Türkiye'de Perakende Tedarik Zinciri Envanter Sistemlerinin Belirsizlik Altında Yönetimi için Planlama Modelleri ve Çözüm Yöntemleri

Proje No: 108K027

Araş. Gör. Mustafa ÇİMEN
Dr. Şule TARIM
Yrd. Doç. Dr. Ayşegül TAŞ
Prof. Dr. Brahim HNICH
Prof. Dr. Ş. Armağan TARIM

HAZİRAN 2010

ANKARA

Önsöz

Bu proje, tedarik zinciri envanter yönetimi alanında önemli teorik sorulara cevap aramanın yanında, ülkemizin en dinamik ve hızlı büyüyen sektörlerinin başında yer alan perakende sektöründe verimlilik artışına dönük yeni teknolojilerin ve stratejilerin geliştirilmesini amaçlamıştır.

TÜBİTAK-1001 araştırma projeleri kapsamında desteklenen bu SOBAG-108K027 projesi Prof. Dr. Ş. Armağan Tarım (Hacettepe Üniversitesi, İşletme Bölümü) tarafından yürütülmüştür. Proje ekibinde yer alan diğer araştırmacılar Prof. Dr. Brahim Hnich (İzmir Ekonomi Üniversitesi, Bilgisayar Mühendisliği Bölümü) ve Yrd. Doç. Dr. Ayşegül Taş (Çankaya Üniversitesi, İşletme Bölümü) ve projeye destek vermiş olan bursiyerler Dr. Şule Tarım ve Araş. Gör. Mustafa Çimen'dir (Hacettepe Üniversitesi, İşletme Bölümü). Proje ekibinde yer almamakla birlikte, Prof. Roberto Rossi (Wageningen University, Hollanda), Prof. Steve Prestwich (UCC, İrlanda), Dr. Mustafa Doğru (Bell Labs, İrlanda), Dr. Ulaş Özen (Bell Labs, İrlanda) tüm proje boyunca çalışmalara katkı veren ve ortak çalışma yürütülen araştırmacılardır.

İçindekiler

Tablo listesi

Özet

Bu proje, tedarik zinciri envanter yönetimi alanında önemli teorik sorulara cevap aramanın yanında, ülkemizin en dinamik ve hızlı büyüyen sektörlerinin başında yer alan perakende sektöründe verimlilik artışına dönük yeni teknolojilerin ve stratejilerin geliştirilmesini amaçlamıştır. Tedarik zinciri yönetimi alanındaki araştırma faaliyetlerinin uygulamada sonuç alabilmesi için iş dünyasının çevresini tanımlayan temel unsurları dikkate alması gereklidir. Bu çerçevede, belirsizlik günümüz iş dünyasını tanımlayan en kritik faktörler arasındadır ve tedarik zinciri yönetimi araştırmalarının merkezinde olmalıdır. Belirsizliğin planlamacılar için karmaşıklık yaratan bir faktör olduğuna dair fikir birliği bulunmasına rağmen mevcut planlama sistemleri veri için nadiren belirlenimsiz (non-deterministic) yaklaşım kullanmaktadır. Bu proje perakende tedarik zincirleri için belirsizlik altında envanter planlaması sorununu ele almıştır. Tedarik zinciri envanter araştırmalarının uygulamada beklenen etkiyi gösterebilmesi için büyük ölçekli stokastik karar problemlerinin çözülmesi gerekmektedir. Ancak küçük boyutlu stokastik problemler bile hesaplama bakımından zorluklar taşımaktadır. Bu kapsamda talep belirsizliği altında beklenen maliyetleri azaltmaya dönük tedarik zinciri envanter politikaları geliştirilmiş, politika parametrelerinin hesaplanmasında kullanılacak yeni tedarik zinciri envanter modelleri kurulmuş ve kurulan modelleri etkin şekilde çözmek için algoritmalar geliştirilmiştir.

Anahtar kelimeler: Perakendecilik; Tedarik Zinciri Yönetimi; Envanter Yönetimi; Stokastik Modelleme

Abstract

Planning Models and Solution Methods for Retail Supply Chain Inventory Management Under Uncertainty in Turkey

This project aims at addressing important theoretical questions in supply chain inventory management, besides developing new technologies and strategies for productivity increase in retail industry, which is one of the most dynamic and rapidly growing industries in Turkey. Supply chain research must address, or at a minimum be compatible with, important aspects of the business environment. Uncertainty is among the most critical elements of the business environment, and requires significant attention by the supply chain management research. While uncertainty is universally recognized as a complicating factor for planners and schedulers, planning and scheduling systems seldom support non-deterministic views of data. This project produces solutions for inventory planning under uncertainty for retail supply chains. It is clear that supply chain research, in order to be relevant, must tackle large-scale stochastic combinatorial decision problems. Stochastic combinatorial models often pose serious computational challenges for even small sized problems. To serve this purpose, we developed new supply chain inventory management policies to minimize expected costs, built new supply chain inventory management models to compute policy parameters and designed efficient algorithms to solve these models.

Keywords: Retailing; Supply Chain Management; Inventory Management; Stochastic Modelling

1. Giriş

Bu proje, tedarik zinciri envanter yönetimi alanında önemli teorik sorulara cevap aramanın yanında, ülkemizin en dinamik ve hızlı büyüyen sektörlerinin başında yer alan perakende sektöründe verimlilik artışına dönük yeni teknolojilerin ve stratejilerin geliştirilmesini amaçlamaktadır. ABD Ticaret Bakanlığı istatistiklerine göre, 2004 yılında ABD'de perakendecilerin taşıdıkları stok değeri 450 milyar dolardır. Stok seviyelerinin servis düzeyinde bir azalmaya yol açmadan indirilmesi sonucunda elde edilecek olan tasarruf rekabetin yoğun olduğu bu sektör için hayati önemdedir. Envanter seviyelerinin yüksekliği sebebiyle perakende sektöründe seyrek olarak stoksuz kalındığı düşünülebilir; oysa ki, yapılan araştırmalar bunun doğru olmadığını göstermiştir. Bir araştırmaya göre sıradan bir günde bir süpermarkette sunulan ürünlerin %8.2'si için stoksuz kalındığı belirlenmiştir. Sözü edilen stoksuz kalma durumu tüm satışların %6.5'na karşılık gelmektedir. ABD için bulunan stoksuz kalma yüzdelerinin Türkiye perakende sektörü için de geçerli olduğunun varsayılması halinde sadece stoksuz kalma sebebiyle uğranılan yıllık satış kaybının 6 milyar doları bulacağı anlaşılır ki bu da konunun önemini açıkca ortaya koymaktadır.

Tedarik zinciri yönetimi alanındaki araştırma faaliyetlerinin uygulamada sonuç alabilmesi için iş dünyasının çevresini tanımlayan temel unsurları dikkate alması gereklidir. Belirsizlik günümüz iş dünyasını tanımlayan en kritik faktörler arasındadır ve tedarik zinciri yönetimi araştırmalarının merkezinde olmalıdır. Belirsizliğin planlamacılar için karmaşıklık yaratan bir faktör olduğuna dair fikir birliği bulunmasına rağmen mevcut planlama sistemleri veri için nadiren belirlenimsiz (non-deterministic) yaklaşım kullanmaktadır. Birçok araştırmacı tedarik zinciri yönetimi alanında belirsizliğin etkin şekilde dikkate alınamayışının olumsuz etkisini giderek daha çok hissettiklerini belirtmekte ve araştırma sonuçlarının uygulamada beklenen etkiyi yaratmamasının temel sebebi olarak bunu görmektedir. Bu proje perakende tedarik zincirleri için belirsizlik altında envanter planlaması sorununu ele almıştır.

Yukarıda yapılan açıklamalar çerçevesinde, tedarik zinciri envanter araştırmalarının uygulamada beklenen etkiyi gösterebilmesi için büyük ölçekli stokastik karar problemlerinin çözülmesi gerekmektedir. Ancak küçük boyutlu stokastik problemler bile hesaplama bakımından zorluklar taşımaktadır. Bu alanda bazı başarılar elde edilmiş olsa da araştırılmayı ve çözülmeyi bekleyen birçok soru bulunmaktadır. Bu araştırma projesinin hedeflerinin başında uygulamacılar için tedarik zinciri envanter sistemlerinde belirsizlikle baş etmeye dönük bir planlama çerçevesinin geliştirilmesi ve belirsizlik altında tedarik zinciri envanter yönetimi araştırmalarına yeni bir yaklaşım kazandırmak bulunmaktadır. Bu hedefe dönük olarak (*i*) talep belirsizliği ve (*ii*) tedarik süresi riski altında beklenen maliyetleri azaltmaya dönük tedarik zinciri envanter politikaları geliştirilmiştir; (*iii*) politika parametrelerinin hesaplanmasında kullanılacak yeni tedarik zinciri envanter modelleri kurulmuştur; (*iv*) kurulan modelleri etkin şekilde çözmek için algoritmalar geliştirilmiştir.

Yukarıda verilen amaçlara Yönetim Bilimi/Yöneylem Araştırması (YB/YA) ve Yapay Zeka (YZ) alanlarında geliştirilen yeni tekniklerin Matematiksel Programlama ve Kısıt Programlama (Constraint Programming) çerçevesi altında bir araya getirildiği hibrit uygulamalarla ulaşılmıştır.

2. Tanımlar ve Kapsam

Perakendecilik, tüketicinin ihtiyaç duyduğu malların çoğunlukla sabit satış noktalarından, küçük parti büyüklüklerinde ve nihai tüketim için pazarlanması faaliyetlerinin bütünüdür. Bu tanım çerçevesinde perakendeciler gıda maddeleri, giyim eşyası, mobilya, ev eşyası, madeni eşya, cam, ilaç ve ıtriyat, kereste ve inşaat malzemesi, kitap ve kırtasiye gibi çok farklı alanlarda faaliyet gösterirler. Bu kapsamdan da açıkca görüleceği üzere sektör olarak ekonomide önemli ağırlıkları vardır.

ABD Ticaret Bakanlığı istatistiklerine göre, 2004 yılında ABD'de perakende sektörünün satış hacmi 3500 milyar doları geçmiştir. Bu satış hacmini gerçekleştirmek için perakende tedarik zincirinde taşınan toplam envanterin değeri 1200 milyar doları aşmıştır. Bu envanterin tedarik zincirindeki dağılımı şöyledir: imalatçı seviyesinde 430 milyar dolar; toptancı/dağıtımcı seviyesinde 320 milyar dolar; ve perakende seviyesinde 450 milyar dolardır. Envanter maliyetlerinin işletmeler için önemli bir maliyet kalemi olduğu düşünüldüğünde aynı servis düzeyinin daha az envanter ile sağlanmasının taşıdığı önem ortaya çıkmaktadır.

Yukarıda verilen envanter rakamlarının yüksekliği sebebiyle perakende sektöründe seyrek olarak stoksuz kalındığı düşünülebilir; oysa ki, yapılan araştırmalar bunun doğru olmadığını göstermiştir. Andersen Consulting tarafından Coca-Cola için yapılan bir araştırma (Coca-Cola Research Council/Andersen Consulting 1996) sıradan bir günde bir süpermarkette sunulan ürünlerin %8.2'si için stoksuz kalındığını belirlemiştir. Reklamı yapılan ürünler için bu değer %15.0'e yükselmektedir. Sözü edilen stoksuz kalma durumu tüm satışların %6.5'na karşılık gelmektedir. Alternatif ürünler sunarak tüketicinin talebinin karşılanması halinde dahi perakenciler toplam satışların %3.1'i kadar bir potansiyel hasılatı yeteri kadar stok tutmamak sebebiyle kaybetmektedirler (Lee 2003).

Türkiye için benzer veriler bulunmasa da perakende sektörünün büyüklüğünden hareketle çıkarılacak sonuç değişmemektedir. "Planet Retail" (http://www.planetretail.net/) tarafından sağlanan en son bağımsız verilere göre, Türk perakende sektörünün cirosu 2006 yılında 137 milyar dolar olarak gerçekleşmiş ve 2010'a kadar sektörün 199 milyar dolara ulaşması beklenmektedir. Türkiye'de perakende sektörü ekonomiye yaklaşık 6.7 milyar dolar tutarında bir katma değer yaratmakta ve yine yaklaşık olarak 2.5 milyon kişiyi istihdam etmektedir ki buna göre perakende sektörünün tüm ekonomi üzerindeki etkisi, toplam Türkiye üretiminin %3.5'i ve istihdamın ise %12'si olacaktır. Bu rakamlar perakende sektörünün Türk ekonomisi üzerindeki ağırlığını açıkça gözler önüne sermektedir ve bu sektörde envanter planlamasıyla sağlanacak tasarrufun firma ve ulusal ekonomi bazında ne derece önemli olduğunu göstermektedir. ABD için bulunan stoksuz kalma yüzdelerinin Türkiye perakende sektörü için de geçerli olduğunun varsayılması halinde sadece stoksuz kalma sebebiyle uğranılan yıllık satış kaybının 6 milyar doları bulacağı anlaşılır ki bu da konunun önemini açıkca ortaya koymaktadır. Stoksuz kalmanın veya gereğinden fazla stok bulundurmanın maliyetli olması hangi ürünün siparişinin ne zaman ve ne miktarda verilmesi gerektiği sorusunu perakende sektörünün temel sorunlarından birisi haline getirmektedir.

Ancak, yukarıda ifade edilen perakendeci seviyesinde envanter planlaması sorununu perakende tedarik zincirinin planlamasından bağımsız olarak düşünmek mümkün değildir. Bugünün küreselleşen piyasalarında yaşanan yoğun rekabet, yeni ürünler için giderek düşen ürün yaşam süreleri ve tüketicilerin yükselen beklentileri firmaların tedarik zincirlerini tekrar düzenlemelerine yol açmıştır ve perakende sektörü bunun bir istisnası değildir. Bu çabanın arkasında “firmaların değil, onların ait oldukları tedarik zincirlerinin rekabet ediyor” olması tesbiti yatmaktadır. Tedarik zinciri yönetimi sistemin etkinliğinde rol oynayan tedarikçiden imalatçıya, ana depolardan dağıtım merkezlerine, perakendecilere ve dükkanlara kadar bütün aktörleri dikkate alarak, tüketicinin gereksinimlerine cevap verecek ürünün üretilmesine ve tüketiciye sunulmasına imkan sağlar. Tedarik zinciri yönetiminde hedef tüm sistemin verimli ve maliyet etkin olmasını sağlamaktır; amaç taşıma ve dağıtım maliyetlerinden ham madde, ara ürün ve nihai ürün stok maliyetlerine kadar katlanılan toplam maliyeti en aza indirgemektir. Tedarik zinciri yönetiminin esas olarak bir şebekeye ilişkin planlama, uygulama ve kontrol merkezli tanımlanması sebebiyle zincire dahil olan tüm firmaların stratejik, taktik ve operasyonel düzeydeki faaliyetlerinin dikkate alınmasını gerektirir. Bu faaliyetler dağıtım şebekesinin tasarımı, üretim planlaması, envanter kontrolü, envanter ve taşımanın koordinasyonu, araç filosu yönetimi gibi geniş bir yelpazede verilecek kararlarla birbirlerine bağlıdır. Bahsedilen bu karar problemlerinin her biri kendi başına çözümü zor problemlerden olan kombinatöryel eniyileme (optimizasyon) problemi sınıfına girmektedir ve hepsinin birlikte eşanlı olarak çözümü mümkün değildir. Önerilen bu proje bir ilk adım olarak sadece perakende tedarik zincirleri için envanter planlaması sorununa gerçekci varsayımlar altında çözüm üretilmeye çalışılmıştır.

Tedarik zinciri yönetimi alanındaki araştırma faaliyetlerinin uygulamada sonuç alabilmesi için iş dünyasının çevresini tanımlayan temel unsurları dikkate alması gereklidir. Belirsizlik günümüz iş dünyasını tanımlayan en kritik faktörler arasındadır ve tedarik zinciri yönetimi araştırmalarının merkezinde olmalıdır. Rassal tüketici davranışı, eksik bilgi, hatalı veri, açık olmayan veya eksik tanımlamalar “tedarik zinciri yönetiminde belirsizlik” temasını vurgulayan rassal faktörlerden yalnızca birkaçıdır. Ürün yaşam sürelerinin giderek kısalması kullanılabilecek tarihsel verinin de azalmasına ve böylece belirsizliğin artmasına yol açmaktadır. Bu durum ise özellikle perakende tedarik zincirlerinde belirsizliğin modellenmesinin önemini artırmaktadır. Mevcut rekabetçi ortamda belirsizliği yönetmeye yardımcı olacak ve değişen piyasa koşullarına hızlı ve güvenilir şekilde tepki vermeyi sağlayacak yeni yaklaşımlara perakende sektörünün ihtiyacı vardır (Fisher, Raman and McClelland 2000).

Belirsizliğin planlamacılar için karmaşıklık yaratan bir faktör olduğuna dair fikir birliği bulunmasına rağmen mevcut planlama sistemleri veri için nadiren belirlenimsiz (non-deterministic) yaklaşım kullanmaktadır (Wu, et al. 1999). Buna ilişkin bir örnek olarak İşletmecilik alanından Kurumsal Kaynak Planlaması (Enterprise Resource Planning, ERP) ve diğer kurumsal bilgi ve planlama sistemlerinin nadiren rassallığa ilişkin bilgi kullanmaları verilebilir. Yine Wu ve diğerlerine göre, birçok araştırmacı tedarik zinciri yönetimi alanında belirsizliğin etkin şekilde dikkate alınamayışının olumsuz etkisini giderek daha çok hissettiklerini belirtmekte ve araştırma sonuçlarının uygulamada beklenen etkiyi yaratmamasının temel sebebi olarak bunu görmektedir. Bu bağlamda, bu proje perakende tedarik zincirleri için envanter planlaması sorununa belirsizlik faktörlerini hesaba katarak çözüm üretmeye çalışmıştır.

Yukarıda yapılan açıklamalar çerçevesinde, tedarik zinciri envanter araştırmalarının uygulamada beklenen etkiyi gösterebilmesi için büyük ölçekli stokastik karar problemlerinin çözülmesi gerekmektedir. Ancak küçük boyutlu stokastik problemler bile hesaplama bakımından zorluklar taşımaktadır. Bu alanda bazı başarılar elde edilmiş olsa da araştırılmayı ve çözülmeyi bekleyen birçok soru bulunmaktadır. Bu araştırma projesinin hedeflerinin başında uygulamacılar için tedarik zinciri envanter sistemlerinde belirsizlikle baş etmeye dönük bir planlama çerçevesinin geliştirilmesi ve belirsizlik altında tedarik zinciri envanter yönetimi araştırmalarına yeni bir yaklaşım kazandırmak bulunmaktaydı. Bu hedefe ulaşmak için talep belirsizliği ve tedarik süresi riski altında beklenen maliyeti azaltmaya dönük tedarik zinciri envanter politikaları geliştirilmiş; strateji parametrelerinin hesaplanmasında kullanılacak yeni tedarik zinciri envanter modelleri kurulmuştur; kurulan modelleri etkili şekilde çözmek için algoritmalar geliştirilmiştir.

3. İlgili Yazın

Bu araştırma projesinin ana temasını oluşturan perakende tedarik zincirleri üzerine yayınlanan araştırmalar ve istatistikler konunun makro ve mikro ölçekte önemini açık şekilde ortaya koymaktadır. ABD'de perakende sektörünün satış hacmi 2004 yılında 3500 milyar doları aşmıştır (ABD Ticaret Bakanlığı istatistikleri, http://www.census.gov/econ/www/). Bu satış hacmini gerçekleştirmek için 1200 milyar dolarlık envantere ihtiyaç duyulmuştur. Bu envanterin 450 milyarlık kısmı tedarik zincirinin en alt seviyesi olan perakendeciler tarafından tutulmuştur. Diğer yandan envanter rakamlarının yüksekliğine rağmen perakende sektöründe stoksuz kalınmayla çok sık karşılaşılmaktadır. Andersen Consulting tarafından Coca-Cola için yapılan bir araştırma (Coca-Cola Research Council/Andersen Consulting 1996) sıradan bir günde bir süpermarkette sunulan ürünlerin %8.2'si için stoksuz kalındığını belirlemiştir. Reklamı yapılan ürünler için bu değer %15.0'e yükselmektedir. Sözü edilen stoksuz kalma durumu tüm satışların %6.5'na karşılık gelmektedir. Alternatif ürünler sunarak tüketicinin talebinin karşılanması halinde dahi perakenciler toplam satışların %3.1'i kadar bir potansiyel hasılatı yeteri kadar stok tutmamak sebebiyle kaybetmektedirler (Lee 2003). Türkiye için benzer veriler bulunmasa da perakende sektörünün büyüklüğünden hareketle çıkarılacak sonuç değişmeyecektir.

Fisher ve diğerlerine göre son 20 yılda indirimli satışların toplam satışlar içindeki payının %8 seviyesinden %33'lere yükselmesinin temel nedeni perakende sektöründe tüm gerekli verilerin bilgisayar ortamında tutuluyor olmasına rağmen bunların uygun şekilde işlenip kullanılmamasıdır. Bu çalışmada özellikle perakende sektöründe ortalama kar marjlarının %2-3 seviyelerinde olduğu hatırlatılıp, bunun yanında stoksuz kalma sebebiyle kaybedilen hasılatın %10'unun ne kadar büyük olduğu gözler önüne serilmektedir. Fisher ve diğerlerinin (Fisher, Raman and McClelland 2000) perakende tedarik zincirlerinin performanslarını artırmak için iyileştirmenin gerekli görüldüğü üç alandan birisi belirsizlik altında envanter planlamasıdır.

Chen ve diğerleri (Chen, Frank and Wu, US retail and wholesale inventory performance from 1981 to 2004 2007) de perakende sektöründe yaptıkları ve 23 yıllık veriyi kapsayan araştırmaları sonucunda yüksek envanter seviyesi ile çalışan firmaların hisse senedi ortalama getirilerinin uzun vadede düşük olduğunu görmüşlerdir. Bir diğer çalışmalarında (Chen, Frank and Wu, What Actually Happened to the Inventories of American Companies Between 1981 and 2000? 2005) ise 1981-2000 arasındaki 19 yıllık dönemde imalat sektöründe ortalama envanter taşıma süreleri 96 günden 81 güne inerken (yıllık %2 seviyesinde bir azalış), perakende sektöründe stokta tutulan nihai ürünlerde hiçbir azalış gözlenmemiştir. Bu sonuçlar perakendeci seviyesinde envanter planlamasında bir başarı kazanılamadığını göstermektedir.

Enslow tarafından yapılan ve Aberdeen Group tarafından yayınlanan (Enslow 2004) raporda günümüzde firmaların %60'dan fazlasının envanter planlamasında uygun planlama araçlarını kullanmadıklarını ve bu firmaların sahip olmaları gerekenden %15-30 daha fazla envanterle çalıştığı ve servis düzeylerinin de düşük olduğu belirtilmiştir.

Tedarik zinciri yönetimi alanındaki araştırma faaliyetlerinin uygulamada sonuç alabilmesi için iş dünyasının çevresini tanımlayan temel unsurları dikkate alması gereklidir. Belirsizlik günümüz iş dünyasını tanımlayan en kritik faktörler arasındadır ve tedarik zinciri yönetimi araştırmalarının merkezinde olmalıdır. Wu ve diğerlerine (Wu, et al. 1999) göre belirsizliğin planlamacılar için karmaşıklık yaratan bir faktör olduğuna dair fikir birliği bulunmasına rağmen mevcut planlama sistemleri veri için nadiren belirlenimsiz (non-deterministic) yaklaşım kullanmaktadır. Buna ilişkin bir örnek olarak İşletmecilik alanından Kurumsal Kaynak Planlaması (Enterprise Resource Planning, ERP) ve diğer kurumsal bilgi ve planlama sistemlerinin nadiren rassallığa ilişkin bilgi kullanmaları verilebilir. Yine Wu ve diğerlerine göre, birçok araştırmacı tedarik zinciri yönetimi alanında belirsizliğin etkin şekilde dikkate alınamayışının olumsuz etkilerini giderek daha çok hissettiklerini belirtmekte ve araştırma sonuçlarının uygulamada beklenen etkiyi yaratmamasının temel sebebi olarak görmektedir. Bu nokta gerçekleştirilen araştırmanın esasını teşkil etmekte ve proje perakende tedarik zincirleri için envanter planlaması sorununa belirsizlik faktörlerini dikkate alarak çözüm üretmektedir.

Belirsizlik altında envanter planlamasına ilişkin literatür incelendiğinde ise (bakınız (Graves, Rinnooy Kan and Zipkin 1993), (Porteus 2002), (Zipkin 2000) ve (de Kok and Graves 2003)) birçok farklı envanter politikasının uygulamada yer bulduğu görülecektir. Bunlar arasında (s,S), (s,Q), (R,S) gibi sipariş noktası (s), sipariş miktarları (Q), sipariş aralıkları (R), envanter yükseltme seviyesi (S) parametrelerini sabitleyerek envanter kontrolü yapan politikalar öne çıkmaktadır.

Tek ürün ve tek stok noktası durumu için –doğrusal elde stok tutma ve stoksuz kalma maliyetleri ile sabit sipariş maliyeti varsayımı altında– Scarf (Scarf 1959) tarafından ispat edildiği üzere (s,S) politikası optimal maliyetli politikadır. Ancak bu varsayımlar, tanım gereği, hemen hiçbir perakende tedarik zincirine uygun değildir. Bunun birçok sebebi bulunsa da temelde birçok stok ve üretim merkezinin dikkate alınması gereği ile birden çok ürün için planlama yapılması ihtiyacı (s,S) politikasının artık optimal politika olma özelliğini ortadan kaldırır. Bu gevşetilmiş varsayımlar altında (s,S) politikasının pratikte uygulanabilirliği de kolay görünmemektedir. Hatta tek ürün tek stok noktası için planlama yapmak gerektiği zaman bile birçok firma ileriye dönük olarak hiçbir planlamaya imkan vermeyen (s,S) politikası yerine (R,S) tipi politikaları tercih etmektedir (bakınız (Silver, Pyke and Peterson, Inventory Management and Production Planning and Scheduling 1998)). Envanter alanında temel kaynak niteliğinde olan Silver ve diğerlerinde işaret edildiği gibi özellikle birden çok ürün için eşanlı sipariş vermenin sipariş maliyetlerini düşürdüğü durumlarda koodinasyon imkanı sunan (R,S) politikasının diğer politikalara üstün olduğu ifade edilmiştir. Bu açıklamalar çerçevesinde (R,S) politikası perakende envanter sistemleri için cazip bir planlama platformu sunmaktadır.

Yukarıda yapılan açıklamalardan anlaşılacağı üzere perakende tedarik zinciri envanter planlamasının temel karakteristiğini çok ürünlülük, çok merkezlilik ve belirsizlik ifade eder. Belirsizliği konu alan envanter yazını incelendiğinde hemen tamamının durağanlık (stationarity) varsayımında bulunduğu görülür. Oysa bir çok sektörde –bunların başında perakende sektörü gelmektedir– talep durağan olmayan (non-stationary) bir desen gösterir. Dolayısıyla yapılan araştırmaların uygulamada beklenen etkiyi yaratabilmesi için belirsizlik durağan olmayan bir yapıda tanımlanmalıdır. (R,S) politikası ile ilgili literatür incelendiğinde durağan olmayan talep varsayımında bulunan ilk yayının Silver'a (Silver, Inventory control under a probabilistic time varying demand pattern 1978) ait olduğu görülür.

Silver (Silver, Inventory control under a probabilistic time varying demand pattern 1978) stokastik dinamik parti büyüklüğü problemine sezgisel bir yaklaşım önerir. Bu çalışma, dönemsel ortalama maliyeti hesaplayan Silver-Meal (Silver and Meal, A heuristic for selecting lot size requirements for the case of a deterministic time-varying demand rate with discrete opportunities for replenishment 1973) sezgisel yaklaşımının stokastik sürümüdür. Silver'in sezgisel yaklaşımı üç adımdan oluşmaktadır. İlk adım, ne zaman sipariş verileceğini, ikinci adım siparişin kaç dönemi kapsayacağını, üçüncü adım ise sipariş miktarını bulmaya yöneliktir. Bu çalışmayı takip eden bir diğer sezgisel yöntem ise Askin (Askin 1981) tarafından önerilmiştir.

Durağan olmayan talep altında (R,S) politikası için ilk matematiksel programlama modeli (statik-dinamik belirsizlik stratejisi olarak adlandırılmıştır) Bookbinder ve Tan (Bookbinder and Tan 1988) tarafından önerilmiştir. Bu stratejiye göre planlama ufku başında tedarik dönemleri Wagner-Whitin (Wagner and Whitin 1958) tipi bir dinamik programlama modeli ile belirlenmekte ve belirlenen bu dönemler için envanter yükseltme düzeyleri doğrusal programlama ile bulunmaktadır. Bookbinder ve Tan'nın çalışması, sipariş dönemlerini ve bu dönemlerdeki beklenen sipariş miktarlarını iki adımda hesaplayan bir diğer sezgisel yaklaşımdır.

Bookbinder ve Tan'ın kullandıkları varsayımlar altında problemin ilk optimal çözümü Tarim ve Kingsman (Tarim and Kingsman, The stochastic dynamic production/inventory lot-sizing problem with service-level constraints 2004) tarafından geliştirilen kesinlik dengi (certainty equivalent) karışık tamsayı programlama (MIP) modeli ile bulunmuştur. Bu formülasyon yardımıyla optimallikten taviz verilmeden sipariş dönemleri ve envanter yükseltme seviyeleri eşanlı olarak bulunabilmektedir. Bookbinder-Tan ve Tarim-Kingsman çalışmalarının ortak varsayımları her ikisinde de stoksuz kalma maliyeti yerine servis düzeyi kısıtının kullanılmasıdır. Tarim ve Kingsman bu çalışmalarını takiben servis düzeyi kısıtı yerine stoksuz kalma maliyetini dikkate alarak (R,S) politikası için politika parametrelerini hesaplayacak doğrusal olmayan bir model kurmuşlar ve daha sonra bu model için parçalı doğrusal bir tamsayı programlama modeli önermişlerdir (Tarim and Kingsman, Modelling and Computing Policies for Inventory Systems with Non-Stationary Stochastic Demand 2006). Geliştirilen bu kombinatöryel optimizasyon modellerinin önemli bir envanter planlama problemine çözüm önermesine karşın uygulamada ancak küçük ölçekli uygulamaların sonuçlandırılması mümkün olmuştur. Hesaplamada karşılaşılan bu güçlüğü aşmak üzere Tarim ve Smith (Tarim and Smith, Constraint Programming for Computing Non-Stationary (R,S) Inventory Policies 2008) Yapay Zeka alanında geliştirilen Kısıt Programlama (Constraint Programming, CP) tekniğini kullanarak çok daha hızlı çözüm elde etmeyi başarmışlardır. Bunu takiben Tarim ve diğerleri tarafından geliştirilen bir MIP/CP hibriti modelle durağan olmayan talep altında tek ürün ve tek stok noktası için tüm pratik uygulamalarda makul sürede planlama yapmak mümkün olmuştur (Tarim, Rossi, et al. 2008).

Buraya kadar verilmiş olan literatürden de açıkça görüleceği üzere çok ürün, eşanlı siparişler, olasılıksal tedarik süreleri ve çok stok/üretim noktasının dikkate alındığı (R,S) tipi politikalar üzerine çalışmalar ise ilk defa bu proje kapsamında yapılmıştır. Bu araştırma projesinin uygulamada ve teoride büyük bir boşluğu doldurduğu açıktır.

Geliştirilen modeller en zor optimizasyon problemi sınıfına giren doğrusal olmayan stokastik kombinatöryel optimizasyon modelleri türündedir. Bu modeller için kullanılan yaklaşımlar arasında dekompozisyon algoritmaları (örneğin Tarim ve Miguel (Tarim and Miguel, A Hybrid Benders' Decomposition Method for Solving Stochastic Constraint Programs with Linear Recourse 2006)), senaryo indirgeme yöntemleri (örneğin Kleywegt ve diğerleri (Kleywegt, Shapiro and de Mello 2002) ve Santoso ve diğerleri (Santoso, et al. 2005)), sezgisel yaklaşımlar (örneğin Van Hentenryck ve Bent (Van Hentenryck and Bent 2006)) ve hibrit yöntemler (örneğin (Hnich, et al. 2004)) bulunmaktadır.

4. Türkiye’de Perakendecilik ve Envanter Yönetimi

Türkiye’de faaliyet gösteren perakendeci firmaların tedarik zinciri envanter planlamasına ilişkin öncelikli sorunlarını ve kullanmakta oldukları karar destek sistemlerini belirlemek, ve mevcut envanter planlama yaklaşımlarının bir fotoğrafını çekebilmek amacıyla br saha çalışması gerçekleştirilmiştir. Bu çalışmanın kapsamı sektörü temsilen süpermarket düzeyindeki perakendeci firmalarla sınırlandırılmıştır. Perakende sektöründe faaliyet gösteren firmaların çeşitliliği ve sayılarının çokluğu, öngörülen proje bütçesi ve proje süresi kısıtları altında, bu tür bir sınırlandırmaya gidilmesini zorunlu kılmıştır. Saha çalışmasına konu edilecek olan firmalar “Soysal Türkiye Perakende Kataloğu -- 2005” (S. Soysal: İstanbul, 2005) süpermarketler bölümünde listelenen firmalardan seçilmiştir.

İlk olarak perakende tedarik sistemlerine ilişkin mevcut karar destek sistemlerinin envanterinin çıkarılmasına dönük olarak kapsamlı bir araştırma yapılmıştır. Araştırma sonuçları mevcut sistemlerin fonksiyonelliklerini özetleyen bir rapor (Rapor-I) halinde ekte (Ek-1) sunulmuştur. Bu araştırma neticesinde halihazırda kullanılan sistemlerinin (bir istisna haricinde) envanter planlaması sürecinde hiçbir belirsizlik unsurunu dikkate almadıkları ortaya çıkmıştır. Talep belirsizliğini dikkate alan tek bir yazılımında sadece durağan talep varsayımı altında çalıştığı görülmüştür. İncelemeye ilişkin ayrıntılar aşağıda özetlenmiştir.

Görüşmelerde sorulan başlıca sorular şunlardır:

- İlgili ERP yazılımı stokta tutulan kalem bazında stok takibi yapabiliyor mu?
- Yazılım talep tahmini yapabiliyor mu? Cevap eğer evet ise talep tahmini tek dönemli (tek bir hafta/ay/yıl) mi yoksa çok dönemli (birden çok hafta/ay/yıl) mi yapılıyor?
- Yapılan bu tahminler nokta tahmini şeklinde mi gerçekleşiyor, yoksa tahminleme hataları da tespit edilebiliyor mu?
- Yazılım kullanıcıya ne zaman ve ne kadar sipariş verileceğini söyleyebiliyor mu?
- Yazılım güvenlik stoğu hedefi sunabiliyor mu?
- Güvenlik stoğu belirlemede kullanılan yöntemler/kurallar/yaklaşımlar nelerdir?
- Güvenlik stoğu düzeyi optimal biçimde hesaplanabiliyor mu?

Görüşmelerde, yukarıdaki soruların ışığında mevcut ERP yazılımlarının hangi ihtiyaçları ne ölçüde giderebildikleri belirlenmeye çalışılmıştır. Bu görüşmelerin sonucunda elde edilen bilgiler aşağıda Tablo 1'de toplu olarak sunulmuştur.

Görüşmeler yapılmadan önce mevcut ERP yazılımlarının stok takibi ve talep tahmini yapabildikleri tahmin ediliyordu. Fakat birçoğunun gerek deterministik gerekse stokastik planlama yönünden yetersiz kalacağı düşünülmekteydi. Görüşmeler sonucunda sadece SAS şirketinin yazılımının durağan stokastik planlama yapabildiği tespit edilmiştir. Birkaç büyük yazılım firmasının deterministik planlama yapabildikleri, geriye kalan firmaların ise tahmin yapabilmekten öteye gidemedikleri ortaya çıkmıştır.

Araştırma dâhilindeki hiçbir şirketin durağan olmayan talep altında stokastik planlama yapamaması, bu yönde yapılacak çalışmaların önemini ve gerekliliğini göstermiştir. Bununla birlikte çok uluslu olan yazılım şirketlerinin ürünlerinin genellikle büyük şirketlerin kullandığı, yerel bazda olan yazılım şirketlerinin ise küçük ve orta büyüklükteki şirketlere yönelik ürün geliştirdikleri tespit edilmiştir.

ERP yazılımların birçoğu öncelikle imalat sanayiinde faaliyet gösteren işletmelerin ihtiyaçlarını karşılamaya dönük olarak hazırlanmıştır. Perakende sektörüne yönelik çözümler de, bu paketlere ilave edilmiş veya ayrı bir paket halinde piyasaya sunulmuştur. Bu yüzden birçoğunun eksiği bulunması ile birlikte günden güne bu eksiklerin giderilmeye çalışılmakta olduğu görülmüştür.

Proje kapsamında ele alınan perakende tedarik zinciri envanter yönetimi optimizasyon modelleri durağan olmayan talep varsayımını taşımaktadır. Bu raporda sunulan bulgular, geliştirilen modellerin ve çözüm yöntemlerinin bahsi geçen çabalara büyük destek vermenin de ötesine geçeceği ve bu tartışmaların merkezine oturacağı bekletimizi destekler niteliktedir.

				Planlama		Planlama	
Tablo 1		Öngörü		**Deterministik**		**Stokastik**	
	İzleme	Nokta	Hata	Heuristik	Optimal	Durağan	Durağan Olmayan
SAP R/3	✓	✓	✓	✓	✓	×	×
Oracle	✓	✓	✓	✓	✓	×	×
INFOR ERP(Baan)	✓	✓	✓	×	×	×	×
IFS Application	✓	✓	✓	×	×	×	×
SAGE MAS 500	✓	✓	✓	×	×	×	×
CANIAS ERP	✓	✓	✓	×	×	×	×
SAS	✓	✓	✓	✓	✓	✓	×
Fusion@6	✓	✓	×	×	×	×	×
LOGO Unity	✓	✓	×	×	×	×	×
OBASE	✓	✓	×	×	×	×	×
Wolvox	✓	✓	✓	×	×	×	×
Micro	✓	✓	×	×	×	×	×
Compiere	✓	×	×	×	×	×	×
ERP5	✓	×	×	×	×	×	×
GNU ERP	✓	×	×	×	×	×	×
SQL Ledger	✓	×	×	×	×	×	×
OFBiz	✓	×	×	×	×	×	×
Opentaps	✓	×	×	×	×	×	×
Openbravo	✓	×	×	×	×	×	×
OpenERP	✓	×	×	×	×	×	×
OpenPro	✓	×	×	×	×	×	×

Türkiye'nin süpermarket zincirlerinin tamamına yakınıyla görüşülmüş ve envanter planlaması için kullandıkları sistemler hakkında bilgi edinilmiştir. Görüşmeler sektörde belirsizlik altında envanter planlamasına dönük herhangi bir sistematik veya "ad hoc" uygulamanın olmadığını göstermiştir. Ayrıca firmalar ellerinde bulunan yazılımların mevcut fonksiyonlarını dahi çoğu zaman kullanmadıklarını, sadece birer envanter seviyesi izleme aracı olarak bu yazılımlardan istifade ettiklerini söylemişlerdir. Buna ilişkin rapor (Rapor-II) ekte (Ek-2) sunulmuştur.

Bu raporun amacı Türkiye perakende sektöründe faaliyet gösteren süpermarketlerin tedarik zinciri yönetimi için kullandıkları envanter yönetim yazılımlarını belirlemek ve tedarik zinciri yönetiminde bu yazılımlardan ne ölçüde faydalandıklarını tespit etmektir.

Araştırma kapsamında 31 firma ile görüşme yapılmıştır. Elde ettiğimiz bulgular temelde sektör tarafından kullanılan yazılımların etkin şekilde belirsizlikle baş edebilecek fonksiyonelliğe sahip olmadıklarını ve çoğu zamanda bu yazılımların sunduğu temel fonksiyonların bile firmalar tarafından uygun organizasyonel süreçler ve uzmanlık bulunmadığı için kullanılmadığını göstermektedir.

Genel olarak baktığımızda, sadece 2 süpermarket zinciri yazılımlarının talep tahmini yapabildiğini ve tahmin hatasına ilişkin bilgi sunduğunu belirtmiştir. 9 süpermarket yazılımlarının sadece talep tahmini yapabildiğini bildirmiştir. 1 süpermarket ise ERP yazılımı kullanmadıklarını belirtmiştir.

Yukarıdaki tabloya baktığımızda hiçbir süpermarketin deterministik planlama yapmadığı görülmektedir. Buradan yola çıkarak, günümüz Türkiye perakende sektöründe faaliyet gösteren süpermarketlerde yönetici deneyimlerinin, sipariş kararlarında hala en önemli faktör olduğuna ulaşabiliriz. Ayrıca yaptığımız görüşmelerde ulaştığımız bir başka sonuç ise, yazılımların birçoğunun süpermarketlerde çok görülen mevsimsel satış, indirim, kampanya gibi durumlara özel çözümlerinin olmayışıdır. Bu da sipariş miktarlarının belirlenmesinde hala yöneticilerin neden kişisel tecrübelerine dayanarak karar verdiklerini bir noktaya kadar açıklamaktadır.

Süpermarketlerin belirsizlik altında ihtiyaçlarına cevap verecek yeni ve kapsamlı envanter planlama sistemlerine ihtiyaç olduğu açıktır. Öte yandan bu yazılımların var olması halinde dahi etkin şekilde kullanılabileceğine dair bir bulgu yoktur. Bu noktada yöneticilerin gerekli eğitimden geçirilmesi, bilinçlendirilmesi ve teknik ekibin yazılımların fonksiyonellikleri hakkında bilgilendirilmesi gerekliliği ortadadır.

Bu incelemenin sonucunda karar destek sistemi yazılımı üreten firmaların durağan olmayan belirsiz talep altında planlamaya yapmaya dönük ürünlerinin bulunmadığı ve süpermarket zincirlerinin de bu hususta tanımlı bir politika izlemedikleri görülmüştür.

Bu sonuçlar araştırma projemiz kapsamında geliştirilen modellerin ve çözüm tekniklerinin önemli bir boşluğu dolduracağına işaret etmektedir.

5. Envanter Yönetimi için Yeni Yaklaşımlar

Perakende tedarik zincirlerinde belirsizlik altında envanter planlamasının taşıdığı teorik ve teknik güçlüklere çözüm üretmek bu projenin temel amacıdır. Bu bölümde bu güçlüklerin nasıl ele alındığı ve kullanılan temel modelleme ve çözüm paradigmalarının neler olduğu konusunda teknik ayrıntıya girmeden özet bilgi sunulmuştur. Yapılan her bir çalışmanın ayrıntılı şekilde sunumu eklerde verilmiştir.

Proje kapsamında yürütülen teorik çalışmalar iki ana grupta tasnif edilebilir. Bu gruplardan ilki *belirsizlik altında karar almaya dönük olarak yeni teknikler*in geliştirilmesiyle ilgilidir. Bu gruptaki çalışmalar genel amaçlı olup daha çok belirsizliğin rol oynadığı karar problemlerinin modellenmesine ve çözümüne ilişkin kullanıcıya platformlar sunmaktadır. İkinci grup çalışma da ise *belirsizlik altında envanter yönetimi ve planlaması* problemi dikkate alınmıştır. Aşağıda ilk olarak geliştirilen, belirsizliği yönetmeye dönük genel amaçlı, teknikler ve platformlar sunulmuş, ardından envanter ve tedarik zinciri yönetimi alanındaki çalışmalarımız özetlenmiştir.

Burada özetlenecek olan ilk çalışma belirsizlik altında karar almaya dönük olarak tasarlanmış bir teknik olan “stokastik kısıt programlama” (Stochastic Constraint Programming, SCP) için etkin çözüm yöntemlerinin geliştirilmesini hedeflemektedir. Kısıt programlama Yapay Zeka alanında geliştirilen bir tekniktir ve kombinatöryel karar problemlerinin çözümünde kullanılmaktadır. SCP’nin Yöneylem Araştırması temel tekniklerinden olan Matematiksel Programlama’ya karşı en belirgin üstünlüğü doğrusal olmayan stokastik kombinatöryel eniyileme problemlerinin modellenmesine ve çözümüne imkan vermesidir. Perakende tedarik zinciri envanter planlamasının bu tür bir stokastik kombinatöryel optimizasyon problemi olması bu modelleme yaklaşımını proje çalışması için anahtar hale getirmiştir. Stokastik Kısıt Programlama yardımıyla kurulacak modellerin perakende tedarik zinciri envanter planlaması alanında uygulanabilirliğini sağlamak için etkin çözüm algoritmalarının geliştirilmesi gerekmiştir. Bu algoritmaların geliştirilmesi araştırma projesinin teorik kapsamını oluşturan ana iş paketlerinde ele alınmıştır. “Cost-Based Domain Filtering for Stochastic Constraint Programming” başlığı altında yapılan bu çalışmalarda Yapay Zeka ve Yöneylem Araştırması alanından seçilen bazı teknikler melez olarak karar değişkenlerinin tanım kümelerinin filtre edilmesinde kullanılmış ve böylece çözüm süreleri önemli ölçüde kısaltılmıştır. Lecture Notes in Computer Science, LNCS 5202’de yayınlanan ve TÜBİTAK desteğini belirten bu çalışmanın tam metni Ek-3’de sunulmuştur. Bu çalışma araştırma projesinin konusu olan talep belirsizliği altında envanter yönetimi gibi stokastik kombinatöryel optimizasyon problemlerinin çözümü için kullanılabilecek temel bir tekniğin etkin şekilde çözümünü sağlamıştır. Makalenin giriş kısmında genel motivasyon ve çalışma konusu verilmiştir. İkinci bölümde “kısıt sağlama problemi” (constraint satisfaction problem) nin stokastik versiyonu formal olarak tanımlanmış ve semantiği verilmiştir. Bu semantiğin optimizasyon amaçlı olan kısıt sağlama problemi içinde geçerli olduğu ifade edilmiştir. Üçüncü bölümde belirsizlik taşıyan bir problemde belirsizliği dikkate alarak çözüm almanın sağlayacağı fayda fonksiyonundaki artışa ilişkin bazı önermelerde bulunulmuştur. Bu önermeler Jense ve Edmundson-Madansky tipi sınırların bir ifadesidir. Dördüncü bölüm kısıt programlama için yeni geliştirilen bir kısıt olan “global optimization chance-constraint” in sunumunu yapmaktadır. Bu kısıt özellikle servis düzeyi altında envanter planlaması göz önünde bulundurularak tasarlanmıştır. Bu bölümde ayrıca tanımlanan “global optimization chance-constraint” için etkin hesaplama yöntemleri önerilmiştir. Beşinci bölümde “sırt çantası” probleminin (knapsack problem) stokastik varyansyonu ve bir stokastik çizelgeleme problemi üzerinden önerilen tekniklerin etkinliği sınanmıştır. Sonuçlar incelenen problemlerin çozüm süresinde 100 kata varan iyileşmelerin olduğunu göstermiştir.

Benzer amaçlı olarak yürütülen bir başka çalışma ise “A Steady-State Genetic Algorithm with Resampling for Noisy Inventory Control” başlıklı çalışmadır. Bu çalışmada stokastik kısıt programlama yerine Yapay Zeka alanından bir başka teknik olan Genetik Algoritmalar kullanılmıştır. Projede çalışılan (R,S) envanter kontrol modelinin çözümü ve böylece sipariş kontrol parametrelerinin hesaplamasına dönük olarak yeni bir teknik önerilmiştir. Yapılan nümerik testler geliştirilen bu tekniğin stokastik envanter planlaması için son derece etkili olduğunu göstermiştir. Bu çalışma Lecture Notes in Computer Science serisinde LNCS 5199 numara ile yayınlanmış ve tam metni Ek-4’de sunulmuştur. Yine bu çalışma da makalede TÜBİTAK desteğini not etmektedir. Makalenin ilk bölümünde Genetik Algoritmalar ve diğer Evrimsel Hesap Algoritmalarına ilişkin bir yapısal problemden bahsedilmiş ve buna ilişkin yeni bir çözüm önerisinde bulunulmuştur. Bu problem kromozomların uygunluğunun direkt olarak belirlenememesi, fakat örneklem üzerinden bulunmaya çalışılmasıdır. Bu yaklaşım kaçınılmaz olarak “gürültü”ye (bir tür hataya) sebep olmaktadır. Bu bölümde ayrıca ilgili literatüre yer verilmiştir. İkinci bölümde önerilen algoritma verilmiştir. Bu algoritmaya “greedy average sampling” adı verilmiştir. Bu algoritmada hesaplamada ek olarak yeni bir parametrenin belirlenmesi gerekmektedir. Ancak diğer tekniklerde olduğu gibi büyük ana kütle ile çalışma ve gürültünün dağılımı üzerine kısıtlayıcı varsayımlarda bulunma gerekliliği yoktur. Üçüncü bölümde geliştirilen algoritmanın denemelerinin yapılacağı stokastik envanter problemi tanımlanmıştır. Dördüncü bölümde nümerik deneylere ilişkin sonuçlar sunulmuştur. Deneyler önerilen algoritmanın literatürde bilinen ve yaygın olarak kullanılan dört tekniğin hepsinden iyi sonuçlar verdiğini göstermiştir. Bu çalışma proje kapsamında ele alınan envanter problemlerinden bir tanesi için önerilen genel bir tekniğin etkinliğini göstermiştir. Diğer bir deyişle, bu çalışmada incelenen bir envanter problemine sadece bu probleme özgü olmayan, fakat genel amaçlı olarak kullanılabilecek bir teknikle çözüm bulunmuştur.

Ek-4’te sunulan çalışmanın bir varyasyonu ise Pekiştirmeli Öğrenme (Reinforcement Learning) algoritmaları ve Genetik algoritmalarına bir alternatif olarak Cultural algoritmaların denenmesi üzerinedir. SARSA gibi pekiştirmeli öğrenme algoritmaları ve genetik algoritmalar “kısmi olarak gözlenebilen Markov karar süreçleri”’nin (Partially Observable Markov Decision Processes) analizinde kullanılan temel tekniklerdir. Bu tür Markov süreçleri stokastik envanter planlamasında önemli rol oynamaktadırlar. Bu yüzden bahsi geçen süreçlerin analizinin etkin olarak yapılabilmesi stokastik envanter sistemlerinin de kolaylıkla çözümü anlamına gelecektir. “A Cultural Algorithm for POMDPs from Stochastic Inventory Control” başlıklı çalışmada (R,S) envanter kontrol politikası POMDP olarak modellenmiş ve çözümünde ilk olarak SARSA ve genetik algoritmalar kullanılmıştır. Ardından bu makalede önerilen “cultural” algoritma ile envanter kontrol parametrelerinin hesaplaması yapılmıştır. Elde edilen sonuçlar “cultural” algoritmanın her iki alternatiften de iyi sonuç verdiğini göstermiştir. Bu çalışma yukarıda bahsedilen çalışmalar gibi belirsizlik altında karar almaya dönük olarak geliştirilen bir stokastik kombinatöryel optimizasyon tekniği olarak görülmelidir. Aynı zamanda genel amaçlı olan bu tekniğin stokastik bir envanter problemine etkin çözüm sağladığı gösterilmiştir. Çalışma Lecture Notes in Computer Science, LNCS 5296'da yayınlanmıştır. Çalışmanın tam metni Ek-5'te verilmiştir.

Envanter yönetimi alanında gerçekleştirilen ve talep belirsizliğini dikkate alan çalışmaların çok büyük bir kısmı durağan talep varsayımında bulunmaktadır. Doğası itibarıyla perakendecilik sektöründe talep dönemsellik göstermektedir. Dolayısıyla perakende sektöründe kullanılacak olan envanter planlama sistemlerinin bu talep karakteristiklerini göz önüne alması gerekmektedir. Proje başlangıcında gerçekleştirilen saha çalışmasının bulguları çerçevesinde, daha önce yapılması planlanan "halihazırda kullanılan stratejilerin performanslarının ölçülmesi" çalışmasında süpermarket zincirlerinde belirsizlikle baş etmeğe dönük tanımlı ve genel geçer herhangi bir politika tespit edilemediği için süpermarket zincirlerinin uygulamaları yerine bu zincirler için üretilen yazılımların kullandıkları politikaların performansının incelemesi yapılmıştır. Mevcut envanter planlama yazılımlarına bakıldığında sadece SAS'ın, o da durağan talep varsayımı altında, talep belirsizliğini dikkate aldığı ortaya çıkmıştır. Bu çerçevede "durağana indirgenmiş durağan olmayan talep" yaklaşımının performansı optimal politika ile karşılaştırılmıştır. Böylece planlama yazılımları arasında en gelişmiş model tabanı bulunanın performansı hakkında yorum yapmak mümkün olmuştur. Şu anda "Omega" dergisinde hakem sürecinde olan bu çalışma Ek-6'da sunulmuştur. Bu çalışma, özetle, talep belirsizliğinin dönemsellik arz ettiği durumda -- ki, bu perakende sektöründe karşılaşılan temel talep durumudur – literatürde yaygın olarak incelenmiş bulunan durağan envanter yönetim politikalarının optimal maliyetlerinin, gerçek talep deseni olan durağan olmayan stokastik süreçler dikkate alındığında ve bunun için envanter kontrolü yapıldığındaki maliyetlerden çok daha yüksek olduğunu göstermiştir. Çalışma durağan olmayan talebin durağan varsayılarak envanter yönetiminin optimalden uzak sonuçlar verdiğini göstermiştir ve gerçek talep deseninin kullanılmasının önemine işaret etmiştir.

Perakende tedarik zincirlerinin yönetiminde dikkate alınması gereken en önemli faktörlerden biri kullanılacak olan politikaların belirsizliğe karşı duyarlılığıdır. Duyarlılığı yüksek olan politikalar küçük dalgalanmalara aşırı tepki vererek tedarik zincirinin bütününde dalgalanmaya sebep olmaktadırlar. Bu yüzden "sağlam" (robust) politikaların geliştirilmesi perakende tedarik zincirleri açısından önem taşımaktadır. Bu kapsamda yayınlanan "Finding reliable solutions: event-driven probabilistic constraint programming" çalışması güvenilir ve sağlam politikaların belirsizlik altında geliştirilmesini sağlayan yeni bir planlama platformudur. "Event-driven probabilistic constraint programming – EDP-CP" adı verilen bu çerçeve stokastik kısıt programlama mantığına dayanmaktadır ve bu çerçevede daha önce hiçbir modelleme paradigması ile ifadesi direkt olarak mümkün olmayan problemlerin modellemesi ve çözümü yapılabilmektedir. EDP-CP politikaların duyarlılığı ve katlanılması beklenen maliyet arasındaki ödünleşmelerin (trade-offs) incelenmesinde önemli rol oynayacak bir teknik olarak görülmektedir. Bu çalışma Annals of Operations Research'de yayınlanmıştır ve TÜBİTAK desteğini belirtmektedir. Bu makalenin tam metni Ek-7'de sunulmuştur. Makalenin ilk bölümünde bu makalede sunulan EDP-CP tekniğinin yakından ilgili olduğu şans kısıtlı stokastik programlama gibi tekniklerle olan benzer ve farklı yönleri tartışılmıştır. İkinci bölümde bu tür bir modelleme ve çözüm platformuna neden ihtiyaç duyulduğu bir tedarik zinciri/dağıtım planlaması problemi üzerinden verilmiştir. Farklı modelleme paradigmalarının bu problemi nasıl ele alabileceği ayrıntılı olarak tartışılmıştır. Üçüncü bölümde EDP-CP platformu formal olarak tanımlanmıştır. Dördüncü bölümde bu platform kullanılarak kurulan modellerin çözümüne ilişkin yaptığımız çalışmalar sunulmuştur. Bu tip modeller için önerdiğimiz tam çözüm algoritması maalesef problemlerin doğası gereği hızlı çözüm üretememektedir. Bu sakıncayı ortadan kaldırmak için, optimallikten fedakarlıkta bulunma koşuluyla, bir yaklaşık sonuç üreten senaryo indirgeme tekniği takip eden bölümde (Bölüm 5) verilmiştir. Bölüm altı, farklı örnek problemler üzerinden modelleme ve çözüm yöntemini açıklamaktadır. Bu bölümde ele alınan problemler: olasılıksal tedarik zinciri planlama problemi, olasılıksal çizelgeleme problemi ve olasılıksal üretim planlama/sermaye bütçelemesi problemidir. Yedinci bölümde ilgili literatür üzerine bir tartışma yapılmıştır.

EDP-CP yaklaşımının ilham aldığı çalışma Liu-Iwamura'nın "bağımlı sanş programlama" (dependent chance programming) tekniğidir. Ancak bu tekniğin tasarımında ortaya çıkan bazı yanlışlıklar ve eksikler bu proje çalışmaları sırasında fark edilmiş ve bunun için bir düzeltme yayınlanmıştır. Bu düzeltme "A note on Liu–Iwamura's dependent-chance programming" başlığıyla European Journal of Operational Research'te yayınlanmıştır. Bu makale Ek-8'te sunulmuştur. Özetle, Liu ve Iwamura tarafından önerilen modelleme yaklaşımında tüm kararlar arasındaki ilişki karar değişkenlerinin değerinden bağımsız olarak varsayılmıştır. Ancak bunun doğru olmayacağı birçok durum vardır. Ve çalışmada aykırı bir örnek üzerinden Liu-Iwamura'nın varsayımlarının yanlış sonuç verdiği gösterilmiştir. Diğer tüm çalışmalarda olduğu gibi bu çalışmada TÜBİTAK desteğini belirtmektedir.

Bir kısmının proje öncesi çalışmalara dayandığı "A multi-objective stochastic programming approach for supply chain design considering risk" başlıklı çalışma International Journal of Producton Economics dergisinde yayınlanmıştır. Bu makalede, belirsizlik altında tedarik zinciri tasarımı için çok amaçlı bir stokastik programlama yaklaşımı geliştirilmiştir. Talepler, arzlar, işlem süreleri, nakliye süreleri, stoksuz kalma ve kapasite artırma maliyetlerinin tamamı rassal değişken olarak alınmıştır. "Sağlam" (robust) bir model kurabilmek için, konvansiyonel tedarik zinciri tasarımı problemine iki yeni amaç fonksiyonu eklenmiştir. Böylece, çok-amaçlı olarak kurulan model (i) şimdiki yatırım ve gelecekte ortaya çıkacak işlem, taşıma, stoksuz kalma ve kapasite genişletme maliyetlerinin beklenen değerinin toplamını minimize etmek, (ii) toplam maliyetin varyansını minimize etmek, ve (iii) finansal riskin, ya da belirlenmiş bütçenin aşılması olasılığının minimize etmek hedeflerini gerçekleştirmektedir. Tam olarak güvenilemeyen arz ve belirsizlik sonrası kapasite genişletme durumlarının dikkate alındığı bir model önerilmiştir. Bu modelin çözümü için "goal attainment" tekniği kullanılarak Pareto-optimal çözümler elde edilmiştir. Bu çalışmanın tam metni Ek-9'da verilmiştir. Çalışma TÜBİTAK desteğini belirtmektedir.

Projenin ana temaları olan envanter yönetimi ve belirsizlik çerçevesinde yaptığımız çalışmalardan bir diğeri de Journal of Combinatorial Optimization dergisinde yayınlanan "Scheduling internal audit activities: a stochastic combinatorial optimization problem" başlıklı makaledir. Bu makale esas itibariyle bir envanter problemini ele almamaktadır. Ancak ele alınan problem, ki bir muhasebe denetim problemidir, envanter modelleriyle ilişkinlendirilmiştir. Bu ilişkilendirme konu bağlamında olmayıp tamamen modelleme tabanlı bir analojiye dayanmaktadır. Diğer bir deyişle, bir muhasebe denetim problemi sanki bir envanter problemiymiş gibi düşünülüp, envanter planlaması için geliştirilen modellerle çözülmüştür. Bu makale esas olarak yaptığımız envanter çalışmalarının sadece envanter alanı ile sınırlı olmayıp çok daha uzak alanlarda da belirsizlik altında karar alma problemlerinin çözümünde kullanılabileceğini göstermektedir. Makalenin basılmış kopyası Ek-10'da sunulmuştur.

Araştırmalarımızın temel hedeflerinden olan stokastik envanter planlaması modelleri için etkin çözüm yöntemlerinin geliştirilmesine yönelik olarak gerçekleştirdiğimiz çalışmalardan bir diğeri de Ek-11'de sunmuş olduğumuz "A state space augmentation algorithm for the replenishment cycle inventory policy" başlıklı çalışmadır. Bu makale International Journal of Production Economics dergisinde yayına kabul edilmiştir. Çalışma envanter politikaları arasında önemli bir yer tutan ve dört temel politikadan biri olan (R,S) envanter politikasının en genel hali olan durağan olmayan talep altında çözümüne dönük yeni yöntemlerin geliştirilmesini konu edinmiştir. Diğer temel politikalar (s,S), (s,Q) ve (R,s,S) dir. (R,S) için kullanılan karışık tamsayı programlama modelinin ancak küçük ve orta büyüklükteki problemler için makul sürede çözüm verebilmesi, büyük endüstriyel ölçekteki envanter problemlerinin çözümüne dönük yeni tekniklerin geliştirilmesi mecburiyetini doğurmuştur. Çalışma bu ihtiyaca binaen yapılmıştır. Kullanılan temel teknik dinamik programlamadır. Dinamik programlama konvansiyonel şekliyle kullanıldığında bu problem için etkili bir çözüm yöntemi değildir. Ancak bu çalışmada gösterildiği üzere, bazı filtreleme ve "augmentation" (birleştirme) teknikleri kullanılarak en büyük (R,S) modelleri için dahi optimal politika parametrelerinin hesaplanması kolaylıkla yapılabilmektedir. Bu tekniğin ardında yatan temel fikir durağan olmayan (R,S) modelinin esnetilmiş (relaxed) halinin bir şebeke olarak gösterilebilmesi ve bu şebekenin en kısa yol problemine karşılık gelmesidir. Bu en kısa yol probleminin çözümü mümkün bir çözüm vermeyebilir; çünkü esnetilmiş probleme karşılık gelmektedir. Bu şebeke üzerinde ilk olarak orijinal şebekeyi küçültmek üzere alt-optimal oldukları ispat edilebilen yayların filtrelenmesi yapılmıştır. Daha sonra elde edilen indirgenmiş şebeke optimal çözümü verecek şekilde yeniden kurulmuştur. Bu teknikle elde edilen nihai şebeke mevcut en kısa yol algoritmaları ile son derece hızlı şekilde çözülebilmektedir. Yapılan nümerik deneylerde 200 dönemlik problemlerin 1 saniyenin altında bir sürede çözüldüğü gösterilmiştir. Karışık tamsayı programlama formülasyonunun 50 dönemlik problemleri çözmesinin saatler aldığı düşünülürse elde edilen çözüm hızının büyüklüğü daha iyi anlaşılmaktadır.

Bu araştırma projesinde kullanılan temel envanter yönetim politikası (R,S) dir. Bilindiği üzere (R,S) politikasında planlama ufkunun başında gelecekteki sipariş zamanları (R) sabitlenmekte ve sabitlenen bu sipariş zamanlarındaki sipariş miktarları gerçekleşen talebe bağlı olarak belirlenmektedir. Herhangi bir sipariş döneminde verilecek olan sipariş miktarı o dönemin sipariş yükseltme noktası (order-up-to-level) ve mevcut envanter seviyesi arasındaki farktır. Sipariş yükseltme noktaları da planlama ufku başında belirlenmekte ve daha sonra değiştirilmemektedir. Bu politikanın beklenen en düşük maliyeti veremeyebileceği bilinmektedir. Beklenen en düşük maliyetli politikanın (s,S) politikası olduğu 1960'dan beri bilinmektedir. (s,S) politikasında siparişler envanter seviyesi s değerinin altına düştüğünde ve envanter seviyesini S değerine yükseltecek kadar verilmektedir. Bu politikanın beklenen en düşük maliyetli politika olmasına rağmen, (s,S) politikası uygulamada çok rağbet gören bir politika değildir. Bunun en büyük sebebi gelecekteki sipariş zamanlarını ve beklenen sipariş miktarlarını şimdiden ön görmenin mümkün olamamasıdır. Dolayısıyla bu tip bir politika sistem sinirliliğine (nervousness) açıktır. Ek-12'de sunulan çalışmada (s,S) ve (R,S) politikalarının maliyet ve sinirlilik açısından kıyaslaması yapılmıştır. Sonuçları özetlemek gerekirse, (R,S) ve (s,S) arasındaki maliyet farkı ortalamada çok yüksek değildir; öte yandan sinirlilik bakımından (s,S) çok kötü bir performans göstermektedir. Bu çalışma (R,S) politikasının uygulamada, özellikle perakendecilik gibi talebin durağan olmadığı ve belirsizlik taşıdığı sektörlerde, kullanımının önemine vurgu yapmaktadır. Geçtiğimiz günlerde bu çalışma International Journal of Production Economics dergisinde yayınlanmak üzere kabul edilmiştir.

Bu bölümün başında da belirtildiği gibi proje kapsamında yapılan teorik çalışmalar iki ana grupta tasnif edilebilir: belirsizlik altında karar almaya dönük olarak yeni tekniklerin geliştirilmesi ve belirsizlik altında envanter yönetimi ve planlaması. Belirsizlik altında karar almaya dönük olarak yeni tekniklerle ilgili olarak yaptığımız bir literatür taraması ve sınıflaması Ek-13'te verilmiştir. Perakende tedarik zincirlerine ilişkin büyük boyutlu stokastik kombinatöryel karar problemlerinin çözümünde kullanılacak olan yaklaşımın verimli, etkin ve ölçeklenebilir olması gerekir. Bu proje çalışmasının teorik kısmının hedeflerinden biri de kombinatöryel problemlerin çözümünde kullanılan temel tekniklerden "kısıt programlama", "karışık tamsayı programlama" ve "dinamik programlama"nın tek başlarına kullanımı yerine tek bir çerçeve altında üçünün bir melezinin (hibritinin) kullanımının denenmesidir. Buna ilişkin çalışmanın literatür incelemesi davet üzerine yapılmış olup, Kısıt Programlama-Yapay Zeka ve Yöneylem Araştırması alanı ortak kümesinde geliştirilen belirsizlik altında kara almaya dönük melez modelleri kapsamaktadır. İkinci bölümde belirsizlik altında karar almaya ilişkin temel kavramlar örnek problemler üzerinden verilmiştir. Kullanılan problemler tek-aşamalı stokastik sırt çantası problemi ve bunun çok-aşamalı varyasyonudur. Temel stokastik programlama kavramları bu bölümde verilmiştir. Üçüncü bölümde Kısıt Programlama-Yapay Zeka ve Yöneylem Araştırması melezlerinin kullanıldığı bir problem kümesi derlenip sunulmuştur. Bu problemler: stokastik kuyruk kontrol problemi; şartlı görev grafilerinin çizelgelemesi; stokastik rezervasyon problemi; olasılıksal sürelerle iş istasyonu çizelgelemesi; iki aşamalı stokastik eşleştirme problemi; üretim/envanter yönetimi; stokastik "template" tasarımı; denetim faaliyetlerinin planlaması; stokastik sıralama. Bölüm dörtte, kısıt programlama ve yöneylem araştırmasıyla belirsizlik altında karar alma için kullanılan melez yaklaşımlar incelenmiştir. Bunlar arasında stokastik boolgil sağlama (stochastic boolean satisfiability), olasılıksal kısıt sağlama problemi, olay-tabanlı olasılıksal kısıt programlama, stokastik kısıt programlama sayılabilir. Takip eden beşinci bölümde var olan platformların bir sınıflaması yapılmıştır. Bölüm altıda stokastik mantık yürütmeye dayanan yaklaşımlar tartışılmıştır. Bölüm yedi ise yeniden formüle edilme esasına dayalı teknikleri incelemiştir. Bölüm sekiz örnekleme tabanlı teknikleri tartışmaktadır. Son olarak bölüm dokuz stokastik dinamik programlama gibi ilgili tekniklere yer vermiştir. Bu çalışma bir kitap bölümü olarak Springer serisinde basılacaktır.

Buraya kadar yapılan çalışmalarda belirsizlik unsuru olarak sadece talep ele alınmıştır. Yaptığımız iki ayrı çalışmada ise tedarik sürelerindeki belirsizlik taleple birlikte ele alınmıştır. Takdir edileceği üzere sadece talep belirsizliği kendi başına büyük zorluk yaratırken, bunun üzerine tedarik sürelerindeki belirsizliği de eklemek problemin karmaşıklığını daha da artırmıştır. Bu iki çalışmanın ilkinde ("Constraint-based local search for inventory control under stochastic demand and lead time") iki yeni sezgisel yaklaşım bu zor problemin çözümü için önerilmiştir. Önerilen bu sezgisel yaklaşımların etkinliği üçük problemlerin tam çözümünün bulunması ve sonuçların sezgisel metotlarla elde edilenlere kıyaslaması yoluyla yapılmıştır. Nümerik sonuçlar her iki sezgisel yaklaşımında son derece etkili olduğunu bize göstermiştir. Bu çalışma alanının önemli dergilerinden "INFORMS Journal on Computing" de hakem sürecindedir. Hakemlerden sadece küçük düzeltme alan bu çalışmanın yakın bir zamanda kabul edilmesini beklemekteyiz. Bu çalışmanın son hali Ek-14'te sunulmuştur.

Yukarıda bahsedilen tedarik sürelerindeki belirsizliğe ilişkin olarak yaptığımız ikinci çalışma “Computing the non-stationary replenishment cycle inventory policy under stochastic supplier lead-time” International Journal of Production Economics’te yayınlanmak üzere kabul edilmiştir. Bu çalışma (R,S) envanter politikası izlendiğinde ve servis düzeyi kısıtı altında optimal politika parametrelerinin hesaplanmasına dönük bir yaklaşım önermektedir. Talep ve tedarik sürelerindeki belirsizlik her iki halde de durağan olmayan stokastik tiptedir. Çözüm yönteminin özü bir başka çalışmamızda tasarlamış olduğumuz “global chance-constraint” tabanlı formülasyondur. Ayrıca bu kısıt filtreleme algoritmalarıyla desteklenmiştir. Sonuçlar tedarik süresindeki belirsizliğin ne kadar belirleyici olduğunu ve problem varsayımları arasında tedarik süresinin deterministik (belirlenimli) kabul edilmesinin uygun olmayan koşullarda aslında optimalden büyük ölçüde sapan sonuçlara yol açtığını göştermiştir. Bu çalışma Ek-15’te verilmiştir.

Raporda bahsedilecek son çalışmada esas olarak çok-aşamalı envanter sistemleri ele alınmıştır. Talep riski altında stok bulundurma, stoksuz kalma ve sabit sipariş maliyetlerinin minimizasyonu küçük boyutlu problemler (en çok 3 aşamalı ve 10 periyotlu) için stokastik programlama yardımıyla optimal olarak gerçekleştirilmiştir. Tahmin edilebileceği gibi aşama sayısının veya planlama ufkunda dikkate alınan dönem sayısının artması halinde stokastik programlama yardımıyla sonuç elde edilmesi imkansız hale gelmektedir. Hatta matematiksel modelin bilgisayarın hafızasına yüklenmesi dahi mümkün olamamaktadır. Ancak stokastik programlama yardımıyla küçük problemler için bulunan çözümler önerilen yeni çözüm tekniklerinin performanslarının değerlendirilmesinde kullanılmıştır. Bu noktada Yapay Zeka tekniklerinden yapay sinir ağı (artificial neural network) kullanılarak olası tüm senaryolar kompakt bir şekilde ifade edilmiş ve daha sonra ağ bir simülasyon tabanlı evrimsel (evolutionary) algoritma yardımıyla eğitilmiştir. Deneysel olarak bu yaklaşımla yüksek kalitede sonuçların elde edilebileceği ve stokastik programlama yardımıyla modellenmesi mümkün olmayan sistemler için tedarik planlarının bulunabileceği gösterilmiştir. Bazı periyodik yapılar kullanılarak aslında yüzlerce döneme sahip problemlerin bile çok hızlı bir şekilde çözüldüğü gözlenmiştir. Bu çalışma “Neuroevolutionary inventory control in multi-echelon system” baslığıyla Ek-16’da verilmiştir. International Journal of Production Research dergisinde hakem sürecinde olan bu çalışma için küçük bir düzeltme istenmiştir. Bu düzeltme su sıralarda yapılmaktadır ve en geç yıl sonuna kadar bu makalenin de basılmak üzere kabul edilmesi beklenmektedir.

Sonuç

Bu proje, tedarik zinciri envanter yönetimi alanında önemli teorik sorulara cevap aramanın yanında, ülkemizin en dinamik ve hızlı büyüyen sektörlerinin başında yer alan perakende sektöründe verimlilik artışına dönük yeni teknolojilerin geliştirilmesini sağlamıştır.

Yapılan çalışmaların ilk aşamasında Türkiye perakende sektöründe faaliyet gösteren süpermarketlerin tedarik zinciri yönetimi için kullandıkları envanter yönetim yazılımları belirlenmiş ve tedarik zinciri yönetiminde bu yazılımlardan ne ölçüde faydalandıklarını tespit edilmiştir. Türkiye'de faaliyet gösteren 31 süpermarket zinciri ile görüşülmüştür. Görüşmeler yapılmadan önce kullanımda olan ERP yazılımlarının stok takibi ve talep tahmini yapabildikleri tahmin edilmekte ve fakat gerek deterministik (belirlenimli) gerekse stokastik (belirlenimsiz) planlama yönünden yetersiz kalacakları düşünülmekteydi. Görüşmeler sonucunda sadece bir tek ERP yazılımının durağan (stationary) stokastik planlama yapabildiği tespit edilmiştir. Birkaç büyük yazılım firmasının ürününün deterministik planlama yapabildikleri, geriye kalan yazılımların ise tahmin yapabilmekten öteye gidemedikleri ortaya çıkmıştır. Araştırmaya dâhil edilen hiçbir ERP yazılımının durağan olmayan (non-stationary) talep altında stokastik planlama yapamaması, bu yönde yapılacak çalışmaların önemini ve gerekliliğini göstermiştir. Bununla birlikte çok uluslu olan yazılım şirketlerinin ürünlerinin genellikle büyük şirketlerin kullandığı, yerel bazda olan yazılım şirketlerinin ise küçük ve orta büyüklükteki şirketlere yönelik ürün geliştirdikleri tespit edilmiştir. ERP yazılımlarının birçoğu öncelikle imalat sanayiinde faaliyet gösteren işletmelerin ihtiyaçlarını karşılamaya dönük olarak hazırlanmıştır. Perakende sektörüne yönelik çözümler de, bu paketlere ilave edilmiş veya ayrı bir paket halinde piyasaya sunulmuştur. Bu yüzden birçoğunun eksiği bulunması ile birlikte günden güne bu eksiklerin giderilmeye çalışılmakta olduğu görülmüştür. Elde ettiğimiz bulgular temelde sektör tarafından kullanılan yazılımların etkin şekilde belirsizlikle baş edebilecek fonksiyonelliğe sahip olmadıklarını ve çoğu zamanda bu yazılımların sunduğu temel fonksiyonların bile firmalar tarafından uygun organizasyonel süreçler ve uzmanlık bulunmadığı için kullanılmadığını göstermektedir. Genel olarak baktığımızda, sadece 2 süpermarket yazılımının talep tahmini yapabildiğini ve tahmin hatasına ilişkin bilgi sunduğunu belirtmiştir. 9 süpermarket yazılımlarının sadece talep tahmini yapabildiğini bildirmiştir. 1 süpermarket ise ERP yazılımı kullanmadıklarını belirtmiştir. Bu tabloya baktığımızda hiçbir süpermarketin deterministik planlama yapmadığı görülmektedir. Buradan yola çıkarak, günümüz Türkiye perakende sektöründe faaliyet gösteren süpermarketlerde yönetici deneyimlerinin sipariş kararlarında hala en önemli belirleyici olduğu sonucuna ulaşabiliriz. Ayrıca yaptığımız görüşmelerde ulaştığımız bir başka sonuç ise, yazılımların birçoğunun süpermarketlerde çok görülen mevsimsel satış, indirim, kampanya gibi durumlara özel çözümlerinin olmayışıdır. Bu da sipariş miktarlarının belirlenmesinde hala yöneticilerin neden kişisel tecrübelerine dayanarak karar verdiklerini bir noktaya kadar açıklamaktadır.

Süpermarketlerin belirsizlik altında ihtiyaçlarına cevap verecek yeni ve kapsamlı envanter planlama sistemlerine ihtiyaç olduğu açıktır. Öte yandan bu yazılımların var olması halinde dahi etkin şekilde kullanılabileceğine dair bir bulgu yoktur. Bu noktada yöneticilerin gerekli eğitimden geçirilmesi, bilinçlendirilmesi ve teknik ekibin yazılımların fonksiyonellikleri hakkında bilgilendirilmesi gerekliliği ortadadır.

Proje çalışmasının ikinci aşamasında tespit edilen ihtiyaçların giderilmesine dönük olarak modelleme ve çözüm algoritması geliştirme faaliyetleri gerçekleştirilmiştir. Ele alınan perakende tedarik zinciri envanter yönetimi optimizasyon modelleri durağan olmayan talep varsayımını taşımaktadır. Bunun da ötesine geçilmiş ve tedarik sürelerindeki belirsizlikte durağan olmayan form varsayımı altında modellemeye dahil edilmiştir. Sadece tek stok noktası ile sınırlı kalınmamış çok-aşamalı stok sistemleri için de modelleme ve çözüm yöntemleri üzerine çalışılmıştır.

Proje kapsamında yürütülen teorik çalışmalar iki ana grupta tasnif edilebilir. Bu gruplardan ilki *belirsizlik altında karar almaya dönük olarak yeni teknikler*in geliştirilmesiyle ilgilidir. Bu gruptaki çalışmalar genel amaçlı olup daha çok belirsizliğin rol oynadığı karar problemlerinin modellenmesine ve çözümüne ilişkin kullanıcıya platformlar sunmaktadır. İkinci grup çalışmada ise *belirsizlik altında envanter yönetimi ve planlaması* problemi dikkate alınmıştır. Bu kapsamda yaptığımız onaltı makale çalışmanın eklerinde sunulmuştur.

Tedarik zinciri yönetimi alanındaki araştırma faaliyetlerinin uygulamada sonuç alabilmesi için iş dünyasının çevresini tanımlayan temel unsurları dikkate alması gereklidir. Bu çerçevede, belirsizlik günümüz iş dünyasını tanımlayan en kritik faktörler arasındadır ve tedarik zinciri yönetimi araştırmalarının merkezinde olmalıdır. Belirsizliğin planlamacılar için karmaşıklık yaratan bir faktör olduğuna dair fikir birliği bulunmasına rağmen mevcut planlama sistemleri veri için nadiren belirlenimsiz (non-deterministic) yaklaşım kullanmaktadır. Bu proje perakende tedarik zincirleri için belirsizlik altında envanter planlaması sorununu ele almıştır ve çeşitli modelleme platformları ve çözüm yöntemleri sunmuştur. Tedarik zinciri envanter araştırmalarının uygulamada beklenen etkiyi gösterebilmesi için büyük ölçekli stokastik karar problemlerinin çözülmesi gerekmektedir. Ancak küçük boyutlu stokastik problemler bile hesaplama bakımından zorluklar taşımaktadır. Bu kapsamda talep belirsizliği ve tedarik süresi belirsizliği altında beklenen maliyetleri azaltmaya dönük tedarik zinciri envanter politikaları geliştirilmiş, politika parametrelerinin hesaplanmasında kullanılacak yeni tedarik zinciri envanter modelleri kurulmuş ve kurulan modelleri etkin şekilde çözmek için algoritmalar geliştirilmiştir.

Referanslar

Askin, R. G. “A procedure for production lot-sizing with probabilistic dynamic demand.” *AIIE Transactions*, 1981: 132-137.

Bookbinder, J. H., and J. Y. Tan. “Strategies for the Probabilistic Lot-Sizing Problem with Service-Level Constraints.” *Management Science*, 1988: 1096-1108.

Chen, H., M. Z. Frank, and O. Q. Wu. “US retail and wholesale inventory performance from 1981 to 2004.” *Manufacturing and Service Operations Management*, 2007: 430-456.

Chen, H., M. Z. Frank, and O. Q. Wu. “What Actually Happened to the Inventories of American Companies Between 1981 and 2000?” *Management Science*, 2005: 1015-1031.

Coca-Cola Research Council/Andersen Consulting. *Where to Look for Incremental Sales Gains: The Retail Problem of Out-of-Stock Merchandise.* Atlanta, GA.: The Coca-Cola Research Council, 1996.

de Kok, A. G., and S. C. Graves. *Supply Chain Management: Design, Coordination and Operation, Handbooks in Operations Research and Management Science, Vol.11.* Elsevier, 2003.

Enslow, B. *Supply Chain Inventory Strategies Benchmark Report.* Technical Report, Aberdeen Group Teknik Raporu (http://www.aberdeen.com/), 2004.

Fisher, M. L., A. Raman, and A. McClelland. “Rocket Science Retailing Is Almost Here -- Are You Ready?” *Harvard Business Review*, 2000: 115-124.

Graves, S. C., A. H. G. Rinnooy Kan, and P. H. Zipkin. *Logistics of Production and Inventory, Handbooks in Operations Research and Management Science, Vol.4.* North-Holland, 1993.

Hnich, B., Z. Kiziltan, I. Miguel, and T. Walsh. “Hybrid Modelling for Robust Solving.” *Annals of Operation Research*, 2004: 19-39.

Kleywegt, A. J., A. Shapiro, and T. H. de Mello. “The Sample Average Approximation Method for Stochastic Discrete Optimization.” *SIAM Journal on Optimization*, 2002: 479-502.

Lee, C. B. *Demand Chain Optimization: Pitfalls and Key Principles.* Evant White Paper Series, 2003.

Porteus, E. L. *Foundations of Stochastic Inventory Theory.* Stanford University Press, 2002.

Santoso, T., S. Ahmed, M. Goetschalckx, and A. Shapiro. “A stochastic programming approach for supply chain network design under uncertainty.” *European Journal of Operational Research*, 2005: 96-115.

Scarf, H. “Optimality of (s,S) Policies in the Dynamic Inventory Problem.” In *Mathematical Methods in the Social Sciences*, by K. J. Arrow, S. Karlin and P. Suppes, 197-202. Stanford University Press, 1959.

Silver, E. A. “Inventory control under a probabilistic time varying demand pattern.” *AIIE Transactions*, 1978: 371-379.

Silver, E. A., and H. A. Meal. “A heuristic for selecting lot size requirements for the case of a deterministic time-varying demand rate with discrete opportunities for replenishment.” *Production and Inventory Management Journal*, 1973: 64-74.

Silver, E. A., D. F. Pyke, and R. Peterson. *Inventory Management and Production Planning and Scheduling.* John Wiley & Sons, 1998.

Tarim, S. A., and B. G. Kingsman. “Modelling and Computing Policies for Inventory Systems with Non-Stationary Stochastic Demand.” *European Journal of Operational Research*, 2006: 581-599.

Tarim, S. A., and B. G. Kingsman. “The stochastic dynamic production/inventory lot-sizing problem with service-level constraints.” *International Journal of Production Economics*, 2004: 105–119.

Tarim, S. A., and B. M. Smith. “Constraint Programming for Computing Non-Stationary (R,S) Inventory Policies.” *European Journal of Operational Research*, 2008.

Tarim, S. A., and I. Miguel. “A Hybrid Benders' Decomposition Method for Solving Stochastic Constraint Programs with Linear Recourse.” *Lecture Notes in Artificial Intelligence LNAI3978*, 2006: 133-148.

Tarim, S. A., R. Rossi, B. Hnich, and S. Prestwich. “Cost-based Filtering Techniques for Stochastic Inventory Systems under Service Level Constraints.” *Constraints*, 2008.

Van Hentenryck, P., and R. Bent. *Online Stochastic Combinatorial Optimization.* MIT Press, 2006.

Wagner, H. M., and T. M. Whitin. “Dynamic version of the economic lot size model.” *Management Science*, 1958: 89-96.

Wu, S. D., R. Roundy, R. H. Storer, and L. A. Martin-Vega. “Manufacturing logistics research: taxonomy and directions.” Technical Report No:1254, School of Operations Research and Industrial Engineering, Cornell University, 1999.

Zipkin, P. H. *Foundations of Inventory Management.* McGraw-Hill, 2000.

TÜBİTAK

PROJE ÖZET BİLGİ FORMU

Proje No: 108K027

Proje Başlığı: Türkiye'de Perakende Tedarik Zinciri Envanter Sistemlerinin Belirsizlik Altında Yönetimi için Planlama Modelleri ve Çözüm Yöntemleri

Proje Yürütücüsü ve Araştırmacılar:

Prof. Dr. Ş. Armağan TARIM (Yürütücü)
Prof. Dr. Brahim HNICH (Araştırmacı)
Yrd. Doç. Dr. Ayşegül TAŞ (Araştırmacı)

Projenin Yürütüldüğü Kuruluş ve Adresi:

Hacettepe Üniversitesi, İşletme Bölümü, Beytepe Kampüsü, Ankara

Destekleyen Kuruluş(ların) Adı ve Adresi:

Yok

Projenin Başlangıç ve Bitiş Tarihleri:

01 Temmuz 2008 – 30 Haziran 2010

Öz

Tedarik zinciri yönetimi alanındaki araştırma faaliyetlerinin uygulamada sonuç alabilmesi için belirsizlik gibi iş dünyasının çevresini tanımlayan temel unsurları dikkate alması gereklidir. Bu proje perakende tedarik zincirleri için belirsizlik altında envanter planlaması sorununu ele almıştır. Talep belirsizliği altında beklenen maliyetleri azaltmaya dönük tedarik zinciri envanter politikaları geliştirilmiş, politika parametrelerinin hesaplanmasında kullanılacak yeni tedarik zinciri envanter modelleri kurulmuş ve kurulan modelleri etkin şekilde çözmek için algoritmalar geliştirilmiştir.

Anahtar Kelimeler:

Perakendecilik; Tedarik Zinciri Yönetimi; Envanter Yönetimi; Stokastik Modelleme

Fikri Ürün Bildirim Formu Sunuldu mu? □ □ **Gerekli Değil** ☒

Projeden Yapılan Yayınlar:

Uluslararası Dergiler/Kitap Bölümü:

"Finding Reliable Solutions: Event-Driven Probabilistic Constraint Programming," S. A. Tarim, B. Hnich, S. Prestwich and R. Rossi, Annals of Operations Research, 171, pp.77-99, 2009.

"A Note on Liu-Iwamura's Dependent-Chance Programming," R. Rossi, S. A. Tarim, B. Hnich, S. Prestwich, C. Guran, European Journal of Operational Research, 198, pp.983–986, 2009.

"A Multi-Objective Stochastic Programming Approach for Supply Chain Design Considering Risk," A. Azaron, K. N. Brown, S. A. Tarim, M. Modarres, International Journal of Production Economics, 116, pp.129-138, 2008.

"Scheduling Internal Audit Activities: A Stochastic Combinatorial Optimization Problem," R. Rossi, S. A. Tarim, B. Hnich, S. Prestwich, S. Karacaer, Journal of Combinatorial Optimization, 19, pp.325-349, 2010.

"A State Space Augmentation Algorithm for the Replenishment Cycle Inventory Policy," R. Rossi, S. A. Tarim, B. Hnich, S. Prestwich, International Journal of Production Economics, to appear, 2010.

“An Investigation of Setup Stability in Non-Stationary Stochastic Inventory Systems," O. A. Kilic, S. A. Tarim, International Journal of Production Economics, to appear, 2010.

"Computing the Non-Stationary Replenishment Cycle Inventory Policy under Stochastic Supplier Lead-Times," R. Rossi, S. A. Tarim, B. Hnich, S. Prestwich, International Journal of Production Economics, to appear, 2010.

Invited Paper: "A Survey on CP-AI-OR Hybrids for Decision Making under Uncertainty," B. Hnich, R. Rossi, S. A. Tarim, S. Prestwich, to appear, 2010.

"Cost-based Domain Filtering for Stochastic Constraint Programming," R. Rossi, S. A. Tarim, B. Hnich, S. Prestwich, Lecture Notes in Computer Science, Springer-Verlag, LNCS 5202, pp. 235-250, 2008.

"A Steady-State Genetic Algorithm With Resampling for Noisy Inventory Control," S. Prestwich, S. A. Tarim, R. Rossi and B. Hnich, Lecture Notes in Computer Science, Springer-Verlag, LNCS 5199, pp. 559-568, 2008.

"A Cultural Algorithm for POMDPs from Stochastic Inventory Control," S. D. Prestwich, S. A. Tarim, R. Rossi, B. Hnich, Lecture Notes in Computer Science, Springer-Verlag, LNCS 5296, pp. 16-28, 2008.

“The Cost Of Using Stationary Inventory Policies When Demand Is Non-Stationary,” H. Tunç, O. A. Kılıç, S. A. Tarim, B. Ekşioğlu, Omega’da hakem sürecinde (1. raund).

“Constraint-based Local Search for Inventory Control under Stochastic Demand and Lead time,” R. Rossi, S. A. Tarim, R. Bollapragada, INFORMS Journal on Computing’te hakem sürecinde (2. raund).

"Neuroevolutionary Inventory Control in Multi-Echelon Systems," S. D. Prestwich, S. A. Tarim, R. Rossi, B. Hnich, Lecture Notes in Artificial Intelligence, Springer-Verlag, LNAI 5783, pp.402-413, 2009.

Uluslararası Bildiriler:

“Neuroevolutionary Inventory Control in Multi-Echelon Systems”, S. Prestwich, S.A. Tarim, R. Rossi, B. Hnich, 1st International Conference on Algorithmic Decision Theory (ADT-09) October 21-23, 2009 Venice, Italy.

“Synthesizing Filtering Algorithms for Global Chance-Constraints”, B. Hnich, R. Rossi, S.A. Tarim, S. Prestwich, 5th International Conference on Principles and Practice of Constraint Programming (CP-09) September 21-24, 2009, Lisbon, Portugal.

“Evolving Parameterised Policies for Stochastic Constraint Programming”, S. Prestwich, S.A. Tarim, R. Rossi, B. Hnich, 15th International Conference on Principles and practice of Constraint Programming (CP-09) September 21-24, 2009, Lisbon, Portugal.

"A Relaxation Approach for Solving the Stochastic Lot Sizing Problem with Service Level Constraints" S.A. Tarim, U. Ozen, M.K. Dogru, R. Rossi, INFORMS Annual Meeting, October 11-14, 2009, San Diego, CA.

"Boosting Partial Symmetry Breaking by Local Search", S. Prestwich, B. Hnich, H. Simonis, R. Rossi and S.A. Tarim, 9th International Workshop on Symmetry and Constraint Satisfaction Problems (SYMCON 2009), September 21-24, 2009, Lisbon, Portugal, in conjunction with CP 2009.

"Synthesizing Filtering Algorithms in Stochastic Constraint Programming", B. Hnich, R. Rossi, S.A. Tarim, S. Prestwich, 40th Annual Conference of the Italian Operational Research Society (AIRO 2009) September 8-11, 2009, Siena, Italy.

"Stochastic Constraint Programming by Neurevolution with Filtering," S. Prestwich, S. A. Tarim, R. Rossi, B. Hnich, 7th International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) techniques in Constraint Programming, Bologna, Italy, June 14-18, 2010.

SOBAG-108K027

Türkiye'de Süpermarket Tedarik Zinciri Envanter Yönetimi için
Kullanılan Sistemler Üzerine Değerlendirmeler

RAPOR I:

Yazılım Firmalarıyla Görüşmeler

Hazırlayanlar:

Sedat Belbağ
Mustafa Çimen
Dr. Şule Tarım
Yrd. Doç. Dr. Ayşegül Taş
Prof. Dr. Ş. Armağan Tarım

29.12.2008

SOBAG-108K027 RAPOR-I 29.12.2008

Türkiye'de Süpermarket Tedarik Zinciri Envanter Yönetimi için Kullanılan Sistemler Üzerine Değerlendirmeler I

1. Giriş

Tedarik zinciri bir firma açısından, hammaddenin üreticiden alınması ile başlayıp, son tüketiciye ulaşmasını sağlayan süreçlerin toplamıdır. Tedarik zinciri içerisinde imalat, satın alma, satış, dağıtım gibi birçok faaliyeti barındırmaktadır. Ayrıca tedarik zinciri tedarikçileri, nakliyecileri, üretim tesisleri, dağıtım merkezleri, perakendecileri içeren bir kurumsal ağ topluluğudur. Bu sebeplerden ötürü tedarik zincirinin düzgün işlemesi işletmeler için çok önemlidir. Aksi takdirde şirketler, tedarik zincirinden kaynaklanan sorunlar sebebiyle yüksek maliyet, siparişlerde gecikme, yanlış ürün siparişi, hatalı sipariş miktarları gibi olumsuz durumlarla yüz yüze gelebilmektedir. Bu tür sorunlar şirkete hem maliyet açısından büyük yük getirmekte hem de şirketin hizmet ya da ürün sunduğu tüketicinin gözündeki marka değerini etkilemektedir. Şirketler, karşılaşabilecekleri sorunları en aza indirebilmek amacıyla zaman içinde sistemlerine bilgisayar yazılımlarını entegre etmişlerdir. İlk olarak işletmelerin sadece envanter takibine imkan veren MRP (Malzeme İhtiyaç Planlaması) yazılımları kullanılmaya başlanmıştır. Daha sonra şirketler bu programların yetersiz kalması üzerine ERP (Kurumsal Kaynak Planlaması) yazılımlarını kullanmayı tercih etmişlerdir.

Kurumsal Kaynak Planlaması (ERP), işletmelerin mal ve hizmet üretimlerini planlamalarına imkan veren, üretim, finans, pazarlama, insan kaynakları gibi modüller içeren, yönetim yazılımlarına verilen genel isimdir. Başta çokuluslu firmalar olmak üzere birçok işletme tarafından kullanılan bu yazılımlar verimlilik artışı sağlamada kritik role sahiptirler. Son yıllarda ERP yazılımlarının üretim, hizmet, finans, taşımacılık, kamu gibi hemen hemen her sektörde kullanımı yaygın hale gelmiştir. Bizim çalışmamız açısından ERP yazılımlarının temel amacı şirketlerin sahip olduğu tedarik zincirlerinin işleyişini düzenlemektir. Şirketler, ERP yazılımlarını kullanarak tedarik zincirinde gerçekleşmesi muhtemel sorunların etkilerini azaltmak istemektedirler.

Günümüzde belirsizlik, tedarik zinciri yönetiminde şirketlerin başa çıkması gereken çok önemli bir faktör olarak görülmektedir. Belirsizlik faktörü özellikle envanter planlamasının temel işlev olduğu perakendecilik gibi sektörlerde dikkatle ele alınmalıdır. Buna rağmen birçok planlama sistemi nadiren belirsizlik unsurlarını içeren planlama yöntemlerini kullanmaktadır.

Bu raporun ilk amacı Türkiye'de perakende sektöründe kullanılan tedarik zinciri envanter planlama sistemlerinin mevcut durumunun detaylı bir resmini ortaya koymaktır. Bu inceleme sonucunda perakende sektöründe, özellikle süpermarket zincirlerinde, farklı ihtiyaçlara cevap verecek ve hâlihazırda kullanıma sunulmuş olan çözümlerin envanteri çıkarılacaktır. Perakende sektörü için farklı işlevselliğe sahip çözümler üreten yerli ve yabancı yazılım firmalarının ürünlerinin bir karşılaştırması ve işletme yönetimi açısından bir değerlendirmesi bu bağlamda yapılmış olacaktır.

Araştırmanın ilk aşamasında dünyanın belli başlı ülkelerinde faaliyet gösteren ve kurumsal kaynak planlaması için yazılım sunan çok uluslu yazılım şirketleri ile birlikte yerli ERP yazılım şirketleri tespit edilmiştir. Bunlara açık kaynak kodlu ERP programları da ilave edilerek liste son haline getirilmiştir. Böylece kurumsal kaynak planlamasında kullanıcılara çözüm sunan başlıca şirketler çalışmaya dahil edilmiştir. Araştırma kapsamına alınan çok uluslu yazılım şirketleri Türkiye'de kurumsal kaynak planlama paketleri kullanılan şirketlerle sınırlı tutulmuştur. Türkiye'de kurulmuş ve kurumsal kaynak planlaması için yazılım üreten şirketlerin tamamına yakınına bu listede yer verilmiştir. Son olarak açık kaynak kodlu ERP yazılımlarının ise tamamı listede kendine yer bulmuştur.

Araştırmanın ikinci aşamasında mevcut kurumsal kaynak planlaması yazılımlarının tedarik zinciri yönetimine olan katkısını ve envanter planlaması bakımından kullanıcılara ne tür imkanlar sunduklarını belirleyebilmek için ERP yazılımları geliştiren şirketlerle bağlantıya geçilmiştir. Görüşmeler elektronik posta, telefon görüşmesi ya da yüz yüze görüşme yöntemlerinden bir veya birkaçı kullanılarak yapılmıştır. Yapılan bu görüşmelerle mevcut ERP yazılımlarının tahmin ve planlama konusundaki yetenekleri tespit edilmeye çalışılmıştır.

Görüşmelerde sorulan başlıca sorular şu şekilde sıralanabilir;

1- İlgili ERP yazılımı stokta tutulan kalem bazında stok takibi yapabiliyor mu?
2- Yazılım talep tahmini yapabiliyor mu? Cevap eğer evet ise talep tahmini tek dönemli (tek bir hafta/ay/yıl) mi yoksa çok dönemli (birden çok hafta/ay/yıl) mi yapılıyor?
3- Yapılan bu tahminler nokta tahmini şeklinde mi gerçekleşiyor, yoksa tahminleme hataları da tespit edilebiliyor mu?
4- Yazılım kullanıcıya ne zaman ve ne kadar sipariş verileceğini söyleyebiliyor mu?
5- Yazılım güvenlik stoğu hedefi sunabiliyor mu?
6- Güvenlik stoğu belirlemede kullanılan yöntemler/kurallar/yaklaşımlar nelerdir?
7- Güvenlik stoğu düzeyi optimal biçimde hesaplanabiliyor mu?

Görüşmelerde, yukarıdaki soruların ışığında mevcut ERP yazılımlarının hangi ihtiyaçları ne ölçüde giderebildikleri belirlenmeye çalışılmıştır. Bu görüşmelerin sonucunda elde edilen bilgiler raporun sonunda yer alan Tablo 1'de toplu olarak sunulmuştur.

2. ERP Yazılımı Sunan Başlıca Çok Uluslu Şirketler

SAP AG

SAP şirketi 1972 yılında IBM'de çalışan beş mühendisin ortak girişimi sonucunda kurulmuştur. Bir yıl sonra ilk finansal muhasebe programı geliştirilmiştir. Daha sonra bu program, R/1(R gerçek zamanlı veri işlemeyi temsil eder) sistemi olarak anılacaktır. 70'lerin sonuna doğru, SAP veritabanı ve diyalog kontrol sistemlerinde yapılan çalışmalar sonucunda SAP R/2 sistemi ortaya çıkmıştır. Ayrıca SAP R/2'nin bir diğer özelliği de farklı dil ve para kurlarına sahip çok uluslu şirketlerin ihtiyaçlarına karşılık verebilecek şekilde geliştirilmiş olmasıdır. 1988 yılında SAP şirketinin ismi SAP AG olarak değiştirilmiştir. SAP AG şirketi 1992 yılında R/2 sisteminden R/3 sistemine geçiş yapmıştır. Bu sistemin sahip olduğu kullanıcı-sunucu konsepti, birbiri ile bağlantılı veritabanlarının tutarlı bir şekilde kullanılmasını sağlamaktadır. Ayrıca sistem, farklı tedarikçilerin bilgisayarlarından gelen onaylama isteklerine cevap verebilmektedir [23].

SAP şirketinin kurumsal kaynak planlama programının ismi SAP ERP'dir. Bu program SAP Business Suite isimli beş kurumsal uygulamadan oluşan paketin birinci ürünüdür. Diğerleri SAP CRM (Müşteri İlişkileri Yönetimi), SAP PLM (Ürün Ömür Döngüsü Yönetimi), SAP SCM (Tedarik Zinciri Yönetimi) ve SAP SRM (Tedarikçi İlişkileri Yönetimi) olarak sıralanır.

Günümüzde 1500'ten fazla SAP ortağı, 25 endüstri odaklı iş çözümü ve 120 ülkede yaklaşık 47,800 müşterisi ile SAP şirketi dünyanın büyük yazılım şirketlerinden biridir. SAP şirketinin ana rekabet alanı Kurumsal Kaynak Planlaması Endüstrisi'dir. Ancak şirket, pazarlama ve satış yazılımı, üretim ve depo yazılımları, tedarik zinciri ve lojistik yazılımı alanlarında da yer almaktadır.

Oracle

Oracle şirketi Larry Ellison tarafından 1977 yılında Software Development Laboratories ismi ile kurulmuştur. Daha sonra sırasıyla 1979 yılında Relational Software Inc. ve 1982 yılında Oracle Systems olarak son halini almıştır. Oracle şirketi 1979 yılında Oracle veritabanı yazılımının ilk versiyonu olan Oracle 2'yi piyasaya sürmüştür. Oracle son olarak 2007 yılında veritabanı yazılımın 11g versiyonunu çıkarmıştır.

Günümüzde Oracle, büyük yazılım devlerinden biri haline gelmiştir. Orcale şirketi, SAP AG ile 1988 yılında başlayan, Sap R/3 sisteminin Oracle veritabanlarına entegrasyonunu sağlayan ve yaklaşık on yıl süren bir ortaklık geçirmiştir. Ancak şu an iki şirket birbirinin rakibi haline gelmiştir[20].

Oracle şirketi temelini veritabanı programının üstüne kurmakla birlikte zaman içinde başka alanlara yönelik programlar da hazırlamıştır. Oracle bankacılık, iletişim, perakende, endüstri uygulamaları gibi birçok farklı alana yönelik programlar oluşturmaktadır. Bunların arasında en önemli olanlar, Müşteri İlişkileri Yönetimi (CRM), Tedarik Zinciri Yönetimi (SCM), Üretim ve Kurumsal Kaynak Planlaması (ERP) paketleridir.

Infor ERP (Baan)

Baan firması Jan Baan tarafından 1978 yılında Hollanda'da, finansal ve yönetimsel danışmalık servisleri sağlamak amacı ile kurulmuştur. Baan yazılımı, şöhretini teknik mimarisine ve 4GL diline borçludur. Boeing firmasının açtığı ihaleyi kazanmaları önemli başarılarından biridir. Ancak şirketin halka arzından sonra sermaye piyasasına satış rakamları yönetim tarafından abartılı olarak verilmiştir. Bu durum keşfedildiğinde ise 1998 yılında Baan hisseleri ani bir düşüş yaşamıştır. Bunun üzerine şirket 2000 yılında ekonomik sebeplerden dolayı Invensys şirketine satılmış sonrasında ise 2006 yılında Infor Global Solutions Şirketi tarafından satın alınmıştır [2].

Baan şirketi kurulduğundan itibaren kurumsal kaynak planlaması yazılımı üzerine odaklanarak bütün gücünü bu konu üstünde harcamıştır. Baan ERP yazılımı hizmet yönetimi, yalın üretim, kalite yönetimi, süreç yönetimi ve tedarik zinciri yönetimi gibi birçok alanda kullanıcıya destek sağlamaktadır.

IFS

IFS şirketi, 1983 yılında nükleer santral kontrolünü sağlayacak bir yönetim yazılımı geliştirilmesi için İsveç'te kurulmuştur. Özellikle üretim, teknoloji ve kamu sektörlerine odaklanmıştır.

IFS Uygulamaları (IFS Applications) şirketlere, bütün modülleri ile uygulanan ve 60'dan fazla bileşene sahip bir yazılımdır. IFS Applications'ı SAP gibi diğer ERP paketlerinden ayıran fark, bu yazılımın sunduğu Web hizmetleri modülünün önceden yüklenmiş olarak satın alınmasıdır [10].

The Sage Group

Sage, David Goldman ve Graham Wylie tarafından 1981 yılında Newcastle şehrinde, küçük ölçekli firmalar için muhasebe ve tahmine yönelik yazılım geliştirmek amacı ile kurulmuştur. Şirket 1984 yılında CP/M işletim sistemini kullanan Sage yazılımını piyasaya sürmüştür. 2001 yılında müşteri ilişkileri yönetimi alanında yazılım üretmeye başlamıştır.

Sage şirketinin, muhasebe, üretim ve insan kaynakları bölümlerinin ihtiyaçlarına yönelik birçok yazılımı bulunmaktadır. Şirket, Sage Mas 500 isimli ERP yazılımı ile müşterilerine finans, envanter yönetimi, sipariş yönetimi, müşteri ilişkileri yönetimi ve insan kaynakları yönetimi konularında destek hizmeti vermektedir [29].

IAS (Industrial Application Software)

IAS Bilgi İşlemleri Danışmanlık A.Ş. 1989 yılında gerek küçük ve orta ölçekli, gerekse büyük ölçekli şirketlerin kurumsal ihtiyaçlarını karşılamak amacı ile Almanya'da kurulmuştur. Yazılım geliştirme ofisi İstanbul'da bulunan şirket İtalya, İngiltere, Hollanda, Fransa gibi birçok Avrupa ülkesine ilaveten Çin ve Tayvan gibi uzak doğu ülkelerine proje yönetimi, danışmanlık, eğitim ve özel yazılım geliştirme hizmetleri sunmaktadır [5].

Şirketin sunduğu CANIAS ERP paketi standart yapısında lojistik, üretim ve kapasite kontrol, muhasebe ve finans, fiilî maliyetlendirme, müşteri ilişkileri yönetimi, insan kaynakları yönetimi, döküman yönetimi, bakım yönetimi gibi modüllerden oluşmaktadır.

SAS

SAS Institute 1976 yılında Anthony Barr, James Goodnight, John Sall and Jane Helwig tarafından yazılım üreticisi olarak Amerika'da kuruldu. SAS aslında "istatistiksel analiz sisteminin" kısaltması olmasına rağmen daha sonra marka ismi haline dönüşmüştür.

SAS şirketinin ana ürünü IBM sistemli bilgisayarlarda çalışan çok sayıda modülleri içeren istatistiksel analiz sunmaktadır. Fakat özellikle Windows programının kullanımının yaygınlaşmasından sonra kendi yazılımlarını Windows'a uyarlamıştır. SAS müşteri ilişkileri yönetimi ve iş zekâsına yönelik modüller içermektedir. Program dördüncü nesil programlama diline sahiptir [25].

Havacılık, otomobil, bankacılık, iletişim, üretim, perakende gibi birçok sektöre hizmet veren şirket raporlama, sorgulama, veri madenciliği, öngörü, optimizasyon konularında firmalara yazılım desteği sunmaktadır.

3. ERP Yazılımı sunan başlıca Türk firmaları

LOGO Yazılım

Logo yazılım 1984 yılında, Türkiye'de kişisel bilgisayarlar için mühendislik yazılımları geliştirmek üzere kurulmuştur. Logo, bugün hepsi bilişim teknolojilerine odaklı, yedi şirketten oluşan bir teknoloji grubu haline gelmiştir. Logo'nun en yaygın ürün ve hizmetleri KOBİ'ler için özel olarak tasarlanmış verimlilik ve rekabetçilik çözümleridir. Bu çözümler arasında çeşitli büyüklüklere göre özel tasarlanmış kurumsal kaynak yönetimi, tedarik ve talep zinciri otomasyonu, müşteri ilişkileri yönetimi, iş süreçleri tasarımı danışmanlığı ürün ve hizmetleri sayılabilir.

Logo şirketinin geliştirdiği en son ürün kurumsal kaynak planlaması için hazırlanan Logo Unity On Demand yazılımıdır. Logo Unity, Java programlama dili ile yazılmış olup IBM DB2, MySQL ve Oracle veritabanları ile çalışabilmektedir [13].

Netsis

1991 yılında kurulan Netsis, farklı sektör ve farklı ölçekteki firmalara çağdaş ve uluslararası kriterlere uygun kurumsal iş yazılımları geliştirmektedir. Yüzde yüz Türk sermayeli bir kuruluş olan Netsis bugün İstanbul, Ankara, İzmir'deki Türkiye bölge ofisleri, İzmir Yüksek Teknoloji Enstitüsü Teknoparkı'ndaki ana yazılım üssü ve Türkiye'ye dağılmış, sayısı 400'ü bulan iş ortağı kanalı ile tüm ülke geneline hizmet sunmaktadır.

Netsis firmasının başlıca yazılımları kurumsal kaynak planlaması Fushion@6 ve küçük ölçekli işletmelere yönelik hazırlanmış Entegre@6'dır. Fushion@6 serisi müşteri ilişkileri yönetimi, tedarik zinciri yönetimi, finans yönetimi ve insan kaynakları yönetimi gibi konularda işletmelere destek sağlamaktadır [15].

OBASE

OBASE 1995 yılında yazılım ve danışmanlık hizmetleri vermek amacıyla kurulmuştur. Müşterilerine veri entegrasyonu, iş zekâsı, kurumsal performans yönetimi, müşteri ilişkileri yönetimi gibi hizmetler sunmaktadır. OBASE bilgi yönetimine yönelik yazılımları ile perakende, telekomünikasyon, ulaşım, finans, üretim, sigorta, lojistik, ilaç, tekstil gibi sektörlerde faaliyet göstermektedir.

OBASE Retailer, Detailer, Muhasebe ve Finansman Yönetimi, Personel Yönetimi ve Reporter yazılımları ile firmalara destek sağlamaktadır. Retailer ve Detailer gibi perakendeciliğe yönelik paketler, bu sektördeki firmalara sipariş, alım-satım, stok yönetimi, talep yönetimi konularında hizmet sunmaktadır [17].

Akınsoft

Akınsoft şirketi 1995 yılında Konya'da kurulmuştur. Yurt içi ve yurt dışında çeşitli ülkelere kurumsal yazılım alanında çözüm önerileri sunmaktadır. Firma, sunduğu programları ticari, sektörel, web tabanlı ve DOS tabanlı olarak ayrılmaktadır. Akınsoft şirketlere ERP başta olmak üzere MRP, MRPII, CRM gibi yazılım çeşitleri ile destek vermektedir. Şirketin Wolvox olarak adlandırdığı ERP yazılımı malzeme yönetimi, malzeme ihtiyaç planlaması, kapasite ihtiyaç planlaması, maliyet muhasebesi, genel muhasebe, üretim kontrol ve kalite kontrol sistemleri, planlama, bütçe, bakım, insan kaynakları ve ön muhasebe modüllerini kapsamaktadır [1].

Mikro Yazılım Evi

Mikro Yazılım Evi, merkezi İstanbul'da bulunan genelde küçük ve orta ölçekli işletmelere yönelik bilişim çözümleri sunan bir firmadır. Ürünleri ERP, Retail ve Standart seri olmak üzere üç sınıfa ayrılmıştır. ERP sınıfının içinde öne çıkan başlıca yazılımları, şirketlerin büyüklük ve ihtiyaçlarına göre uyarlanmış my-ERP, ERP9000 ve MRPII 9000'dir. Perakendecilere özel çözümler sunan Retail sınıfının içinde Retail 9000, Ofis ve Microshop ürünleri bulunmaktadır. Firmanın kurumsal kaynak planla yazılımı olan ERP9000 paketi üretim kaynak planlaması, satın alma ve satış, stok yönetimi, finans, personel yönetimi, muhasebe modüllerine sahiptir.

4. Açık Kodlu ERP Yazılımları

SQL Ledger

SQL Ledger küçük ve orta ölçekli firmaların temel ihtiyaçlarını karşılamak üzere geliştirilmiş, muhasebe ve üretime yönelik bir açık kodlu ERP yazılımıdır. Programın özellikleri ulusal muhasebe standartlarına uyarlanabilirlik açısından yeterlidir. Muhasebe bölümü yanında alım, satım ve envanter yönetimi için de fonksiyonları mevcuttur. Fonksiyonları ve kullanıcı ara yüzü internet üzerinden test edilebilmektedir. Windows, Linux ve Mac OS X gibi işletim sistemlerinde çalışabilmektedir. Oracle, IBM DB2 ve Postgre SQl gibi veri tabanlarını kullanabilmektedir [9].

GNUe(Enterprise) ERP

GNUe yazılımı, form, rapor ve iş akış şemaları bütün veritabanı sürücülerinde çalışabilen ve XML yapıda olan, bütünsel bir ERP yazılım paketi sunmaya odaklanmıştır. Yazılımın mimarisi ve ara yüzü açık kodlara dayalı çeşitli programlama dilleriyle geliştirilmiştir [8]. Paketsel çözümleri güncel olmakla birlikte, tüm ERP modülleri sürekli geliştirilme durumundadır. Neredeyse bütün işletim sistemlerinde çalışabilen yazılım, PostgreSQL, MySQL, MaxDB, Oracle IBM DB2, Sybase veritabanlarını kullanabilmektedir. Python dili ile yazılmış olan program, diğer yazılım dilleri ile geliştirilmesine izin vermektedir [6].

ERP5

ERP5, 300'den fazla işçi kapasitesine sahip ve uluslararası dağıtım alanlarında faaliyet gösteren tekstil endüstrisindeki şirketlerin, organizasyonel yapısını ve karar alma sürecini desteklemek amacıyla geliştirilmiştir. Birçok modülü olan programda yer alan başlıca modüller şunlardır:

- Ticaret: Alım, satım, sipariş ve depolama fonksiyonları
- Ürün veri yönetimi: Ürün tanımı, çeşitleri, kategorizasyonunu sağlar
- Malzeme ihtiyaç planlaması
- Müşteri ilişkileri yönetimi
- Muhasebe
- İnsan kaynakları

ERP5 Windows, Linux ve Mac Os X işletim sistemlerinde çalışabilmektedir. Programın yazılım dili Python'dur. Açık kodlu olarak oluşturulan Zope veritabanını kullanmaktadır [7].

Opentaps

Opentaps programı, kurumsal seviyede ayarlanabilir iş uygulamaları geliştirme imkânı sunabilmektedir. Odak noktası e-ticaret olmasına rağmen ERP alanında da gelişme kaydetmektedir. Programın hedef grupları arasında perakendeci, dağıtımcı ve üreticiler yer almaktadır. E-ticaret, ürün kataloğu, sipariş emri, envanter ve depo yönetimi, üretim, müşteri hizmetleri, müşteri ilişkileri yönetimi ve pazarlama yönetimi için modülleri bulunmaktadır. Veri modelleri planlama, öngörü ve bütçeleme özelliklerine sahiptir [11].

Opentaps programı Windows, Linux, MAC OS X işletim sistemlerinde çalışabilmektedir. Veritabanları olarak MySQL, MaxDB, PostgreSQL, Oracle, Microsoft SQL, IBM DB2'yi kullanabilmektedir. Yazılım dili olarak Java ve XML'i kullanan program, 39 ayrı dil desteği sunmaktadır [3].

Compiere

Compiere perakende, dağıtım ve hizmet alanlarında faaliyet gösteren küçük ve orta ölçekli işletmelere yönelmiştir. Müşteri işlemleri (ön-satış, satış, sipariş ve diğer aktiviteler), tedarikçi ilişkileri, muhasebe ve kontrol, hizmet yönetimi ve temel proje yönetimi fonksiyonları mevcuttur. Şu anda en popüler açık kodlu ERP yazılımlarının başında gelen Compiere, şimdiye kadar 900,000 den fazla kullanıcı tarafından indirilmiştir.

Compiere yazılımı Windows, Linux ve Mac OS X işletim sistemlerinde çalışabilmektedir. Java programla dili ile geliştirilmiş program, Oracle ve Sybase veritabanlarını kullanabilmektedir. Ayrıca 26 farklı dil desteğine ve 4 çeşit muhasebe düzenine sahiptir [21].

OpenERP

OpenERP'nin ara yüzünde birçok grafik ve diyagram tipleri mevcuttur. Program işçi sayısı 150'ye kadar olan ticaret, dağıtım ve hizmet sektöründe faaliyet gösteren küçük ve orta ölçekli işletmeleri hedeflemektedir. Muhasebe, müşteri ilişkileri yönetimi, alım-satım, insan kaynakları, pazarlama, malzeme ihtiyaç planlaması, envanter kontrolü, proje yönetimi için modülleri bulunmaktadır. Ara yüz ve e-ticaret için bazı özel amaçlı modüller de OpenERP yazılımının içine entegre edilmiştir.

OpenERP programı XML, Excel, Pdf ve Open Office ile oluşturulmuş ara yüzleri kullanabilmektedir. Windows, Linux ve Mac OS X işletim sistemlerinde çalışabilen program PostgreSQL veritabanını kullanmaktadır. Python programlama dili ile yazılmıştır. 11 faklı dil seçeneği bulunmaktadır [9].

OpenPro

OpenPro, web üzerinde açık kodlu olarak tamamlanmış ERP yazılımını 1999 yılında kullanıcılarına sunmuştur. OpenPro yazılımında müşteri ilişkileri yönetimi, tedarik zinciri yönetimi, e-ticaret, finans ve muhasebe, insan kaynakları modülleri bulunmaktadır. Esnek bir yapıya sahip olan yazılım Windows, Linux, Unix, Sun, IBM işletim sistemlerinde çalışabilmektedir. Ms-SQL, MySQL, Oracle, IBM DB2, Postgre SQL veritabanlarını kullanabilmektedir. OpenPro yazılımı php programlama dili ile yazılmış olup, ara yüzü kullanıcıların isteği doğrultusunda yeniden tasarlanabilme özelliğine sahiptir [19].

Openbravo

Openbravo yazılımının temeli doksanların ortalarında Navara Üniversitesi'nde okuyan Nicolas Serrano ve Ismael Ciordia isimli iki öğrencinin, üniversitenin yönetim sistemini düzenleyecek bir yazlım oluşturmaları ile atılmıştır. 2001 yılına gelindiğinde kurumsal yönetim sistemlerine uygun hale getirilen Openbravo ERP kullanıma sunulmuştur.

Openbarvo ERP ve Openbarvo Pos olmak üzere iki yazılım üzerine odaklanılmıştır. Openbravo ERP veri yönetimi, satın alım yönetimi, depo yönetimi, üretim yönetimi, satış yönetimi, finans ve muhasebe, proje yönetimi modüllerini içermektedir. Java dili ile geliştirilmiş program XML, Html ve Javascript'i de desteklemektedir [18].

5. Değerlendirme Kriterleri

Araştırmaya konu edinilen ERP yazılımlarının sunduğu özelliklerden yola çıkılarak hazırlanan Tablo 1'de, tedarik zinciri ve envanter planlaması için önemli olan belli başlı kriterler belirlenmiştir. Tablonun satırlarını ERP yazılımları, sütunlarını ise belirlenen kriterler oluşturmaktadır. Böylece bu yazılımların sundukları temel envanter planlama işlevselliği özet olarak sunulmuştur. Tabloda yer alan kriterler şunlardır;

i) **İzleme:** Bu kriter, yazılım paketinde kalem bazında stok takibi yapılıp yapılmadığını göstermektedir. Yani bir parçanın satın alımdan itibaren, firma içindeki hareketlerinden çıkışına kadar olan süreci, kullanıcıya gösterebilme özelliğidir.

ii) **Öngörü (Tahmin):** Yazılımın ürünler için talep öngörüsü yapabilme kapasitesini göştermektedir. Bu kriter nokta tahmini ve tahmin hatası olmak üzere ikiye ayrılmıştır. Nokta tahmin bir kalem için belirli bir zaman diliminde gerçekleşen talep miktarının tahminidir. Her hangi bir olasılıksal bilgi içermez. Tahmin hatası kriteri, yazılımın stokastik tahminleme yapıp yapmadığını gösterir. Tahmin hatası, gerçekleşecek talebin, tahmini değerden olası sapma miktarına ilişkin bir hata değeridir.

iii) **Deterministik Planlama:** Yazılım paketinin, kullanıcı ihtiyaçları doğrultusunda talep, fiyat, tedarik süresi hakkında bilgilerin kesin olarak bilindiği varsayılarak planlama yapmasıdır. Burada yazılım ile ilgili, iki önemli noktaya dikkat edilmektedir. Birincisi yazılımın bu planlamayı bir "heuristik" (sezgisel yaklaşım) kullanarak yapması, diğeri ise bu planlamanın optimal olup olmadığıdır.

iv) **Stokastik Planlama:** Yazılımın talep, fiyat, tedarik süresi ve miktarının yalnızca olasılıksal dağılımlarının bilindiği varsayılarak kullanıcının ihtiyacı doğrultusunda ileriye dönük planlama yapabilme kapasitesidir. Burada yazılım stoksuz kalma durumunu engellemek için taşınması gereken güvenlik stoğunu belirlemektedir. Bu kriter durağan ve durağan olmayan stokastik planlama olmak üzere ikiye ayrılmıştır. Durağan stokastik planlama kapsamında talep gibi bir belirsizlik unsurunun dönemler arasında olasılık dağılımının farklılık göstermediği varsayımı yapılmaktadır. Durağan olmayan stokastik planlamada ise bunun aksine dağılımların dönemsellik gösterdiği varsayılmaktadır ve bu yüzden öncekine göre daha geneldir ve perakende sektörü gibi dönemselliğin önemli olduğu sektörlerde kullanılması gerekmektedir.

Oluşturulan bu tablo sayesinde mevcut kurumsal kaynak planlama yazılımlarının envanter planlaması bakımından işlevselliklerinin bir envanteri çıkartılmıştır. Böylece ERP yazılımlarında ulaşılan noktanın bir resmi ortaya konmuştur.

6. Bulgular

Bu kısımda önceki bölümde verilen kriterler çerçevesinde ERP yazılımlarının perakende tedarik zinciri yönetimi alanına dönük olarak sundukları işlevselliklerin bir değerlendirmesi sunulmuştur.

6.1. Çok Uluslu ERP Yazılım Şirketleri

Bu yazılımlar karmaşık yapıdaki organizasyonların sorunlarını çözmeye yönelik olduğundan, genellikle büyük şirketler tarafından tercih edilmektedirler. Uzmanların yardımı ile şirket bünyesine entegre edildikten sonra şirketin işleyişine yardımcı olmaktadırlar. Ayrıca verimli bir şekilde kullanılabilmeleri için deneyimli elemanlar tarafından kullanılmaları gerekmektedir. Gerek uzman kişi gereksinimi gerekse yüksek kurulum masraflarından dolayı küçük ve orta büyüklükteki şirketler tarafından kullanılma oranları düşüktür.

SAP R/3

SAP AG firmasının ERP paketi olan SAP R/3, kullanıcılarına birçok yönden destek sağlamaktadır. Yazılımın talep planlama, üretim, sipariş, RFID, dağıtım ve depo modülleri bulunmaktadır. Envanter planlaması bakımdan kalem bazında stok takibi yapabilmektedir. Talep tahmini, geçmiş veriler kullanılarak üstel düzeltme, hareketli ortalama, doğrusal regresyon yöntemleri ile yapılmaktadır.

SAP R/3 yazılımı deterministik planlama yapabilen yazılımlardan biridir. Tahmin aşamasında elde edilen veriler (*i*) Groff prosedürü, (*ii*) Parça Dönem Dengesi (Part Period Balancing), (*iii*) Dinamik Parti Büyüklüğü (Dynamic Lot Sizing) yöntemleri içinde kullanılarak deterministik optimizasyon gerçekleştirmektedir [27].

Oracle

Oracle firmasının ERP paketi kalem bazında stok takibi yapabilmektedir. Oracle talep tahminini "Demantra" isimli modül yapmaktadır. Bu modül esas olarak Bayesgil yaklaşım kullanmaktadır. Bayesgil tahmin yaklaşımı, geçmiş satış verileri ile planlanan satış verilerinin birleştirilmesi temeline dayanmaktadır. Geçmiş verilerin bulunmadığı zamanlarda bile tahmin yapabilmektedir. Bu durum, daha önce planlanan satış verileri ile satış anında elde edilen verilerin kullanılması ile gerçekleşmektedir [28].

Oracle ERP yazılımı kullanıcılara deterministik planlama sunabilen bir diğer pakettir. Sipariş yenileme optimizasyonu paketi; satış hacmini, tahmini verilerin uygunluğu, mevsimsel değişimleri, kullanıcı kısıtlarını dikkate alarak, stokların optimal seviyesini belirlemeye çalışmaktadır. Satışlardaki ve ürün çeşitliliğindeki çokluk ya da azlığa göre öneriler sunabilen yazılım, (*i*) "MinMax", (*ii*) "Dynamic", (*iii*) "Poisson", (*iv*) "TimeSupply" ve (*v*) "Hybrid" isimli sipariş yenileme yöntemlerini kullanmaktadır [16].

IFS Applications

IFS şirketinin ERP yazılımı kalem bazında stok takibine imkân vermektedir. Talep tahmini yapabilen yazılım hareketli ortalama, en küçük kareler regresyonu, üstel düzlemeli hareketli ortalama, Brown modeli, Bayesgil ve çoklu regresyon yöntemlerinden faydalanmaktadır. Ayrıca ortalamadan sapma, ortalama hatası, ortalama hata yüzdesi, Theil U-istatistiği yöntemlerini kullanarak tahmin hatalarını belirleyebilmektedir. IFS ERP yazılımı kullanıcısına ne zaman ve ne kadar sipariş vermesi gerektiğini bildiren bir modüle sahip değildir [12].

Infor ERP (Baan)

INFOR ERP yazılımı stok takibi yapabilme özelliğine sahiptir. Yazılım nokta talep tahmininin yanında, tahminleme hatalarını da belirleyebilmektedir. INFOR ERP paketi kullanıcılarına planlama imkânı sunmamaktadır [30].

Sage MAS 500

Sage MAS 500 ERP yazılımı stokta tutulan kalem bazında stok takibi yapmaktadır. Buna ilaveten çeşitli istatistiksel yöntemler kullanarak talep tahmini yapabilmektedir. Talep tahmininde olası hata oranlarını hesaplayabilmektedir. Sage MAS 500 ERP yazılımında planlama olanağı bulunmamaktadır [22].

CANIAS ERP

CANIAS ERP paketi stok takibi yapabilme özelliğine sahiptir. Talep öngörüsünü çeşitli istatistiksel yöntemlerin yardımı ile gerçekleştirmektedir. Ayrıca tahminlemede olası hata paylarını hesaba katabilmektedir. Bu yazılım kullanıcıya planlama bakımından bir çözüm sunamamaktadır. Güvenlik stoğu hesaplaması mevcut değildir [4].

SAS

SAS ERP yazılımının kalem bazında stok takibi yapabilme kapasitesi vardır. Yazılım talep tahmini yapabilmesine olanak kılan “High Performance Forecasting 2.3” isimli bir modüle sahiptir. Bu modülün kullandığı yöntemler; mevsimsel modeller, yerel modeller (Winters metodu, basit yapısal model), ARIMA zaman serileri, Croston metodu olarak sıralanmaktadır [24].

SAS ERP, araştırmada incelediğimiz ERP paketleri arasında durağan stokastik planlama yapabilme özelliğine sahip tek yazılımdır. Yazılım durağan stokastik planlamayı “Inventory Optimization 1.3” modülü ile yapmaktadır. Programın bu tür planlama için

SS(s,S), BS(base stock), NQ(s, nQ), RQ politikaları olmak üzere dört çeşit yöntem kullanmaktadır. Modül, talep ve tedarik sürelerinin ortalamasını ve varyansını hesaplayarak tercih edilen politika doğrultusunda hesaplama yapmaktadır. Ayrıca kullanıcı için ceza maliyeti ve servis düzeyi hesaplamaları yapılarak güvelik stoğu hedefi belirlenebilmektedir [26].

6.2. Türkiye'de Geliştirilen ERP Yazılımları

Türkiye'de kurulmuş yazılım şirketlerinin kullanıcıya sunduğu ERP paketlerinin çoğu, küçük ve orta ölçekli şirketlerin ihtiyaçları doğrultusunda hazırlanmıştır. Mevcut ERP paketlerinin büyük çoğunluğu imalat sanayiinde faaliyet gösteren işletmelere yöneliktir. Fakat bir kısmının bankacılık, tekstil, gıda, ilaç, perakende gibi diğer sektörlere de özel çözümleri bulunmaktadır. ERP paketleri finans, envanter, muhasebe, insan kaynakları, müşteri ilişkileri yönetimi, depo yönetimi, tedarik zinciri yönetimi gibi konularda destek sağlamaktadır.

Türkiye'de ERP yazılımlarının kullanımı günden güne artış göstermektedir. Şirketlerde bu yazılımları kullanabilecek düzeyde bilgiye sahip çalışan sayısının düşük olması, yazılımların en etkin şekilde kullanılmasının önündeki en büyük engeldir.

Netsis Fushion@6

Netsis firmasının sunduğu Fushion@6 ERP paketi kalem bazında stok takibi yapabilmektedir. Yazılım hareketli ortalama, trend modeli, mevsimsel model, ağırlıklandırılmış hareketli ortalama gibi çeşitli istatistiksel yöntemler kullanarak talep tahmini yapabilmektedir. Talep tahminleri için olası hataları hesaplayamamaktadır. Yazılımın herhangi bir yöntem ile planlama yapma özelliği bulunmamaktadır [14].

LOGO Unity

LOGO Unity ERP yazılımı ürün bazında stok takibi yapabilmektedir. İstatistiksel yöntemlerden faydalanarak talep tahmini yapabilme desteği sunan paket, taleplerdeki hata paylarını tespit etmemektedir. Planlama yapma desteği bulunmamaktadır.

OBASE

OBASE ERP paketi kalem bazında stok takibi yapabilme özelliğine sahiptir. Yazılım geçmiş verileri (normalize edilmiş ortalama satış miktarlarını) kullanarak talep tahmini yapabilmektedir. Güvelik stoğu miktarını kullanıcı belirlemektedir. Tahminleme hatalarını hesaplayamayan yazılım, planlama bakımından kullanıcıya destek vermemektedir.

Mikro ERP

Bu yazılım stok takibi yapılmasına olanak sağlamaktadır. Ürünler için tek dönemli talep tahmini yapılabilmektedir. Yapılan bu talep tahminleri satış hareketleri ve kritik seviyeler dikkate alınarak yapılan nokta tahmini ile gerçekleşir. Güvenlik stoğu seviyesini belirleme yöntemi bulunmamakla birlikte, bunu kullanıcıların kendilerinin belirlemesi gereklidir. İleriye dönük bir planlama yapma olanağı yoktur.

Wolvox E-Business

Wolvox E-Business kullanıcılarına stok takibi yapabilme özelliği sunmaktadır. Yazılım kullanıcıya çok dönemli talep tahmini yapmasının yanında bu tahminlerde olacak olası tahminleme hatalarını içermektedir. Planlama özelliği bulunmayan yazılım güvenlik stoğu hedefi sunmamaktadır.

6.3. Açık Kodlu ERP Yazılımları

Araştırma dâhilindeki açık kaynak kodlu ERP yazılımlarının çoğunluğu, küçük ve orta büyüklükteki işletmelerin sorunlarına çözüm getirme amaçlıdır. Bu yazılımlar muhasebe, tedarik zinciri yönetimi, müşteri ilişkileri yönetimi, stok, finans, insan kaynakları gibi birçok bölüm için hazırlanmış modüllere sahiptirler. Bu tür yazılımların en büyük avantajı, açık kodlu olduklarından sürekli gelişime elverişli olmasıdır. Kullanıcılar tarafından eksik görülen noktaların, bu yazılımların forumlarında belirtilerek kısa zamanda giderilmesi mümkündür. Ayrıca bu tür ERP paketleri internet üzerinden elde edilerek kısa zamanda sisteme entegre edilebilir.

Stok yönetimi açısından baktığımızda ise incelenen bütün açık kodlu ERP yazılımları, parça bazında stok takibi yapabilme özelliğine sahiptir. Fakat diğer taraftan, bu kategorideki hiçbir yazılım ne talep tahmini ne de herhangi bir şekilde planlama yapamamaktadır.

7. Sonuç

Görüşmeler yapılmadan önce mevcut ERP yazılımlarının stok takibi ve talep tahmini yapabildikleri tahmin ediliyordu. Fakat birçoğunun gerek deterministik gerekse stokastik planlama yönünden yetersiz kalacağı düşünülmekteydi. Görüşmeler sonucunda sadece SAS şirketinin yazılımının durağan stokastik planlama yapabildiği tespit edilmiştir. Birkaç büyük yazılım firmasının deterministik planlama yapabildikleri, geriye kalan firmaların ise tahmin yapabilmekten öteye gidemedikleri ortaya çıkmıştır.

Araştırma dâhilindeki hiçbir şirketin durağan olmayan talep altında stokastik planlama yapamaması, bu yönde yapılacak çalışmaların önemini ve gerekliliğini göstermiştir. Bununla birlikte çok uluslu olan yazılım şirketlerinin ürünlerinin genellikle büyük şirketlerin kullandığı, yerel bazda olan yazılım şirketlerinin ise küçük ve orta büyüklükteki şirketlere yönelik ürün geliştirdikleri tespit edilmiştir.

ERP yazılımların birçoğu öncelikle imalat sanayiinde faaliyet gösteren işletmelerin ihtiyaçlarını karşılamaya dönük olarak hazırlanmıştır. Perakende sektörüne yönelik çözümler de, bu paketlere ilave edilmiş veya ayrı bir paket halinde piyasaya sunulmuştur. Bu yüzden birçoğunun eksiği bulunması ile birlikte günden güne bu eksiklerin giderilmeye çalışılmakta olduğu görülmüştür.

Projemiz kapsamında önümüzdeki dönemlerde ele alınacak olan perakende tedarik zinciri envanter yönetimi optimizasyon modelleri durağan olmayan talep varsayımını taşıyacaktır. Bu raporda sunulan bulgular, geliştirilecek olan modellerin ve çözüm yöntemlerinin bahsi geçen çabalara büyük destek vermenin de ötesine geçeceği ve bu tartışmaların merkezine oturacağı bekletimizi destekler niteliktedir.

TABLO 1

		Öngörü		Planlama Deterministik		Planlama Stokastik	
	İzleme	Nokta	Hata	Heuristik	Optimal	Durağan	Durağan Olmayan
SAP R/3	EVET	EVET	EVET	EVET	EVET	HAYIR	HAYIR
Oracle	EVET	EVET	EVET	EVET	EVET	HAYIR	HAYIR
INFOR ERP(Baan)	EVET	EVET	EVET	HAYIR	HAYIR	HAYIR	HAYIR
IFS Application	EVET	EVET	EVET	HAYIR	HAYIR	HAYIR	HAYIR
SAGE MAS 500	EVET	EVET	EVET	HAYIR	HAYIR	HAYIR	HAYIR
CANIAS ERP	EVET	EVET	EVET	HAYIR	HAYIR	HAYIR	HAYIR
SAS	EVET	EVET	EVET	EVET	EVET	EVET	HAYIR
Fusion@6	EVET	EVET	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR
LOGO Unity	EVET	EVET	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR
OBASE	EVET	EVET	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR
Wolvox	EVET	EVET	EVET	HAYIR	HAYIR	HAYIR	HAYIR
Micro	EVET	EVET	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR
Compiere	EVET	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR
ERP5	EVET	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR
GNU ERP	EVET	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR
SQL Ledger	EVET	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR
OFBiz	EVET	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR
Opentaps	EVET	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR
Openbravo	EVET	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR
OpenERP	EVET	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR
OpenPro	EVET	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR	HAYIR

Kaynakça

[1] Akınsoft, 2008, kaynak: http://www.akinsoft.com.tr/as/hakkimizda/hakkimizda.php

[2] Baan ERP History, 2008, kaynak: http://en.wikipedia.org/wiki/Baan

[3] Basil Argasosy: OfBiz An Insider View, 2005, kaynak: http:ofbizwiki1.gointegral.com/Wiki.jsp?page=OFBizInsiderTutorial.

[4] CaniasERP Envanter Yönetim Modülü, 2007, Canias.

[5] Canias ERP Kurumsal, 2008, kaynak: http://www.ias.com.tr/enterprise/hr/about-us.html

[6] Carter, J., 2005, "GNUe Forms: A Developer's Introduction 0.5.4", kaynak: http://www.gnuenterprise.org/tools/forms/docs/Developers-Guide.pdf

[7] Deldycke, K., 2005, "ERP5 Tutorial: Develop your own ERP with ERP5 Business Templates 0.9", kaynak: http://www.erp5.org/sections/documentation/articles/erp5_developer_tutor3829/downloadFile/file/Tutorial-Kevin-en.pdf

[8] GNUe Brochure, 2002, kaynak: http://www.gnu.org/software/gnue/brochures/Brochure.pdf

[9] Herzog, T., 2006, "A Comprasion of Open Source ERP Systems", Vienna University of Economics and Business Administration.

[10] IFS, 2008, kaynak: http://en.wikipedia.org/wiki/Industrial_and_Financial_Systems

[11] Jones, D. E., 2005, Getting and using OfBiz, kaynak: http://www.ofbiz.org/docs/GettingAndUsingOFBiz.pdf,

[12] IFS Demand Planning Brochure, 2004, IFS Applications.

[13] Logo, 2008, kaynak: http://tr.wikipedia.org/wiki/Logo_Business_Solutions

[14] Netsis Ana Üretim Planı, 2006, Netsis, kaynak: http://www.netsis.com.tr/dokumanlar.aspx.

[15] Netsis Tarih, 2008, kaynak: http://www.netsis.com.tr/netsishakkinda.aspx

[16] O'Hara, G., 2007, Oracle Retail Replenishment Optimization, Manual, Oracle.

[17] Obase, 2008, kaynak: http://www.obase.com/obase.aspx?page=1

[18] Openbravo ERP, 2008, kaynak: http://www.openbravo.com/product/erp/features.

[19] OpenPro ERP, 2008, kaynak: http://www.openpro.com/erp.html

[20] Oracle History, 2008, kaynak: http://www.oracle.com/corporate/story.html

[21] Pink, K., Janke, J., Wassmer, A., 2005, "Compiere User Manual Version 2.52e", kaynak: www.compiere.org.

[22] Sage Mas 1000 Brochure, 2007, Sage Limited, kaynak: http://www.sage.co.uk

[23] SAP AG History, 2008, kaynak: http://en.wikipedia.org/wiki/SAP_(company)

[24] SAS High-Performance Forecasting 2.3, 2007, Manual, SAS Institute INC.

[25] SAS History, 2008, kaynak: http://www.sas.com/presscenter/bgndr_history.html

[26] SAS Inventory Optimization 1.3, 2006, User Guide, SAS Institute INC.

[27] Theis, S., 2004, SAP SCM 4.1 Supply Chain Planning.

[28] The Bayesian Approach to Forecasting, 2006, Manual, Oracle White Paper.

[29] The Sage Group, 2008, kaynak: http://en.wikipedia.org/wiki/The_Sage_Group

[30] Wagner, G., 2000, "Understanding the Enterprice Information System Technology of BaanERP", Institut für Informatik Freie Universitat Berlin.

SOBAG-108K027

Türkiye'de Süpermarket Tedarik Zinciri Envanter Yönetimi için Kullanılan Sistemler Üzerine Değerlendirmeler

RAPOR II:

Süpermarket Zincirleriyle Görüşmeler

Hazırlayanlar:

Sedat Belbağ
Mustafa Çimen
Dr. Şule Tarım
Yrd. Doç. Dr. Ayşegül Taş
Prof. Dr. Ş. Armağan Tarım

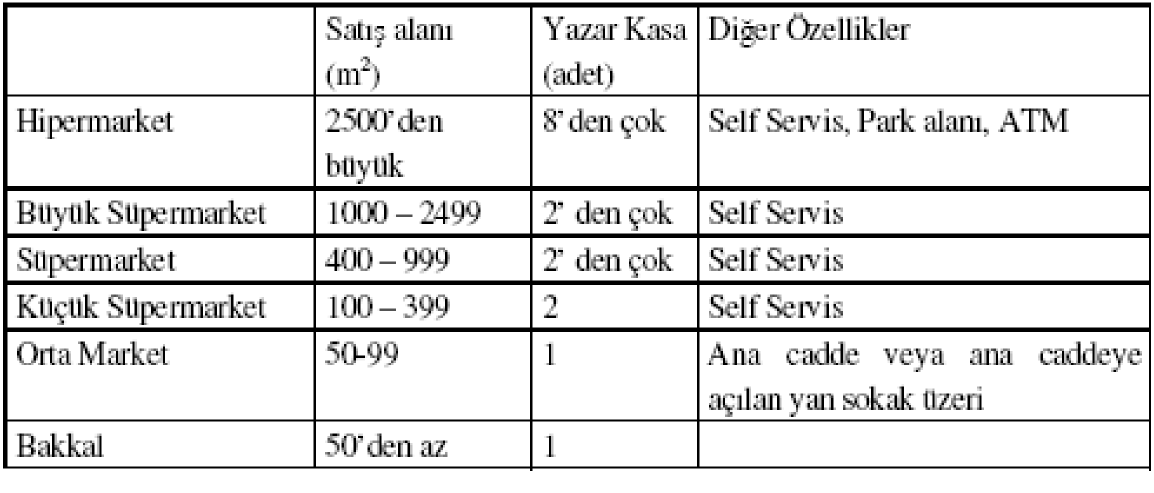
29.12.2008

Türkiye'de Süpermarket Tedarik Zinciri Envanter Yönetimi için Kullanılan Sistemler Üzerine Değerlendirmeler II

1. Giriş

Perakendecilik, tüketicinin ihtiyaç duyduğu malların çoğunlukla sabit satış noktalarından, küçük parti büyüklüklerinde ve nihai tüketim için pazarlanması faaliyetlerinin bütünüdür. Bu tanım çerçevesinde perakendeciler gıda maddeleri, giyim eşyası, mobilya, ev eşyası, madeni eşya, cam, ilaç ve ıtriyat, kereste ve inşaat malzemesi, kitap ve kırtasiye gibi çok farklı alanlarda faaliyet gösterirler. Perakendeci kuruluşlar kapalı alan büyüklüğüne göre bakkal, market, büyük market, küçük süpermarket, büyük süpermarket ve hipermarket şeklinde sıralanabilir. Tablo 1 perakendeci kuruluşun büyüklüğü ile sunduğu hizmetlerin niceliği arasında doğrusal bir ilişki olduğunu göstermektedir [12].

Tablo 1 – Süpermarketlerin Sınıflanması



	Satış alanı (m^2)	Yazar Kasa (adet)	Diğer Özellikler
Hipermarket	2500'den büyük	8' den çok	Self Servis, Park alanı, ATM
Büyük Süpermarket	1000 – 2499	2' den çok	Self Servis
Süpermarket	400 – 999	2' den çok	Self Servis
Küçük Süpermarket	100 – 399	2	Self Servis
Orta Market	50-99	1	Ana cadde veya ana caddeye açılan yan sokak üzeri
Bakkal	50'den az	1	

"Planet Retail" [30] tarafından sağlanan verilere göre, Türk perakende sektörünün cirosu 2006 yılında 137 milyar dolar olarak gerçekleşmiş ve 2010'a kadar sektörün 199 milyar dolara ulaşması beklenmektedir. Türkiye'de perakende sektörü ekonomiye yaklaşık 6.7 milyar dolar tutarında bir katma değer yaratmakta ve yine yaklaşık olarak 2.5 milyon kişiyi istihdam etmektedir ki buna göre perakende sektörünün tüm ekonomi üzerindeki etkisi, toplam Türkiye üretiminin %3.5'i ve istihdamın ise %12'si olacaktır. Bu rakamlar perakende sektörünün Türk ekonomisi üzerindeki ağırlığını açıkça göz önüne sermekte ve bu sektörde envanter planlamasıyla sağlanacak tasarrufun firma ve ulusal ekonomi bazında ne derece önemli olduğunu göstermektedir.

Andersen Consulting tarafından Coca-Cola için yapılan bir araştırma [10] sıradan bir günde bir süpermarkette sunulan ürünlerin %8.2'si için stoksuz kalındığını belirlemiştir. Reklamı yapılan ürünler için bu değer %15.0'e yükselmektedir. Sözü edilen stoksuz kalma durumu tüm satışların %6.5'na karşılık gelmektedir. Alternatif ürünler sunarak tüketicinin talebinin karşılanması halinde dahi perakendeciler toplam satışların %3.1'i kadar bir potansiyel hasılatı yeteri kadar stok tutmamak sebebiyle kaybetmektedirler [8]. ABD için bulunan stoksuz kalma yüzdelerinin Türkiye perakende sektörü için de geçerli olduğunun varsayılması halinde sadece stoksuz kalma sebebiyle uğranılan yıllık satış kaybının 6 milyar doları bulacağı anlaşılır ki bu da konunun önemini açıkça ortaya koymaktadır. Stoksuz kalmanın veya gereğinden fazla stok bulundurmanın maliyetli olması hangi ürünün siparişinin ne zaman ve ne miktarda verilmesi gerektiği sorusunu perakende sektörünün temel sorunlarından birisi haline getirmektedir.

Perakendeci seviyesinde envanter planlaması sorununu perakende tedarik zincirinin planlamasından bağımsız olarak düşünmek mümkün değildir. Bugünün küreselleşen piyasalarında yaşanan yoğun rekabet, yeni ürünler için giderek düşen ürün yaşam süreleri ve tüketicilerin yükselen beklentileri firmaların tedarik zincirlerini tekrar düzenlemelerine yol açmıştır ve perakende sektörü bunun bir istisnası değildir.

Günümüzde gelişen teknoloji ile birlikte kurumsal perakende sektöründe envanter takibini yapabilmek için bilgisayar yazılımları kullanılmaktadır. Kurumsal kaynak planlama yazılımlarının Türkiye süpermarket zincirlerinde kullanımı tarihi yakın bir geçmişe dayanmaktadır. Bu sebeple Türkiye'deki süpermarket zincirlerinin kullandıkları yazılımların ne olduğu, bu yazılımların envanter planlaması konusunda firmalara ne ölçüde destek verdiğine dair bir envanter henüz çıkartılmamıştır.

Bu raporda Türkiye'de faaliyet gösteren perakendeci firmaların tedarik zinciri envanter planlamasına ilişkin kullanmakta oldukları karar destek yazılımları belirlenmeye çalışılmıştır. Bunun için araştırma, mevcut envanter yaklaşımlarını ortaya koyabilmek amacıyla sektörü temsilen süpermarket düzeyindeki firmalar ile sınırlandırılmıştır. Perakende sektöründe faaliyet gösteren firmaların çeşitliliği ve sayılarının çokluğu, bu tür bir sınırlamayı gerekli kılmaktadır. Araştırmada konu edilen firmalar "Soysal Türkiye Perakende Kataloğu 2005" süpermarketler bölümünde listelenen süpermarket zincirleri arasından seçilmiştir.

2. Türkiye'de Perakende Sektörü

Türkiye'de perakendecilik sektörü 1950'lerde Gima ve Migros'un kuruluşu ile gelişmeye başlamıştır. Perakende sektörünün asıl atılım yaptığı yıllar ise 90'lardır. Öncesinde bakkalların hâkimiyetinde olan perakendecilik sektörü, 1990'lı yıllardan sonra daha büyük alana ve ürün çeşitliliğine sahip olan marketlere doğru kaymaya başlamıştır. Tablo 2'de Türkiye perakende sektöründe faaliyet gösteren şirketlerin yıllara göre niceliği gözlenmektedir. Bu tabloya göre hipermarket ve süpermarketlerin şube sayıları yıllara göre hızla artış göstermektedir. Marketlerdeki artış süpermarketlere göre düşükken, bakkalların sayısı son yıllarda azalma yönünde bir eğilime sahiptir [20].

Tablo 2 – Perakendeci Şirketlerin Yıllara Göre Mağaza Sayıları

Sınıflandırma	Satış Alanı(m2)	1994	1998	1999	2003
Hipermarket	2500 ve üstü	14	100	105	160
Süpermarket(Büyük)	1000 - 2499	58	178	227	350
Süpermarket(Orta)	400 - 999	187	487	596	920
Süpermarket(Küçük)	100 - 399	773	1.370	1.493	2.070
Market	50 - 99	9.176	12.196	13.247	16.000
Bakkal	49 ve altı	66.925	155.420	148.925	131.000
Toplam		77.133	169.751	164.593	150.500

Perakendecilik sektöründeki büyüme trendine bakılacak olursa, 2001 yılından 2005'e kadar, gıda perakendecileri arasında süpermarketler ve indirim mağazalarının pazar payları hızla büyümüştür. Son yıllarda, özellikle indirim mağazaları ve süpermarketlerin pazar paylarındaki yükseliş eğilimi göze çarpmaktadır. Bu eğilimin nedeni olarak, bu market türlerinin tüketicilerin evlerine yakın olması ve alışveriş için daha az zaman gerekmesi gibi kolaylıklarından kaynaklandığı söylenebilir. Bakkallar ise, pazar paylarındaki düşüş eğilimine rağmen hala en yüksek paya sahiptir[27].

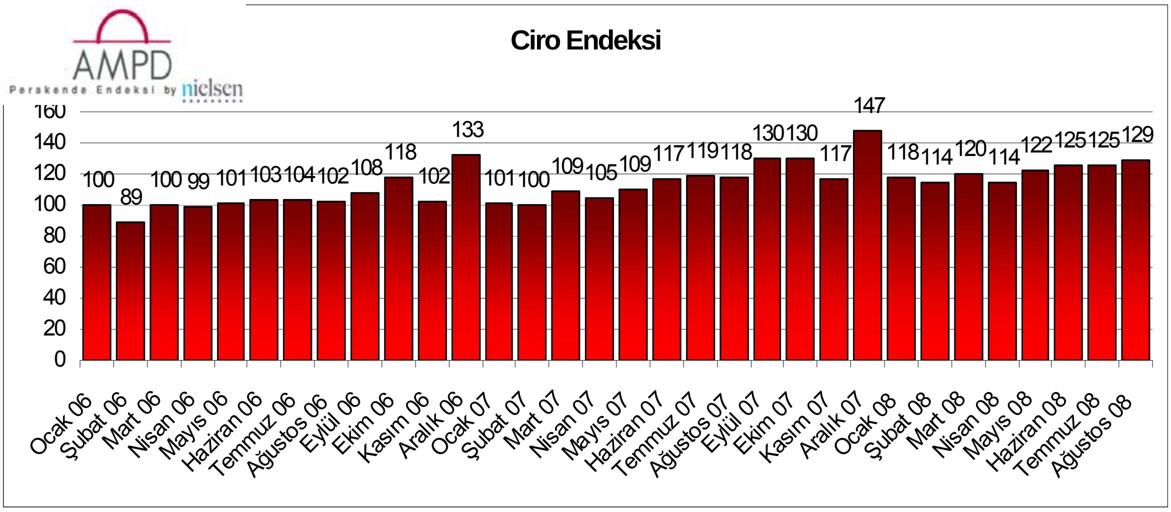
Tablo 3 - Türkiye'de Gıda Perakendeciliğinde Pazar Payları (%)

	2001	2002	2003	2004	2005
Hipermarketler	2,8	2,9	2,9	3,1	3,2
Süpermarketler	19,3	19,8	20,8	23,1	24,2
İndirim Mağazaları	3,2	3,7	4	5	5,6
Marketler	1,1	1,1	1,1	1	1
Bakkallar	47,4	46,4	45,3	43,1	42
Şarküteriler	16,8	16,8	16,7	16	15,6
Diğer	9,4	9,3	9,2	8,7	8,4
Toplam	100	100	100	100	100

Tablo 3 perakende sektöründe şirketlerin kategorik olarak Türkiye'deki pazar paylarını göstermektedir. Tablodan görüleceği üzere süpermarketlerin pazar payları yıllar geçtikçe büyürken, market ve bakkalların pazar payları küçülmektedir. Bu yöndeki değişimin kaynağı süpermarketlerde birçok ürünün bir arada bulunması ve bunların sunduğu kaliteli hizmet anlayışı gösterilebilir.

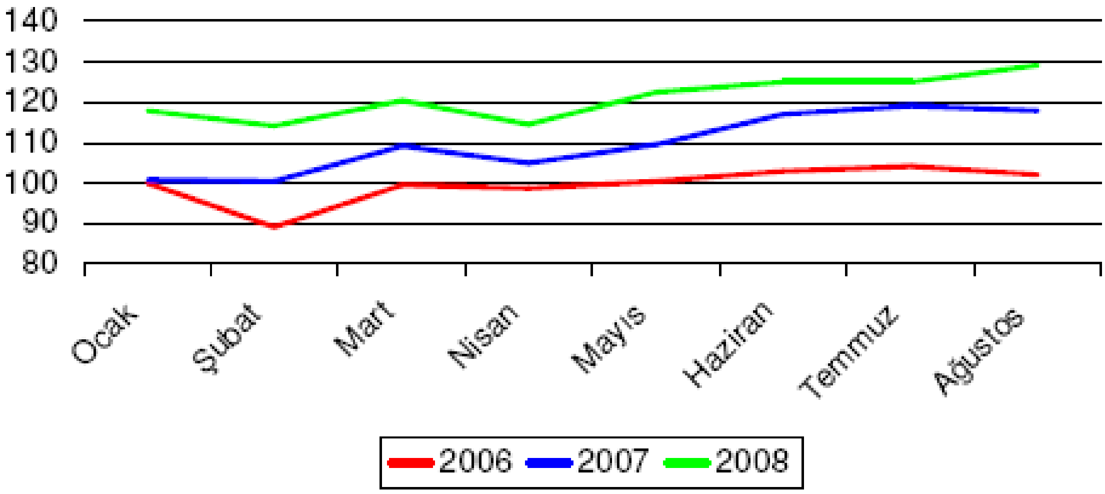
Süpermarketlerin elde ettikleri yıllık cirolarının toplamına bakacak olursak, 2006 yılının ocak ayında 100,000 YTL, 2007 yılında 133,000 YTL ve 2008 yılının Ağustos ayında ise 129,000 YTL olduğu görülmektedir (Grafik 2, Grafik 3). Bu rakamlar perakende sektörünün önemini bir kez daha ortaya koymaktadır.

Grafik 2 – Süpermarketlerin Aylara Göre Toplam Ciroları



Kaynak: Alışveriş Merkezi ve Perakendeciler Derneği

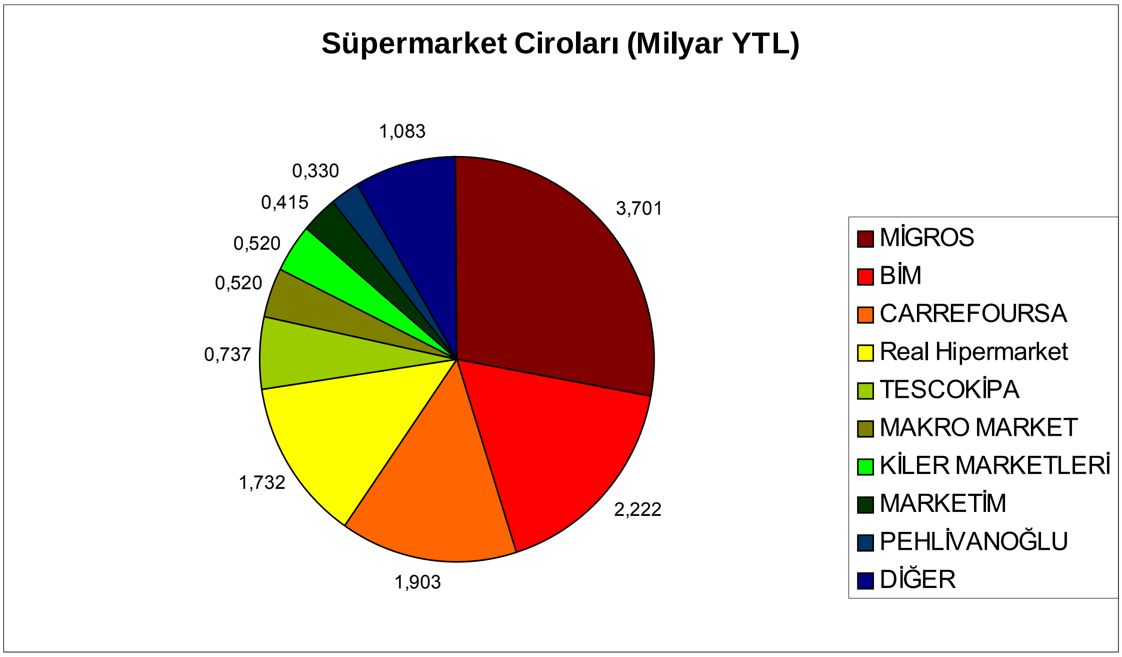
Grafik 3 – Süpermarketlerin Yıllara Göre Toplam Ciroları



Kaynak: Alışveriş Merkezi ve Perakendeciler Derneği

Türkiye sıralamasında 2006 yılının şirket bazındaki ciro rakamlarına baktığımızda Migros, BİM, Carrefour, Real Hipermarket ve Tesco-Kipa şirketlerinin pazarda elde edilen toplam cironun büyük bir kısmı oluşturduğu görülmektedir (Grafik 4).

Grafik 4 – Perakende Sektöründe Süpermarketlerin Ciroları



Perakende sektörü günden güne artan ivme ile büyümesine devam etmektedir. Bu sektörün içindeki süpermarketler için, ciroları ve mağaza sayıları dikkate alınarak sektörün yapı taşları haline geldikleri söylenebilir. Bu yüzden süpermarketlerin tedarik zincirlerini nasıl yönettikleri ve yönetirken hangi yazılımlardan faydalandıklarının incelenmesi önemli hale gelmiştir. Araştırmaya konu olan mağazalar, gerek ciro gerekse mağaza sayısı bakımından perakende sektöründe önemli yere sahip olan mağaza zincirlerinden seçilmiştir. Araştırmadaki şirketlerin son durumları, bu yönde hazırlanmış olan Soysal Marketler Kataloğu 2008'de özet şeklinde görülmektedir (Tablo 4) [34].

Tablo 4 – Türkiye'de faaliyet gösteren belli başlı süpermarket zincirleri

Süpermarket Listesi	*Toplam Çalışan Sayısı*	*Mağaza Sayısı*	*Toplam Satış Alanı*	*Toplam Kasa Sayısı*
ADESE	2.216	113	80.160	353
ALTUNBİLEKLER	750	30	14.000	90
BAHAR MARKETLER ZİNCİRİ	100	6	3.800	18
BEĞENDİK	1.300	16	100.000	135
BİLDİRİCİ GIDA	320	11	8.000	48
BİM	10.394	1.920	*	*
CARREFOURSA	*	129	*	*
ÇAĞRI HİPERMARKET	*	14	20.000	195
ERİMPAŞ	170	4	9.000	26
ESENLİK	286	11	5.100	49
GOP PAZAR MARKETLERİ	185	14	5.405	34
GROSERİ	570	12	9.825	78
GÜN SÜPERMARKET	470	13	12.000	54
MAKROMARKET	3.252	88	86.365	510
MARKETİM	2.221	93	68.100	238
MOPAŞ HİPERMARKET	1.100	4	2.500	15
ÖZHAN MARKETLER ZİNCİRİ	440	16	10.253	54
PEHLİVANOĞLU	*	94	38.000	297
PEKDEMİR	850	21	14.000	102
RAMMAR MARKETLER ZİNCİRİ	750	25	32.000	96
REAL HİPERMARKET	2.215	11	97.505	454
SİNCAP MARKETLER ZİNCİRİ	321	14	10.940	45
ŞAYPA	850	19	20.000	180
TESCOKİPA	5.300	80	198.000	*
ÜÇLER SÜPERMARKET	150	4	2.200	17
YİMPAŞ	2.019	37	235.000	390
YUNUS MARKETLER ZİNCİRİ	919	27	21.240	145

*Hakkında bilgi bulunmamaktadır.

Migros Türk A.Ş.

Migros Türk A.Ş., 1954 yılında İsviçre Migros Kooperatifler Birliği ve İstanbul Belediyesi'nin girişimleri ile kurulmuştur. 1975 yılında çoğunluk hisseleri Koç firması tarafından satın alınmıştır. Bu gelişmeden sonra hızla büyümeye giren firma, 1988 yılında İstanbul dışındaki ilk şubesini İzmir'de açmıştır. Migros, 1996 yılında Azerbaycan'daki altyapı çalışmalarını sonlandırarak, ilk yurt dışı mağazası Ramstore'u Bakü'de açmıştır. 2000 yılında Kangurum isimli sanal alışveriş merkezini kurarak tüketiciye internet üzerinden de ulaşmaya başlamıştır. Migros, 2005 yılında önemli perakende zincirlerinden biri olan Tansaş'ı da bünyesine katarak büyümesini sürdürmüştür. 2007 yılında Koç grubu tarafından satışa çıkarılmıştır.14 Şubat 2008'de Koç Holding hisselerini Moonlight Capital S.A adlı İngiliz şirketine satmıştır.

Günümüzde yurtiçinde 900'den fazla mağazası bulunan Migros, Türkiye'nin en önemli süpermarket zincirlerinden biridir. Migros 803.000 m^2 kapalı alanda, 4.000 personelle ve 75.000 üzerinde ürünle hizmet vermektedir [24].

CarrefourSA

15 Haziran 1963'te Fransa'da kurulan Carrefour, Türkiye'deki ilk mağazasını 1993 yılında İstanbul-İçerenköy'de açmıştır. 1996 yılında Sabancı Holding ile ortaklık kuran şirketin, Türkiye'deki ismi CarrefourSA olarak değişmiştir. İlk alışveriş merkezini bir yıl sonra Adana'da hizmete sunmuştur. 2005 yılında Gima ve Endi firmalarını bünyesine katarak büyümesini sürdürmüştür.

Şirket hipermarket, süpermarket ve ucuzluk marketleri formatlarında, Türkiye'nin 31 ilinde ve Anadolu'nun 16 ilçesinde 470 mağaza ile faaliyet göstermektedir. Türkiye'nin önemli süpermarket zincirlerinden biri olan firma 7500 çalışana sahiptir [9].

BİM

Açılımı Birleşik Mağazalar A.Ş. olan BİM, ilk şubesini 1995 yılında İstanbul’da hizmete açmıştır. Mağaza büyüklükleri 200–600 m^2 arasında değişmektedir. Ürünlerin mümkün olan en düşük fiyatta ve düşük maliyette halka sunulduğu “hard discount” modelini benimsemiştir. Bu yönden çalışma şekli, Avrupa’da faaliyet gösteren ALDI şirketine çok benzemektedir. Ürünler diğer süpermarketlerin aksine raflar yerine kutular içinde sunularak maliyetler düşürülmektedir. Tercihen öz markalardan yaklaşık 600 ürün ile sınırlı ürün çeşidine sahiptir. BİM, 11.736 çalışanı ve 2.175 mağaza sayısı ile Türkiye süpermarket zincirleri içinde pazarda önemli bir paya sahiptir [7].

YİMPAŞ

Yimpaş Holding, 1982 yılında Yozgat'ta bir grup girişimci tarafından kurulmuştur. Yimpaş Mağazalar Zinciri Türkiye'de 31 ilde 43 mağazanın yanında, Avrupa ve Türk Cumhuriyetleri'ndeki mağazaları ile geniş bir kitleye hizmet sunmaktadır. Yimpaş Mağazalar Zincirinin veri tabanına göre yaklaşık 50.000 adet ürün çeşidi bulunmakta ve bu portföyün %30'u gıda ürünlerini içermektedir. Şirket 2001 yılında Yimpaş PROMA adı altında süpermarketler açmaya başlamıştır [38].

TANSAŞ

Şirket Tansa adıyla, halka ucuz et ve kömür sağlamak hedefiyle 1973 yılında İzmir’de kurulmuştur. 1976 yılında Tanzim Satışlar Müdürlüğü çatısı altında, Konak'ta ilk mağazası açılmıştır. 15 Aralık 1986 tarihinde Tansaş, İzmir Büyükşehir Belediyesi İç ve Dış Ticaret A.Ş. adını almıştır. 1996 yılında, İzmir Büyükşehir Belediyesi'ne ait Tansaş hisselerinin %32.98'i halka açılmıştır. 1999 yılında şirket hisselerinin büyük bölümü Doğuş Grubu tarafından satın alınmıştır. 2002 yılı içinde, Macrocenter'ı kendi bünyesine katmıştır. %78.1 oranındaki hissesi Koç Holding ve Migros tarafından satın alınan Tansaş, 10 Kasım 2005’te Migros Türk A.Ş. bünyesine katılmıştır.

Tansaş, 2007 yılı içinde 18 yeni mağaza açarak, toplam mağaza sayısını 253'ye çıkartmış, toplam faaliyet alanını 330,870 m^2'ye ulaştırmıştır. Ege, Marmara, Akdeniz, İç Anadolu ve Batı Karadeniz olmak üzere toplam 5 coğrafi bölgede hizmet vermektedir [36].

Kiler Marketleri

Kiler Group'un temelleri Nahit, Vahit ve Ümit Kiler tarafından 1983 yılında atılmıştır. Perakendecilik sektörü ile iş hayatına başlayan şirket başka sektörlerde de atılımlar gerçekleştirmiştir. Hizmet sunduğu mağazalarını üç konsepte ayırarak 0–600 metrekare ölçekli mağazalarında 2500 çeşit, 600–1500 metrekare ölçekli mağazalarında 7500 çeşit, 1500 metrekare ve üzeri mağazalarında 15000–20000 çeşit ürünü müşterilerine ulaştırmaktadır. Türkiye genelinde toplamda 130, yurtdışında 2 adet olan mağaza sayısını 2006 yılı sonuna kadar 150'ye çıkarmayı hedeflemektedir [21].

Real Hipermarket

Real, 1992 yılında Divi, Basar, Continent, Esbella ve Real-Kauf gibi küçük markaların Metro Group tarafından satın alınarak tek bir marka haline dönüştürülmesi sonucu kurulmuştur. Real merkezi Düsseldorf'ta bulunan ve sadece Almanya'da 350 adet satış mağazası bulunan, Metro Group'un yan kuruluşlarından birisidir. 2007 yılı rakamlarıyla Real firmasının Türkiye'de 11 şubesi bulunmaktadır. Real gıda maddelerinin yanı sıra ev eşyaları, elektronik aletler, kitaplar, medya ürünleri, kırtasiye, spor malzemeleri gibi ürünlerle Türkiye dâhil olmak üzere dünyanın birçok ülkesinde şubeler açmıştır. Ürünlerin üzerine yapıştırılan gizli etiketler sayesinde kasalardan geçmeyen ürünlerin alarm vermesini sağlayan sistemin, ilk denendiği kuruluşlardan biridir [32].

Rammar Grosmarket

Rammar 1996 yılında küçük çaplı yatırımların birleşimi ile İstanbul'da faaliyete geçmiştir. 25 şubesi ile İstanbul'un her iki yakası ve Trakya'daki müşterilerine ulaşmaya devam etmektedir. 2007 yılını 25 şube ve 750 personeliyle bitiren Rammar Marketler Zinciri, 2008 yılında şube sayısını 35'e, çalışan sayısını ise 1.100'e yükseltmeyi hedeflemektedir [31].

Adese

Adese şirketi, 1994 yılında İttifak Holding tarafından perakende sektörüne girmek amacıyla kurulmuştur. Şirket 2007 yılı sonunda indirim marketleri zinciri olan Mercek Gıda ile birleşmiştir. Yazılımlarını ve teknik alt yapısını kendi bünyesinde geliştirerek tamamen öz kaynaklarını kullanmaktadır. Adese, Mayıs 2008 itibariyle, yedi şehirde 119 mağazaya ve 2850 çalışana sahiptir. 2007 sonu itibariyle cirosu 450 milyon YTL olarak hesaplanmıştır [1].

Makromarket

Makromarket Mağazalar Zinciri 1991 yılında Ankara'da kurulmuştur. Bugün itibariyle 9 ilde bulunan 106 mağazası ve 3950 personeli ile hizmet sunmaktadır. Hafta içi ortalama 110.000–125.000, hafta sonu 135.000–150.000 müşteriye hitap eden Makromarket; 55 bini aşan ürün çeşidi ile tüketicisine hizmet vermektedir [22].

Mopaş Hipermarket

İstanbul Bayrampaşa'da üç tane market ve Rami'de toptan mağaza işletmesiyle hizmete başlayan Mopaş Marketler Birliği, 1996'da üç küçük marketi devrederek Moda'da ilk MOPAŞ mağazasını açtı. Bugün İstanbul Anadolu yakasında 31, Bursa bölgesinde 10 şube olmak üzere toplam 41 mağaza ve Pendik/ Kurtköy'de 4500m2'lik alan üzerine kurulu Genel Müdürlük& Antrepo binaları ile hizmet vermektedir [25].

Tescokipa

Kipa (Kitle Pazarlama Ticaret ve Gıda Sanayi A.Ş.), 17 Ağustos 1992 tarihinde 100 ortaklı bir girişim olarak İzmir'de kurulmuştur. Kipa ilk hipermarketini 18 Ekim 1994 tarihinde İzmir'in Bornova ilçesinde hizmete açmıştır. 1 Kasım 2003 tarihinde dünyanın önde gelen perakende devlerinden Tesco ile birleşerek Tesco Kipa ismini almıştır. Kipa 2006 yılında ilk küçük formatlı ekspres mağazası olan Eşrefpaşa Kipa Ekspres'i İzmir'de açmıştır. Tesco Kipa, İzmir'de açtığı 11 ekspres mağazadan sonra, Antalya'da Şarampol Kipa Ekspres ve Ali Çetinkaya Kipa Ekspres mağazaları aynı gün hizmete açmıştır [37].

Pehlivanoğlu

Pehlivanoğlu 14 Haziran 1980 tarihinde İzmir'in Üçkuyular semtinde ilk mağazasını açarak faaliyete geçmiştir. Şirket 1999 yılının Ağustos ayında unvanını, Pehlivanoğlu Marketçilik Gıda Pazarlama San. ve Tic. A.Ş. olarak değiştirmiştir. Şirket, ana iş kolu olan gıda sektörünün yanı sıra; İnşaat, Turizm, Bilgi İşlem, Özel Güvenlik, Ambalaj, Makine, Depoculuk, Taşıma, Sigorta Acenteliği gibi farklı sektörlerde de faaliyet göstermektedir. Pehlivanoğlu Şirketler Grubu, bugün itibariyle Ege bölgesi ve çevresinde 79 mağazaya, 1223 personele sahip bir şirket haline gelmiştir [28].

Akyurt Alışveriş Merkezleri

Akyurt, 1983 yılında Ankara-Keçiören'de Akyurt Besi Çiftliği'ni kurarak perakende alanında ilk yatırımını yapmıştır. 1987 yılında Demetevler'de 2'nci mağazasını hizmete açmıştır. 1996 yılında branş mağazacılıktan, zincir süpermarket mağazacılığa geçiş yapmıştır. 2003–2004 yıllarında Etimesgut ve Demetevler Mağazalarını faaliyete geçirmiştir. Akyurt şirketi 2008'de Güneşevler mağazasını açarak Ankara içinde 23. mağazasına ulaşmıştır [2].

Yunus Marketler Zinciri

Bir aile şirketi olarak ticari geçmişi 1989 yılına dayanan Yunus Market İşletmeleri'nin "Yunus Marketler Zinciri" markasıyla perakende sektörüne girişi 1995 yılındadır. Faaliyetlerine Ankara'da başlayan Yunus Marketler Zinciri, 28 şubeye ve 30.000 m^2 satış alanına sahiptir. Şirketin Ankara dışındaki ilk mağazası Nisan 2008'de Düzce'de açılmıştır. Yunus Market İşletmeleri zaman içinde bünyesine perakende sektörünün dışında; inşaat, metal ve gıda sektörlerinde faaliyet gösteren 3 grup şirketi daha katmıştır [39].

Beğendik Mağaza İşletmeleri

Beğendik Mağaza İşletmeleri Ticaret ve Sanayi A.Ş. perakende sektöründeki faaliyetlerine 1986 yılında Kayseri'de başlamıştır. Ankara'daki ilk mağazasını 1993 yılında Kocatepe'de açan firma, bir yıl sonra Akköprü'de ikici şubesini faaliyete sokmuştur. 1995'te İstanbul Carousel ve 1997'de İzmit'te olmak üzere iki şube daha açarak büyümeyi sürdürmüştür. Firma, gıda maddeleri ile birlikte tekstil, ev eşyaları, restoran gibi hizmet işletmelerini de içeren "Bölümlü Mağaza" tarzını benimsemektedir [5].

IYAŞ

İyaş (Isparta Gıda Sanayi Ve Ticaret A.Ş.) 1997 yılında 10 ortaklı bir şirket olarak Isparta'da kurulmuştur. 14 Mart 1998 tarihinde Isparta'da ilk alışveriş merkezini hizmete sokmuştur. Firma gıda sektöründe üretim ve pazarlama faaliyetlerini gerçekleştirmektedir. Antalya İyaş'ı ve Dinar Alışveriş Merkezi'ni açarak büyümesini sürdürmüştür [19].

Sincap Marketler Zinciri

Sincap Marketler Zinciri, Temmuz 2002'den bu yana Konya'da faaliyet göstermektedir. Şirket 5 yıl içerisinde, Konya'da daha önce marketçilik yapan Karipek, Özhatunsaraylılar, Ananas, Özkent ve Osmanlı şirketlerinin mağazalarını satın alarak büyüme sürecini sürdürmüştür. 2007 yılı sonunda şirket 14 mağaza, 321 personel ve yıllık 50 milyon YTL'lik ciroya ulaşmıştır [33].

ŞAYPA

Şaypa Marketler Zinciri ilk şubesini 1995 yılında Bursa'nın Yeşilyayla mahallesinde faaliyete geçirmiştir. 10 yıllık süre içerisinde Bursa dâhilinde 20 mağaza sayısına ulaşmıştır. Şaypa toplamda 20 bin metrekarelik bir alana ve 850 çalışana sahiptir. Çevre illerde yeni şubeler açarak büyümeyi hedeflemektedir [35].

Bahar Marketler

Bahar Gıda A.Ş. 1985 yılında Merzifon'da kurulmuştur. Marketçilik alanında Merzifon, Suluova, Havza ve Ünye ilçelerinde olmak üzere 6 şube ile faaliyet göstermektedir. Marketçiliğin yanında un üretimi, hayvancılık, inşaat ve petrol ofisi işletmeciliği de yapmaktadır [4].

Özhan Marketler Zinciri

Özhan Gıda 1980'li yıllarda toptancılıkla piyasaya giriş yapmıştır. 1992 yılında Mudanya yolu mağazasını açarak perakende sektörüne girmiştir. 2008 yılı itibariyle Bursa içinde 19 mağazası ile yerel bazda hizmet vermektedir [26].

Altunbilekler

Altunbilekler Marketler Zinciri 1983 yılında Ankara'nın Keçiören ilçesinde semt marketçiliği ile faaliyetlerine başlamıştır. Günümüzde şirket, Ankara içine yayılmış 32 adet süpermarkete sahip bir zincir haline gelmiştir. Şirket toplamda 10.100m^2'lik satış alanına ve 800'e yakın personele sahiptir [3].

Groseri

Groseri, 1989 yılının Haziran ayında Ersin Özdemir, Mehmet Ali Önür ve Levent Uğurses tarafından Adana'da kurulmuştur. Şirket 2007 yılında 12 şube sayısına ulaşmıştır. Yerel bir zincir olan Groseri, bölgesel olarak büyüme hedeflemektedir [17].

Pekdemir

Pekdemir 1979 yılında küçük bir market olarak Denizli'de açılmıştır. 15 Eylül 2000 yılında Denizli'nin ilk yerel hipermarketini açan firma, 2005 yılında Umpaş'ın üç mağazasını da bünyesine katarak büyümesini sürdürmüştür. Şu an da 18 mağaza, 22.000m^2'lik satış alanına ve 1000'e yakın çalışana sahip yerel marketler zinciri haline gelmiştir [29].

Çağrı Hipermarket

Çağrı Gıda, 1980'lerde İstanbul Üsküdar'da hizmete başlamıştır. 1990'larda büyüyen firma ismini Çağrı Hipermarket olarak değiştirmiştir. İstanbul iline odaklanmış şirket, il genelinde 16 mağazasında 900 kişilik personeli ile hizmet sunmaktadır [13].

Gün Süpermarket

1995 yılında Mukaddes Gün tarafından Denizli ilinde kurulmuştur. 1999 yılı ortalarında Beltaş şirketinin mağazalarını alarak mağaza sayısını yediye çıkarmıştır. 2006 yılında Kuyucak, Sarayköy ve Nazilli ilçelerinde şubeler açılmıştır. Firma bölgesel bazda mağaza zinciri olmayı hedeflemektedir [18].

Erimpaş

Erimpaş şirketi 1997 yılında Ahmet Tanoğlu tarafından Erzincan şehrinde kurulmuştur. Erzincen içerisinde bulunan 3 şubesi ve yaklaşık 200 çalışanı ile hizmet vermektedir.

Kim Marketleri

Kim Marketleri 1997 Kiler Marketleri'nden aldığı franchising ile İstanbul'da hizmete başlamıştır. Açılımı "Kazançlı İstanbul Mağazaları" olan Kim, daha sonra Kiler Marketleri'nden ayrılarak kendi markasını oluşturmuştur. Günümüzde Kim Marketlerin İstanbul'da 39, İzmit'te 3 şubesi bulunmaktadır. Firma yaklaşık 1800 çalışanı ile hizmet sunmaktadır.

Bildirici Marketler Zinciri

Bildirici Gıda 1993 yılında Niyazi Bildirici tarafından Ankara'da kurulmuştur. Günümüzde firma 11 şube, 280'e yakın çalışan ve 8100 m^2 satış alanı ile Ankara içerisinde faaliyetlerini sürdürmektedir [6].

Meşhur peynirci

Meşhur Peynirci ticari hayatına 1944 yılında Ankara Samanpazarı'nda toptan gıda satışı ile başlamış, 1978 yılında ilk perakendeci markerini faaliyete geçirmiştir. Günümüzde Ankara içersinde 40'a varan şube sayısı ile yerel bazda hizmetlerini sürdürmektedir [23].

GOP Marketler Zinciri

GOP Market ticari hayatına 1995 yılında, Gaziosmanpaşa'da açılan mağaza ile başlamıştır. GOP Marketleri, şu anda 14 mağazası, toplam 7850 m^2 market satış alanı, 34 kasası ve 219 çalışanı ile tüketicilerine ulaşmaktadır [16].

Esenlik Süper Market Zinciri

Esenlik Süper Market Zinciri, Malatya Belediyesinin %95'ine sahip olduğu Esenlik İmar İnşaat ve Ticaret Ltd. Şti.'nin bir iştirakidir. Firma Malatya'nın çeşitli yerlerinde faaliyet gösteren 11 adet mağazaya sahiptir [14].

Macit Marketler Zinciri

Macit Marketler Zinciri, 31 Mart 1987 yılında Orhan Macit tarafından Ankara şehrinde kurulmuştur. Ankara içerinde 16 şubeye ve yaklaşık 400 çalışana sahip olmaktadır. Firmanın Ankara dışında Çorum şehrinde 1 şubesi bulunmaktadır.

Üçler Süpermarket

Üçler Süpermarket iş hayatına 1974 yılında İstanbul'da başlamıştır. Yerel düzeyde hizmet sunan market, İstanbul'un Avrupa yakasında bulunan 6 şubesi ve yaklaşık 200 çalışanı ile faaliyet göstermektedir.

Çağdaş Marketler Zinciri

Çağdaş Marketler Zinciri, perakendecilik sektörüne 3 Aralık 1987 tarihinde Çağdaş Ucuzluk adıyla Ankara'da açılan 150 m^2'lik mağazasıyla adım attı. Çağdaş Marketler Zinciri, şu anda 22 mağazası ile Ankaralı tüketicilere hizmet vermektedir [11].

GİMSA Marketler Zinciri

Gimsa merkez şubesi Sincan’da olan, 11 mağazaya ve 1000 çalışana sahip Ankara’da kurulmuş bir marketler zinciridir [15].

3. Süpermarket zincirleriyle yapılan görüşmeler

Bu raporda Türkiye'de faaliyet gösteren belli başlı süpermarket zincirlerinin tedarik zinciri envanter planlamasına dönük olarak kullandıkları yazılımların bir envanterinin çıkarılması hedeflenmiştir. Yöntem olarak süpermarketlerin herbiri ile ayrı ayrı yapılandırılmış görüşme yapılması benimsenmiştir. Yapılan görüşmeler, araştırma kapsamındaki şirketlerin genel merkezlerinde çalışan satın alma veya bilgi işlemden sorumlu yetkililer ile gerçekleştirilmiştir. Bu yapılandırılmış görüşmelerde aşağıdaki sorular verilen sırada kullanılmıştır:

- Şirketinizde stok takibini yapacak bir bilgisayar programı kullanıyor musunuz?.

  Eğer evet ise,

  - Hangi programı kullanıyorsunuz?
  - Bu program talep tahmini yapabiliyor mu?
  - Program bir sonraki dönem için talep, fiyat, maliyet ve tedarik süresi tahmini yapıyor mu?

    Eğer evet ise,

    - Program talep tahmini hatasına ilişkin bilgi sunuyor mu?
    - Bu program ürün bazında ne zaman ne miktarda sipariş verilmesi gerektiğini size söylüyor mu?
  - Program ileriye dönük olarak tahmin yapabiliyor mu / sipariş zamanlarını söyleyebiliyor mu?
  - Kullandığınız program tutulması gereken yedek/güvence stok miktarını belirleyebiliyor mu?
  - Sipariş kararlarını verirken programın önerilerini dikkate alıyor musunuz?

Araştırma neticesinde birçok bulguya ulaşılmıştır. Bu sonuçların bazıları ERP yazılımları, bazıları ise süpermarketlerle bağlantılıdır. Bulgular ışığında ERP yazılımlarının süpermarketlerin envanter planlaması açısından ihtiyaçlarını karşılayamadıkları görülmüştür. Süpermarketlerin 22'si kullandığı yazılımlarla, nokta talep tahmini yapabildiklerini bildirmiştir. Bir şirket yazılımlarının talep tahmini hatasına ilişkin bilgi sunabildiğini belirtmiştir. Hiçbir şirket yazılımlarını kullanarak deterministik envanter planlaması yapabildiğini belirtmemiştir. Ayrıca bazı şirketlerin kullandığı yazılımlar, nokta talep tahminine ve tahmin hatasına ilişkin bilgi sunabilse de, programların bu özellikleri kullanıcılar tarafından etkin bir şekilde kullanılamamaktadır. Böylece yazılım sisteme entegre olsa bile istenilen düzeyde destek sağlayamamaktadır. Sonuçta kullanılan yazılımlar stok takibi için kullanırken, aynı yazılımlar planlama yapmak için kullanılmamaktadır. Şu ana kadar tamamlanmış olan şirket görüşmelerinden elde edilmiş olan araştırma bulguları şöyle özetlenebilir:

- Süpermarketlerin çoğunluğu hazır ERP paketleri kullanmaktadır. Bazı süpermarketler ise kendilerine özel yazılımlar hazırlatmış ve kullanmaktadır.
- Hazır ERP paketlerinin çoğu sadece Türkiye'de hizmet veren yazılım şirketleri tarafından oluşturulmuştur.
- ERP paketlerinde envanter yönetimi ile ilgili olan modüllerin büyük çoğunluğu nokta talep tahmini yapabilmektedir. Birkaç programın modülleri tahmin hatasına ilişkin bilgi sunabilmektedir.
- Süpermarketlerin hiçbirinde ERP yazılımı kullanılarak planlama yapılmamaktadır.
- Süpermarketlerde verilen sipariş kararlarında insan faktörü ön plana çıkmaktadır. Yazılımın sunduğu öneriler, yöneticilerin deneyimlerine dayanarak verdiği kararların gölgesinde kalmaktadır.
- Birçok süpermarket zinciri, siparişlerini şubelerinden gelen sipariş miktarlarına göre ayarlamaktadır. Yazılımların bu durumdan kaynaklanabilecek kırbaç etkisine karşı çözümü bulunmamaktadır.
- Güvenlik stoğu için yazılımların bir önerisi bulunmamaktadır. Kullanıcılar güvenlik stoğu miktarını önceki satışlara ve kendi deneyimlerine göre belirleyerek, bu miktarı programa girmektedirler.
- Süpermarketlerde kullanılan yazılımların promosyon, kampanya ve özel günlerdeki satışlar (bayram, yılbaşı vs.) gibi durumlarla başa çıkabilmesini sağlayacak çözümleri bulunmamaktadır.
- Süpermarketler ERP yazılımlarını stok hareketlerini takip etmek için kullanmaktadır.
- Süpermarketler yazılımların planlama yapma konusundaki önemi hakkında yeterli bilgiye sahip değildir.

Aşağıda firma bazında yapılan görüşmelerin sonuçları verilmiştir. Şu ana kadar Migros, Tansaş, Carrefour, Şok ve Tescokipa ile yapılan görüşmeler henüz sonuçlandıralamamıştır. Bu görüşmelerinde tamamlanmasından sonra elde eldilen bulgularla rapor nihai şeklini alacaktır.

Yimpaş: Şirket stok takibi için kendi yazılımları olan Proma isimli programı kullanmaktadır. Yimpaş yazılımın nokta tahmini yapabildiğini, fakat tahmin hatasına ilişkin bir seçenek sunmadığını belirtmiştir. Güvenlik stoğu yazılım tarafından belirlenmemektedir.

BİM: Stok takibi yazılımı olarak SAP R/3 kullanılmaktadır. Daha önceki araştırmadan programın talep tahmini ve planlama yapabildiği bilinmektedir. Ama şirketin bu özellikleri ne oranda kullandığı hakkında detaylı bilgi elde edilememiştir.

REAL Hipermarket: Real Hipermarket, MMS isimli kendi geliştirdikleri bir stok takip programı kullanılmaktadır. Şirketten alınan bilgilere göre program izleme ve nokta talep tahmini işlemlerini yapabilmektedir. Tahmin hatasına ve planlamaya ilişkin kullanıcılara destek sağlamamaktadır. Güvelik stoğu belirleme seçeneği bulunmamaktadır.

Rammar Grosmarket: Şirket stok takibi için Micro isimli ERP programını kullanmaktadır. Yazılımın nokta talep tahmini seçeneği bulunmasına karşın şirket tarafından bu seçenek kullanılmamaktadır. Siparişler firmanın kendi deneyimleri dikkate alınarak belirlenmektedir.

Marketim: Netsis firmasının standart muhasebe ağırlıklı programını kullandığı tespit edilmiştir. Aktif olarak kullandıkları ERP programı bulunmamaktadır.

ADESE: Adese stok takibi için kendi yazılımlarını kullanmaktadır. Şirket yazılımın tek dönemli planlama yaptığını belirtmiştir. Yazılım çok dönemli planlama ve tahmin hatası içinse kullanıcıya bir seçenek sunmamaktadır.

Makro Market: Makro market stok takibi için OBASE programını kullanmaktadır. Şirket bu programı kullanarak talep tahmini yaptığını belirtmiştir. Yazılım talepteki tahmin hatalarına ilişkin bilgi sunmamaktadır. Yazılım en iyi sipariş miktarını kullanıcı için hesaplayamamaktadır.

Mopaş Hipermarket: Stok takibi için Mikro yazılımını kullanmaktadır. Programın talep tahmini yapabilme özelliği bulunmasına rağmen şirket tarafından bu özellik kullanılmamaktadır. Yazılımın güvenlik stoğu belirleme özelliği bulunmamaktadır.

Pehlivanoğlu: Pehlivanoğlu şirketine özel hazırlanan Unix tabanlı bir stok takip programı kullanılmaktadır. Şirketin verdiği bilgilere göre program satış ortalamasına temel alarak talep tahmini yapabilmektedir. Tahmin hatalarına ve planlamaya dair bir seçenek bulunmamaktadır. Asgari stok kullanıcılar tarafından tanımlanmaktadır.

Akyurt Alışveriş Merkezleri: Şirket stok yazılımı olarak Oracle veritabanlı bir program kullanmaktadır. Akyurt firması yazılımın talep tahmini yaptığını bildirmiştir. Tahmin hatasına ve planlamaya ilişkin kullanıcılara destek vermemektedir. Geçmiş verileri kullanarak sipariş için minimum ve maksimum noktaları kullanıcılara bildirmektedir. Bu değerler şirket açısından, yöneticilerin verdiği kararlara göre daha az önemlidir.

Yunus Marketler Zinciri: Stok takibi için Oracle tabanlı Olympos programı kullanılmaktadır. Şirket tarafından yazılımın nokta tahmini yaptığı belirtilmiştir. Diğer yandan yazılım tahmin hatası ve uzun dönem planlama konusunda kullanıcıya destek sağlamamaktadır.

Beğendik: Stok yazılımı olarak Milenium isimli program kullanılmaktadır. Şirket yetkilisi tarafından yazılımın talep tahminlerini tek dönemli ve nokta tahmini şeklinde gerçekleştirdiği belirtilmiştir. Tahmin hatası ve planlama yapma özelliği bulunmamaktadır. Güvenlik stoğu belirleme özelliği bulunmamaktadır.

İyaş: Firma stok takibi için Netsis şirketinin ERP yazılımını kullanmaktadır. Şirket yazılımı kullanarak talep tahmini yaptığını bildirmiştir. Yazılımın, ne zaman ne kadar sipariş verilmesi gerektiği konusunda öneri sunmadığı tespit edilmiştir. Bu kararlar firma tarafından verilmektedir.

Sincap Marketler Zinciri: Firma stok takibi için Mikro yazılımını kullanmaktadır. Fakat şirket, yazılımı talep tahmini açısından yeterli görmediği için firma bünyesindeki bilgi işlem sorumluları ek bir yazılım yazmışlardır. Mikro ve şirketin yazılımı beraber kullanılmaktadır. Bu iki programın kullanılmasına karşın, sipariş miktarlarını yine yöneticilerin belirlediği tespit edilmiştir.

Bahar Marketler: Şirket stok takibi için Logo Gold yazılımını kullanmaktadır. Şirket tarafından yazılımın nokta tahmini yaptığı, buna karşın tahmin hatasına ve planlamaya dair seçenekler içermediği söylenmiştir. Güvenlik stoğu açısından öneri sunmamaktadır. Sipariş kararlarında yöneticilerin belirlediği miktarlar dikkate alınmaktadır.

Şaypa: Şaypa stok takibi için Mikro yazılımını kullanmaktadır. Şirket programı kullanarak talep tahmini yapabilmektedir. Yazılım talep tahmini hatası ve planlama açısından destek sunmamaktadır. Güvenlik stoğu belirlenememektedir.

Özhan Market Zinciri: Stok takibi için Mikro yazılımının Retail 9000 isimli perakendeciler için geliştirilmiş modülünü kullanmaktadır. Şirket tarafından nokta talep tahmini yapılmaktadır. Fakat yazılımın tahmin hatası ve planlama için seçeneklerinin bulunmadığı belirtilmiştir. Güvenlik stoğu kullanıcılar tarafından belirlenmektedir.

Altunbilekler: Firma stok takibini Okyanus isimli yazılım ile gerçekleştirmektedir. Şirket yazılımın talep tahmini yapma özelliği bulunduğunu bildirmiştir. Yazılım sipariş konusunda kullanıcıya bir öneri sunmamaktadır.

Groseri: Stok takibi yazılımları Okyanus isimli programdır. Şirket programı kullanarak talep tahmini yapmadığını belirtmiştir. Sipariş kararları firma yöneticileri tarafından alınmaktadır.

Pekdemir: Şirket stok takibi için Mikro yazılımını kullanmaktadır. Firma yazılımın talep tahmini özelliğini kullandığını belirtmiştir. Talep tahmini hatalarına ve planlamaya ilişkin özellik bulunmamaktadır. Güvenlik stoğunun, genelde ortalama satışlara göre üç günlük stok şeklinde tutulduğu bildirilmiştir.

Çağrı Hipermarket: Çağrı Hipermarket stok takibi için Kalem yazılım şirketinin hazırladığı Santana isimli ürünü kullanmaktadır. Şirket yazılımın talep tahmini yaptığını, tahmin hatası ve planlama seçeneklerinin bulunmadığını belirtmiştir. Güvenlik stoğu kullanıcı tarafından belirlenmektedir.

Gün Süpermarket: Stok takibi yazılımı olarak Netsis Fushion@6 kullanmaktadır. Şirket tarafından talep tahmini yapılmamaktadır. Siparişler satışlarına göre mağaza bazında belirlenmektedir. Güvenlik stoğunu kullanıcı belirlemektedir.

Erimpaş: Stok takibi için Mikro yazılımı kullanılmaktadır. Şirket tarafından yazılım kullanılarak talep tahmini yapıldığı belirtilmiştir. Yazılım tahmin hatası ve planlama bakımından yetersiz kalmaktadır.

Kim Marketleri: Kim Marketleri stok takibi için Workùp E-iş isimli yazılımı kullanmaktadır. Şirket yetkililerince programın talep tahmini ve planlama yapma özelliği bulunmadığı bildirilmiştir.

Bildirici Gıda: Stok takibi için Okyanus isimli yazılım kullanılmaktadır. Firma yazılımı kullanarak talep tahmini yapamadığını belirtmiştir. Güvenlik stoğu kullanıcılar tarafından belirlenip programa girilmektedir.

GOP Pazar Marketleri: Şirket stok takibi için Mikro yazılımını kullanmaktadır. Program talep tahmini yapmasına rağmen bu özelliği firma tarafından kullanılmamaktadır. Sipariş miktarları, şubelerin istekleri doğrultusunda belirlenmektedir. Güvenlik stoğu belirleme seçeneği bulunmamaktadır.

Esenlik: Stok takibi için Netsis firmasının yazılımını kullanılmaktadır. Şirket tarafından talep tahmini yapılmamaktadır. Yazılım sipariş verme konusunda kullanıcıya destek vermemektedir.

Macit Marketler Zinciri: Şirket stok takibi için Oracle veritabanını kullanan Olympos isimli yazılımı kullanmaktadır. Şirket tarafından ne talep tahmini ne de planlama konusunda yazılım kullanılmamaktadır.

Üçler Süpermarket: Üçler Süpermarket stok takibi için OBASE 2.1 programını kullanmaktadır. Şirket tarafından yazılımın nokta talep tahmini yaptığı belirtilmiştir. Yazılımda tahmin hatası ve planlamaya yönelik bir seçenek bulunmamaktadır.

Meşhur Peynirci: Stok takibi için Omega isimli yazılım kullanılmaktadır. Firma yazılımın nokta talep tahmini yaptığını bildirmiştir. Yazılım tahmin hatasına ilişkin bilgi sunmamaktadır. Sipariş kararları kullanıcılar tarafından belirlenmektedir.

Çağdaş Marketler Zinciri: Firma stok takibi için Olympos isimli yazılımı kullanmaktadır. Şirket yazılımın nokta talep tahmini yapabildiğini belirtmiştir. Yazılım tahmin hatasına ilişkin bilgi sunmamaktadır. Güvenlik stoğu hedefi bildirmemektedir. Sipariş kararları tamamen firma yöneticileri tarafından verilmektedir.

4. Sonuç

Bu raporun amacı Türkiye perakende sektöründe faaliyet gösteren süpermarketlerin tedarik zinciri yönetimi için kullandıkları envanter yönetim yazılımlarını belirlemek ve tedarik zinciri yönetiminde bu yazılımlardan ne ölçüde faydalandıklarını tespit etmektir.

Araştırma kapsamında şu ana kadar 31 firma ile görüşme yapılmıştır. Elde ettiğimiz bulgular temelde sektör tarafından kullanılan yazılımların etkin şekilde belirsizlikle baş edebilecek fonksiyonelliğe sahip olmadıklarını ve çoğu zamanda bu yazılımların sunduğu temel fonksiyonların bile firmalar tarafından uygun organizasyonel süreçler ve uzmanlık bulunmadığı için kullanılmadığını göstermektedir.

Genel olarak baktığımızda, sadece 2 süpermarket yazılımlarının talep tahmini yapabildiğini ve tahmin hatasına ilişkin bilgi sunduğunu belirtmiştir. 9 süpermarket yazılımlarının sadece talep tahmini yapabildiğini bildirmiştir. 1 süpermarket ise ERP yazılımı kullanmadıklarını belirtmiştir.

Yukarıdaki tabloya baktığımızda hiçbir süpermarketin deterministik planlama yapmadığı görülmektedir. Buradan yola çıkarak, günümüz Türkiye perakende sektöründe faaliyet gösteren süpermarketlerde yönetici deneyimlerinin, sipariş kararlarında hala en önemli faktör olduğuna ulaşabiliriz. Ayrıca yaptığımız görüşmelerde ulaştığımız bir başka sonuç ise, yazılımların birçoğunun süpermarketlerde çok görülen mevsimsel satış, indirim, kampanya gibi durumlara özel çözümlerinin olmayışıdır. Bu da sipariş miktarlarının belirlenmesinde hala yöneticilerin neden kişisel tecrübelerine dayanarak karar verdiklerini bir noktaya kadar açıklamaktadır.

Süpermarketlerin belirsizlik altında ihtiyaçlarına cevap verecek yeni ve kapsamlı envanter planlama sistemlerine ihtiyaç olduğu açıktır. Öte yandan bu yazılımların var olması halinde dahi etkin şekilde kullanılabileceğine dair bir bulgu yoktur. Bu noktada yöneticilerin gerekli eğitimden geçirilmesi, bilinçlendirilmesi ve teknik ekibin yazılımların fonksiyonellikleri hakkında bilgilendirilmesi gerekliliği ortadadır.

Kaynakça

[1] Adese, 2008, kaynak: http://www.adese.com.tr/Kurumsal.aspx?page=Tarihce

[2] Akyurt Alışveriş Merkezleri, 2008, kaynak: http://www.akyurt.com.tr/turkce/hakkimizda.aspx?id=1

[3] Altunbilekler, 2008, kaynak: http://www.altunbilekler.com.tr/kurumsal.htm

[4] Bahar Marketler, 2008, kaynak: http://www.bahargida.com.tr/hakkimizda.html

[5] Beğendik A.Ş., 2008, kaynak: http://www.begendik.com.tr/begendik/begendik.html

[6] Bildirici Gıda, 2008, kaynak: http://www.bildirici.com.tr/hakkimizda.php

[7] BİM A.Ş., 2008, kaynak: http://www.bim.com.tr/

[8] C. B. Lee, 2003, "Demand Chain Optimization: Pitfalls and Key Principles," Evant White Paper Series.

[9] CarrefourSA, 2008, kaynak: http://www.carrefour.com.tr/hakkinda.asp

[10] Coca-Cola Research Council/Andersen Consulting, 1996, "Where to Look for Incremental Sales Gains: The Retail Problem of Out-of-Stock Merchandise," The Coca-Cola Research Council, Atlanta, GA.

[11] Çağdaş Marketler Zinciri, 2008, kaynak: http://www.cagdasmarketler.com/?x=1

[12] Çatı K., 2007, "Süpermarketlerin Tercih Edilmesinde Etkili Olan Faktörlerin Belirlenmesine Yönelik Bir Araştırma", *Elektronik Sosyal Bilimler Dergisi*, 150-168.

[13] Çağrı Hipermarket, 2008, kaynak: http://www.cagrihipermarket.com/kurumsal.aspx

[14] Esenlik, 2008, kaynak: http://www.esenlik.com.tr/kurumsal.htm

[15] Gimsa, 2008, kaynak: http://www.gimsa.com.tr/turkce/hakkimizda.aspx?id=1

[16] GOP Pazar Mrketleri, 2008, kaynak: http://www.gop.com.tr/hakkimizda.asp

[17] Groseri, 2008, kaynak: http://www.groseri.com.tr/kurumsal.asp

[18] Gün Süpermarket, 2008, kaynak: http://www.gun.com.tr/kurumsal.asp

[19] Iyaş, 2008, kaynak: http://www.iyas.com.tr/Hakkimizda.aspx

[20] Karhan V., 2000, "Sektör Raporu", Strateji Menkul Değerler.

[21] Kiler Marketleri, 2008, kaynak: http://www.kiler.com.tr/kileri_taniyalim.asp

[22] Makromarket, 2008, kaynak: http://www.makromarket.net/kurumsal2.html

[23] Meşhur Peynirci 2008, kaynak: http://www.meshurpeynirci.com.tr/kurumsal.asp

[24] Migros Türk Kurumsal, 2008, kaynak: http://www.migros.com.tr/tarihce.asp

[25] Mopaş 2008. kaynak: http://www.mopasmarket.com/hakkinda.php

[26] Özhan Marketler Zinciri, 2008, kaynak: http://www.ozhan.com.tr/tr/ozhan.asp?id=1

[27] Öztürk İ., 2008, "Girişimcilik Raporu", İktisadi Girişim ve İş Ahlakı Derneği.

[28] Pehlivanoğlu, 2008, kaynak: http://www.pehlivanoglu.com.tr/index_tr.html

[29] Pekdemir Alışveriş Merkezleri, 2008, kaynak: http://www.pekdemir.com.tr/tarihce.aspx

[30] Planet Retail, kaynak: http://www.planetretail.net/

[31] Rammar Grosmarket, 2008, kaynak: http://www.rammar.com.tr/

[32] Real Hipermarketleri, 2008, kaynak: http://www.real.com.tr/062_Real_company.html

[33] Sincap Marketler Zinciri, 2008, kaynak: http://www.sincapmarket.com/?post=tarihce

[34] Soysal Marketler Kataloğu 2008, kaynak: http://www.soysal.com.tr/bolum/21/soysal-magazalar-marketler-katalogu-2008/tr

[35] Şaypa, 2008, kaynak: http://www.saypa.com.tr/

[36] Tansaş, 2008, kaynak: http://www.tansas.com.tr/profil.html

[37] Tescokipa, 2008, kaynak: http://tesco.kipa.com.tr/AboutTesco.aspx?id=1

[38] Yimpaş Mağazalar Zinciri, 2008, kaynak: http://www.yimpas.com.tr/kurumsal.php

[39] Yunus Marketler Zinciri, 2008, kaynak: http://www.yunusmarket.com.tr/?sayfa=kurumsal

Cost-Based Domain Filtering for Stochastic Constraint Programming*

Roberto Rossi[1], S. Armagan Tarim[2], Brahim Hnich[3], and Steven Prestwich[1]

Cork Constraint Computation Centre - CTVR, University College, Cork, Ireland
{r.rossi,s.prestwich}@4c.ucc.ie
Department of Management, Hacettepe University, Ankara, Turkey
armagan.tarim@hacettepe.edu.tr
Faculty of Computer Science, Izmir University of Economics, Turkey
brahim.hnich@ieu.edu.tr

Abstract. Cost-based filtering is a novel approach that combines techniques from Operations Research and Constraint Programming to filter from decision variable domains values that do not lead to better solutions [7]. Stochastic Constraint Programming is a framework for modeling combinatorial optimization problems that involve uncertainty [19]. In this work, we show how to perform cost-based filtering for certain classes of stochastic constraint programs. Our approach is based on a set of known inequalities borrowed from Stochastic Programming — a branch of OR concerned with modeling and solving problems involving uncertainty. We discuss bound generation and cost-based domain filtering procedures for a well-known problem in the Stochastic Programming literature, the static stochastic knapsack problem. We also apply our technique to a stochastic sequencing problem. Our results clearly show the value of the proposed approach over a pure scenario-based Stochastic Constraint Programming formulation both in terms of explored nodes and run times.

1 Introduction

Constraint Programming (CP) [1] has been recognized as a powerful tool for modeling and solving combinatorial optimization problems. CP provides global constraints offering concise and declarative modeling capabilities and efficient domain filtering algorithms. These algorithms remove combinations of values which cannot appear in any consistent solution. Cost-based filtering is an elegant way of combining techniques from CP and Operations Research (OR) [7]. OR-based optimization techniques are used to remove from variable domains values that cannot lead to better solutions. This type of domain filtering can be

* S. Armagan Tarim and Brahim Hnich are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027. Roberto Rossi is supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain-Driven Research (CTVR) and Grant No. 05/IN/I886.

P.J. Stuckey (Ed.): CP 2008, LNCS 5202, pp. 235–250, 2008.

combined with the usual CP-based filtering methods and branching heuristics, yielding powerful hybrid search algorithms. Cost-based filtering is a novel technique that has been the subject of significant recent research.

Stochastic Constraint Programming (SCP) [19] is an extension of CP, in which there is a distinction between decision variables, which we are free to set, and stochastic (or observed) variables, which follow some probability distribution. SCP is designed to handle problems in which uncertainty comes into play. Uncertainty may take different forms: data about events in the past may not be known exactly due to measuring or difficulties in sampling, and data about events in the future may simply not be known with certainty.

In this work we propose a novel approach to performing cost-based filtering for certain classes of stochastic constraint programs. Our approach is based on a well-known inequality borrowed from Stochastic Programming [4], a branch of OR that is concerned with modeling constraint satisfaction/optimization problems under uncertainty. We implemented this approach for two problems in which uncertainty plays a role. In both cases we obtained significant improvements with respect to a pure SCP formulation both in terms of explored nodes and run times.

The rest of the paper is structured as follows. In Section 2 we give the necessary formal background. In Section 3 we review relevant inequalities from Stochastic Programming. In Section 4, we introduce global optimization chance constraints. We describe our empirical results in Section 5 and review related works in Section 6. Finally, we conclude and outline our future work in Section 7.

2 Formal Background

A *Constraint Satisfaction Problem* (CSP) [1] is a triple $\langle V, C, D\rangle$, where $V = \{V_1, \ldots, V_n\}$ is a set of decision variables, D is a function mapping each element of V to a domain of potential values, and C is a set of constraints stating allowed combinations of values for subsets of variables in V. A *solution* to a CSP is an assignment to every variable of a value in its domain, such that all of the constraints are satisfied. We may also be interested in finding a feasible solution that maximizes (minimizes) the value of a given objective function over a subset of the variables. With no loss of generality, we restrict our discussion to maximization problems.

Optimization-oriented global constraints embed an optimization component, representing a proper relaxation of the constraint itself, into a global constraint [7]. This component provides three pieces of information: (a) the optimal solution of the relaxed problem; (b) the optimal value of this solution representing an upper bound on the original problem objective function; (c) a *gradient function* $\mathbf{grad}(V,v)$, which returns for each variable-value pair (V,v) an optimistic evaluation of the profit obtained if v is assigned to V. These pieces of information are exploited both for propagation purposes and for guiding the search.

In [19], a *stochastic CSP* is defined as a 6-tuple $\langle V, S, D, P, C, \theta\rangle$, where V is a set of decision variables and S is a set of stochastic variables, D is a function mapping each element of V and each element of S to a domain of potential values. A decision variable in V is *assigned* a value from its domain. P is a function mapping each element of S to a probability distribution for its associated domain. C is a set of constraints. A constraint $h \in C$ that constrains at least one variable in S is a *chance-constraint.* θ_h is a threshold value in the interval $[0, 1]$, indicating the minimum satisfaction probability for chance-constraint h. Note that a chance-constraint with a threshold of 1 (or without any explicit threshold specified) is equivalent to a hard constraint. A stochastic CSP consists of a number of *decision stages.* A decision stage is a pair $\langle V_i, S_i \rangle$, where V_i is a set of decision variables and S_i is a set of stochastic variables. In an m-stage stochastic CSP, V and S are partitioned into disjoint sets, $V_1, \ldots, V_m$ and $S_1, \ldots, S_m$, and we consider multiple stages, $\langle V_1, S_1 \rangle, \langle V_2, S_2 \rangle, \ldots, \langle V_m, S_m \rangle$. To solve an m-stage stochastic CSP an assignment to the variables in V_1 must be found such that, given random values for S_1, assignments can be found for V_2 such that, given random values for S_2, ..., assignments can be found for V_m so that, given random values for S_m, the hard constraints are satisfied and the chance constraints are satisfied in the specified fraction of all possible scenarios. The solution of an m-stage stochastic CSP is represented by means of a *policy tree* [18]. A policy tree is a set of decisions where each path represents a different possible scenario and the values assigned to decision variables in this scenario. Let $\mathcal{S}$ denote the space of policy trees representing all the solutions of a stochastic CSP. We may be interested in finding a feasible solution, i.e. a policy tree $s \in \mathcal{S}$, that maximizes the value of a given objective function $f(\cdot)$ over the stochastic variables S (edges of the policy tree) and over a subset $\widehat{V} \subseteq V$ of the decision variables (nodes in the policy tree). A *Stochastic COP* is then defined in general as $\max_{s \in \mathcal{S}} f(s)$. In [19] a policy-based view of stochastic constraint programs is proposed. Such an approach has been further investigated in [3]. An alternative semantics for stochastic constraint programs comes from a scenario-based view [4,18]: this solution method consists in generating a scenario-tree that incorporates all possible realizations of discrete stochastic variables into the model explicitly.

3 Value of Stochastic Solutions

Let Ξ be a discrete stochastic (vector) variable whose realizations correspond to the various scenarios. Recall that in the policy-based view of stochastic CP a scenario is a set of edges in the policy tree connecting the root to a leaf. Define

$$\mathrm{P} = \max_{x \in S} z(x, \xi)$$

as the optimization problem associated with one particular scenario $\xi \in \Xi$, where S is a *finite* set, and $z(x, \xi)$ is a real valued function of two (vector) variables x and ξ. Note that in what follows the discussion is dual for minimization problems. In order to simplify the notation used, we will here use the same notation for referring to a problem and to the value of its optimal solution. The meaning will be made clear by the context.

The function $z(x, \xi)$ can be seen as a payoff table that for a given decision x provides the profit with respect to a given scenario ξ having probability $\Pr\{\xi\}$. We may be then interested in computing the optimal solution value to the *recourse problem* [4] RP(P)$= \max_{x \in S} \sum_{\Xi} \Pr\{\xi\} z(x, \xi)$. This can be expressed, by using the expectation operator $\mathbb{E}$, as

$$\mathrm{RP(P)} = \max_{x \in S} \mathbb{E} z(x, \Xi),$$

with an optimal solution x^*.

The *expected value problem*, the deterministic problem obtained by replacing all the stochastic (vector) variables by their expected values, is defined as

$$\mathrm{EV(P)} = \max_{x \in S} z(x, \mathbb{E}[\Xi]).$$

Let us denote by $\widehat{x}$ an optimal solution of the expected value problem, called the *expected value solution*. Anyone familiar with Stochastic Programming or realizing that uncertainty is a fact of life would feel a little insecure about taking decision $\widehat{x}$. Indeed, unless such a decision is independent of Ξ, there is no reason to believe that this decision is even close to the optimal solution of the recourse problem.

For any stochastic maximization (minimization) program, under the assumptions that (i) $z(x, \Xi)$, the profit function, is a concave[1] (convex) function of Ξ and (ii) $\max_{x \in S} z(x, \Xi)$ $(\min_{x \in S} z(x, \Xi))$ exists for all Ξ,

Proposition 1. *EV(P) - RP(P)* ≥ 0 *(EV(P) - RP(P)* ≤ 0*).*

Proof. A proof is given in [2].

It directly follows that EV(P) $\geq$ RP(P) (EV(P) $\leq$ RP(P)). We will base our cost-based filtering strategies on this inequality.[2] Assumption (i) restricts the form of the cost function. As witnessed by much of the Stochastic Programming literature [4,11], many real life applications exhibit such a behavior in the profit (cost) function. Nevertheless, it is often possible to encounter stochastic constraint programs whose objective exhibits a generalized non-convex dependence on the stochastic variables. Note that, although the classical Jensen (Proposition 1) and Edmundson-Madansky type bounds [4], which we will employ in the following sections, or their extensions are generally not available for such problems, tight bounds may still be constructed under mild regularity conditions as discussed in [13]. Assumption (ii) states that Proposition 1 provides a valid bound only when a feasible solution exists and its existence is not affected by the distribution of the stochastic variables. Intuitively, this means that nothing can be inferred by using Proposition 1 if EV(P) is infeasible or, clearly, if RP(P) is infeasible. Assumption (ii) may be violated in problems where

[1] A real-valued function f is *convex* if for any x_1, x_2 in the domain and any $\lambda \in [0, 1]$, $\lambda f(x_1) + (1-\lambda) f(x_2) \geq f(\lambda x_1 + (1-\lambda) x_2)$ [5]. f is *concave* if $-f$ is convex.

[2] Other inequalities are discussed in [4], pp. 140–141. Effective relaxations can be also built on these other inequalities.

chance-constraints appear. We will not discuss how to handle generic chance-constraints and how to produce deterministic equivalent reformulations for them in EV(P): the reader may refer to [6]. In this work we will consider only examples of stochastic COPs that always satisfy assumptions (i) and (ii). In particular, to comply with assumption (ii), we will consider problems for which a feasible solution always exists and for which the chance-constraints are "hard" ($\theta = 1$). Note that "hard" chance-constraints in RP(P) become deterministic in EV(P).

4 Global Optimization Chance-Constraints

Solving stochastic constraint programs is computationally a challenging task. In [19], the computational complexity — membership in PSPACE — of these models is discussed. In [18], the authors proposed a standard way of compiling down these models into conventional (non-stochastic) CP models that can be solved by any available commercial software. This approach employs a scenario-based [4] modelling strategy for representing stochastic variables. Of course this approach has a price since the number of scenarios that need to be considered in order to fully represent the problem grows exponentially with the number of decision stages in the problem. A possible way to overcome this difficulty is to reduce the number of scenarios considered by sampling them, but this obviously affects the completeness of the model. Another possibility consists instead in developing specialized and efficient filtering strategies. For this purpose *global chance-constraints* have been proposed in [16]. These constraints differ from conventional global constraints in the fact that they represent relations among a non-fixed number of decision variables and stochastic variables.

In this work, by creating a parallel with [7], we present *optimization-oriented global chance-constraints* as a way of enhancing the solving process of stochastic constraint programs. Conventional optimization-oriented global constraints perform cost-based filtering by encapsulating in global constraints optimization components representing suitable relaxations of the constraint itself. Similarly optimization-oriented global chance-constraints also encapsulate suitable relaxations of the constraint considered, but in contrast to conventional optimization-oriented global constraints this relaxation may involve stochastic variables.

A *global optimization chance-constraint* provides the same three pieces of information provided by optimization-oriented global constraints. The difference is the fact that in a global optimization chance-constraint we find two stages of relaxations. At the first stage of relaxation, we are mainly involved with the stochastic variables and we exploit well-known inequalities such as the one in Proposition 1 to replace stochastic variables in our stochastic programs with deterministic quantities and to yield a valid relaxation that is a deterministic problem. This deterministic problem, however, may still be computationally very challenging (NP-hard in general). Therefore, a second stage of relaxation may be needed to produce a further relaxation that is computationally more tractable. Finally, as we will see, a global optimization chance-constraint may also provide a valid, and possibly good, solution at each node of the search tree.

Objective:

$$\max\left\{\sum_{i=1}^{k} r_i X_i - c\mathbb{E}\left[\sum_{i=1}^{k} \mathcal{W}_i X_i - q\right]^+\right\}$$

Decision variables:

(1) $X_i \in \{0,1\}$ $\qquad \forall i \in 1,\ldots,k$

Stochastic variables:

$\mathcal{W}_i \rightarrow$ item i weight

Fig. 1. RP(SSKP). Note that $[y]^+ = \max\{y, 0\}$.

In this section and in the following ones we will refer to a running example and we will employ the following problem to better understand the concepts explained. Consider the *Static Stochastic Knapsack Problem* (SSKP) [12]: a subset of k items has to be chosen, given a knapsack of size q into which to fit the items. Each item i has an expected reward of r_i. The size $\mathcal{W}_i$ of each item is not known at the time the decision has to be made, but we assume that the decision maker has an estimate of the probability distribution of $\overline{\mathcal{W}} = (\mathcal{W}_1, \ldots, \mathcal{W}_k)$. A per unit penalty of c has to be paid for exceeding the capacity of the knapsack. By modeling this problem as a one-stage Stochastic COP, the recourse problem RP(SSKP) can be formulated as shown in Fig. 1. The objective function maximizes the trade-off between the reward brought by the objects selected in the knapsack (those for which the binary decision variable X_i is set to 1) and the expected penalty paid for buying additional capacity units in those scenarios in which the low cost capacity q is not sufficient.

Example 1. Consider 5 items, item rewards r_i are $\{10, 15, 20, 5, 25\}$. The discrete probability distribution functions $f(i)$ for the weight of item $i = 1, \ldots, 5$ are respectively, $f(1) = \{10(0.5), 8(0.5)\}$, $f(2) = \{10(0.5), 12(0.5)\}$, $f(3) = \{9(0.5), 13(0.5)\}$, $f(4) = \{4(0.5), 6(0.5)\}$, $f(5) = \{12(0.5), 15(0.5)\}$. The figures in parenthesis represent the probability that an item takes a certain weight. The other problem parameters are $c = 2$, $q = 30$. The optimal solution of the recourse problem selects items $\{2, 3, 5\}$ and has a value of RP(SSKP)=49.

This solution can be obtained by solving a deterministic equivalent conventional constraint program obtained by employing a scenario-based representation [18]. Let $\mathcal{W}_i^j$ be the realized weight of object i in scenario j. We hand-crafted a deterministic equivalent model DetEquiv(RP(SSKP)) for RP(SSKP) following the guidelines in [18]. This model is shown in Fig. 2. Constraint (1) states that Z_j, total excess weight in scenario j, must be greater than the sum of the weights of the objects selected in this scenario minus the low cost capacity q. Constraint (2) restricts the decision variables X_i to be binary. X_i is equal to 1 iff item i is selected in the knapsack. Constraint (3) fixes an upper bound for Z_j; this upper bound is the sum of the weights of all the k objects in scenario j. The objective function maximizes the trade-off between the total reward brought by the objects selected and the sum of penalty costs — weighted by the respective scenario probability — paid for those scenarios where the low cost capacity q is not sufficient.

Objective:

$$\max\left\{\sum_{i=0}^{k} r_i X_i - c\left[\sum_{j=1}^{n} Z_j \Pr\{j\}\right]\right\}$$

Constraints:

(1)	$Z_j \geq \sum_{i=1}^{k} \mathcal{W}_i^j X_i - q$	$\forall j \in 1,\dots,n$
Decision variables:		
(2)	$X_i \in \{0,1\}$	$\forall i \in 1,\dots,k$
(3)	$Z_j \in [0, \sum_{i=1}^{k} \mathcal{W}_i^j]$	$\forall j \in 1,\dots,n$

Fig. 2. DetEquiv(RP(SSKP)). $\Pr\{j\}$ is the probability of scenario $j \in \{1,\dots,n\}$. Note that $\sum_{j=1}^{n} \Pr\{j\} = 1$.

4.1 Expectation-Based Relaxation for Stochastic Variables

The first step in our cost-based filtering strategy consists in applying a relaxation involving the stochastic variables. By applying Proposition 1, if the profit (respectively cost for minimization problems) function satisfies the two assumptions discussed, an upper (lower) bound for the cost of an optimal solution to RP(P) can be obtained by solving EV(P), that is the deterministic problem in which all the stochastic variables are replaced by their respective expected values.

Lemma 1. *The profit function for $RP(SSKP)$ is concave in $\overline{\mathcal{W}}$.*

Proof. When proving concavity w.r.t. $\overline{\mathcal{W}}$ we can ignore the constant term $\sum_{i=1}^{k} r_i X_i$. What remains is $f(\overline{\mathcal{W}}) = -c\mathbb{E}\left[\overline{\mathcal{W}}^T \cdot \overline{X} - q\right]^+$, where "$\cdot$" is the inner product and $\overline{\mathcal{W}}^T$ is vector $\overline{\mathcal{W}}$ transposed. We now prove that $-f(\overline{\mathcal{W}}) = c\mathbb{E}\left[\overline{\mathcal{W}}^T \cdot \overline{X} - q\right]^+$ is convex in $\overline{\mathcal{W}}$. By recalling that a maximum of convex functions is convex [5], this function is clearly convex w.r.t. each element of vector $\overline{\mathcal{W}}$ and it is therefore convex in $\overline{\mathcal{W}}$. This implies that $-f$ is concave in $\overline{\mathcal{W}}$.

Obviously, in RP(SSKP), it is always possible to find a feasible assignment for decision variables, therefore both the assumptions are satisfied for this problem. The expected value problem EV(SSKP) can be obtained by replacing every random variable $\mathcal{W}_i$ in RP(SSKP) with the respective expected value $\mathbb{E}[\mathcal{W}_i]$, thus obtaining a fully deterministic model.

Example 2. Here we solve the problem where the weights of the objects are deterministic and equal to the respective expected weights[3]: $\lfloor\mathbb{E}[f(1)]\rfloor = 9$, $\lfloor\mathbb{E}[f(2)]\rfloor = 11$, $\lfloor\mathbb{E}[f(3)]\rfloor = 11$, $\lfloor\mathbb{E}[f(4)]\rfloor = 5$, $\lfloor\mathbb{E}[f(5)]\rfloor = 13$. This problem provides the first two pieces of information needed by our cost-based filtering method, that is (a) the optimal solution of the relaxed problem and (b) the optimal value of this solution, which represents, according to Proposition 1, an upper bound for the original problem objective function. In our running example this solution selects items $3, 4, 5$ and has a value of EV(SSKP)$= 50$.

[3] As this is a maximization problem, the expected weight of each object is rounded down to the nearest integer ($\lfloor\ \rfloor$) in order to keep the bound provided by the relaxation optimistic.

4.2 Relaxing the Expected Value Problem

It should be noted that, although the expected value problem is easier than the recourse problem, it may still be difficult to solve (NP-hard). For this reason we can further relax the expected value problem in order to obtain a valid bound by solving an easier problem. Let R(EV(P)) be a generic relaxation of EV(P). Then for a maximization problem EV(P)$\leq$ R(EV(P)) holds, therefore R(EV(P)) provides a valid bound for the recourse problem.

In SSKP, for instance, instead of solving to optimality the deterministic (NP-Complete) knapsack problem obtained for the expected value scenario, we may instead solve in linear time its continuous relaxation, thus obtaining Dantzig's upper bound, DUB(EV(SSKP)) [15]. DUB(EV(SSKP)) $\geq$ EV(SSKP) therefore DUB(EV(SSKP)) $\geq$ RP(SSKP). DUB(EV(SSKP)) is a valid upper bound for our recourse problem.

Example 3. To obtain DUB(EV(SSKP)) we order items for profit over expected weight: $\{25/13, 20/11, 15/11, 10/9, 5/5\}$, and we insert items until the first that does not fit completely into the remaining knapsack capacity. Of this last item we take a fraction of the profit proportional to the capacity available. Therefore DUB(EV(SSKP))$= 25 + 20 + (6 * 15/11) = 53.18$.

Obviously at any node of the search tree it is possible to solve the expected value problem taking into account decision variables already assigned. The bound obtained can be used to exclude part of the tree that cannot lead to a better solution.

In [7] the authors discuss filtering strategies based on reduced costs (RC). As we shall see in the next section a similar technique can be adopted for SSKP, provided that an efficient way of obtaining bounds is available for the expected value problem.

4.3 Cost-Based Filtering

In order to perform cost-based filtering, as in RC-based filtering, we need a *gradient function* **grad(**V**,**v**)**, which returns for each variable-value pair (V,v) an optimistic evaluation of the profit obtained if v is assigned to V. This function is obviously problem dependent, but regardless of the strategy adopted in the former section — i.e. whether we are using a relaxation for the expected value problem or solving this problem to optimality — it is possible to specify it and use it to filter provably suboptimal values. In what follows we present a gradient function for SSKP. At each node of the search tree, in order to compute this function, we use a continuous relaxation of the expected value problem similar to the one proposed by Dantzig for the well-known 0-1 Knapsack Problem [15]. We will now define the gradient function for SSKP by reasoning on the expected value problem. Assume that a partial assignment for decision variables is given. Let K be the set of all the items in the problem, $|K| = k$. Let S be the set of items for which a decision has been fixed, with $|S| < k$. Let q^* be the sum of the expected weights of the elements in S that are part of the knapsack. The profit

$\overline{r}$ associated with this assignment is equal to the sum of the profits of the items in the knapsack minus the eventual expected penalty cost $c(q^* - q)$, if $q - q^*$ is negative. Now we consider an element $i \in K/S$. There are two possible options: taking it into the knapsack or not. If we take it, we increase the profit by r_i minus any eventual expected penalty cost we pay if the expected residual capacity is or becomes negative. Finally for every other element in K/S we check if the balance between its profit and the eventual expected penalty gives an overall positive profit and, if so, we add it to the knapsack. This procedure requires at most $O(k)$ steps for each element for which a decision has not yet been taken, therefore it can be applied at each node of the search tree to compute a valid upper bound associated with a certain decision on an item, which therefore may be filtered if suboptimal.

Example 4. We now consider the case in which items 2 and 3 have been selected in the knapsack and item 4 is not selected. We still have to decide on items 1 and 5. The total capacity used is $c^* = 11 + 11 = 22$. The profit $\overline{r}$ brought by items 2 and 3 is 35. We consider the set of the remaining items for which a decision must be taken, $K/S \equiv \{1, 5\}$. Let us reason on item 1: this is a critical item, in fact if taken in the knapsack it will use more capacity than the residual $30 - 22 = 8$ units. If we consider the option of taking this item, then the expected profit is $\overline{r}_1 = 10 - 2 * (30 - 22 - 9) = 8$, there is no more residual capacity and item 5 is therefore excluded in the bound computation since $25 - 4 * 13 \leq 0$. The computed bound is $35 + 8 = 43$. The reasoning is similar for item 5. If we consider the option of taking this item, then the expected profit is $\overline{r}_5 = 25 - 2 * (30 - 22 - 13) = 15$, there is no more residual capacity and item 1 is therefore excluded in the bound computation since $10 - 4 * 9 \leq 0$. The computed bound is $35 + 15 = 50$. Assume now that the current best solution has a value of 46, corresponding to a knapsack that contains elements 3, 4 and 5: then element 1 can be excluded from the knapsack.

Obviously, as discussed in [7] the information provided by the relaxed model (EV(P)), i.e. expected weights, gradient function etc., can be also used to define search strategies. For instance in SSKP we may branch on variables according to a decreasing profit over expected weight heuristic, or selecting the one for which the chosen gradient function gives the most promising value.

4.4 Finding Good Feasible Solutions

In CP, it is critical, in order to achieve efficiency, to quickly obtain a good feasible solution so that cost-based filtering can prune provably suboptimal nodes as early as possible. In Stochastic COPs the EV(P) solution can be often used as a good starting solution in the search process. If such a solution is feasible with respect to RP(P) — in our examples assumption (ii) guarantees this — we can easily compute EEV(P), that is *the expected result of using the EV(P) solution in the recourse problem RP(P).* Furthermore, at every node of the search tree it is possible to adopt a variable fixing strategy and compute the EV(P) solution with respect to such a node, that is the best possible EV(P) solution incorporating the partial decisions represented by the given node of the search tree. This provides a full assignment for decision variables in RP(P) at each point of the search. By using this assignment, we can again easily compute EEV(P). In this case EEV(P) is the cost of a feasible, and possibly good, solution for RP(P) incorporating the partial assignment identified by the current node explored in the search tree.

Example 5. In our SSKP example the solution of the expected value problem, EV(SSKP), selects items 3, 4 and 5 in the optimum knapsack. This solution is clearly feasible for RP(SSKP). We can therefore compute EEV(SSKP)= 46. This is, of course, a good lower bound for the objective function value.

5 Experimental Results

In this section we report our computational experience on two one-stage stochastic COPs, the SSKP and the Stochastic Sequencing with Release Times and Deadlines (SSEQ). In our experiments we used Choco 1.2, an open source solver written in Java [14]. We ran our experiments on an Intel(R) Centrino(TM) CPU 1.50GHz with 2Gb of RAM.

5.1 Static Stochastic Knapsack Problem

We created a Choco CP model for DetEquiv(RP(SSKP)), and we implemented for it a global optimization chance-constraint incorporating the filtering discussed in the former sections. To recall, within this constraint at each node of the search tree the stochastic variables are replaced by their respective expected values. Then, after fixing decision variables according to the partial solution associated with the given search tree node, EV(SSKP) is solved and the bound obtained is used to prune suboptimal parts of the search tree. Furthermore cost-based filtering is performed as explained in Section 4.3. Finally EEV(P), the *expected result of using the EV(P) solution in the recourse problem*, is computed at each node of the search tree and used as a valid lower bound (profit of a feasible solution). In fact RP(SSKP) satisfies assumption (ii) for Proposition 1, therefore the solution of EV(SSKP) is feasible for RP(SSKP).

In our experiments we adopted a randomly generated test bed similar to the one proposed in [12]. There are three sets of instances considered: the first set has $k = 10$, the second set has $k = 15$ and the third has $k = 20$ items. For all the instances, item random weights, W_i, from which scenarios are generated, are independent and normally distributed with probability distribution function $N(\mu_i, \sigma_i)$. The expected weights, μ_i, are generated from the uniform (20,30) distribution, and the weight standard deviations, σ_i, are generated from the uniform (5,10) distribution. Rewards r_i are generated from the uniform (10,20) distribution. The per unit penalty is $c = 4$, while the available low cost capacity is $q = 250$ for 20 items, $q = 187$ for 15 items, and $q = 125$ for 10 items. We randomly generated, using simple random sampling, sets of scenarios having different sizes: $\{100, 300, 500, 1000\}$. Scenarios are equally likely. The variable selection heuristic branches first on items with lower profit over expected weight

Table 1. Experimental results for SSKP. Comparison between a pure SCP approach (SCP) and an SCP model enhanced with optimization-oriented global-chance constraints (SCP-OO), times are in seconds. In each line we indicated in bold the best performance in terms of run time and explored nodes.

Instance		Time		Nodes	
k	Scenarios	SCP	SCP-OO	SCP	SCP-OO
10	100	**0.4**	0.5	916	**100**
10	300	1.3	**0.5**	2630	**59**
10	500	2.4	**0.2**	4237	**8**
10	1000	7.2	**2.4**	6227	**120**
15	100	2.5	**0.3**	4577	**11**
15	300	15	**2.3**	10408	**252**
15	500	33	**1.1**	9982	**75**
15	1000	150	**6.3**	16957	**222**
20	100	70	**10**	102878	**1024**
20	300	250	**13**	85073	**953**
20	500	860	**9.5**	129715	**225**
20	1000	3200	**240**	134230	**7962**

ratio. The value selection tries first not to insert an item into the knapsack. In Table 1 we report our computational results. In all the instances considered our approach outperforms a pure SCP model in terms of explored nodes: the maximum improvement reaches a factor of 576.5. Run times are also shorter in our approach for almost all the instances. An exception is observed for the smallest instance, where the cost of filtering domains is not compensated by the payoff in terms of reduction of the search space. The maximum speed-up observed for run times reaches a factor of 90.5.

5.2 Stochastic Sequencing with Release Times and Deadlines

We consider a specific sequencing problem similar to the one considered by Hooker et. al [9]. Garey and Johnson [8] also mention this problem in their list of NP-hard problems and they refer to it as “Sequencing with Release Times and Deadlines” (SSEQ). An optimization version of this scheduling problem was also described in [10]. The problem consists in finding a feasible schedule to process a set I of k orders (or jobs) using a set M of n parallel machines. Processing an order $i \in I$ can only begin after the release date r_i and must be completed at the latest by the due date d_i. Order i can be processed on any of the machines. The processing time of order $i \in I$ on machine $m \in M$ is P_{im}. The model just described is fully deterministic, but we will now consider a generalization of this problem to the case where some inputs are uncertain. For convenience we will just consider uncertain processing times $\mathcal{P}_{im}$ for order $i \in I$ on machine $m \in M$. Instead of simply finding a feasible plan we now aim to minimize the expected total tardiness of the plan (the deterministic version of this problem is known as “Sequencing to minimize weighted tardiness” [8] and it is NP-hard). A solution for our SSEQ problem consists in an assignment for the jobs on the machines and in a total order between jobs on the same machine. In such a plan, a job will be processed on its release date if no other previous job is still processing,

Objective:

$\min\left\{\sum_{i=1}^{k} \mathbb{E}\left[\mathcal{C}_i - d_i\right]^+\right\}$

Constraints:

(1) $S_{ab} + S_{ba} \leq 1$ $\quad\forall a, b \in 1, \ldots, k,\ a \neq b$

(2) $X_{am} + X_{bm} \leq S_{ab} + S_{ba} + 1$ $\quad\forall a, b \in 1, \ldots, k,\ a \neq b,\ \forall m \in 1, \ldots, n$

(3) $\sum_{m=1}^{n} X_{im} = 1$ $\quad\forall i \in 1, \ldots, k$

(4) $\mathcal{C}_i - \sum_{m=1}^{n} \mathcal{P}_{im} X_{im} \geq r_i$ $\quad\forall i \in 1, \ldots, k$

(5) $S_{ab} = 1 \rightarrow \mathcal{C}_b \geq \mathcal{C}_a + \sum_{m=1}^{n} \mathcal{P}_{bm} X_{bm}$ $\quad\forall a, b \in 1, \ldots, k,\ a \neq b$

Decision variables:

(6) $X_{im} \in \{0, 1\}$ $\quad\forall i \in 1, \ldots, k,\ \forall m \in 1, \ldots, n$

(7) $S_{ab} \in \{0, 1\}$ $\quad\forall a, b \in 1, \ldots, k,\ a \neq b$

Stochastic variables:

$\mathcal{P}_{im}$: processing time of job i on machine m

Auxiliary variables:

$\mathcal{C}_i$: stochastic completion time of job i.

Fig. 3. RP(SSEQ). Note that $[y]^+ = \max\{y, 0\}$. $\mathbb{E}$ denotes the expectation operator.

or as soon as the previous job terminates. The recourse problem RP(SSEQ) can be formulated as a one-stage Stochastic COP. This is shown in Fig. 3.

Decision variable X_{im} takes value 1 iff job i is processed on machine m, decision variable S_{ab} takes value 1 iff job a is processed before job b. Constraints (1) and (2) enforce a total order among jobs on the same machine. Constraint (3) enforces that each job must be processed on one and only one machine. Constraint (4) states that the (stochastic) completion time, $\mathcal{C}_i$, of a job i minus its (stochastic) duration $\mathcal{P}_{im}$ on the machine on which it is processed must be greater than or equal to its release date r_i, where $\mathcal{C}_i$ is an auxiliary variable used for simplifying notation. Let $I_m \equiv \{\mathcal{J}_{1m}, \mathcal{J}_{2m}, \ldots, \mathcal{J}_{qm}\} \subseteq I$ be the ordered set of jobs assigned to machine m. $\mathcal{C}_{\mathcal{J}_{qm}}$ is defined recursively as $\mathcal{C}_{\mathcal{J}_{qm}} = \max\{r_{\mathcal{J}_{qm}}, \mathcal{C}_{\mathcal{J}_{(q-1)m}}\} + \mathcal{P}_{\mathcal{J}_{qm}m}$, and $\mathcal{C}_{\mathcal{J}_{0m}} = 0$. Constraint (5) states that if two jobs a and b are processed on the same machine and if a is processed before b, that is $S_{ab} = 1$, then the (stochastic) completion time of job a plus the (stochastic) duration of job b on the machine on which it is processed must be less than or equal to the (stochastic) completion time of job b. Finally, the objective function minimizes the sum of the expected tardiness of each job. The tardiness is defined as $\max\{0, \mathcal{C}_i - d_i\}$. The cost function to be minimized can easily be proved convex in the random job durations. The expected total tardiness is in fact minimized for n machines. Job completion times on different machines are independent, therefore if we prove convexity for machine $m \in M$, then it directly follows that the cost function of the problem is also convex[4]. The cost function for machine m can be expressed as $\mathbb{E}\left[\sum_{i \in I_m} (\mathcal{C}_i - d_i)^+\right]$.

Lemma 2. *The expected total tardiness for machine m is convex in the uncertain processing times $\mathcal{P}_{im}$.*

Proof. Maximum of convex functions is convex. $\mathcal{C}_{\mathcal{J}_{1m}} = r_{\mathcal{J}_{1m}} + \mathcal{P}_{\mathcal{J}_{1m}m}$ is convex: it follows that $\mathcal{C}_i$ for any $i \in I_m$ is convex, since function "max" is a convex function. Therefore the objective function is convex.

[4] Note that the sum of convex functions is convex [5].

Objective:

$$\min\left\{\sum_{i=1}^{k}\sum_{v=1}^{w}\Pr\{w\}\,[C_i^v - d_i]^+\right\}$$

Constraints:

(1) $S_{ab} + S_{ba} \leq 1$ $\quad\forall a, b \in 1, \ldots, k,\ a \neq b$

(2) $X_{am} + X_{bm} \leq S_{ab} + S_{ba} + 1$ $\quad\forall a, b \in 1, \ldots, k,\ a \neq b,\ \forall m \in 1, \ldots, n$

(3) $\sum_{m=1}^{n} X_{im} = 1$ $\quad\forall i \in 1, \ldots, k$

and $\forall v \in 1, \ldots, w$

(4) $C_i^v - \sum_{m=1}^{n} \mathcal{P}_{im}^v x_{im} \geq r_i$ $\quad\forall i \in 1, \ldots, k$

(5) $S_{ab} = 1 \to C_b^v \geq C_a^v + \sum_{m=1}^{n} \mathcal{P}_{bm}^v X_{bm}$ $\quad\forall a, b \in 1, \ldots, k,\ a \neq b$

Decision variables:

(6) $X_{im} \in \{0, 1\}$ $\quad\forall i \in 1, \ldots, k,\ \forall m \in 1, \ldots, n$

(7) $S_{ab} \in \{0, 1\}$ $\quad\forall a, b \in 1, \ldots, k,\ a \neq b$

(8) $C_i^v \in \{0, \max_{i=1,\ldots,k} r_i +$
$\sum_{t=1}^{k}(\max_{m=1,\ldots,n} \pi_{tm}^v)\}$ $\quad\forall i \in 1, \ldots, k,\ \forall v \in 1, \ldots, w$

Fig. 4. DetEquiv(RP(SSEQ)). Note that $[y]^+ = \max\{y, 0\}$. $\Pr\{v\}$ is the probability of scenario $v \in \{1, \ldots, w\}$. Note that $\sum_{v=1}^{w} \Pr\{v\} = 1$.

In RP(SSEQ) a feasible solution can be found for any given set of stochastic job lengths, therefore both the assumptions are satisfied for this problem. We handcrafted a deterministic equivalent model DetEquiv(RP(SSEQ)) shown in Fig. 4 for the RP(SSEQ) following the guidelines of scenario-based approach described in [18]. In this model, $\mathcal{P}_{im}^v$ is the deterministic length of job i on machine m in scenario v and C_i^v is the deterministic completion time of job i in scenario v.

Finally, as discussed for SSKP, we can obtain the expected value problem EV(SSEQ) by replacing every stochastic variable $\mathcal{P}_{im}$ in RP(SSEQ) with the respective expected value $\mathbb{E}[\mathcal{P}_{im}]$. Since all the chance-constraints in RP(SSEQ) are "hard", they are retained in EV(SSEQ) and they become deterministic.

We implemented DetEquiv(RP(SSEQ)) in Choco and we coded an optimization-oriented global chance-constraint which exploits the expected value problem both in order to generate valid bounds at each node of the search tree and to filter provably suboptimal values from decision variable domains. At each node of the search tree, we consider the associated partial assignment for decision variables X_{im} and S_{ab} and we fix decision variables in EV(SSEQ) according to it. Then we solve EV(SSEQ) with respect to the remaining decision variables that have not been assigned. This provides a lower bound for the cost of a locally optimal solution associated with the node considered. This bound can be used for pruning suboptimal nodes. Furthermore at any given node, after performing variable fixing in EV(SSEQ) for every variable X_{im} and S_{ab} already assigned, all the remaining binary variables X_{im} that have not been assigned yet can be forward checked by fixing the respective value to 1, by solving EV(SSEQ) with this new decision fixed, and by employing the new bound provided.

In order to generate instances for our experiments, we adopted release times, deadlines and deterministic processing times from the first two "hard" instances proposed in [9], the one with 3 jobs and 2 machines and the one with 7 jobs and 3 machines. In each scenario, we generated processing times uniformly distributed in $[1, 2 * J_{im}]$, where J_{im} is the deterministic processing time required for job i on machine m for the instance considered. We considered different number of

Table 2. Experimental Results for SSEQ. Comparison between a pure SCP approach (SCP) and an SCP model enhanced with optimization-oriented global-chance constraints (SCP-OO), times are in seconds. In each line we indicated in bold the best performance in terms of run time and explored nodes.

Instance			Time		Nodes	
Jobs	Machines	Scenarios	SCP	SCP-OO	SCP	SCP-OO
3	2	10	**0.3**	**0.3**	203	**48**
3	2	30	1.3	**0.6**	701	**133**
3	2	50	3.2	**1.1**	927	**418**
3	2	100	12	**3.5**	1809	**838**
7	3	10	**180**	866	57688	**1723**
7	3	30	1800	**880**	186257	**5293**
7	3	50	3300	**1100**	212887	**6586**
7	3	100	14000	**1200**	277804	**8862**

scenarios in $\{10, 30, 50, 100\}$. Scenarios are equally likely in terms of probability. The variable selection heuristic branches first on binary decision variables. The value selection tries increasing values in the domain. In Table 2 we report the results observed with and without the improvement brought by our cost-based filtering approach.

It should be noted that in this case, in contrast to the approach employed for SSKP, we only relax stochastic variables and we do not employ a relaxation for the deterministic equivalent problem, which therefore remains NP-hard. Recall that in SSKP we adopted Dantzig's relaxation to efficiently obtain a bound for the deterministic equivalent problem. A direct consequence of this is that, while in the SSKP example the improvement is significant both in terms of explored nodes and run times for all the instances, in this example the run time improvement starts to be significant (a factor of 11.6) only for the largest instance (7 jobs and 3 machines) and for a high number of scenarios (100 scenarios). This is due to the fact that at every node of the search tree we solve a difficult problem (though far easier than the original stochastic constraint program) to obtain bounds and perform cost-based filtering. In terms of explored nodes, however, we obtain a significant improvement for every instance — the maximum improvement factor is of 32.3 — since the bounds generated are tight.

6 Related Work

This paper extends the original work by Focacci et al. [7] on optimization-oriented global constraints. It also extends the original idea of global chance-constraints [16] to optimization problems. It should be noted that dedicated cost-based filtering techniques for stochastic combinatorial optimization problems have been presented in [17], but these techniques are specialized for inventory control problems, while those here presented can be applied to a wider class of stochastic constraint programs. On the other hand this work also builds on known inequalities borrowed from Stochastic Programming [2,4] usually exploited for relaxing specific classes of stochastic programs and obtaining good bounds or approximate solutions. Nevertheless Stochastic Programming models are typically formulated as dynamic programs or MIP models. In both cases these bounds are not exploited for filtering decision variable domains as in our approach and they cannot be used for guiding the search.

7 Conclusions

We proposed a novel strategy to performing cost-based filtering for certain classes of stochastic constraint programs, under the assumptions that (i) the objective function is concave or convex in the stochastic variables, and (ii) the existence of a feasible solution is not affected by the distribution of the stochastic variables. This strategy is based on a known inequality borrowed from Stochastic Programming. We applied this technique to two combinatorial optimization problem involving uncertainty from the literature. Our results confirm that orders-of-magnitude improvements in terms of explored nodes and run times can be achieved. In the future, we aim to apply cost-based filtering to multi-stage Stochastic COPs, define strategies to handle generic chance-constraints, which are currently ruled out by our assumptions, and to extend the approach to other valid inequalities such as Edmundson-Madansky [4] or to suitable inequalities for non-convex problems [13]. Finally, we plan to exploit the information provided by optimization-oriented global chance-constraints to define search strategies.

References

1. Apt, K.: Principles of Constraint Programming. Cambridge University Press, Cambridge (2003)
2. Avriel, M., Williams, A.C.: The value of information and stochastic programming. Operations Research 18(5), 947–954 (1970)
3. Balafoutis, T., Stergiou, K.: Algorithms for stochastic csps. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 44–58. Springer, Heidelberg (2006)
4. Birge, J.R., Louveaux, F.: Introduction to Stochastic Programming. Springer, New York (1997)
5. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
6. Charnes, A., Cooper, W.W.: Deterministic equivalents for optimizing and satisficing under chance constraints. Operations Research 11(1), 18–39 (1963)
7. Focacci, F., Lodi, A., Milano, M.: Optimization-oriented global constraints. Constraints 7, 351–365 (2002)
8. Garey, M.R., Johnson, D.S.: Computer and Intractability. A guide to the theory of NP-Completeness. Bell Laboratories, Murray Hill, New Jersey (1979)
9. Hooker, J.N., Ottosson, G., Thorsteinsson, E.S., Kim, H.J.: On integrating constraint propagation and linear programming for combinatorial optimization. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI 1999), pp. 136–141. The AAAI Press/MIT Press, Cambridge (1999)
10. Jain, V., Grossmann, I.E.: Algorithms for hybrid milp/cp models for a class of optimization problems. INFORMS Journal on computing 13, 258–276 (2001)

11. Kall, P., Wallace, S.W.: Stochastic Programming. John Wiley & Sons, Chichester (1994)
12. Kleywegt, A.J., Shapiro, A., Homem-De-Mello, T.: The sample average approximation method for stochastic discrete optimization. SIAM Journal of Optimization 12(2), 479–502 (2001)
13. Kuhn, D.: Generalized bounds for convex multistage stochastic programs. Lecture Notes in Economics and Mathematical Systems, vol. 584
14. Laburthe, F.: The OCRE project team. Choco: Implementing a cp kernel. Technical report, Bouygues e-Lab, France (1994)
15. Martello, S., Toth, P.: Knapsack Problems. John Wiley & Sons, NY (1990)
16. Rossi, R., Tarim, S.A., Hnich, B., Prestwich, S.: A global chance-constraint for stochastic inventory systems under service level constraints. Constraints 13(4) (2008)
17. Tarim, S.A., Hnich, B., Rossi, R., Prestwich, S.: Cost-based filtering techniques for stochastic inventory control under service level constraints. Constraints (forthcoming) (2008)
18. Tarim, S.A., Manandhar, S., Walsh, T.: Stochastic constraint programming: A scenario-based approach. Constraints 11(1), 53–80 (2006)
19. Walsh, T.: Stochastic constraint programming. In: Proceedings of the 15th ECAI. European Conference on Artificial Intelligence. IOS Press, Amsterdam (2002)

A Steady-State Genetic Algorithm with Resampling for Noisy Inventory Control*

Steven Prestwich[1], S. Armagan Tarim[2], Roberto Rossi[1], and Brahim Hnich[3]

[1] Cork Constraint Computation Centre, University College, Cork, Ireland
s.prestwich@cs.ucc.ie, r.rossi@4c.ucc.ie
[2] Department of Management, Hacettepe University, Turkey
armagan.tarim@hacettepe.edu.tr
[3] Faculty of Computer Science, Izmir University of Economics, Turkey
brahim.hnich@ieu.edu.tr

Abstract. Noisy fitness functions occur in many practical applications of evolutionary computation. A standard technique for solving these problems is fitness resampling but this may be inefficient or need a large population, and combined with elitism it may overvalue chromosomes or reduce genetic diversity. We describe a simple new resampling technique called Greedy Average Sampling for steady-state genetic algorithms such as GENITOR. It requires an extra runtime parameter to be tuned, but does not need a large population or assumptions on noise distributions. In experiments on a well-known Inventory Control problem it performed a large number of samples on the best chromosomes yet only a small number on average, and was more effective than four other tested techniques.

1 Introduction

In many real-world applications of Genetic Algorithms (GAs) and other Evolutionary Computation algorithms, the fitness function is *noisy*: that is, the fitness of a chromosome cannot be computed directly but must be averaged over a number of samples. Examples include the learning of randomised games such as Backgammon, human-computer interaction, and simulation problems for which we wish to evolve a robust plan. The standard deviation of the sample mean of a random variable with standard deviation σ is $\sigma/\sqrt{n}$ where n is the number of samples, so a large number of samples may be needed for very noisy fitness functions.

Several techniques for handling fitness noise in EAs are surveyed in [4,13]: the use of sampling to obtain an average fitness reduces noise; increasing the population size makes it harder for an unfit chromosome to displace a fitter one

* S.A. Tarim and B. Hnich are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027. R. Rossi is supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain-Driven Research (CTVR) and Grant No. 05/IN/I886.

G. Rudolph et al. (Eds.): PPSN X, LNCS 5199, pp. 559–568, 2008.

(a point also made by [10]) and can be viewed as a form of implicit averaging; and *rescaled mutation* samples distant points in the search space then moves a small distance toward them. [5] propose regression to estimate the fitness of neighbouring chromosomes. [1] vary sample rates across both chromosomes and generations in a generational GA. [18] record fitness levels in a search history, and use a stochastic model of fitness levels to locate new points in the search space. [3] use a threshold selection heuristic for accepting chromosomes. [17] adapt the sampling rate to different regions of the search space, a technique they call *dynamic resampling*. [19] use a Bayesian approach to sampling called Optimal Computing Budget Allocation, which assumes normally distributed noise.

A popular approach is to use a *Noisy Genetic Algorithm* (NGA) which computes the fitness of each chromosome by averaging over a number of samples [9,11,14,15]. Following [1] we shall refer to this as *static sampling*, and refer to this algorithm as NGAs. NGAs wastes considerable time evaluating unpromising chromosomes, but it can be improved by linearly increasing the number of samples with search time, starting from a low value [21,27]. We shall refer to this as *incremental sampling* and the resulting algorithm as NGAi. However, though NGAs and NGAi have been used to solve real problems, they may not be the most efficient approach. It is pointed out in [22] that a reduction in noise is not necessary for *every* chromosome, only for the *best* ones. Of course, this entails discovering which are the best chromosomes without performing a large number of samples, but poor chromosomes might become apparent after just a few samples.

An alternative technique is to *resample* chromosome fitness: that is, some chromosomes are allowed to survive for more than one generation, and their fitness is periodically recomputed to refine the estimate. Various heuristics may be used to decide when to discard a chromosome. [22] experiments with averaging over a small number of samples, and guiding resampling by a statistical test which assumes Gaussian noise but is considered to be robust under non-Gaussian noise. [12] uses the standard deviation of the fitness to correct for its noise, again under assumptions on noise distribution. Resampling and the common heuristic of *elitism* do not always combine well. [6] show that, with an elitist GA, the probabilistic method of [12] is inferior to a resampling approach. [2] show that, in Evolutionary Strategies that allow fitness values to survive for more than one generation, failure to resample can lead to systematic overvaluation of chromosomes. [8] found that, when applying co-evolutionary learning to the noisy task of learning how to play Backgammon, more sampling can have a bad effect on the learning besides incurring overhead. It causes less fit chromosomes to be pruned more quickly which reduces genetic diversity too drastically, especially with small populations. Despite these drawbacks, resampling and elitism have been successfully combined. [25] describe an extension of the Simple (generational) GA that maintains a list of the fittest solutions found so far, while increasing the number of samples as search proceeds as in NGAi; they also increase the population size during search.

Another successful resampling elitist GA is the *Kalman-extended Genetic Algorithm* (KGA) [23], designed for problems whose fitness is both noisy and nonstationary. It adapts its sampling rate for each chromosome individually, based on techniques from Kalman filtering. Removing the nonstationary aspects of KGA yields a steady-state algorithm that evaluates the fitness of each new chromosome just once before adding it to the population, then replaces the least-fit population member by the new chromosome. Alternate iterations are devoted to resampling chromosomes that are already in the population. The current fitness estimate of a chromosome is the mean over all its samples. In KGA a chromosome is selected for resampling according to its current fitness estimate and how many times it has already been sampled (which is a measure of the fitness uncertainty): choose the chromosome with fewest samples, among those whose fitness estimates are greater than the population fitness mean minus the population fitness standard deviation. The intuition behind this approach is that unfit chromosomes with high fitness estimate based on only a few samples will be resampled, and their low fitness will become apparent. We shall refer to this as *Kalman sampling*.

In this paper we investigate resampling strategies for the steady-state (therefore elitist) GENITOR algorithm [26]. Our aim is to find a simple resampling strategy that can be used with a steady-state GA, does not assume any noise properties, does not require a large population, resamples fit chromosomes many times to avoid overvaluation, yet on average uses only a few samples per chromosome. We find it necessary to introduce a new runtime parameter that requires manual tuning, but this might be automated in future work. We demonstrate our technique on a well-known problem from Inventory Control. Section 2 describes our algorithm, Section 3 describes the problem we attempt to solve, Section 4 presents experimental results, and Section 5 concludes the paper.

2 The Algorithm

We use a single GA in our experiments: a basic version of GENITOR [26] without refinements such as a gene to determine crossover probability. GENITOR is a steady-state GA that, at each iteration, selects two parent chromosomes, breeds a single offspring by (optional) crossover followed by mutation, evaluates it, and uses it to replace the least-fit member of the population. We use random parent selection, and standard uniform crossover applied with a crossover probability 0.5: if it is not applied then a single parent is selected and mutated. In our problem (described below) each gene can take any of 100 integer values, plus a special value denoted by NULL. Because of the special nature of the NULL value we select it with probability 0.5, otherwise randomly select one of the 100 integer values. Mutation is applied to a chromosome once with probability 0.5, twice with probability 0.25, three times with probability 0.125, and so on. A small population of size 30 is used. We assume that at least U samples are required to obtain a sufficiently reliable fitness estimate, and in experiments we will use the large value $U = 1000$. Thus we face the challenge of sampling effectively without incurring the drawbacks described above: inefficiency, lack of genetic diversity, or overvaluation, while using only a small population.

This is our basic GA but we have yet to specify a sampling strategy to cope with fitness noise. We will compare five resampling strategies, three of which are well-known: static sampling (as in NGAs) in which we take U samples for each chromosome, incremental sampling (as in NGAs) in which we take a variable number of samples per chromosome that linearly increases from 1 to U during the GA execution, and Kalman sampling (as in KGA). The other two strategies are new.

Our first new strategy tries to combine the rapid convergence of Kalman sampling with the reliability of static sampling. It applies Kalman sampling but with a number $S \geq 1$ of samples to initialise and resample chromosomes, with the best value of S to be determined by experiment. We shall refer to this as *Kalman averaged sampling* and our GA with this sampling scheme as KASGA. It is inspired by a note in [1] stating that if the fitness variance in the population is small compared to the noise variance then a GA will make no progress, and it becomes necessary to increase the sample rate. It is also inspired by the use of a small number of samples for evolutionary algorithms in [22].

Our second new strategy also takes S samples each time a chromosome is selected for (re)sampling, but it resamples the chromosome with highest fitness, ignoring chromosomes that already have U samples. Note that if $S < U$ then there is always at least one chromosome with fewer than U samples: the most recently created chromosome, which only has S samples. Note also that we normally choose S to be a divisor of U to avoid unnecessary resampling, but this is not strictly required. We shall call this scheme *greedy averaged sampling* because it greedily resamples the most promising chromosome, based on current fitness estimates. Combining this with the GA we obtain a new algorithm we shall call the Greedy Average Sample GA (GASGA). This is our main contribution and it is summarised in Figure 1.

```
GASGA($S, P, U$)
  create population of size $P$
  evaluate population using $S$ samples
  while not(termination condition)
    select two parents
    breed one offspring O
    evaluate O using $S$ samples
    replace least-fit chromosome by O
    select fittest chromosome F with #samples $< U$
    re-evaluate F using $S$ samples
  output fittest chromosome
```

Fig. 1. GASGA pseudo-code

3 An Inventory Control Problem with Uncertainty

The problem we consider is as follows. Given a planning horizon of N periods and a demand for each period $t \in \{1, \ldots, N\}$, which is a random variable with a given probability density function; we assume that these distributions are normal, though this is not required by our GA. Demands occur instantaneously at the beginning of each time period and are *non-stationary* (can vary from period to period), and demands in different periods are independent. A fixed delivery cost a is incurred for each order, a linear holding cost h is incurred for each product unit carried in stock from one period to the next, and a linear stockout cost s is incurred for each period in which the net inventory is negative (it is not possible to sell back excess items to the vendor at the end of a period). The aim is to find a replenishment plan that minimizes the expected total cost over the planning horizon.

Different inventory control policies can be adopted to cope with this and other problems. A policy states the rules used to decide when orders are to be placed and how to compute the replenishment lot-size for each order. (The term *policy* here refers to the *form* of the plan, whereas in some fields such as Artificial Intelligence a policy refers to an *actual* plan. We use the term in both senses, and the meaning should be clear from the context.) One possibility is the *replenishment cycle policy* (R, S) [20]. With non-stationary demands this policy takes the form (R^n, S^n) where R^n denotes the length of the n^{th} replenishment cycle and S^n the order-up-to-level for replenishment. In this policy a wait-and-see strategy is adopted, under which the actual order quantity for replenishment cycle n is determined only after the demand in former periods has been realized. The order quantity is computed as the amount of stock required to raise the closing inventory level of replenishment cycle $n-1$ up to level S^n. To provide a solution we must populate both the sets R^n and S^n for $n = \{1, \ldots, N\}$. The (R, S) policy yields plans of higher cost than optimal but has been formulated to reduce *nervousness* in inventory control, and is more often used in practice.

There are more efficient algorithms which are guaranteed to yield optimal policies (under reasonable simplifying assumptions) so a GA would not be applied to precisely this problem in practice. However, if we complicate the problem in simple but realistic ways, for example by adding order capacity constraints or dropping the assumption of independent demands, these efficient algorithms become unusable. In contrast, a GA can be used almost without modification. Thus the problem is useful as a representative of a family of more complex problems.

The replenishment cycle policy can be modelled as follows. Each chromosome represents a single policy, each gene corresponds to a period n, an allele specifies the order-up-to level or the lack of an order (denoted here by the special value NULL) for that period, and a chromosome's fitness is the inverse of the total cost incurred by the policy that it represents. For our experiments we allow 100 different order-up-to levels, linearly spaced in the range 1–300. Thus each gene has 101 alleles. These parameters were chosen as suitable for the instances we tested.

4 Experiments

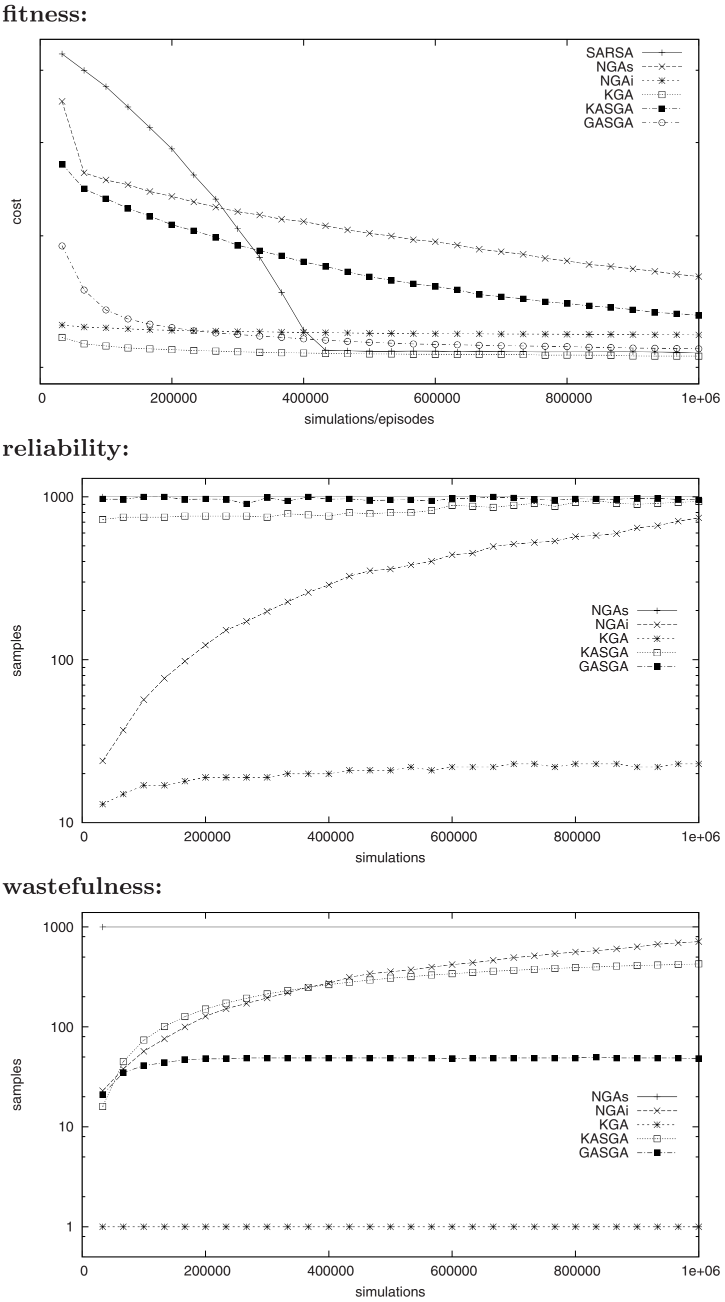
We obtained results using several problem parameter settings, and in each case found the same relationships between the algorithms. For this reason, and because of limited space, we present results for only one instance: 100 periods, stationary demands with mean 50 and standard deviation 10 in all periods, and cost parameters $h = 1$, $a = 400$ and $s = 10$. Problems with 100 periods are very hard: none of the methods we test can find the optimal policy within several hours (nor did attempts using Mixed Integer Programming and Reinforcement Learning algorithms). The optimal policy has an expected total cost of 19,561 with replenishment every 4 periods (starting from the first period) and order-up-to levels of 205 deduced from the cyclic nature of the problem (which is not exploited by the algorithms we test).

We will compare several GAs using three metrics: the *fitness* of the selected chromosome, the *reliability* of the selected chromosome measured by the number of samples used to compute the fitness, and the *wastefulness* of the GA measured by the number of samples used to estimate the fitness of discarded chromosomes. Almost every chromosome is discarded at some point during search, so the wastefulness is an approximation to the average number of samples used per chromosome. Ideally we aim for a GA with high fitness and reliability, but low wastefulness. In our experiments we aim for a reliability of $U = 1000$. The results are shown in Figure 2.

The fitness graph also shows results for the SARSA(λ) Reinforcement Learning algorithm [24] for comparison, as the problem can be modelled as an episodic Partially Observable Markov Decision Process in which a *state* is the period, an *action* is either the choice of an order-up-to level or the lack of an order (NULL) in a period, and a *reward* (undiscounted) is minus the total cost incurred in a period. We use an ϵ-greedy heuristic, varying ϵ inversely with time as recommended in [24], and tuning the α, λ parameters by the common method of hill-climbing in parameter space. All state-action values were initialised to 0, as the use of optimistic initial values encourages early exploration [24].

Because there is a range of Pareto-optimal solutions among the chromosomes of a GA, varying from high fitness based on few samples to low fitness based on many samples, we have a problem: how should different GAs be compared? We are interested in fit solutions based on many samples, so for each GA we shall select the chromosome with the greatest value of samples/cost. The results are as follows.

The graphs show that NGAs has high reliability, but it converges quite slowly and has high wastefulness as it uses exactly 1000 samples for every chromosome. NGAi has much better fitness than NGAs. It reaches this fitness rapidly but then make little further progress, perhaps because of its increasing wastefulness. However, it achieves NGAs's reliability by the end of the run, and only matches its wastefulness by the end of the run. Note that the reliability does not quite reach 1000 samples: there is a delay between (i) increasing the number of samples to a given number, and (ii) obtaining a chromosome whose fitness is both high and based on that number of samples. This delay would not occur in a generational GA, in which no chromosome survives to the next generation. We should perhaps use a generational GA to evaluate incremental sampling, as this was the form of GA used in the Noisy GA work, but in this paper our aim is to



fitness:

reliability:

wastefulness:

Fig. 2. Experimental results

compare several sampling techniques on the same (steady-state) GA. However, a generational GA will presumably exhibit similar wastefulness.

KGA has excellent fitness but very low reliability. Though KGA has given good results on other problems, here no chromosome survives long enough to achieve a sufficient number of samples. This is caused by the high fitness noise in our problem: as chromosomes are resampled their estimated fitnesses fluctuate significantly, and over many iterations the fittest chromosome is not much more likely to survive than any other. Our problem is very noisy, with the fitness standard deviation not much less than the mean, and KGA seems unsuitable for such problems. KASGA is a marked improvement over KGA. Increasing S until the reliability is approximately 1000 samples, we reach a value $S = 250$. The graphs show that KASGA has better fitness than NGAs but no other algorithm, probably because of its fairly high wastefulness (approximately 400 samples per chromosome). But it does have high reliability, making it more usable than KGA.

GASGA outperforms KASGA and the other algorithms. Again increasing S until reliability is approximately 1000, this time we reach a value of only $S = 25$. The graphs show that GASGA has higher fitness than any other GA (other than KGA). GASGA is also less wasteful than any other GA (other than the unreliable KGA): though it finds high-fitness solutions using 1000 samples, it uses only 39 samples per chromosome on average. This is exactly what we aimed for: a GA that achieves high fitness and reliability but low wastefulness.

As noted above, in further experiments using different problem parameters we obtained the same relationships among the GAs. The only difference was the SARSA(λ) result: on this instance it found a solution that was approximately as good as that found by GASGA, on others it found better solutions, and on others it found worse solutions. This illustrates the known fact that Reinforcement Learning and Evolutionary Computation are rival approaches to some problems, and neither dominates the other over all instances [16].

GASGA should find application to many problems with noisy fitness functions. The required number of samples can be chosen by considering the required solution accuracy and the observed variance in solution fitness. Parameter S must currently be tuned by hand: too small a value causes GASGA to behave like KGA, and it never obtains a reliable solution; too large a value causes it to behave like NGA, and it converges slowly. We tried automating S by maintaining it at a level that only just generates a chromosome with 1000 samples, but this forced it to a higher value than necessary (over 100); automation of S is a topic for future work.

5 Conclusion

We designed a simple new resampling strategy for steady-state GAs that makes no assumptions about fitness noise distributions (though problems with different distributions will probably require the parameter values to be tuned differently), does not require a large population, provides a high level of reliability, yet takes a low number of samples on average. Incorporated into GENITOR and applied to a problem from classical Inventory Control, it gave better results than four other sampling strategies. In future work we will evaluate GASGA on other problems with noisy fitness functions such as perception [7], image registration [9,15], network design [27] and remediation design [11].

None of the algorithms we tested are able to find optimal policies for the inventory problem so it is a challenging benchmark for Evolutionary Computation, and in further experiments we also found it to be hard for Reinforcement Learning and Mixed Integer Programming. This makes it an interesting benchmark despite its simplicity, and in future work we will add features such as order capacity constraints.

References

1. Aizawa, A., Wah, B.: Scheduling of Genetic Algorithms in a Noisy Environment. Evol. Comput. 2(2), 97–122 (1994)
2. Arnold, D.V., Beyer, H.-G.: Local Performance of the (1+1)-ES in a Noisy Environment. IEEE Trans. Evolutionary Computation 6(1), 30–41 (2002)
3. Beielstein, T., Markon, S.: Threshold Selection, Hypothesis Tests, and DOE Methods. In: Congress on Evolutionary Computation, pp. 777–782. IEEE Press, Los Alamitos (2002)
4. Beyer, H.-G.: Evolutionary Algorithms in Noisy Environments: Theoretical Issues and Guidelines for Practice. Computer Methods in Applied Mechanics and Engineering 186(2–4), 239–267 (2000)
5. Branke, J., Schmidt, C., Schmeck, H.: Efficient Fitness Estimation in Noisy Environments. In: Genetic and Evolutionary Computation Conference, pp. 243–250. Morgan Kaufmann, San Francisco (2001)
6. Bui, L.T., Abbass, H.A., Essam, D.: Fitness Inheritance for Noisy Evolutionary Multi-Objective Optimization. In: Genetic and Evolutionary Computation Conference, Washington DC, USA. ACM Press, New York (2005)
7. de Croon, G., van Dartel, M.F., Postma, E.O.: Evolutionary Learning Outperforms Reinforcement Learning on Non-Markovian Tasks. In: Workshop on Memory and Learning Mechanisms in Autonomous Robots, 8th European Conference on Artificial Life, Canterbury, Kent, UK (2005)
8. Darwen, P.J.: Computationally Intensive and Noisy Tasks: Coevolutionary Learning and Temporal Difference Learning on Backgammon. Congress on Evolutionary Computation (2000)
9. Fitzpatrick, J.M., Grefenstette, J.J.: Genetic Algorithms in Noisy Environments. Machine Learning 3, 101–120 (1988)
10. Goldberg, D.E., Deb, K., Clark, J.H.: Genetic Algorithms, Noise, and the Sizing of Populations. Complex Systems 6, 333–362 (1992)
11. Gopalakrishnan, G., Minsker, B.S., Goldberg, D.: Optimal Sampling in a Noisy Genetic Algorithm for Risk-Based Remediation Design. In: World Water and Environmental Resources Congress, ASCE (2001)
12. Hughes, E.J.: Evolutionary Multi-objective Ranking with Uncertainty and Noise. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 329–343. Springer, Heidelberg (2001)
13. Jin, Y., Branke, J.: Evolutionary Optimization in Uncertain Environments — a Survey. IEEE Transactions on Evolutionary Computation 9(3), 303–317 (2005)

14. Miller, B.L.: Noise, Sampling, and Efficient Genetic Algorithms. PhD thesis, University of Illinois, Urbana-Champaign (1997)
15. Miller, B.L., Goldberg, D.E.: Optimal Sampling for Genetic Algorithms. Intelligent Engineering Systems Through Artificial Neural Networks 6, 291–298 (1996)
16. Moriarty, D.E., Schultz, A.C., Grefenstette, J.J.: Evolutionary Algorithms for Reinforcement Learning. Journal of Artificial Intelligence Research 11, 241–276 (1999)
17. Di Pietro, A., While, L., Barone, L.: Applying Evolutionary Algorithms to Problems With Noisy, Time-Consuming Fitness Functions. In: Congress on Evolutionary Computation, pp. 1254–1261. IEEE, Los Alamitos (2004)
18. Sano, Y., Kita, H.: Optimization of Noisy Fitness Functions by Means of Genetic Algorithms Using History of Search With Test of Estimation. In: Congress on Evolutionary Computation, pp. 360–365. IEEE, Los Alamitos (2002)
19. Schmidt, C., Branke, J., Chick, S.E.: Integrating Techniques from Statistical Ranking into Evolutionary Algorithms. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 752–763. Springer, Heidelberg (2006)
20. Silver, E.A., Pyke, D.F., Peterson, R.: Inventory Management and Production Planning and Scheduling. John-Wiley and Sons, New York (1998)
21. Smalley, J.B., Minsker, B., Goldberg, D.E.: Risk-Based In Situ Bioremediation Design Using a Noisy Genetic Algorithm. Water Resour. Res. 36(10), 3043–3052 (2000)
22. Stagge, P.: Averaging Efficiently in the Presence of Noise. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 188–197. Springer, Heidelberg (1998)
23. Stroud, P.D.: Kalman-Extended Genetic Algorithm for Search in Nonstationary Environments with Noisy Fitness Functions. IEEE Transactions on Evolutionary Computation 5(1), 66–77 (2001)
24. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
25. Then, T.W., Chong, E.K.P.: Genetic Algorithms in Noisy Environments. In: 9th IEEE International Symposium on Intelligent Control, Columbus, Ohio, USA, pp. 225–230 (1994)
26. Whitley, D., Kauth, J.: GENITOR: A Different Genetic Algorithm. In: Rocky Mountain Conference on Artificial Intelligence, Denver, CO, USA, pp. 118–130 (1988)
27. Wu, J., Zheng, C., Chien, C.C., Zheng, L.: A Comparative Study of Monte Carlo Simple Genetic Algorithm and Noisy Genetic Algorithm for Cost-Effective Sampling Network Design Under Uncertainty. Advances in Water Resources 29, 899–911 (2006)

A Cultural Algorithm for POMDPs from Stochastic Inventory Control

S.D. Prestwich[1], S.A. Tarim[2], R. Rossi[1], and B. Hnich[3]

[1] Cork Constraint Computation Centre, Ireland
[2] Department of Management, Hacettepe University, Turkey
[3] Faculty of Computer Science, Izmir University of Economics, Turkey
s.prestwich@cs.ucc.ie, armagan.tarim@hacettepe.edu.tr,
r.rossi@4c.ucc.ie, brahim.hnich@ieu.edu.tr

Abstract. Reinforcement Learning algorithms such as SARSA with an eligibility trace, and Evolutionary Computation methods such as genetic algorithms, are competing approaches to solving Partially Observable Markov Decision Processes (POMDPs) which occur in many fields of Artificial Intelligence. A powerful form of evolutionary algorithm that has not previously been applied to POMDPs is the cultural algorithm, in which evolving agents share knowledge in a belief space that is used to guide their evolution. We describe a cultural algorithm for POMDPs that hybridises SARSA with a noisy genetic algorithm, and inherits the latter's convergence properties. Its belief space is a common set of state-action values that are updated during genetic exploration, and conversely used to modify chromosomes. We use it to solve problems from stochastic inventory control by finding memoryless policies for nondeterministic POMDPs. Neither SARSA nor the genetic algorithm dominates the other on these problems, but the cultural algorithm outperforms the genetic algorithm, and on highly non-Markovian instances also outperforms SARSA.

1 Introduction

Reinforcement Learning and Evolutionary Computation are competing approaches to solving Partially Observable Markov Decision Processes, which occur in many fields of Artificial Intelligence. In this paper we describe a new hybrid of the two approaches, and apply it to problems in stochastic inventory control. The remainder of this section provides some necessary background information. Section 2 describes our general approach, an instantiation, and convergence results. Section 3 describes and models the problems. Section 4 presents experimental results. Section 5 concludes the paper.

1.1 POMDPs

Markov Decision Processes (MDPs) can model sequential decision-making in situations where outcomes are partly random and partly under the control of the agent. The states of an MDP possess the *Markov property*: if the current state of the MDP at time t is known, transitions to a new state at time $t+1$ are independent of all previous states. MDPs can be solved in polynomial time (in the size of their state-space) by modelling

M.J. Blesa et al. (Eds.): HM 2008, LNCS 5296, pp. 16–28, 2008.

them as linear programs, though the order of the polynomials is large enough to make them difficult to solve in practice [14]. If the Markov property is removed then we obtain a Partially Observable Markov Decision Process (POMDP) which in general is computationally intractable. This situation arises in many applications and can be caused by partial knowledge: for example a robot must often navigate using only partial knowledge of its environment. Machine maintenance and planning under uncertainty can also be modelled as POMDPs.

Formally, a POMDP is a tuple $\langle S, A, T, R, O, \Omega \rangle$ where S is a set of states, A a set of actions, Ω a set of observations, $R : S \times A \rightarrow \Re$ a reward function, $T : S \times A \rightarrow \Pi(S)$ a transition function, and $\Pi(\cdot)$ represents the set of discrete probability distributions over a finite set. In each time period t the environment is in some state $s \in S$ and the agent takes an action $a \in A$, which causes a transition to state s' with probability $P(s'|s,a)$, yielding an immediate reward given by R and having an effect on the environment given by T. The agent's decision are based on its observations given by $O : S \times A \rightarrow \Pi(\Omega)$.

When solving a POMDP the aim is to find a *policy*: a strategy for selecting actions based on observations that maximises a function of the rewards, for example the total reward. A policy is a function that maps the agent's observation history and its current internal state to an action. A policy may also be *deterministic* or *probabilistic*: a deterministic policy consistently chooses the same action when faced with the same information, while a probabilistic policy might not. A *memoryless* (or *reactive*) policy returns an action based solely on the current observation. The problem of finding a memoryless policy for a POMDP is NP-complete and exact algorithms are very inefficient [12] but there are good inexact methods, some of which we now describe.

1.2 Reinforcement Learning Methods

Temporal difference learning algorithms such as Q-Learning [32] and SARSA [25] from Reinforcement Learning (RL) are a standard way of finding good policies. While performing Monte Carlo-like simulations they compute a *state-action value* function $Q : S \times A \rightarrow \Re$ which estimates the expected total reward for taking a given action from a given state. (Some RL algorithms compute instead a *state value* function $V : S \rightarrow \Re$.)

The SARSA algorithm is shown in Figure 1. An *episode* is a sequence of states and actions with a first and last state that occur naturally in the problem. On taking an action that leads to a new state, the value of the new state is "backed up" to the state just left (see line 8) by a process called *bootstrapping*. This propagates the effects of later actions to earlier states and is a strength of RL algorithms. (The value γ is a *discounting factor* often used for non-episodic tasks that is not relevant for our application below: we set $\gamma = 1$.) A common *behaviour policy* is ϵ-greedy action selection: with probability ϵ choose a random action, otherwise with probability $1 - \epsilon$ choose the action with highest $Q(s,a)$ value. After a number of episodes the state-action values $Q(s,a)$ are fixed and (if the algorithm converged correctly) describe an optimum policy: from each state choose the action with highest $Q(s,a)$ value. The name SARSA derives from the tuple (s, a, r, s', a').

RL algorithms have convergence proofs that rely on the Markov property but for some non-Markovian applications they still perform well, especially when augmented with an *eligibility trace* [10,16] that effectively hybridises them with a Monte Carlo

```
initialise the $Q(s,a)$ arbitrarily
repeat for each episode
(  $s \leftarrow$ initial state
   choose action $a$ from $s$ using a behaviour policy
   repeat for each step of the episode
   (  take action $a$ and observe $r, s'$
      choose action $a'$ from $s'$ using a behaviour policy
      $Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma Q(s',a') - Q(s,a)]$
      $s \leftarrow s'$, $a \leftarrow a'$
   )
)
```

Fig. 1. The SARSA algorithm

algorithm. We will use a well-known example of such an algorithm: SARSA(λ) [25]. When the parameter λ is 0 SARSA(λ) is equivalent to SARSA, when it is 1 it is equivalent to a Monte Carlo algorithm, and with an intermediate value it is a hybrid and often gives better results than either. Setting $\lambda > 0$ boosts bootstrapping by causing values to be backed up to states before the previous one. (See [30] for a discussion of eligibility traces, their implementation, and the relationship with Monte Carlo algorithms.) There are other more complex RL algorithms (see [13] for example) and it is possible to configure SARSA(λ) differently (for example by using *softmax* action selection instead of ϵ-greedy, and different values of α for each state-action value [30]), but we take SARSA(λ) as a representative of RL approaches to solving POMPDs. (In fact it usually outperforms two versions of Q-learning with eligibility trace — see [30] page 184.)

1.3 Evolutionary Computation Methods

An alternative approach to POMDPs is the use of Evolutionary Computation (EC) algorithms such as Genetic Algorithms (GAs), which sometimes beat RL algorithms on highly non-Markovian problems [3,19]. We shall use the most obvious EC model of POMDPs, called a *table-based representation* [19]: each chromosome represents a policy, each gene a state, and each allele (gene value) an action.

The GA we shall use is based on GENITOR [33] but without the refinements of some versions, such as genetic control of the crossover probability. This is a *steady-state* GA that, at each iteration, selects two parent chromosomes, breeds a single offspring, evaluates it, and uses it to replace the least-fit member of the population. Steady-state GAs are an alternative to *generational* GAs that generate an entire generation at each iteration, which replaces the current generation. Maintaining the best chromosomes found so far is an *elitist* strategy that pays off on many problems. Parent selection is random because of the strong selection pressure imposed by replacing the least-fit member. We use standard *uniform crossover* (each offspring gene receives an allele from the corresponding gene in a randomly-chosen parent) applied with a crossover probability p_c: if it is not applied then a single parent is selected and mutated, and the resulting chromosome replaces the least-fit member of the population. Mutation is applied to a chromosome once with probability p_m, twice with probability p_m^2, three times with probability p_m^3, and so on. The population size is P and the initial population contains random alleles.

Nondeterminism in the POMDP causes noise in the GA's fitness function. To handle this noise we adopt the common approach of averaging the fitness over a number of samples S. This technique has been used many times in *Noisy Genetic Algorithms* (NGAs) [4,6,17,18]. NGAs are usually generational and [1] show that elitist algorithms (such as GENITOR) can systematically overvalue chromosomes, but such algorithms have been successful when applied to noisy problems [29]. We choose GENITOR for its simplicity.

1.4 Hybrid Methods

Several approaches can be seen as hybrids of EC and RL. *Learning Classifier Systems* [8] use EC to adapt their representation of the RL problem. They apply RL via the EC fitness function. *Population-Based Reinforcement Learning* [11] uses RL techniques to improve chromosomes, as in a memetic algorithm. The paper is an outline only, and no details are given on how RL values are used, nor are experimental results provided. *GAQ-Learning* [15] uses Q-Learning once only in a preprocessing phase, to generate $Q(s, a)$ values. A memetic algorithm is then executed using the $Q(s, a)$ values to evaluate the chromosomes. *Q-Decomposition* [26] combines several RL agents, each with its own rewards, state-action values and RL algorithm. An arbitrator combines their recommendations, maximising the sum of the rewards for each action. It is designed for distributed tasks that are not necessarily POMPDs. Global convergence is guaranteed if the RL algorithm is SARSA but not if it is Q-Learning. In [9] a GA and RL are combined to solve a robot navigation problem. The greedy policy is applied for some time (until the robot encounters difficulty); next the GA population is evaluated, and the fittest chromosome used to update the state-action values by performing several RL iterations; next a new population is generated in a standard way, except that the state-action values are used probabilistically to alter chromosomes; then the process repeats. Several other techniques are used, some specific to robotics applications, but here we consider only the RL-EC hybrid aspects.

2 A Cultural Approach to POMDPs

A powerful form of EC is the *cultural algorithm* (CA) [21], in which agents share knowledge in a *belief space* to form a consensus. (The *belief space* of a CA is distinct from the *belief space* of a POMDP, which we do not refer to in this paper.) These hybrids of EC and Machine Learning have been shown to converge more quickly than EC alone on several applications. CAs were developed as a complement to the purely genetic bias of EC. They are based on concepts used in sociology and archaeology to model cultural evolution. By pooling knowledge gained by individuals in a body of cultural knowledge, or belief space, convergence rates can sometimes be improved. A CA has an *acceptance function* that determines which individuals in the population are allowed to *adjust* the belief space. The beliefs are conversely used to *influence* the evolution of the population. See [22] for a survey of CA applications, techniques and belief spaces. They have been applied to constrained optimisation [5], multiobjective optimisation [2], scheduling [24] and robot soccer [23], but to the best of our knowledge they have not been applied to POMDPs, nor have they utilised RL.

2.1 Cultural Reinforcement Learning

We propose a new cultural hybrid of reinforcement learning and evolutionary computation for solving POMDPs called *CUltural Reinforcement Learning* (CURL). The CURL approach is straightforward and can be applied to different RL and EC algorithms. A single set of RL state-action values $Q(s, a)$ is initialised as in the RL algorithm, and the population is initialised as in the EC algorithm. The EC algorithm is then executed as usual, except that each new chromosome is altered by, and used to alter, the $Q(s, a)$, which constitute the CA belief space. On generating a new chromosome we replace, with some probability p_l, each allele by the corresponding greedy action given by the modified $Q(s, a)$ values. Setting $p_l = 0$ prevents any learning, and CURL reduces to the EC algorithm, while $p_l = 1$ always updates a gene to the corresponding $Q(s, a)$ value, and CURL reduces to SARSA(λ) without exploration. We then treat the modified chromosome as usual by the EC algorithm: typically, fitness evaluation and placement into the population. During fitness evaluation the $Q(s, a)$ are updated by bootstrapping as usual in the RL algorithm, but the policy followed is that specified by the modified chromosome. Thus in CURL, as in several other CAs [22], *all* chromosomes are allowed to adjust the belief space. There is no ϵ parameter in CURL because exploratory moves are provided by EC.

We may use a steady-state or generational GA, or other form of EC algorithm, and we may use one of the Q-Learning or Q(λ) algorithms to update the $Q(s, a)$, but in this paper we use the GENITOR-based NGA and SARSA(λ). The resulting algorithm is outlined in Figure 2, in which SARSA(λ,α,O) denotes a SARSA(λ) episode with a given value of the α parameter, following the policy specified by chromosome O while updating the $Q(s, a)$ as usual. As in NGA the population in randomly initialised and fitness is evaluated using S samples. Note that for a deterministic POMDP only one sample is needed to obtain the fitness of a chromosome, so we can set $S = 1$ to obtain a CURL hybrid of SARSA(λ) and GENITOR.

```
CURL($S$,$P$,$p_c$,$p_m$,$\alpha$,$\lambda$,$p_l$):
(  create population of size $P$
   evaluate population using $S$ samples
   initialise the $Q(s,a)$
   while not(termination condition)
   (  generate an offspring O using $p_c$,$p_m$
      update O using $p_l$ and the $Q(s,a)$
      call SARSA($\lambda$,$\alpha$,O) $S$ times to estimate O fitness
          and bootstrap the $Q(s,a)$
      replace least-fit chromosome by O
   )
   output fittest chromosome
)
```

Fig. 2. CURL instantiation

2.2 Convergence

For POMDPS, unlike MDPs, suboptimal policies can form local optima in policy space [20]. This motivates the use of global search techniques such as EC, which are less likely to become trapped in local optima, and a hybrid such as CURL uses EC to directly explore policy space. CURL also uses bootstrapping to perform small changes to the policy by hill-climbing on the $Q(s, a)$ values. Hill-climbing has often been combined with GAs to form *memetic algorithms* with faster convergence than a pure GA, and this was a motivation for CURL's design. However, if bootstrapping is used then optimal policies are not necessarily stable: that is, an optimal policy might not attract the algorithm [20]. Thus a hybrid might not be able to find an optimal policy even if it escapes all local optima. The possible instability of optimal policies does not necessarily render such hybrids useless, because there might be optimal or near-optimal policies that *are* stable, but convergence is a very desirable property.

Fortunately, it is easy to show that if $p_l < 1$ and the underlying EC algorithm is convergent then so is CURL: if $p_l < 1$ then there is a non-zero probability that no allele is modified by the $Q(s, a)$, in which case CURL behaves exactly like the EC algorithm. This is not true of all hybrids (for example [9]). The GA used in the CURL instantiation is convergent (to within some accuracy depending on the number of samples used), because every gene in a new chromosome can potentially be mutated to an arbitrary allele. Therefore the CURL instantiation is convergent.

2.3 Note

Ideally, we should now evaluate CURL on standard POMDPs from the literature, but we shall postpone this for future work. The work in this paper is motivated by the need to solve large, complex inventory control problems that do not succumb to more traditional methods. In fact we know of no method in the inventory control literature that can optimally solve our problem in a reasonable time (at least, the constrained form of the problem: see below). We shall therefore test CURL on POMDPs from stochastic inventory control. We believe that the problem we tackle has not previously been considered as a POMDP, but we shall show that it is one.

3 POMDPs from Stochastic Inventory Control

The problem is as follows. We have a planning horizon of N periods and a demand for each period $t \in \{1, \ldots, N\}$, which is a random variable with a given probability density function; we assume that these distributions are normal. Demands occur instantaneously at the beginning of each time period and are *non-stationary* (can vary from period to period), and demands in different periods are independent. A fixed delivery cost a is incurred for each order (even for an order quantity of zero), a linear holding cost h is incurred for each product unit carried in stock from one period to the next, and a linear stockout cost s is incurred for each period in which the net inventory is negative (it is not possible to sell back excess items to the vendor at the end of a period). The aim is to find a replenishment plan that minimizes the expected total cost over the planning

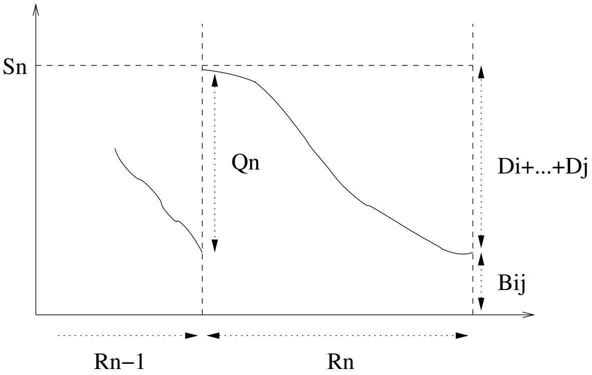


Fig. 3. The (R, S) policy

horizon. Different inventory control policies can be adopted to cope with this and other problems. A policy states the rules used to decide when orders are to be placed and how to compute the replenishment lot-size for each order.

3.1 Replenishment Cycle Policy

One possibility is the *replenishment cycle policy* (R, S). Under the non-stationary demand assumption this policy takes the form (R^n, S^n) where R^n denotes the length of the n^{th} replenishment cycle and S^n the order-up-to-level for replenishment. In this policy a strategy is adopted under which the actual order quantity for replenishment cycle n is determined only after the demand in former periods has been realized. The order quantity is computed as the amount of stock required to raise the closing inventory level of replenishment cycle $n-1$ up to level S^n. To provide a solution we must populate both the sets R^n and S^n for $n = \{1, \ldots, N\}$. The (R, S) policy yields plans of higher cost than the optimum, but it reduces planning instability [7] and is particularly appealing when items are ordered from the same supplier or require resource sharing [27]. Figure 3 illustrates the (R, S) policy. R^n denotes the set of periods covered by the nth replenishment cycle; S^n is the order-up-to-level for this cycle; Q_n is the expected order quantity; $D_i + \ldots + D_j$ is the expected demand; B_{ij} is the buffer stock required to guarantee service level α.

Though both RL and EC have been applied to a variety of inventory control problems, some of them POMDPs [31], neither seems to have been applied to this important problem. There are more efficient algorithms which are guaranteed to yield optimal policies (under reasonable simplifying assumptions) so RL and EC would not be applied to precisely this problem in practice. However, if we complicate the problem in simple but realistic ways, for example by adding order capacity constraints or dropping the assumption of independent demands, then these efficient algorithms become unusable. In contrast, RL and EC algorithms can be used almost without modification. Thus the problem is useful as a representative of a family of more complex problems.

Note that the inventory control term *policy* refers to the *form* of plan that we search for (such as the (R, S) policy), whereas a POMDP policy is a *concrete* plan (such as the (R, S) policy with given (R^n, S^n) values). We use the term in both senses but the meaning should be clear from the context.

3.2 POMDP Model

The replenishment cycle policy can be modelled as a POMDP as follows. Define a *state* to be the period n, an *action* to be either the choice of an order-up-to level or the lack of an order (denoted here by a special action N), and a *reward* r_n to be minus the total cost incurred in period n. The rewards are *undiscounted* (do not decay with time), the problem is *episodic* (has well-defined start and end states), the POMDP is *nondeterministic* (the rewards are randomised), and its solution is a policy that is *deterministic* and *memoryless* (actions are taken solely on the basis of the agent's current observations). This problem is non-Markovian but has an underlying MDP. Suppose we include the current stock level (suitably discretised or approximated) in the state. We then have the Markov property: the current stock level and period is all the information we need to make an optimal decision. But the (R, S) policy does not make optimal decisions: instead it fixes order-up-to levels independently of the stock level.

The problem is slightly unusual as a POMDP for two reasons. Firstly, all actions from a state n lead to the same state $n + 1$ (though they have different expected rewards): different actions usually lead to different states. Secondly, many applications are non-Markovian because of limited available information, but here we *choose* to make it non-Markovian by discarding information for an application-specific reason: to reduce planning instability. Neither feature invalidates the POMDP view of the problem, and we believe that instances of the problem make ideal benchmarks for RL and EC methods: they are easy to describe and implement, hard to solve optimally, have practical importance, and it turns out that neither type of algorithm dominates the other.

There exist techniques for improving the performance of RL algorithms on POMDPs, in particular the use of forms of memory such as a belief state or a recurrent neural network. But such methods are inapplicable to our problem because the policy would not then be memoryless, and would therefore not yield a replenishment cycle policy. The same argument applies to stochastic policies, which can be arbitrarily more efficient than deterministic policies [28]: for our inventory problem we require a deterministic policy. Thus some powerful RL techniques are inapplicable to our problem.

4 Experiments

We compare SARSA(λ), the NGA and CURL on five benchmark problems. The instances are shown in Table 1 together with their optimal policies. Each policy is specified by its planning horizon length R and its order-up-to-level S, and the expected cost of the policy per period is also shown, which can be multiplied by the number of periods to obtain the expected total cost of the policy. For example instance (3) has the optimal policy $[159, N, N, 159, N, N, 159, \ldots]$. However, the policy is only optimal if the total number of periods is a multiple of R, and we choose 120 periods as a common multiple of $R \in \{1, 2, 3, 4, 5\}$. This number is also chosen for hardness: none of the three

Table 1. Instances and their optimum policies

				demand	demand			cost/	cost/120
#	h	s	a	mean	std dev	R	S	period	periods
(1)	1	10	50	50	10	1	63	68	8160
(2)	1	10	100	50	10	2	112	94	11280
(3)	1	10	200	50	10	3	159	138	16560
(4)	1	10	400	50	10	4	200	196	23520
(5)	1	10	800	50	10	5	253	279	33480

algorithms find optimal policies within 10^8 simulations (a Mixed Integer Programming approach also failed given several hours). We varied only the a parameter, which was sufficient to obtain different R values (and different results: see below). We allow 29 different order-up-to levels at each period, linearly spaced in the range 0–280 at intervals of 10, plus the N no-order option, so from each state we must choose between 30 possible actions. This range of order-up-to levels includes the levels in the optimum policies for all five instances. Of course if none of the levels coincides with some order-up-to-level in an optimal policy then this prevents us from finding the exact optimum policy. But even choosing levels carefully so that the exact values are reachable does not lead to optimal policies using the three algorithms.

As mentioned above, this problem can be solved in polynomial time because of its special form, which is how we know the optimum policies. We therefore also generate five additional instances (1c,2c,3c,4c,5c) by adding an order capacity constraint to instances (1,2,3,4,5) respectively, simply by choosing an upper bound below the level necessary for the optimum policy. For each instance the 30 levels are linearly spaced between 0 and $\lfloor 0.8S \rfloor$ (respectively 54, 89, 127, 156 and 223). This problem is NP-hard and we know of no method that can solve it to optimality in a reasonable time. We therefore do not know the optimum policies for these instances, only that their costs are at least as high as those without the order constraints.

We tailored NGA and CURL for our application by modifying the mutation operator: because of the special nature of the N action we mutate a gene to N with 50% probability, otherwise to a random order-up-to level. This biased mutation improves NGA and CURL performance. We also tailored SARSA and CURL for our application. Firstly, we initialise all state-action values to the optimistic value of 0, because the use of optimistic initial values encourages early exploration [30]. Secondly, we experimented with different methods for varying ϵ, which may decay with time using different methods. [3,16] decrease ϵ linearly from 0.2 to 0.0 until some point in time, then fix it at 0.0 for the remainder. [30] recommend varying ϵ inversely with time or the number of episodes. We found significantly better results using the latter method, under the following scheme: $\epsilon = 1/(1 + \epsilon' e)$ where e is the number of episodes so far and ϵ' is a fixed coefficient chosen by the user. For the final 1% of the run we set $\epsilon = 0$ so that the final policy cost reflects that of the greedy policy (after setting ϵ to 0 we found little change in the policy, so we did not devote more time to this purely greedy policy).

Each of the three algorithms has several parameters to be tuned by the user. To simulate a realistic scenario in which we must tune an algorithm once then use it many times,

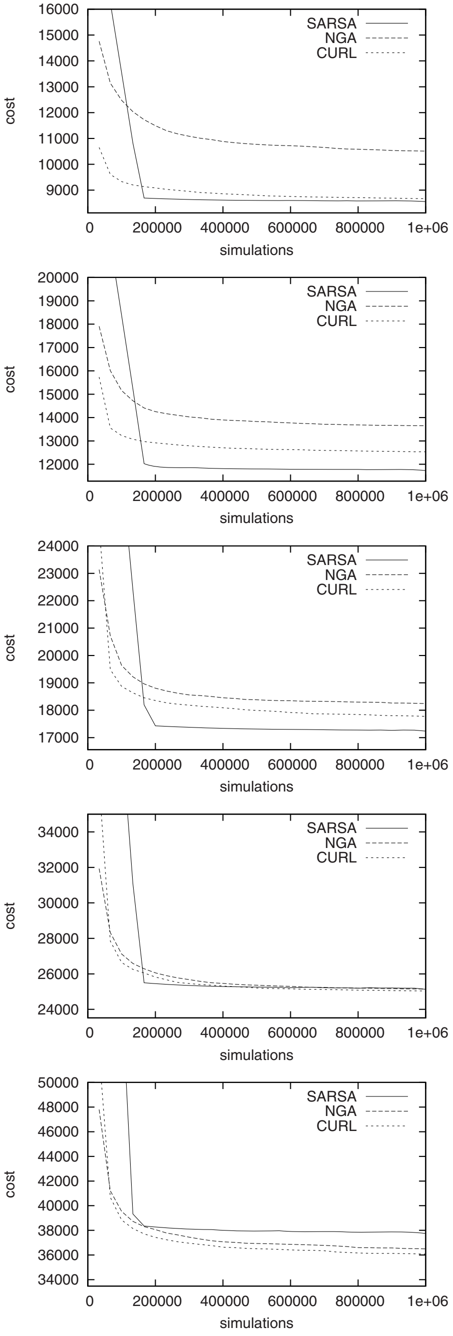


Fig. 4. Instances (1,2,3,4,5)

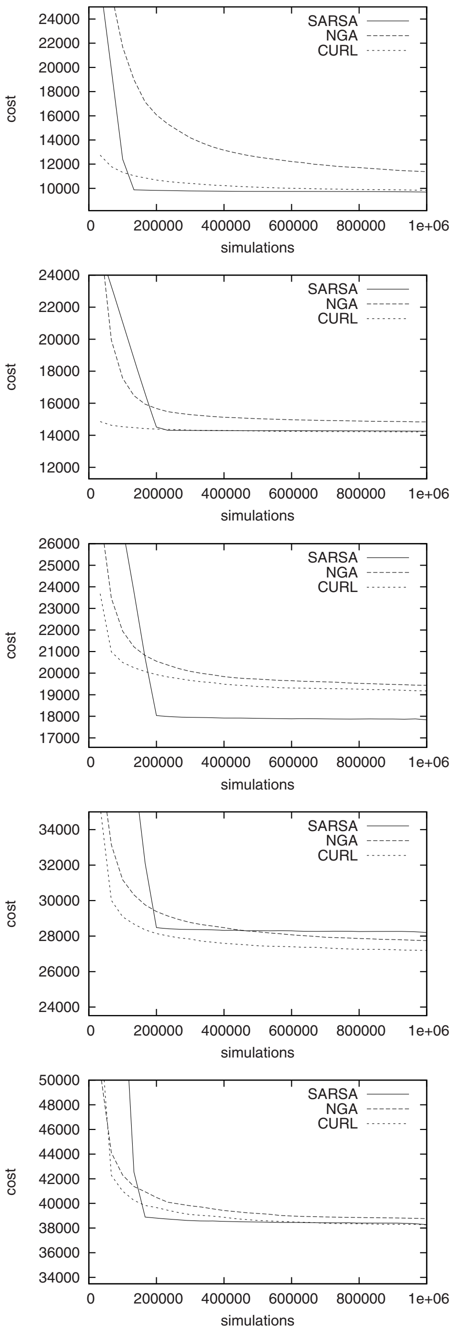


Fig. 5. Instances (1c,2c,3c,4c,5c)

we tuned all three to a single instance: the middle instance (3) without an order capacity constraint. For SARSA(λ) we tuned $\epsilon', \alpha, \lambda$ by the common method of hill-climbing in parameter space to optimise the final cost of the evolved policy, restricted to λ values $\{0.0, 0.1, \ldots, 0.9, 1.0\}$ and ϵ', α values $\{0.1, 0.03, 0.01, 0.003, \ldots\}$. This process led to $\alpha = 0.003$, $\epsilon' = 0.001$ and $\lambda = 0.7$. We chose NGA settings $p_c = p_m = 0.5$ and $P = S = 30$ for each instance: performance was robust with respect to these parameters, as reported by many GA researchers. To tune CURL we fixed the GA parameters as above, set $\lambda = 0$, and applied hill-climbing to the remaining CURL parameters, restricted to $p_l \in \{1.0, 0.3, 0.1, 0.03, \ldots\}$, to obtain $\alpha = 0.1$, $p_l = 0.3$. Using $\lambda > 1$ did not make a significant difference to performance (though it necessitated different values for α and p_l): it might be necessary for deterministic problems in which we do not evaluate chromosome fitness over several simulations, but here we have $S = 30$ simulations per chromosome in which to perform bootstrapping so we use the more efficient SARSA(0).

Figures 4 and 5 plots the performances of the algorithms on the instances. The SARSA(λ) cost is an exponentially-smoothed on-policy cost (the policy actually followed by the algorithm during learning). The NGA and CURL costs are those of the fittest chromosome. All graph points are means over 20 runs. We use the number of SARSA(λ) episodes or GA simulations as a proxy for time, and allow each algorithm 10^6 episodes or simulations. This slightly biases the results in favour of SARSA(λ): one of its episodes takes approximately three times longer than a simulation because of its eligibility trace. But there may be faster implementations of SARSA(λ) than ours so we use this implementation-independent metric.

The graphs show that neither SARSA(λ) nor NGA dominates the other over all instances, though SARSA(λ) is generally better (this might be caused by our choice of instances). However, CURL is uniformly better than NGA, and therefore sometimes better than SARSA(λ) also. Previous research into EC and RL on POMDPS has shown that neither dominates over all problems, but that EC is better on highly non-Markovian problems, so we assume that the problems in which NGA beats SARSA(λ) are highly non-Markovian. This implies that CURL is a very promising approach to such POMDPs, though further experiments are needed to confirm this pattern.

It might be suspected that the biased mutation technique unfairly aids NGA and CURL: but adding this technique to SARSA(λ) worsens its performance. Unlike RL algorithms, EC algorithms can benefit from application-specific mutation and recombination operators, and these can also be used in CURL. The current CURL implementation uses a simple table-based representation of the POMDP, which is often the worst choice [19], so we believe that there is a great deal of room for improvement.

5 Conclusion

Reinforcement Learning (RL) and Evolutionary Computation (EC) are competing approaches to solving POMDPs. We presented a new Cultural Algorithm (CA) schema called CURL that hybridises RL and EC, and inherits EC convergence properties. We also described POMDPs from stochastic inventory theory on which neither RL nor EC dominates the other. In experiments a CURL instantiation outperforms the EC algorithm, and on highly non-Markovian instances it also outperforms the RL algorithm. We believe that CURL is a promising approach to solving POMDPs, combining EC and RL algorithms with little modification.

This work is part of a series of studies in solving inventory problems using systematic and randomised methods. In future work we intend to develop CURL for more complex inventory problems, and for more standard POMDPs from the Artificial Intelligence literature.

Acknowledgement. This material is based in part upon works supported by the Science Foundation Ireland under Grant No. 05/IN/I886, and under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain-Driven Research (CTVR). S. A. Tarim and B. Hnich are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027.

References

1. Arnold, D.V., Beyer, H.-G.: Local Performance of the (1+1)-ES in a Noisy Environment. IEEE Trans. Evolutionary Computation 6(1), 30–41 (2002)
2. Becerra, R.L., Coello, C.A.C.: A Cultural Algorithm with Differential Evolution to Solve Constrained Optimization Problems. In: Lemaître, C., Reyes, C.A., González, J.A. (eds.) IBERAMIA 2004. LNCS (LNAI), vol. 3315, pp. 881–890. Springer, Heidelberg (2004)
3. de Croon, G., van Dartel, M.F., Postma, E.O.: Evolutionary Learning Outperforms Reinforcement Learning on Non-Markovian Tasks. In: Workshop on Memory and Learning Mechanisms in Autonomous Robots, 8th European Conference on Artificial Life, Canterbury, Kent, UK (2005)
4. Fitzpatrick, J.M., Grefenstette, J.J.: Genetic Algorithms in Noisy Environments. Machine Learning 3, 101–120 (1988)
5. Gao, F., Cui, G., Liu, H.: Integration of Genetic Algorithm and Cultural Algorithms for Constrained Optimization. In: King, I., Wang, J., Chan, L.-W., Wang, D. (eds.) ICONIP 2006. LNCS, vol. 4234, pp. 817–825. Springer, Heidelberg (2006)
6. Gopalakrishnan, G., Minsker, B.S., Goldberg, D.: Optimal Sampling in a Noisy Genetic Algorithm for Risk-Based Remediation Design. In: World Water and Environmental Resources Congress. ASCE (2001)
7. Heisig, G.: Comparison of (s,S) and (s,nQ) Inventory Control Rules with Respect to Planning Stability. International Journal of Production Economics 73, 59–82 (2001)
8. Holland, J.H.: Adaptation. In: Progress in Theoretical Biology IV, pp. 263–293. Academic Press, London (1976)
9. Iglesias, R., Rodriguez, M., Sánchez, M., Pereira, E., Regueiro, C.V.: Improving Reinforcement Learning Through a Better Exploration Strategy and an Adjustable Representation of the Environment. In: 3rd European Conference on Mobile Robots (2007)
10. Jaakkola, T., Singh, S.P., Jordan, M.I.: Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems. In: Advances in Neural Information Processing Systems 6. MIT Press, Cambridge (1994)
11. Kovacs, T., Reynolds, S.I.: A Proposal for Population-Based Reinforcement Learning. Technical report CSTR-03-001, Department of Computer Science, University of Bristol (2003)
12. Littman, M.: Memoryless Policies: Theoretical Limitations and Practical Results. In: 3rd Conference on Simulation of Adaptive Behavior (1994)

13. Littman, M.L., Cassandra, A.R., Kaelbling, L.P.: Learning Policies for Partially Observable Environments: Scaling Up. In: International Conference on Machine Learning (1995)
14. Littman, M., Dean, T., Kaelbling, L.: On the Complexity of Solving Markov Decision Problems. In: 11th Conference on Uncertainty in Artificial Intelligence, pp. 394–402 (1995)
15. Liu, H., Hong, B., Shi, D., Ng, G.S.: On Partially Observable Markov Decision Processes Using Genetic Algorithm Based Q-Learning. In: Advances in Neural Networks, pp. 248–252. Watam Press (2007)
16. Loch, J., Singh, S.P.: Using Eligibility Traces to Find the Best Memoryless Policy in Partially Observable Markov Decision Processes. In: 15th International Conference on Machine Learning, pp. 323–331 (1998)
17. Miller, B.L.: Noise, Sampling, and Efficient Genetic Algorithms. PhD thesis, University of Illinois, Urbana-Champaign (1997)
18. Miller, B.L., Goldberg, D.E.: Optimal Sampling for Genetic Algorithms. In: Intelligent Engineering Systems Through Artificial Neural Networks, vol. 6, pp. 291–298. ASME Press (1996)
19. Moriarty, D.E., Schultz, A.C., Grefenstette, J.J.: Evolutionary Algorithms for Reinforcement Learning. Journal of Artificial Intelligence Research 11, 241–276 (1999)
20. Penrith, M.D., McGarity, M.J.: An Analysis of Direct Reinforcement Learning in non-Markovian Domains. In: 15th International Conference on Machine Learning, pp. 421–429. Morgan Kaufmann, San Francisco (1998)
21. Reynolds, R.G.: An Introduction to Cultural Algorithms. In: 3rd Annual Conference on Evolutionary Programming, pp. 131–139. World Scientific Publishing, Singapore (1994)
22. Reynolds, R.G.: Cultural Algorithms: Theory and Applications. New Ideas in Optimization, pp. 367–377. McGraw-Hill, New York (1999)
23. Reynolds, R.G., Chung, C.: A Cultural Algorithm Framework to Evolve Multiagent Cooperation With Evolutionary Programming. In: Angeline, P.J., McDonnell, J.R., Reynolds, R.G., Eberhart, R. (eds.) EP 1997. LNCS, vol. 1213, pp. 323–333. Springer, Heidelberg (2006)
24. Rivera, D.C., Becerra, R.L., Coello, C.A.C.: Cultural Algorithms, an Alternative Heuristic to Solve the Job Shop Scheduling Problem. Engineering Optimization 39(1), 69–85 (2007)
25. Rummery, G.A., Niranjan, M.: On-line Q-learning Using Connectionist Systems. Technical report CUED/F-INFENG/TR 166, Cambridge University (1994)
26. Russell, S.J., Zimdars, A.: Q-Decomposition for Reinforcement Learning Agents. In: 20th International Conference on Machine Learning, pp. 656–663. AAAI Press, Menlo Park (2003)
27. Silver, E.A., Pyke, D.F., Peterson, R.: Inventory Management and Production Planning and Scheduling. John-Wiley and Sons, New York (1998)
28. Singh, S., Jaakkola, T., Jordan, M.: Learning Without State-Estimation in Partially Observable Markovian Decision Processes. In: 11th International Conference on Machine Learning, pp. 284–292. Morgan Kaufmann, San Francisco (1994)
29. Stroud, P.D.: Kalman-Extended Genetic Algorithm for Search in Nonstationary Environments with Noisy Fitness Functions. IEEE Transactions on Evolutionary Computation 5(1), 66–77 (2001)
30. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
31. Treharne, J.T., Sox, C.R.: Adaptive Inventory Control for Nonstationary Demand and Partial Information. Management Science 48(5), 607–624 (2002)
32. Watkins, C.J.C.H.: Learning From Delayed Rewards. PhD thesis, Cambridge University (1989)
33. Whitley, D., Kauth, J.: GENITOR: A Different Genetic Algorithm. In: Rocky Mountain Conference on Artificial Intelligence, Denver, CO, USA, pp. 118–130 (1988)

THE COST OF USING STATIONARY INVENTORY POLICIES WHEN DEMAND IS NON-STATIONARY

Non-stationary stochastic demands are very common in industrial settings with seasonal patterns, trends, business cycles, and limited-life items. In such cases, the optimal inventory control policies are also non-stationary. However, due to high computational complexity, non-stationary inventory policies are not usually preferred in real life applications. In this paper, we investigate the cost of using the optimal stationary policy as an approximation for the optimal non-stationary one. We show that, using stationary policies can be very expensive depending on the magnitude of demand variability and we provide some insight on cases where stationary policies provide good approximations to non-stationary policies.

1. Introduction

Stochastic inventory control systems have been studied extensively under various assumptions on demand. Nevertheless, the literature reflects a clear dichotomy between inventory models with stationary and non-stationary demands. The former assumes a steady demand process, whereas the latter assumes a demand process that varies in time. Non-stationary demand is very common in various industrial settings with seasonal patterns, trends, business cycles, and limited-life items. Furthermore, as product life cycles are becoming shorter, demand which evolves over product life cycles never follow stationary patterns (Graves and Willems, 2008).

One major theme in the continuing development of inventory theory is the incorporation of more realistic demand assumptions into inventory models. Consequently, one would expect increasing number of studies concerned with non-stationary inventory models. However, there is a limited literature assuming non-stationary demand, whereas it is vast for stationary demand. A search on the ISI Web of Knowledge since the year 2000 with *stationary* and *inventory* keywords gives 221 published papers, whilst this figure is only 29 for *non-stationary* and *inventory.* Furthermore, there is a large number of papers assuming stationary demand without using *stationary* as a keyword. This disparity is mainly due to the ill structure of non-stationary problems from a theoretical point of view and the complexity inherent in non-stationary models from a computational point of view. Silver et.al.

(1998) point out that non-stationary demand is too complicated for routine use in practice. Furthermore, as Kurawalwala and Matsuo (1996) stated, the unique characteristics of non-stationary demand preclude the use of traditional forecasting methods not designed for this environment and raise a need for tailor-made forecasting methods. Consequently, stationary policies have always been preferred to non-stationary policies in many real life applications for the sake of their relative simplicity.

In spite of all the above mentioned issues related to non-stationary inventory policies, when demand is non-stationary, a stationary policy is an approximation to the optimal non-stationary one, and hence, is sub-optimal with respect to total expected cost. To the best of authors' knowledge, no work has been done that can be used as a guideline to compute the cost of using stationary policies when demand is non-stationary. This research investigates the magnitude of this sub-optimality under various settings. We establish our analysis by using the (s,S) inventory control policy. (s,S) policy is proven to be optimal both in stationary and non-stationary demand cases, and therefore, constitutes an inherent frame of reference. Our contribution is two-fold. First, we show that using stationary policies can be very expensive depending on the extent of demand variability as well as other factors. Secondly, we provide some insight on cases where stationary models provide good approximations to non-stationary models.

In the remainder of this section, we concisely review related literature. In Section 2, we present the key assumptions of the inventory problem considered. In Section 3 we present algorithms used to compute the stationary and the non-stationary (s,S) policies. In Section 3, we present the experimental design and computational results. Finally, in Section 4, we draw general conclusions and provide some managerial insights.

Most of the research in inventory literature assumes either a stationary or a non-stationary demand, and develop models and policies accordingly. Therefore, it is difficult to refer to any research addressing the cost performance of stationary policies when demand is non-stationary. However, we believe that it is necessary to briefly discuss the key literature in order to ease the exposition of the remaining sections.

One of the most exciting developments in the inventory theory is Scarf's (1960) proof of the optimality of (*s*,*S*) policies. (s,S) policies are characterized by two critical numbers s_n and S_n for each period n, such that, the inventory is replenished up to a target level S_n whenever the initial inventory position is lower than (or equal to) a re-order level s_n. Scarf (1960) showed the optimal value function satisfies a condition, which he called K-convexity, and demonstrated a procedure for establishing the optimal policy parameters in a backward recursion. Nevertheless, there was no known way of computing the critical numbers at that time (Karlin, 1960). Following Scarf (1960), Iglehart (1963) demonstrated the optimality of (*s,S*) policies in infinite horizon inventory problems with stationary demand. He showed that optimal policy parameters converge to two numbers s and S in this case. Iglehart's work has been followed by a large number of researchers (see e.g. Veinott and Wagner, 1965; Archibald and Silver, 1978; Stidham, 1977; Sahin, 1982,1983; Federgruen and Zipkin, 1984; Zheng and Federgruen, 1991) aiming at efficiently computing optimal policy parameters using the stationary analysis approach. However, not much work has been done for computing non-stationary (*s,S*) policies. A few authors addressed the inventory problem with non-stationary demands and proposed heuristics based on non-optimal policies (see e.g. Silver, 1978; Askin, 1981; Morton and Pentico, 1995). Bollapragada and Morton (1999) developed an efficient myopic heuristic for computing the non-stationary (*s,S*) policy where consecutive periods are grouped and converted to stationary problems. In this paper, we consider the inventory problem addressed in Scarf (1960) and investigate the cost efficiency of stationary and non-stationary inventory policies.

2. Problem definition and solution procedures

Our aim is to investigate the cost performance of stationary policies under non-stationary demand. For the sake of completeness, we compare the optimal non-stationary policy with the best possible stationary policy. We use the (*s,S*) policy as a frame of reference since it is proven to be optimal both in stationary and non-stationary demand cases.

Throughout the paper it is assumed that, the demand, d_t in period t, is considered as a random variable with known probability density function, $g_t(d_t)$, and is assumed to occur instantaneously at the beginning of each period. The mean rate of demand may vary from period to period. Demands in different time periods are assumed independent. A fixed holding cost h is incurred on any unit carried in inventory over from one period to the next. Demands occurring when the system is out of stock are assumed to be backordered, and satisfied immediately the next replenishment order arrives. A fixed shortage cost p is incurred for each unit of demand backordered. A fixed procurement (ordering or set-up) cost K is incurred each time a replenishment order is placed. For convenience, without loss of generality, the initial inventory level and the unit procurement cost are set to zero, and it is assumed that there is no replenishment lead-time.

2.1. Non-stationary (*s*,*S*)

Scarf (1960) developed the concept of K-convexity and proved that under the aforementioned assumptions the optimal inventory policy follows an (*s,S*) rule. He provided a dynamic programming formulation to compute the optimal (*s,S*) levels for each period. Obviously, (*s,S*) levels are not constant for different periods in non-stationary problems. Thus, parameters of a non-stationary policy can be represented as (s_t,S_t) for period *t*. The dynamic program proposed by Scarf is given in Equation 1 where $C_{t,n}(x)$ represents the optimal expected total cost for periods *t* through *n* of an inventory system with *x* units of initial inventory. *K* is the fixed ordering cost, and $L_t(x)$ is the expected holding and penalty costs charged in period *t* when starting inventory is *x* units. S_t and s_t represent the order up to level and the reorder point for period *t*, respectively.

$$C_{t,n}(x) = \min\ \{L_t(x) + \mathrm{E}(C_{t+1}(\,x - \xi\,)),\ K + L_t(S_t) + \mathrm{E}(C_{t+1}(S_t - \xi\,))\} \qquad (1)$$

Scarf's formulation required extensive computational power beyond the limitations of its time. As a matter of fact, there was no known way of computing policy parameters at that time (Karlin, 1960). Relatively recently, Bollapragada and Morton (1999) were able to compute the (s_t,S_t) levels by exploiting the property of K-convexity in solving the above dynamic program. In this paper, we employed their approach to determining optimal (s_t,S_t) values.

In order to introduce the non-stationary (*s*,*S*) policy clearly we give two simple examples. Figure 1 demonstrates the optimal non-stationary (s,S) policies for two different problems. In the first graph expected demand pattern is generated randomly, so (*s*,*S*) levels fluctuate pursuant to demand pattern from period to period. In the second graph demand pattern is stationary. As it can be observed from the graph, policy parameters are stationary except at the end of the planning horizon. The deviation in the last period is the *end of horizon effect*.

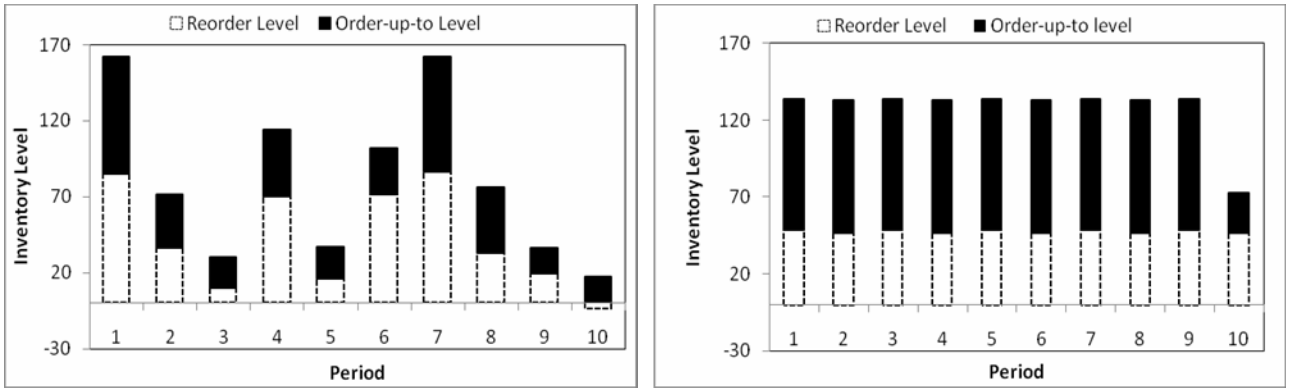


Figure 1 The optimal non-stationary (s,S) policies for two different problem instances: The bottom and the top levels of black columns represent reorder points and order-up-to levels respectively

2.2. Best Representative Stationary (s,S)

One may think of two possible approaches to obtain the best stationary policy for a non-stationary demand pattern. The first approach is to find a stationary demand distribution which best fits the original demand with the given inventory system. Once the best stationary demand distribution is determined, the corresponding stationary policy can be computed using the well-known Zheng-Federgruen (1991) algorithm. The second approach is to find a stationary policy which provides the minimum cost for the actual non-stationary demand. To the best of our knowledge, there is no published work in the literature neither on determining stationary demand distribution nor on computing the optimal policy parameters of a stationary (*s,S*) policy. Therefore we settle to employ an exhaustive search procedure. The aforementioned approaches differ in their search spaces, such that, the former approach requires a search on various demand patterns, whereas the latter requires a search on various policy parameters. Since characterizing the search space of the first approach is rather difficult compared to the second one, we employ the second approach, and compute the best stationary policy through a two-dimensional search procedure on policy parameters s and S ($s<S$). The expected cost of examined (*s,S*) pair is computed by means of the recursive formulation given in equation 1. The elimination of the end-of-horizon effect is discussed in the next section.

3. Computational Study

The experiment design, results and their interpretation are crucial to understanding the applications of stationary policies under the non-stationary demand environments. In the next subsections these will be given in detail.

3.1 Experiment Design

In the experiment design phase we concentrate on: (i) end-of-horizon effects, and (ii) designation of demand and cost parameters.

Stationary policies are exposed to end-of-horizon effects when the planning horizon is finite. As mentioned earlier, non-stationary ones have the ability to adjust policy parameters based on periodic oscillations. However, the stationary policy does not have such an ability to dampen the end of horizon effect on the expected cost. In order to make a fair assessment of the cost performance of stationary policies we adopt the method used in Bollapragada and Morton (1999). They use a simple, yet an effective method based on plugging in the optimal non-stationary plan at the end of the planning horizon. That is, although a stationary plan is used through out the planning horizon, the stock at the end of the horizon is managed using the optimal non-stationary parameters. Hence, the end of horizon effect is eliminated from the results. Following Bollapragada and Morton (1999), we set the planning horizon to 70 periods and use the optimal non-stationary policy parameters in the last 18 periods so as to eliminate the end of horizon effect.

We use a test set of 108 problem instances in order to investigate the effect of various demand and cost parameters on the cost performances of non-stationary and approximate stationary policies. Demand is characterized by a demand pattern π, i.e. a sequence of mean demand values (d^{pi}={d^{\pi}_1,d^{\pi}_2,…d^{\pi}_N}) through the planning horizon, and a fixed coefficient of variation, cv. Note that, a fixed coefficient of variation implies higher standart deviation for larger demands. The cost parameters are: the ordering cost per replenishment, the penalty cost per unit backlog per period, and the holding cost per unit of excess inventory per period. We construct our test set using the following values;

Unit holding cost: $h = \{1\}$
Unit penalty cost: $p = \{2, 5, 10\}$
Fixed ordering cost: $K = \{50, 200, 500\}$
Demand coefficient of variation: $cv = \{0.05, 0.15, 0.25\}$

As shown in Figure 1, we employ four different demand patterns: stationary, seasonal, life cycle, and erratic (Berry, 1972). The average demand per period is 100 units for each demand pattern. Seasonal and life cycle demand patterns are very common in practice. Stationary and erratic patterns represent the extreme cases of the stationary and the non-stationary demand dichotomy.

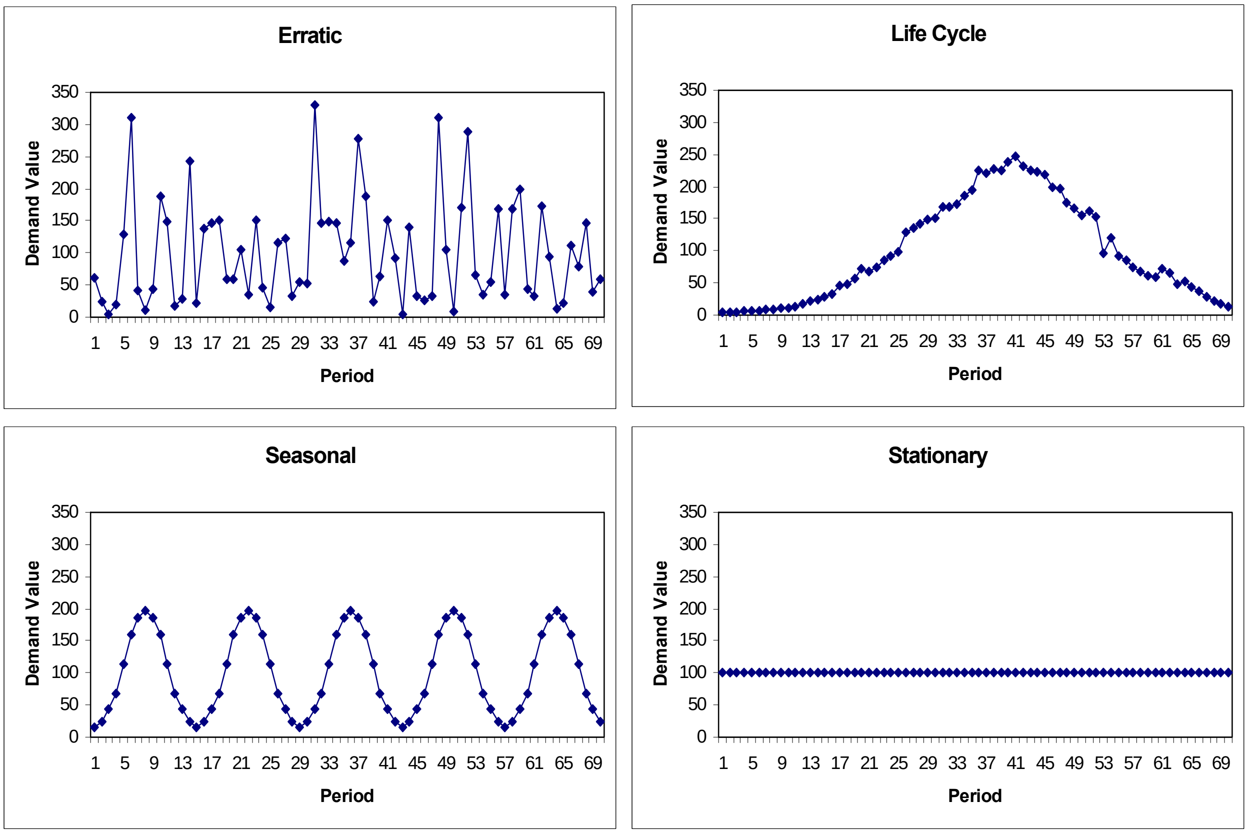


Figure 1: Demand Patterns

An illustrative example: In order to demonstrate the method, we provide an illustrative example. Figure 2 illustrates the results of the problem instance-5 from the test set. This instance is characterized by a seasonal demand pattern with a coefficient of variation of 0.15. The cost parameters are; $K = 50$, $p = 5$, h=1. The first graph plots the policy parameters of the optimal non-stationary policy; the second one plots the best stationary policy in which the last 18 (?) periods are controlled using the optimal non-stationary policy.

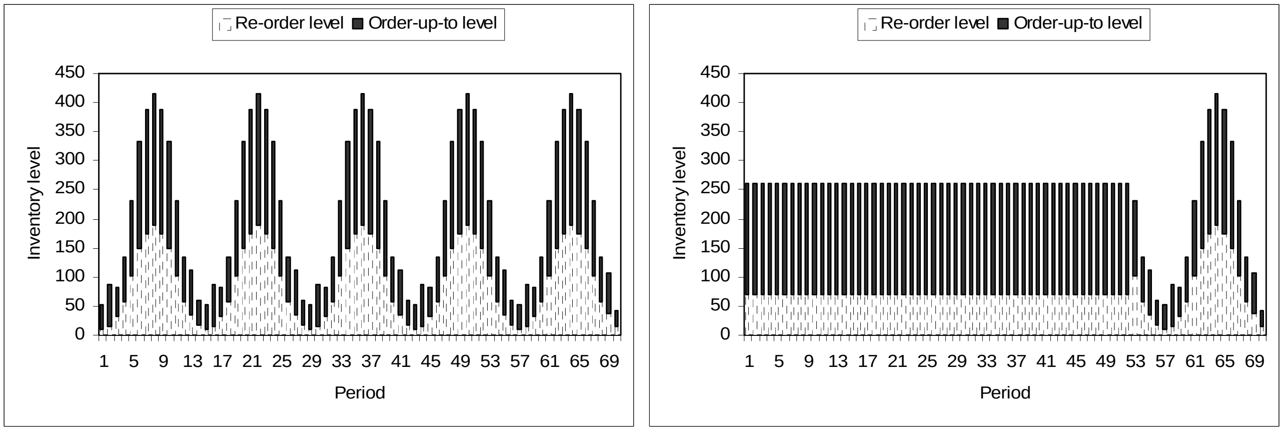


Figure 2: (s_t,S_t) and (s,S) levels of the instance 5 with seasonal demand pattern provided by the optimal non-stationary and the best stationary policies

3.2 Numerical results and insights

We conduct an extensive numerical study to measure the cost performance of the best stationary policy. The results for the 108 instances (i.e., 27 instances for each of the four demand patterns) are presented in Table 2 in the appendix. The cost difference, Δ, for an instance is calculated as follows:

$$\Delta = \frac{SP - OP}{OP} \times 100 \tag{2}$$

where *SP* and *OP* are the expected total costs of the system under the best stationary and the optimal non-stationary policies, respectively.

Effects of demand pattern:

We start our analysis by investigating the results related to each demand pattern. Let us define \$\delta^{\pi}\$ as the average \$\delta\$ of all 27 instances characterized by demand pattern \$\pi\$. Obviously, \$\delta^{\pi}\$ is closely related to characteristics of the demand pattern; more specifically the non-stationarity of \$\pi\$. Let \$\mu^{pi}\$ and \$\sigma^{\pi}\$ be the mean and the standart deviation of the demand pattern \$\pi\$.

Note that μ^{π} is identical for all demand patterns, since the total average demand is standardized. We use σ as the measure of non-stationarity, and examine how it bonds to δ. Table 1 provides the summary results.

	Erratic	Seasonal	Life Cycle	Stationary
Δ	75.39	27.11	45.67	0.00
σ	81.68	65.06	78.16	0.00
μ	100.00	100.00	100.00	100.00

Table 1: Average expected cost discrepancies with respect to demand patterns

According to Table 1 even at first glance, one can easily claim that cost of neglecting non-stationarity is dramatically expensive especially in erratic and life cycle demand patterns. One would expect that when π is near-stationary then δ^{pi} will be close to zero. Our results confirm this expectation. There is no difference between the cost performance of stationary and non-stationary policies when demand is stationary. Similarly, for mean demand patterns with low variances, such as the seasonal pattern, the sub-optimality is rather small, whereas, for demand patterns characterized by high volatility, such as, life cycle and erratic patterns, the sub-optimality is extremely high. In summary, a higher variation in the mean demand pattern, which defines the magnitude of non-stationarity, resulted in higher Δ. In other words, the optimal non-stationary policy converges to a stationary policy as σ^j approaches zero.

Although the results given in Table 2 provides some general insights on the costs of neglecting non-stationarity,, understanding the effects of demand variation, i.e. the coefficient of variation, and cost parameters, such as ordering and penalty costs is also important for a thorough analysis. In the following subsections we discuss the effects of these problem parameters.

Effects of coefficient of variation:

Figure 3 shows the average cost difference for each demand pattern as the coefficient of variation is increased from 0.05 to 0.15 and further to 0.25. Note that, given cost differences reflect, boost the average cost difference of all problems where the reference parameter, i.e. demand pattern, is the same. Each point in the graph corresponds to the average cost difference for a fixed coefficient of variation and a given demand pattern. As can be seen from Figure 1, larger coefficient of variation tends to decrease the cost difference. In fact, as the coefficient of variation increases the expected total cost of the inventory system increases under both the best stationary and the optimal non-stationary policies. However, the cost increase rate for the optimal non-stationary policy is relatively higher than the same rate for the best stationary one. Hence, an increase in coefficient of variation results in a decrease in the cost difference. Therefore, we can argue that, using a stationary policy may be acceptable in highly uncertain environments where the demand patterns is stationary.

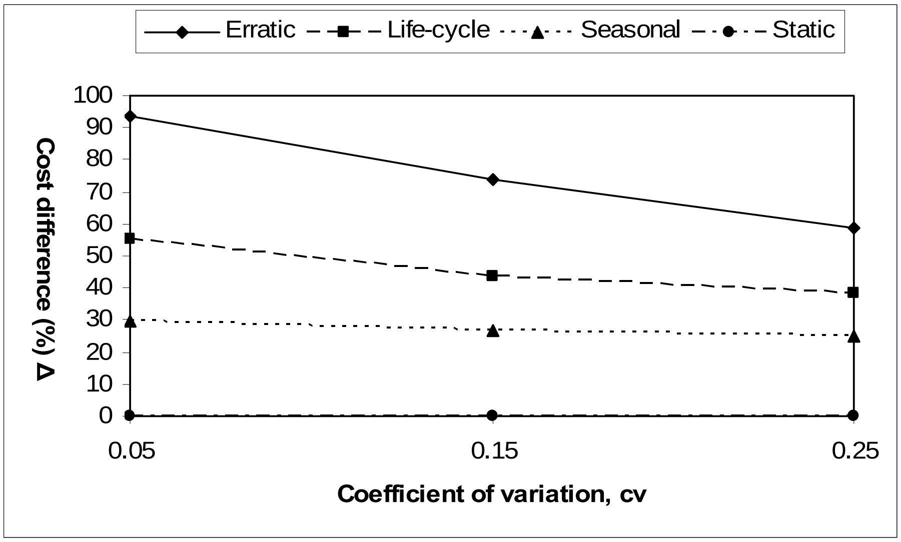


Figure 3: “cv” vs “Delta”

Effects of setup cost:

Figure 4 illustrates the change in “Delta” for each demand pattern as setup cost increases. As can be observed, when the penalty cost and coefficient of variation are kept constant, Δ decreases with the increasing ordering cost. Basically, the ordering cost determines the frequency of orders. Obviously, higher ordering costs encourage the inventory replenishment plan to have fewer replenishments, and leads to longer replenishment cycles where accumulated demand is satisfied by a single replenishment. This dampens the fluctuation of demand through the replenishment cycle, and hence, moderates the non-stationarity of the optimal plan and decreases Δ.

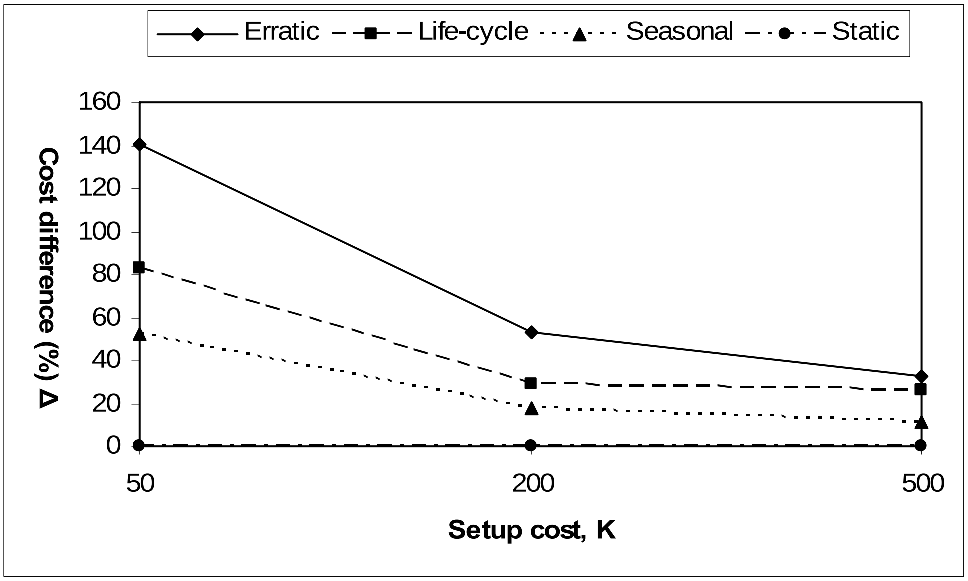


Figure 4: K vs Delta

Effects of penalty cost:

Figure 5 demonstrates the effect of penalty cost for each demand pattern. Unlike ordering costs, penalty costs negatively affect the sub-optimality of stationary policies. The penalty cost basically determines the level of the safety stock. That is, higher penalty costs necessitate larger safety stocks to avoid stock-outs. Since non-stationary policy determines safety stock levels optimally, its cost performance increases with respect to stationary policy with increasing penalty costs.

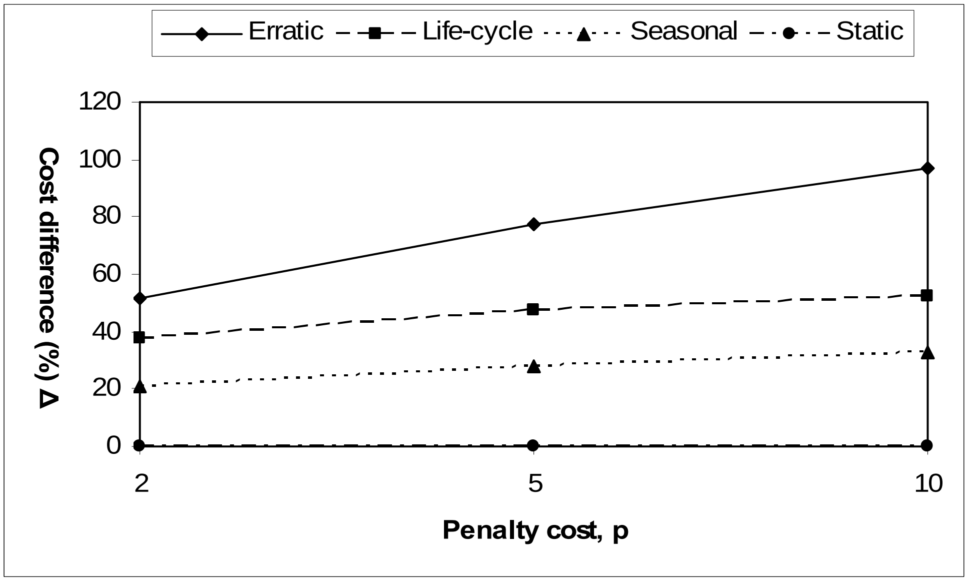


Figure 5: p vs Delta

4. Conclusion

As product life cycles are getting shorter most of the real world problems exhibit non-stationary demand patterns. However, non-stationary inventory policies have not been widespread among neither practitioners nor academics due to their complexity in computation and application. However, when demand is non-stationary, a stationary policy is only an approximation to the optimal non-stationary one, and hence, is sub-optimal with respect to total expected cost. In this paper we analyzed the cost efficiency of using stationary inventory policies when demand is non-stationary. We used (s,S) policy as a frame of reference, and compared the optimal non-stationary (s,S) policy with the best possible stationary (s,S) policy in terms of cost performance.

We conducted an extensive numerical study considering a variety of cases to examine the effects of various organizational parameters and showed that cost of neglecting the non-stationarity of demand is significantly high for the majority of cases. Our numerical study reveals that, the magnitude of the sub-optimality of stationary policies depends heavily on the variation of the demand pattern, i.e. the non-stationarity of demand, among other factors, such as, the coefficient of variation of demand, ordering and penalty costs. In order to characterize the variation of the demand patterns, we considered erratic, life cycle, seasonal and stationary demands. On average, the cost difference between the non-stationary and the stationary policies were; 75% for erratic, 45% for life cycle, 27% for seasonal, and 0% for stationary demands. Our study also provides some insight on cases where stationary models provide good approximations to non-stationary models. More specifically, when demands follow a rather stable pattern with high uncertainty (i.e. variance), stationary policies may be a reasonable substitute for the optimal non-stationary policy. Moreover, the cost performance of stationary policies improves when setup costs are high and penalty costs for stockouts are low.

Altogether, our study underlines the need for careful evaluation when assuming stationary demands. We argue and provide some evidence that it may be very expensive to use stationary policies when actual demand is non-stationary. The recent literature is not sufficient to embed the non-stationary models into real life problems, and hence, modeling non-stationary demands has not yet been addressed satisfactorily in ERP environments. Most, if not all, ERP systems do not support non-stationary demands. As a result, our study also brings up the discussion on the percieved importance of non-stationary demand cases in theory and practice of inventory management.

Bollapragada (2006) stated that uncertainty modeling and management has not been satisfactorily addressed for MRP/ERP environments. In these environments, replenishment orders are typically determined by ad hoc safety stock rules and even stationary demand processes have not been adequately embedded to real life planning tools. Hence, non-stationary inventory policies, which are based on rather sophisticated stochastic processes, have not been used commonly.

Appendix

				% Cost Difference Δ_i			
i	**K**	**p**	**cv**	**Erratic**	**Seasonal**	**Life Cycle**	**Stationary**
1	50	2	0.05	122.31	49.77	89.04	0.00
2			0.15	93.38	40.32	66.77	0.00
3			0.25	71.93	35.27	54.45	0.00
4		5	0.05	191.13	60.63	108.7	0.00
5			0.15	140.89	53.38	80.11	0.00
6			0.25	106.14	47.07	67.07	0.00
7		10	0.05	232.26	72.56	114.73	0.00
8			0.15	175.15	58.22	86.47	0.00
9			0.25	132.05	52.79	74.1	0.00
10	200	2	0.05	42.4	14.02	26.02	0.00
11			0.15	35.25	13.42	22.42	0.00
12			0.25	28.93	12.12	21.14	0.00
13		5	0.05	60.78	17.07	33.66	0.00
14			0.15	53.25	18.11	29.34	0.02
15			0.25	45.72	18.6	26.99	0.00
16		10	0.05	80.63	21.16	38.04	0.00
17			0.15	68.32	22.51	32.66	0.01
18			0.25	59.6	22.92	30.92	0.00
19	500	2	0.05	26.79	6.47	20.22	0.03
20			0.15	24.29	8.22	18.95	0.02
21			0.25	20.3	8.72	18.32	0.00
22		5	0.05	36.67	9.84	30.18	0.00
23			0.15	33.14	11.96	25.95	0.00
24			0.25	28.33	12.23	23.57	0.00
25		10	0.05	47.64	13.4	35.16	0.00
26			0.15	41.71	15.34	30.22	0.00
27			0.25	36.58	15.92	27.75	0.00

Table 2: Cost differences for all problem instances

References

S. Bollapragada, T. E. Morton. A simple heuristic for computing nonstationary (s,S) policies. Operations Research, 47(4):576-584, 1999.

R. Ehrhardt. The power approximation for computing (s,S) inventory policies. Management Science, 25(8):777-786, 1979.

D. L. Iglehart. Optimality of (s,S) policies in the infinite horizon dynamic inventory problem. Management Science, 9(2):259-267, 1963.

H. Scarf. The optimality of (S,s) policies in the dynamic inventory problem. Mathematical Methods in the Social Sciences, Stanford University Press, Stanford, CS, (1960).

A. F. Veinott, H. M. Wagner. Computing optimal (s,S) inventory policies. Management Science, 11(5):525-552, 1965.

H. M. Wagner, T. M. Whitin. Dynamic version of the economic lot size model. Management Science, 5:119-124, 1958.

Y. Zheng, A. Federgruen. Finding optimal (s,S) policies is about as simple as evaluating a single policy. Operations Research, 39(4):654-665, 1991.

Ann Oper Res
DOI 10.1007/s10479-008-0382-6

Finding reliable solutions: event-driven probabilistic constraint programming

S. Armagan Tarim · Brahim Hnich · Steven Prestwich · Roberto Rossi

Abstract Real-life management decisions are usually made in uncertain environments, and decision support systems that ignore this uncertainty are unlikely to provide realistic guidance. We show that previous approaches fail to provide appropriate support for reasoning about reliability under uncertainty. We propose a new framework that addresses this issue by allowing logical dependencies between constraints. Reliability is then defined in terms of key constraints called "events", which are related to other constraints via these dependencies. We illustrate our approach on three problems, contrast it with existing frameworks, and discuss future developments.

Keywords Event-driven · Probabilistic · Constraint programming · Uncertainty

1 Introduction

Real-life management decisions are usually made in uncertain environments. Random behavior such as the weather, lack of essential exact information such as the future demand, incorrect data due to errors in measurement, and vague or incomplete definitions, exemplifies the theme of uncertainty in such environments.

S.A. Tarim
Department of Management, Hacettepe University, Ankara, Turkey
e-mail: armagan.tarim@hacettepe.edu.tr

B. Hnich
Faculty of Computer Science, Izmir University of Economics, Izmir, Turkey
e-mail: brahim.hnich@ieu.edu.tr

S. Prestwich (✉) · R. Rossi
Cork Constraint Computation Centre, University College, Cork, Ireland
e-mail: s.prestwich@cs.ucc.ie

R. Rossi
Centre for Telecommunication Value-Chain Driven Research, Dublin, Ireland
e-mail: rrossi@4c.ucc.ie

It is generally impossible for any set of decisions to satisfy all the constraints under all circumstances. For instance, consider a probabilistic single-item distribution problem in which there are n independent suppliers with their given *probabilistic* supply capacities, and m different customers with known demands. It is realistic to assume that the deliveries are fixed in advance, by consideration of the probabilistic supply capacities. The need to fix the deliveries in advance has been at the heart of many problems such as the buying of raw materials on markets with fluctuating prices (Kingsman 1985). Thus the investigation of modeling approaches and solution algorithms is potentially important not only from a theoretical point of view, but also from the perspective of practical applications. It is quite unrealistic to ask for a plan that satisfies all demand and probabilistic supply constraints, irrespective of the unfolding of uncertainties. In order to deal with the optimization problems with stochastic/fuzzy factors, stochastic programming and fuzzy programming have been greatly developed. The theory of stochastic programming has been summarized by several books such as Sengupta (1972), Vajda (1972), Kall and Wallace (1994) etc.

To address this and related situations, we propose that one should determine in advance a distribution plan that satisfies customer demands as far as possible, under some measure that accurately captures the user's notion of reliability. To address this important class of problems, we take a novel approach and develop a modeling framework that supports more reliable decisions in uncertain environments, yet reduces the cognitive burden on a decision-maker. Our *Event-Driven Probabilistic Constraint Programming* (*EDP-CP*) modeling framework allows users to designate certain probabilistic constraints (such as demand constraints) as *events* whose chance of satisfaction must be maximized, subject to hard constraints (such as a lower bound on profit), and also logical dependencies among constraints (such as the dependency of demand constraints on the satisfaction of the probabilistic supply constraints). We shall show that the EDP-CP framework allows more realistic modeling of some problems than previous approaches.

Complex decision systems are usually multidimensional, multifaceted, multifunctional and multicriteria, and include stochastic or fuzzy factors. With the requirement of considering randomness, appropriate formulations of stochastic programming have been developed to suit the different purposes of management (Fig. 1). The first method dealing with stochastic parameters in stochastic programming is the so-called *expected value model* (Birge and Louveaux 1997), which optimizes the expected objective functions subject to some expected constraints. The second, *chance-constrained programming*, was pioneered by Charnes and Cooper (1959) as a means of handling uncertainty by specifying a confidence level at which it is desired that the stochastic constraint holds. Chance-constrained programming models

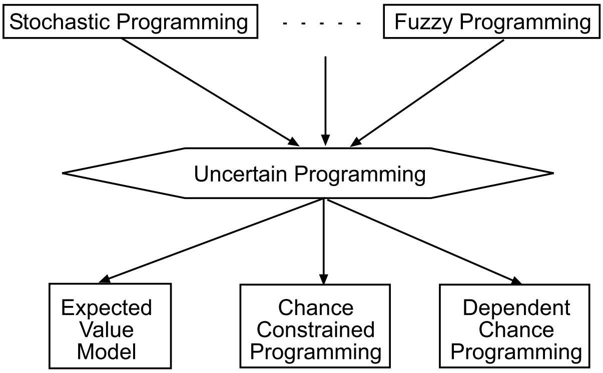


Fig. 1 Techniques for modeling decision problems under uncertainty

can be converted into deterministic equivalents for some special cases, and then solved by some solution methods of deterministic mathematical programming. However it is almost impossible to do this for complex chance-constrained programming models. In order to overcome this dilemma, Liu and Iwamura (1997) proposed a stochastic simulation-based genetic algorithm for solving general chance-constrained programming as well as chance-constrained multi-objective programming, and chance-constrained goal programming, in which the stochastic simulation is employed to check the feasibility of solutions and to handle the objective functions. Sometimes a complex stochastic decision system undertakes multiple tasks called events, and the decision-maker wishes to maximize the chance functions which are defined as the probabilities of satisfying these events. In order to model this type of problem, Liu (1997) provided a theoretical framework of the third type of stochastic programming, called *dependent-chance programming*. Dependent-chance multiobjective programming and dependent-chance goal programming have also been presented (for a more detailed discussion see Liu 1999).

Roughly speaking, dependent-chance programming is aimed at maximizing some chance functions of events in an uncertain environment. In deterministic mathematical programming as well as expected value models and chance-constrained programming, the feasible set is essentially assumed to be deterministic after the real problem is modeled. That is, an optimal solution is always given regardless of whether is it can be performed in practice. However the given solution may be impossible to perform if the realization of uncertain parameters is unfavorable. Thus, the dependent chance-programming model never assumes that the feasible set is deterministic. In fact, the feasible set of dependent chance-programming is described by a so-called *uncertain environment*. Although a deterministic solution is given by the dependent chance-programming model, this solution needs to be performed as far as possible. This special feature of dependent chance-programming is very different from other existing stochastic programming techniques. However, such problems do exist in the real world. Some real and potential applications of dependent chance programming have been presented by Liu and Ku (1993), Liu (1995a, 1995b), Liu and Iwamura (1997), and more recently by Wu et al. (2005). In what follows we will see that the framework we propose, EDP-CP, extends and improves Liu's framework by providing to the user more expressiveness, in order to capture a more realistic and accurate measure of plan reliability, and an exact solution method in contrast to Liu's genetic algorithm.

The rest of this paper is organized as follows. In Sect. 2 we motivate the work. We define the new modelling framework in Sect. 3 and show how to compile any EDP-CP model into an equivalent constraint program in Sect. 4. In Sect. 5 we survey a scenario reduction technique that may be applied to keep the number of possible scenarios under control, references are given to other works adopting the same strategy to reduce the number of scenarios considered. In Sect. 6 we illustrate the flexibility and usefulness of our framework by studying three examples: probabilistic supply chain planning, scheduling, and production planning/capital budgeting. In Sect. 7 we survey related work. Finally, in Sect. 8 we summarise our work and discuss future directions.

2 Motivation

Our motivation for this work comes from an application in the supply chain management area: more precisely, addressing supply and demand uncertainties. The main inherent difficulty in dealing with this class of probabilistic problems is the fact that certain constraints (such as the ones imposing on complete satisfaction of customer demands) may hinge on

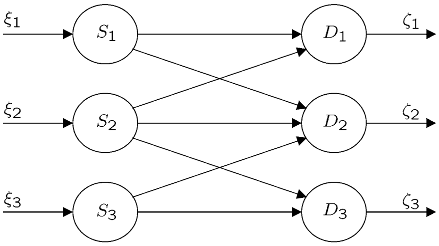


Fig. 2 Distribution problem

the satisfaction of others (such as supply constraints). The problem is particularly interesting when the latter constraints are exposed to uncertainty.

2.1 A motivating example

We provide a concrete example of the distribution problem to motivate the work. Figure 2 depicts a distribution system with three suppliers $S_{1,2,3}$ and three customers $D_{1,2,3}$. The scopes of the suppliers are $S_1 \rightsquigarrow \{D_1, D_2\}$, $S_2 \rightsquigarrow \{D_1, D_2, D_3\}$, $S_3 \rightsquigarrow \{D_2, D_3\}$. The deterministic customer demands are $[8, 7, 4]$. The suppliers' probabilistic capacities are expressed as discrete probability density functions: $f_{S_1} = \{3(0.3), 7(0.5), 12(0.2)\}$, $f_{S_2} = \{6(0.4), 7(0.2), 10(0.4)\}$ and $f_{S_3} = \{3(0.3), 8(0.7)\}$, where values in parentheses represent probabilities. The objective is to obtain the most reliable distribution plan. In the following sections we shall consider a series of models of increasing sophistication. Our running example will emphasize differences between these models.

2.2 Model 1: Naive

Define decision variables $x_{s,c}$ where $s, c \in \{1, 2, 3\}$, denoting the planned supply from supplier s to customer c. Also define random variables ξ_i denoting the uncertain supply available to supplier i. A constant ζ_c denotes the deterministic demand of customer c. Any plan must satisfy the hard constraints

$$\sum_{s \in \mathbf{S}_c} x_{s,c} = \zeta_c$$

where $\mathbf{S}_c$ is the set of suppliers for customer c. There are also probabilistic constraints between decision and random variables:

$$\sum_{c \in \mathbf{C}_s} x_{s,c} \le \xi_s$$

where $\mathbf{C}_s$ is the set of customers for supplier s. These probabilistic constraints are "soft": they may be violated in some scenarios. We therefore do not add them to the model (as with the deterministic constraints), but instead use them to define an objective function:

$$\max \sum_{s} E\left\{ \sum_{c \in \mathbf{C}_s} x_{s,c} \le \xi_s \right\} \tag{1}$$

where $E\{C\}$, the "expectation operator" (Jeffreys 1961), is the sum of the probabilities of the scenarios in which constraint C is satisfied. This model may be viewed as a *Soft Probabilistic CSP*, that is a Probabilistic CSP (Fargier et al. 1995) where some contraints are hard, plus

Table 1 Representative distribution plans and event realization measures (ERM), that is reliability measures, computed by the different models we presented

Plan No.	Planned delivery $S_i \rightsquigarrow D_j$: (i, j)							ERM			
	$(1,1)$	$(1,2)$	$(2,1)$	$(2,2)$	$(2,3)$	$(3,2)$	$(3,3)$	Model 1	Model 2	Model 3	Model 4
1	3	0	5	6	1	1	3	1.7	0.0	0.0	0.0
2	3	5	5	1	1	1	3	1.5	0.6	0.6	0.6
3	0	2	0	2	4	3	0	0.0	0.0	0.0	2.0
4	5	0	3	3	0	4	4	2.4	1.8	2.1	2.1
5	6	0	2	0	4	7	0	2.4	1.8	2.4	2.4

an optimization criterion that we wish to maximise the probability that other probabilistic soft constraints are satisfied.

A drawback of this model is that the objective function does not measure plan reliability in a realistic way. For example, in any scenario in which supplier 2 cannot meet its demands (so that $x_{2,1} + x_{2,2} + x_{2,3} > \xi_2$) we cannot guarantee that *any* customer is supplied. This is therefore a worst-case plan for the given scenario, yet in the above model only one probabilistic constraint is violated under this scenario. A plan in which two or three probabilistic constraints are violated would be assigned a lower objective function value, but would be no less reliable. Worse still, consider a similar problem in which supplier 1 supplies only customer 1, supplier 3 supplies only customer 3, and supplier 2 again supplies customers 1, 2 and 3. A plan in which suppliers 1 and 3 are unable to meet their demands under some scenario would be classed as less reliable than one in which supplier 2 is unable to meet its demand under the same scenario, because more probabilistic constraints are violated. However, the latter plan is less reliable: in the first plan customer 2 is satisfied, but in the second plan no customer is.

In Table 1 we show how this naive model (column "Model 1") classifies reliability of five different plans for our concrete example. In the next sections we will show that other models can give a more accurate and realistic measure of the reliability of these plans.

2.3 Model 2: Dependent-chance programming

To improve the naive model we may define a more intelligent objective function: the reliability of a plan is now the sum of the reliabilities of three *events*, where an event is the satisfaction of a customer:

$$\max \sum_{c} E\left\{ \bigwedge_{s \in \mathbf{S}_c} \left(\sum_{c' \in \mathbf{C}_s} x_{s,c'} \leq \xi_s \right) \right\} \tag{2}$$

where $\wedge$ denotes logical conjunction: $E\{C \wedge C'\}$ is the sum of the probabilities of the scenarios in which both C and C' are satisfied. For example the reliability of satisfaction of customer 1 is the sum of the probabilities of the scenarios in which suppliers 1 and 2 both meet their demands. Under this objective function, our worst-case plan (in which supplier 2 cannot meet its demands) is assigned reliability 0 in the scenario, because the violated probabilistic constraint $x_{2,1} + x_{2,2} + x_{2,3} \leq \xi_2$ affects the reliability of each customer. Allowing logical connectives between constraints allows us to express the problem more accurately. This model is similar to a Dependent-Chance Programming (Liu and Iwamura 1997) approach to a related problem.

Let us observe in Table 1 how this new notion of reliability affects the plans already considered. Note that the new objective function defines a completely new notion of reliability, therefore results provided by Model 1 and 2 are incomparable since Model 1 measures reliability in terms of expected number of suppliers that meet their demand, while in Model 2 the measure refers to the expected number of unsatisfied customers. We shall see that the second notion of reliability reflects a *higher level of expressiveness* and is closer to what is perceived as *reliable* by common sense.

To gain more insight into the notion of reliability captured by Model 2 we now examine two different distribution plans, 1 and 2. These two plans share common decisions, except at $S_1 \rightsquigarrow D_2$ and $S_2 \rightsquigarrow D_2$. Plan 1 (2) requires a capacity value of 3 units (8 units) at S_1 to be feasible, which is available with probability 1.0 (0.2). However if we consider S_2, Plan 1 (2) requires a capacity value of 12 units (7 units) to be feasible. The corresponding probability is 0.0 (0.6), thus Plan 2 is more reliable than Plan 1. It is now easy to see how logical connectives introduced in (2) capture a more intuitive and accurate measure for the reliability of a plan that, as seen, is expressed in terms of expected number of satisfied customers, respectively 0.0 and 0.6 for Plans 1 and 2. Note that the reliability measure in Model 1 classifies Plan 1 as more reliable than Plan 2, since the latter violates more probabilistic constraints. Obviously such a measure is flawed since Plan 1 is never able to reliably satisfy any customer as supplier 2 can not provide 12 units of capacity.

2.4 Model 3: EDP-CP

However, even the second model is flawed. Consider a plan in which $x_{1,1} = 0$ so that customer 1 must receive all supplies from supplier 2. The reliability of the satisfaction of customer 1 should now be independent of the ability of supplier 1 to meet its demand, but in the second model it is still dependent; this point was not considered in Liu and Iwamura (1997). We should therefore refine the objective via further logical connectives between constraints:

$$\max \sum_{c} E\left\{\bigwedge_{s\in\mathbf{S}_c}\left(x_{s,c} \neq 0 \Rightarrow \sum_{c'\in\mathbf{C}_s} x_{s,c'} \leq \xi_s\right)\right\} \tag{3}$$

where $\Rightarrow$ denotes logical implication: $E\{C \Rightarrow C'\}$ is the sum of the probabilities of the scenarios in which either C is violated or C' is satisfied, or both. Because of this modification, under a scenario in which $x_{1,1} = 0$ there is no longer a penalty if

$$\sum_{c'\in\mathbf{C}_1} x_{s,c'} \leq \xi_1$$

is violated. In this case, the reliability of a plan is gauged by an event realization measure which gives equal importance (i.e., equal weights) to satisfying demands completely at D_1, D_2, and D_3.

We now consider Plans 4 and 5. By observing differences between these plans it is easy to see how the further logical connectives introduced in (3) affects reliability of the solutions. In Plan 4, $\{S_1, S_2\} \rightsquigarrow D_1$, $\{S_2, S_3\} \rightsquigarrow D_2$, $S_3 \rightsquigarrow D_3$. In other words, S_3 supplies two customers. In Plan 5 $\{S_1, S_2\} \rightsquigarrow D_1$, but $S_3 \rightsquigarrow D_2$ and $S_2 \rightsquigarrow D_3$. Therefore in both the plans S_1 supplies the same customer D_1, and S_2 supplies two customers (respectively D_1, D_2 and D_1, D_3), while S_3 in Plan 4 supplies two customers (D_2 and D_3) and in Plan 5 only supplies one customer (D_3). If S_1 fails to meet the requirement in both the plans it will affect only D_1 with probability 0.3. S_2 cannot fail to meet the demand in both the plans. But in Plan 4 if

S_3 does not provide 8 units as required, with probability 0.3, it will affect both D_2 and D_3, while in Plan 5 if S_3 does not provide 7 units, with the same probability 0.3, it will affect only D_2. Thus Plan 5 is obviously more reliable than Plan 4. Such a notion is captured by Model 3, which therefore provides a more accurate reliability measure with respect to the former ones we presented. In fact the reader may observe that in Models 1 and 2 Plans 4 and 5 are classified as equally reliable.

2.5 Model 4: EDP-CP

So far, the decision-maker's objective has been to maximize the plan reliability, defined in such a way that all violated plans are treated equally. In other words, plans in which not all customer demand constraints hold are considered equally unreliable, irrespective of the number of customers that are completely satisfied. This obviously constitutes a limit for the first three models presented, since often we may get unrealistic solutions where we try to satisfy every customer achieving a poor overall reliability. An alternative objective could aim to satisfy as many customers as possible, that is to meet as many demand constraints as possible under probabilistic supply constraints. Clearly, this new objective may have a wider application and may lead to more realistic solutions where some customers may be dropped in order to serve the others with higher reliability.

In the first EDP-CP model any plan must satisfy the hard constraints on demands ζ_c, but a plan that reliably satisfies two customers might be more desirable than one that satisfies all three customers less reliably. We can model such a measure of plan reliability by removing the hard constraints and using them in the objective function instead:

$$\max \sum_c E\left\{\left[\bigwedge_{s\in S_c}\left(x_{s,c} \neq 0 \Rightarrow \sum_{c'\in C_s} x_{s,c'} \leq \xi_s\right)\right] \wedge \left(\sum_{s'\in S_c} x_{s',c} = \zeta_c\right)\right\}. \tag{4}$$

A direct consequence of this new objective on optimized plans is that solutions may no longer aim for complete satisfaction of all customers, but most likely a subset of it, with higher reliability. Under this new objective, distribution Plan 3 in Model 4 guarantees complete satisfaction of D_2 and D_3 with a reliability score of 2.0, whereas under the previous models it is assigned reliability score 0.

In Table 1 the column for Model 4 depicts an accurate and realistic classification for the reliability measures of the plans considered in our concrete example.

2.6 A meta-constraint

We believe that our final model is of a form that will apply to many problems. The following sections present a formalization of a modeling framework to express such problems naturally and propose a compilation from the given formalization into a standard constraint program. In Sect. 3.2, we shall introduce a meta-constraint to simplify complex expressions such as those in our final model, so that it can be written in the form

Maximize

$\sum_c E\{e_c : \sum_{s'\in \mathbf{S}_c} x_{s',c} = \zeta_c\}$

given that

$(\forall c)\ (\forall s \in \mathbf{S}_c)\ \textsc{Dependency}(e_c,\ \sum_{c'\in \mathbf{C}_s} x_{s,c'} \leq \xi_s,\ x_{s,c} \neq 0).$

This construct is equivalent to (4), but by expressing the problem in this form, through the new keyword "**given that**", we separate the logical dependencies involving the events from the definition of the events. This way we aim to reduce the cognitive burden on the user. It should be noted that events e_c, although they appear as deterministic constraints,[1] are actually probabilistic. In fact their probabilistic nature is induced by the given dependencies. In practice the satisfaction of customer demands, that is constraints e_c, *depends* on the selected suppliers and on the capacity they can provide. More formally, constraints e_c ($\sum_{s' \in \mathbf{S}_c} x_{s',c} = \zeta_c$) are the *events* whose reliability we wish to maximize, and in each scenario these events are subject to certain *pre-requisite* constraints ($\sum_{c' \in \mathbf{C}_s} x_{s,c'} \leq \xi_s$) and certain *conditions* ($x_{s,c} \neq 0$). Intuitively, if a pre-requisite is unsatisfied in a scenario then the event is also classed as unsatisfied in that scenario; and if a condition is unsatisfied in a scenario then the event is classed as satisfied in that scenario.

3 Event-driven probabilistic constraint programming

In this section we formalise the EDP-CP modeling framework.

3.1 Preliminaries

Recall that a constraint satisfaction problem (CSP) consists of a set of variables $\mathcal{X}$, each with a finite domain of values D_i, and a set of constraints $\mathcal{C}$, each over a subset of $\mathcal{X}$ (denoted by $Scope(C)$) and specifying allowed combinations of values for given subsets of variables. A solution is an assignment of values to the variables satisfying the constraints. A Constraint Optimisation Problem (COP) is a CSP with given objective function over a subset of $\mathcal{X}$ that we wish to maximize or minimize.

Recall that a probabilistic CSP as introduced in Fargier et al. (1995) is defined as a 6-tuple $\langle \mathcal{X}, \mathcal{D}, \Lambda, \mathcal{W}, C, \Pr \rangle$ where:

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of decision variables;
- $\mathcal{D} = D_1 \times \cdots \times D_n$, where D_i is the domain of x_i;
- $\Lambda = \{\lambda_1, \ldots, \lambda_l\}$ is a set of uncertain parameters;
- $\mathcal{W} = W_1 \times \cdots \times W_l$, where W_i the domain of λ_i;
- C is a set of (probabilistic) constraints each involving at least one decision variable (and possibly some uncertain parameters);
- $\Pr : \mathcal{W} \to [0, 1]$ is a probability distribution over uncertain parameters.

In Fargier et al. (1995) a complete assignment of the uncertain parameters (resp. of the decision variables) is called a *world* (resp. a *decision*). The probability that a decision is a solution is the probability of the set of the worlds in which it is a solution.

In its most general form, event-driven probabilistic CP supports uncertain parameters as well as decision variables. A constraint is said to be *probabilistic*, if it involves both decision variables and uncertain parameters. In the rest of this paper we will sometimes refer to classical constraints as deterministic constraints to distinguish them from the probabilistic ones. We will refer to the possible values of an uncertain parameter λ_i as $W(\lambda_i)$ and to the probability of λ_i taking a given value v in $W(\lambda_i)$ as $\Pr(\lambda_i = v)$. As in Fargier et al. (1995), we refer to a complete assignment of uncertain parameters as a *possible world* and denote by $\mathcal{W}$ the set of all possible worlds. We also assume that the probability of each possible world w is given by the probability function $\Pr : \mathcal{W} \to [0, 1]$.

[1] Note that in the general case customer demand ζ_c may also be a random variable.

Definition 1 (Fargier et al. 1995) Given a probabilistic constraint c over decision variables and some uncertain parameters, the reduction of c by world $w \in \mathcal{W}$, denoted by $c_{\downarrow w}$, is the deterministic constraint obtained by setting all its uncertain parameters as in w.

3.2 Modeling framework

In EDP-CP some of the constraints can be designated by the user as *event constraints*. The user's objective is to maximize his/her chances of realizing these events. For instance, in our running example the user may consider the customer demand constraints as events. The objective is then to construct a plan satisfying customer demand constraints as far as possible.

The feasibility of certain event constraints depends on the satisfaction of other constraints. For instance, having a plan that meets the customer demands depends on whether or not the supply constraints are met with such a plan. For this purpose we introduce a new meta-constraint (already described in Sect. 2.6) useful for modeling such situations in our EDP-CP framework, which we refer to as a *dependency meta-constraint*. We first introduce the dependency constraint in the deterministic setting.

Definition 2 $\textsc{Dependency}(e, p, c)$ iff $\mathit{Scope}(e) \cap \mathit{Scope}(p) \neq \emptyset$ & $\mathit{Scope}(c) \subseteq \mathit{Scope}(e) \cap \mathit{Scope}(p)$ & $e \wedge (c \Rightarrow p)$, where e, p, and c are all deterministic constraints.

The $\textsc{Dependency}$ meta-constraint is satisfied *if and only if* e is satisfied and, *if* c holds, p is satisfied. We refer to p as a *pre-requisite* constraint for event constraint e, and c as a *condition* constraint for p. Note that by definition, $\textsc{Dependency}(e, p, c \vee c')$ is equivalent to $\textsc{Dependency}(e, p, c) \wedge \textsc{Dependency}(e, p, c')$, and similarly $\textsc{Dependency}(e, p \wedge p', c)$ is equivalent to $\textsc{Dependency}(e, p, c) \wedge \textsc{Dependency}(e, p', c)$.

We now introduce a measure for event realization in a deterministic setting, and later generalize it to probabilistic events.

Definition 3 Given a deterministic event constraint e with $\mathit{Scope}(e) = \{x_1, \ldots, x_k\}$, an event realization measure $E\{e\}$ on e is a mapping M from $D(x_1) \times \cdots \times D(x_k)$ into $\{0, 1\}$ such that for all $t \in D(x_1) \times \cdots \times D(x_k)$, $M(t) = 1$ iff t satisfies all the $\textsc{Dependency}$ constraints that have e as event constraint argument.

Example 1 Given the meta-constraint $\textsc{Dependency}(e, x_1 \leq 4, x_2 \neq 0)$, an event realization measure on event constraint $e : x_1 + x_2 = 8$, denoted by $E\{e\}$, takes value 1 only when the values v_1 and v_2 assigned to decision variables x_1 and x_2 (resp.) sum to 8 and, if x_2 is different than zero, x_1 is less or equal to 4, otherwise it takes value 0.

When the events are probabilistic constraints, the event realization measure is defined on the set of possible worlds as follows.

Definition 4 Given a probabilistic event constraint e with $\mathit{Scope}(e) = \{x_1, \ldots, x_k\}$ and uncertain parameters $\Lambda = \{\lambda_1, \ldots, \lambda_l\}$, an event realization measure $E\{e\}$ on e is a mapping M from $D(x_1) \times \cdots \times D(x_k)$ into interval $[0, 1]$ such that

$$E\{e\} = \sum_{w \in W(\lambda_1) \times \cdots \times W(\lambda_l)} \Pr(w) E\{e_{\downarrow w}\}.$$

Example 2 An event realization measure on probabilistic constraint $e : x_1 + x_2 \leq \xi$, where ξ is a discrete random variable assuming $\{6(0.2), 8(0.7), 11(0.1)\}$, is denoted by $E\{e\}$ and takes the value 0.8 when $x_1 = 4$ and $x_2 = 3$, and the value 0.1 when $x_1 = 6$ and $x_2 = 3$.

For convenience we shall only considered the "expectation operator" in defining an event realization measure. However, any other relevant operator, such as the nth moment generator (Jeffreys 1961), can be used instead.

The following example demonstrates the use of the DEPENDENCY meta-constraint in a probabilistic setting.

Example 3 In Fig. 2 the event e_1 is the demand constraint for the first customer $e_1 : x_{1,1} + x_{2,1} = 8$, while the pre-requisite constraints are the probabilistic supply constraints $p_1 : x_{1,1} + x_{1,2} \leq \xi_1$, $p_2 : x_{2,1} + x_{2,2} + x_{2,3} \leq \xi_2$, and $p_3 : x_{3,2} + x_{3,3} \leq \xi_3$. Now consider event e_1. From the constraint scopes we see that $Scope(e_1) \cap Scope(p_1) = \{x_{1,1}\}$, $Scope(e_1) \cap Scope(p_2) = \{x_{2,1}\}$ and $Scope(e_1) \cap Scope(p_3) = \emptyset$, so e_1 depends on p_1 and p_2, not p_3. From the problem semantics we should introduce the condition constraints $c_1 : x_{1,1} \neq 0$ and $c_2 : x_{2,1} \neq 0$, to express the fact that there is no dependency relation between e_1 and p_1 if $x_{1,1} = 0$, and that there is no dependency relation between e_1 and p_2 if $x_{2,1} = 0$. Thus we write the dependency meta-constraints DEPENDENCY$(e_1, p_1, x_{1,1} \neq 0)$ and DEPENDENCY$(e_1, p_2, x_{2,1} \neq 0)$.

Equipped with these concepts, we now define EDP-CP as follows.

Definition 5 An EDP-CP is a 9-tuple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \Lambda, \mathcal{W}, \mathcal{E}, \mathcal{C}, \mathcal{H}, \Psi, \Pr \rangle$ where:

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of decision variables;
- $\mathcal{D} = D_1 \times \cdots \times D_n$, where D_i is the domain of X_i;
- $\Lambda = \{\lambda_1, \ldots, \lambda_l\}$ is a set of uncertain parameters;
- $\mathcal{W} = W_1 \times \cdots \times W_l$, where W_i the domain of λ_i;
- $\mathcal{E} = \{e_1, \ldots, e_m\}$ is a set of event constraints. Each e_i may either be probabilistic (involving a subset of $\mathcal{X}$ and a subset of Λ) or deterministic (involving only a subset of $\mathcal{X}$);
- $\mathcal{C} = \{c_1, \ldots, c_o\}$ is a set of dependency meta-constraints. For each dependency meta-constraint c_i : DEPENDENCY(e, p, f) we have $e \in \mathcal{E}$, where p may be either a probabilistic or a deterministic pre-requisite constraint, and f is a deterministic condition constraint;
- $\mathcal{H} = \{h_1, \ldots, h_p\}$ is a set of hard constraints. Each h_i may either be probabilistic (involving a subset of $\mathcal{X}$ and a subset of Λ) or deterministic (involving only a subset of $\mathcal{X}$);
- Ψ is any expression involving the event realization measures on the event constraints in $\mathcal{E}$;
- $\Pr : \mathcal{W} \rightarrow [0, 1]$ is a probability distribution over uncertain parameters.

In Fig. 3 we show a modeling template for EDP-CP.

Example 4 The motivational example of Sect. 2 can be expressed as an EDP-CP $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \Lambda, \mathcal{W}, \mathcal{E}, \mathcal{C}, \mathcal{H}, \Psi, \Pr \rangle$ where:

- $\mathcal{X} = \{x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}, x_{2,3}, x_{3,2}, x_{3,3}\}$;
- $\mathcal{D} = [0..99] \times [0..99] \times [0..99]$;
- $\Lambda = \{\xi_1, \xi_2, \xi_3\}$;
- $\mathcal{W} = \{3(0.3), 7(0.5), 12(0.2)\} \times \{6(0.4), 7(0.2), 10(0.4)\} \times \{3(0.3), 8(0.7)\}$;

Fig. 3 An EDP-CP template

Maximize:

$\Psi(E\{e_1\}, \ldots, E\{e_m\})$

Given that:

dependency meta-constraint c_1

...

dependency meta-constraint c_o

Subject to:

hard constraint h_1

...

hard constraint h_p

- $\mathcal{E} = \{e_1 : x_{1,1} + x_{2,1} = 8, e_2 : x_{1,2} + x_{2,2} + x_{3,2} = 7, e_3 : x_{2,3} + x_{3,3} = 4\}$;
- $\mathcal{C} = \{c_1 : \textsc{Dependency}(e_1, p_1 : x_{1,1} + x_{1,2} \le \xi_1, f_{1,1} : x_{1,1} \neq 0),$
  $c_2 : \textsc{Dependency}(e_1, p_2 : x_{2,1} + x_{2,2} + x_{2,3} \le \xi_2, f_{2,1} : x_{2,1} \neq 0),$
  $c_3 : \textsc{Dependency}(e_2, p_1, f_{1,2} : x_{1,2} \neq 0),$
  $c_4 : \textsc{Dependency}(e_2, p_2, f_{2,2} : x_{2,2} \neq 0),$
  $c_5 : \textsc{Dependency}(e_2, p_3 : x_{3,2} + x_{3,3} \le \xi_3, f_{3,2} : x_{3,2} \neq 0),$
  $c_6 : \textsc{Dependency}(e_3, p_2, f_{2,3} : x_{2,3} \neq 0),$
  $c_7 : \textsc{Dependency}(e_3, p_3, f_{3,3} : x_{3,3} \neq 0)\}$;
- $\mathcal{H} = \{x_{1,1} \ge 0, \ldots, x_{3,3} \ge 0\}$;
- Ψ is $E\{e_1\} + E\{e_2\} + E\{e_3\}$;
- $\Pr(\langle \xi_1 = 3, \xi_2 = 6, \xi_3{=}3\rangle) = 0.036, \ldots, \Pr(\langle \xi_1 = 12, \xi_2 = 10, \xi_3 = 8\rangle) = 0.056.$

Finally, we define optimal solutions to EDP-CPs as follows.

Definition 6 An optimal solution to an EDP-CP $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \Lambda, \mathcal{W}, \mathcal{E}, \mathcal{C}, \mathcal{H}, \Psi, Pr\rangle$ is any assignment S to the decision variables such that:

1. for each $h \in \mathcal{H}$, for each $w \in \mathcal{W}$, $h_{\downarrow w}$ is satisfied; and
2. there exists no other assignment satisfying all the hard constraints with a strictly better value for Ψ, according to the Dependency constraints introduced in the model.

Note that when the total number of worlds is 1 with probability 1, the event realization measure on c is the same as in the deterministic case.

4 Solution methods for EDP-CP

We now show how to map an EDP-CP $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \Lambda, \mathcal{W}, \mathcal{E}, \mathcal{C}, \mathcal{H}, \Psi, Pr\rangle$ into an equivalent classical COP $\mathcal{P}' = \langle \mathcal{X}', \mathcal{D}', \mathcal{C}', \Psi'\rangle$.

4.1 Mapping variables and domains

Algorithm 1 shows how to create the decision variables in $\mathcal{P}'$ starting from $\mathcal{P}$, in two steps. The first step (Line 3) duplicates the decision variables in $\mathcal{P}'$ along with their domains. The

Algorithm 1: Variable-Mapping($\mathcal{X}, \mathcal{D}, \Lambda, \mathcal{W}, \mathcal{E}, \mathcal{C}$):$\langle\ \mathcal{X}', \mathcal{D}'\ \rangle$

```
1 $\mathcal{X}' \leftarrow \emptyset$;
2 $\mathcal{D}' \leftarrow \emptyset$;
3 foreach $x \in \mathcal{X}$ do
      create $x'$ with the same domain as $x$ and add it to $\mathcal{X}'$ ;
4 foreach $e \in \mathcal{E}$ do
      foreach $w \in \mathcal{W}$ do
          create a Boolean $b^e_w$ and add it to $\mathcal{X}'$ ;
```

Algorithm 2: Constraint-Mapping($\mathcal{X}, \mathcal{D}, \Lambda, \mathcal{W}, \mathcal{E}, \mathcal{C}, \mathcal{H}$):$\mathcal{C}'$

```
1 $\mathcal{C}' \leftarrow \emptyset$;
2 foreach $e \in \mathcal{E}$ do
      foreach $w \in \mathcal{W}$ do
          $k \leftarrow e_{\downarrow w} \wedge \left[ \bigwedge_{\{\textsc{Dependency}(\epsilon,p,c) \in \mathcal{C} \mid \epsilon = e\}} (c \Rightarrow p_{\downarrow w}) \right]$;
          add $b^e_w = 1 \leftrightarrow k$ to $\mathcal{C}'$ ;
3 foreach $h \in \mathcal{H}$ do
      foreach $w \in \mathcal{W}$ do
          add $h_{\downarrow w}$ to $\mathcal{C}'$ ;
```

second step (Line 4) introduces a Boolean variable that is used later to represent the truth value of each event e in each possible world w.

4.2 Mapping constraints

Algorithm 2 shows how to create the constraints in $\mathcal{P}'$, again in two steps. In step one (Line 2) we introduce a reification constraint for each event e in each possible world $w \in W$. This ensures that b^e_w is assigned the value 1 iff $e_{\downarrow w}$ is satisfied and, for each $\textsc{Dependency}(e, p, c)$ constraint involving event e, if the given condition c is met, the respective prerequisite $p_{\downarrow w}$ is satisfied. In the second step (Line 3) each probabilistic constraint is transformed into a set of deterministic constraints in $\mathcal{C}'$.

4.3 Mapping the objective function

Finally, the objective function of $\mathcal{P}'$ is the same function Ψ as in $\mathcal{P}$, except that we replace each occurrence of an event measure $E\{e\}$ with

$$\sum_{w \in \mathcal{W}} Pr(w) b^e_w$$

as shown in Algorithm 3.

Algorithm 3: Objective-Function-Mapping($\mathcal{X}, \mathcal{D}, \Lambda, \mathcal{W}, \mathcal{E}, \mathcal{C}, \mathcal{H}, \Psi$):$\Psi'$

```
$\Psi' \leftarrow \Psi$;
foreach $E\{e\} \in \Psi'$ do
    replace $E\{e\}$ with $\sum_{w \in \mathcal{W}} \Pr(w) b_w^e$ ;
```

5 Scenario reduction

In the former section we showed how to compile any EDP-CP program in an equivalent ordinary constraint program. Unfortunately the more scenarios we consider the more decision variables need to be introduced in the model. This may easily lead to large intractable problems when the number of scenarios is high. In Tarim et al. (2006) the authors discuss several scenario sampling techniques to cope with a similar problem arising in a scenario based approach for *stochastic constraint programming*. The purpose of these techniques is to replace a large intractable set of scenarios with a small tractable set so that solving the problem over the small set yields a solution not much different than the solution over the large one. Obviously these technique may also be applied to reduce the number of scenarios considered in EDP-CP. The scenario reduction techniques presented are well known in statistics. Typically they determine a subset of scenarios and a redistribution of probabilities relative to the preserved scenarios. No requirements on the stochastic data process are imposed and therefore the concept is general. However the authors point out that, depending on their sophistication, the reduction algorithms may require different types of data.

The simplest scenario reduction algorithm considers just a single scenario in which stochastic variables take their expected values. This is called the *expected value problem*. In what follows we recall one of the best sampling methods for experimental design, that is *Latin Hypercube Sampling* (LHS) (McKay et al. 1979). This method ensures that a range of values for a variable are sampled. Suppose we want the sample size to be n. We divide the unit interval into n intervals, and sample a value for each stochastic variable exactly once. More precisely, let $f_i(a)$ be the cumulative probability that X_i takes the value a or less, $P_i(j)$ be the jth element of a random permutation P_i of the integers $\{0, \ldots, n-1\}$, and r be a random number uniformly drawn from [0, 1]. Then the jth Latin hypercube sample value for the random variable X_i is:

$$f_i^{-1}\left(\frac{P_i(j)+r}{n}\right).$$

However it should be noted that the sample size n does not guarantee to produce a sample of n scenarios, since a single scenario may be chosen more than once due to, for example, the discreteness of the data.

Techniques like the one illustrated may be applied to reduce the number of scenarios to a reasonable size so that the resulting reduced problem is a tractable one. An example of this will be presented in Sect. 6.2, where LHS is applied to a probabilistic scheduling problem in order to reduce the set of scenarios considered and preserve the quality of the solution provided by the EDP-CP model described.

6 Illustrative examples

In this section we present three illustrative problems and model them using the EDP-CP framework. The first example is a probabilistic supply chain planning problem, which is an extended version of the example of Sect. 2. In this extended version, demand uncertainty, as well as supply uncertainty, is considered. The second example is a probabilistic scheduling problem which generalizes the one proposed in Jain and Grossmann (2001). In this example task durations are uncertain. The third example is a production planning problem with an emphasis on capital budgeting, and assumes that production rates, demands, prices and costs are all uncertain parameters.

6.1 An EDP-CP model for probabilistic supply chain planning

There is a sizeable literature on supply chain modeling under uncertainty (see, for example, de Kok and Graves 2003 and Porteus 2002). Recently, the authors of this work also experienced at first-hand the relevance of modeling supply and demand uncertainties during a research project carried out for a leading international telecommunications company.

Here we adopt a simplified version of the problem, which was presented in Sect. 2.1 and Fig. 2. The objective is to determine the most reliable plan that will meet customers' realised demands at $D_{1,2,3}$ by means of uncertain deliveries from suppliers denoted by $S_{1,2,3}$. It is assumed that (i) the order batch sizes $x_{i,j}$ from supplier i to customer j is not allowed to exceed 6 units, $x_i \leq 6$, (ii) D_3 requires that its order is supplied by only one supplier, $x_{2,3}x_{3,3} = 0$. Scenario parameters are given in Table 2. These parameters can be obtained, for instance, through a sampling method like LHS, which we presented in the former section. Excess supplies from suppliers are stored at customers with a negligible inventory carrying cost until the next order issue.

We consider two possible EDP-CP models for this probabilistic supply chain problem. In the first one we try to find a solution in which all events are realised, while in the second

Table 2 Scenario data

Pr(w)	0.036	0.084	0.018	0.042	0.036	0.084	0.060	0.140	0.030
w	1	2	3	4	5	6	7	8	9
S_1	3	3	3	3	3	3	7	7	7
S_2	6	6	7	7	10	10	6	6	7
S_3	3	8	3	8	3	8	3	8	3
D_1	8	8	8	7	7	7	8	8	8
D_2	7	7	7	5	5	5	5	7	7
D_3	4	6	6	4	4	6	6	4	4

Pr(w)	0.070	0.060	0.140	0.024	0.056	0.012	0.028	0.024	0.056
w	10	11	12	13	14	15	16	17	18
S_1	7	7	7	12	12	12	12	12	12
S_2	7	10	10	6	6	7	7	10	10
S_3	8	3	8	3	8	3	8	3	8
D_1	9	9	9	8	8	8	7	7	7
D_2	5	5	5	5	3	3	3	5	5
D_3	6	6	4	4	6	6	4	4	6

Maximize:

$$
\begin{aligned}
&E\{e_1 : x_{1,1} + x_{2,1} \geq \zeta_1\}\\
&\quad + E\{e_2 : x_{1,2} + x_{2,2} + x_{3,2} \geq \zeta_2\}\\
&\quad + E\{e_3 : x_{2,3} + x_{3,3} \geq \zeta_3\}
\end{aligned}
$$

Given that:

$$
\begin{aligned}
&\textsc{Dependency}(e_1, p_1 : x_{1,1} + x_{1,2} \leq \xi_1, f_{1,1} : x_{1,1} \neq 0)\\
&\textsc{Dependency}(e_1, p_2 : x_{2,1} + x_{2,2} + x_{2,3} \leq \xi_2, f_{2,1} : x_{2,1} \neq 0)\\
&\textsc{Dependency}(e_2, p_1, f_{1,2} : x_{1,2} \neq 0)\\
&\textsc{Dependency}(e_2, p_2, f_{2,2} : x_{2,2} \neq 0)\\
&\textsc{Dependency}(e_2, p_3 : x_{3,1} + x_{3,2} \leq \xi_3, f_{3,2} : x_{3,2} \neq 0)\\
&\textsc{Dependency}(e_3, p_2, f_{2,3} : x_{2,3} \neq 0)\\
&\textsc{Dependency}(e_3, p_3, f_{3,3} : x_{3,3} \neq 0)
\end{aligned}
$$

Subject to:

$$
\begin{aligned}
&0 \leq x_{i,j} \leq 6, \ \forall i, j \in \{1, 2, 3\}\\
&x_{2,3}.x_{3,3} = 0\\
&e_i, \ \forall i \in \{1, 2, 3\}\\
&x_{i,j} \in \mathbb{Z}^{0,+}
\end{aligned}
$$

Fig. 4 An EDP-CP model for probabilistic supply chain planning

this condition is relaxed. The EDP-CP model in Fig. 4 describes the first case. The second case can be simply achieved by dropping e_1–e_3 from the set of hard constraints.

The EDP-CP model is compiled into a standard CP model using the algorithm presented in Sect. 4. The optimal solution is $x_{1,1} = 6$, $x_{1,2} = 1$, $x_{2,1} = 3$, $x_{2,2} = 4$, $x_{2,3} = 0$, $x_{3,2} = 2$, $x_{3,3} = 6$. In the optimal plan $E\{e_1\} = 0.420$, $E\{e_2\} = 0.294$ and $E\{e_3\} = 0.700$, giving an optimal objective function value of 1.414. In other words, this plan guarantees to meet customer demands at $D_{1,2,3}$ with probabilities 42.0%, 29.4% and 70.0%, respectively. This plan aims to satisfy customer demands completely.

In most circumstances it would be more realistic to assume that the event constraints e_1, e_2, and e_3 are not hard constraints and the expected plan should not aim for a complete demand satisfaction. When we drop these hard event constraints, the following plan is optimal under such a relaxation: $x_{1,1} = 6$, $x_{1,2} = 0$, $x_{2,1} = 3$, $x_{2,2} = 3$, $x_{2,3} = 0$, $x_{3,2} = 2$, $x_{3,3} = 6$. The event constraint satisfaction probabilities are now $E\{e_1\} = 0.700$, $E\{e_2\} = 0.476$ and $E\{e_3\} = 0.700$, giving a total of 1.876.

A comparison of two plans shows that there are differences between them at $x_{1,2}$ and $x_{2,2}$. It may not be immediately obvious why we change $x_{1,2}$ from 1 to 0, as in both plans the probability of acquiring the required capacity at S_1 (7 and 6, respectively) is 0.8. The explanation lies in the probability distribution of the uncertain capacity of S_2. Supplier S_2 can provide 6 units with a probability of 1.0, but not 7 units. The second plan exploits this situation and aims for a partial satisfaction at D_2 by providing only 5 units. Thus there is no need for any delivery from S_1 to D_2. The second plan has higher reliability at the expense of partial satisfaction at D_2. It should be noted that there are alternative optimal solutions to this instance.

6.2 An EDP-CP model for scheduling

We consider a specific scheduling problem similar to the one considered by Hooker et al. (1999). This scheduling problem was described in Jain and Grossmann (2001) and it involves finding a least-cost schedule to process a set of orders I using a set of dissimilar parallel machines M. Processing an order $i \in I$ can only begin after the release date r_i and must be completed at the latest by the due date d_i. Order i can be processed on any of the machines. The processing cost and the processing time of order $i \in I$ on machine $m \in M$ are c_{im} and p_{im}, respectively.

The model just described is fully deterministic, but we will now consider a generalization of this problem to the case where some inputs are uncertain. For convenience we will just consider uncertain processing times π_{im} for order $i \in I$ on machine $m \in M$. Nevertheless it is easy to see that EDP-CP can be also employed to model more complicated generalizations of this problem where release dates and due dates are uncertain or processing costs are uncertain.

Scheduling with uncertainty is a topic that has been explored in a variety of fields including artificial intelligence, operations research, fault-tolerant computing and systems. For surveys on the literature see Davenport and Beck (2000), Herroelen and Leus (2005), and Bidot (2005). In Beck and Wilson (2007) a classification of possible approaches for scheduling under uncertainty is summarized. They report three techniques that are usually employed to face uncertainty. In redundancy-based techniques extra resources/time are allocated to every task to cushion the impact of unexpected events during execution. Probabilistic techniques instead tend to build a schedule that optimizes a measure of probabilistic performance, such as expected makespan or expected weighted tardiness. Contingent and policy based approaches typically generate a branching or contingent schedule or, in extreme cases, a policy, that specify a set of actions to be taken when a particular set of circumstances arises. Our EDP-CP approach can be classified as probabilistic under a predefined policy.

Since EDP-CP is meant to model and optimize the reliability of a given plan we will assume in our problem that a fixed budget B is given and that our plan has to meet such a constraint on the costs. Therefore we will no longer look for a least-cost plan, rather we will optimize a reliability measure expressed in terms of events, as it is usual in EDP-CP. The specific event whose probability we wish to maximize is the successful completion of each job within the given time frame defined by its release and due date. Since jobs are scheduled in sequence on each machine dependencies will arise between subsequent jobs. We adopt a specific policy that unschedules a job whether this is not processed within the given due date or before the planned start time of the subsequent job on the respective machine. This policy guarantees that every order will always start at the planned start time, since the respective machine will be free and will start processing it. More complicated EDP-CP models may also consider the case where we aim to minimize total tardiness or total completion time of a given plan. In this cases the realized processing time of an order may affect the scheduling time of subsequent orders. We will not analyze these cases in this example. An EDP-CP model for the problem described is given in Fig. 5. Let us analyze the given model. The objective function maximizes the expected number of tasks completed by the respective due dates. DEPENDENCY constraint states that, when two jobs i, j are executed in sequence on the same machine (condition $\sigma_{ij} = 1$), job i has to be completed by its due date (event e_i is satisfied) and before the start time of job j (pre-requisite $s_j \geq s_i + \sum_{m \in M} \pi_{im} * \delta_{im}$). The hard constraints respectively state that: the start time of job i, s_i, must be no less than the release time r_i for this job; if two jobs i, j are processed on the same machine m and i is processed before j then the start time of i, s_i, must be less than the start time of j, s_j; each

Maximize:

$$\sum_{i\in I} E\{e_i : s_i + \sum_{m\in M} \pi_{im} * \delta_{im} \leq d_i\}$$

Given that:

$$\textsc{Dependency}(e_i, s_j \geq s_i + \sum_{m\in M} \pi_{im} * \delta_{im}, \sigma_{ij} = 1), \forall i, j \in I, i \neq j$$

Subject to:

$$\begin{aligned}
&s_i \geq r_i, \forall i \in I\\
&\sigma_{ij} = 1 \Rightarrow s_i < s_j, \forall i, j \in I, i \neq j\\
&\sum_{m\in M} \delta_{im} = 1, \forall i \in I\\
&\sigma_{ij} + \sigma_{ji} \geq \delta_{im} + \delta_{jm} - 1, \forall m \in M, \forall i, j \in I, i \neq j\\
&\sigma_{ij} + \sigma_{ji} \leq 1, \forall i, j \in I, i \neq j\\
&\sum_{i\in I} (\sum_{m\in M} c_{im} * \delta_{im}) \leq B
\end{aligned}$$

$$\begin{aligned}
&\sigma_{ij} \in \{0, 1\}, \forall i, j \in I\\
&\delta_{im} \in \{0, 1\}, \forall i \in I, \forall m \in M\\
&s_i \in [L_s, L_e], \forall i \in I
\end{aligned}$$

Fig. 5 An EDP-CP model for scheduling

job must be processed on a machine; if two jobs i, j are processed on the same machine m, either i is processed before j, $\sigma_{ij} = 1$, or j is processed before i, $\sigma_{ji} = 1$; the processing costs must be no greater than the given budget B.

We now consider an instance of this problem. We consider 5 orders $\{I_1, \ldots, I_5\}$ on 2 parallel machines $\{M_1, M_2\}$. The uncertain processing times of each order on each machine are shown in Table 3. The release dates for the orders are $[2, 4, 6, 8, 10]$. The due dates are $[16, 13, 30, 41, 35]$. The costs for processing orders on machine M_1 are $[10, 8, 12, 11, 9]$, and on machine M_2 they are $[16, 5, 17, 9, 4]$. The given budget B is 40.

We define

$$L_s = \min_{i\in I} r_i$$

and

$$L_e = L_s + \min_{m\in M} \sum_{i\in I} \lceil \pi_{im} \rceil$$

where $\lceil \pi_{im} \rceil$ is the maximum duration of order $i \in I$ on machine $m \in M$ for every possible world $w \in W$. Therefore

$$\lceil \pi_{im} \rceil = \max_{w\in W} \pi_{im}.$$

In order to solve the proposed scheduling problem we compiled the EDP-CP model into a standard constraint program as described in Sect. 4. This constraint program was solved using OPL Studio 3.7 on an Intel(R) Centrino(TM) CPU 1.50 GHz with 2 Gb RAM. We chose the provided *dichotomic* strategy and *depth-bounded discrepancy search* procedure.

Table 3 Order processing times

	$w \in W$	1		2		3		4		5		6	
	$\Pr\{w\}$	0.1		0.1		0.05		0.1		0.05		0.2	
	$m \in M$	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2
$i \in I$	1	10	14	9	15	11	13	9	14	9	15	11	13
	2	6	8	5	9	7	7	7	8	5	9	7	12
	3	11	16	10	18	12	15	4	16	10	18	14	15
	4	7	9	6	10	8	8	8	9	6	10	8	8
	5	12	17	11	18	13	16	12	17	4	18	13	16

	$w \in W$	7		8		9		10		11		12	
	$\Pr\{w\}$	0.05		0.05		0.1		0.05		0.05		0.1	
	$m \in M$	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2
$i \in I$	1	10	14	9	15	11	13	10	14	9	15	11	13
	2	6	8	15	9	7	7	16	8	5	9	7	7
	3	16	16	10	18	10	15	11	16	10	18	12	15
	4	7	9	6	10	8	8	17	9	6	10	8	8
	5	12	17	11	18	13	16	12	17	11	18	13	6

An optimal solution for the given instance was found in 6.97 seconds, it has a cost of 40 and an overall reliability measure of 4.8, which means that in our plan 4.8 orders over 5 will be, in the average case, processed within the required due date and before the next order scheduled on the same machine. More specifically, the realization measures for each event constraint are: $E\{e_1\} = 100\%$, $E\{e_2\} = 80\%$, $E\{e_3\} = 100\%$, $E\{e_4\} = 100\%$ and $E\{e_5\} = 100\%$. The optimal plan assigns orders $\{1, 3\}$ to M_1 and orders $\{2, 4, 5\}$ to M_2. The start times for the orders are $[2, 4, 13, 31, 13]$.

In order to reduce the size of the model input we will now perform a LHS on the original problem instance presented in Table 3. The original 12 scenarios are then reduced to only 4 sampled scenarios. The reduced instance is presented in Table 4. The optimal solution for the LHS reduced instance was found in 2.08 seconds, it has a cost of 40 and an overall reliability measure of 4.8. More specifically, the optimal plan assigns orders $\{1, 3\}$ to M_1 and orders $\{2, 4, 5\}$ to M_2. The start times for the orders are $[2, 4, 13, 31, 13]$. This is the same optimal plan found for the original problem with 12 scenarios.

We now reduce the number of scenarios to only 2 samples as shown in Table 5. The optimal solution was found in only 0.72 seconds and also in this case it corresponds to the same optimal plan described above.

We finally solved the *expected value problem* in which the random order processing times are replaced with their expected values. The expected times for processing orders on machine M_1 are $[10, 7, 11, 8, 12]$, and on machine M_2 they are $[14, 9, 16, 9, 16]$. The optimal solution for the expected value problem was found in 0.25 seconds, it has a cost of 40 and an overall reliability measure of 3.55. More specifically, the optimal plan assigns orders $\{1, 3\}$ to M_1 and orders $\{2, 4, 5\}$ to M_2. The start times for the orders are $[2, 4, 12, 32, 16]$. This plan is 26.04% less reliable than the previous ones.

As this simple example demonstrates, the expected value approach to probabilistic problems may produce solutions which are far from being close to the optimal solutions, while

Table 4 Order processing times, LHS with 4 samples

	$w \in W$	2		6		7		10	
	$\Pr\{w\}$	0.25		0.25		0.25		0.25	
	$m \in M$	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2
$i \in I$	1	9	15	11	13	10	14	10	14
	2	5	9	7	12	6	8	16	8
	3	10	18	14	15	16	16	11	16
	4	6	10	8	8	7	9	17	9
	5	11	18	13	16	12	17	12	17

Table 5 Order processing times, LHS with 2 samples

	$w \in W$	3		11	
	$\Pr\{w\}$	0.5		0.5	
	$m \in M$	M_1	M_2	M_1	M_2
$i \in I$	1	11	13	9	15
	2	7	7	5	9
	3	12	15	10	18
	4	8	8	6	10
	5	13	16	11	18

a sampling approach usually brings benefits in terms of processing time without sacrificing too much the optimality of the solution produced.

6.3 An EDP-CP model for production planning/capital budgeting

In this section, a production planning problem with an emphasis on capital budgeting is used to demonstrate the flexibility of the proposed modeling framework in dealing with uncertainties.

The production planning/capital budgeting problem assumes that there are $n = 7$ types of products to be produced, under uncertain demands d_i, $i = 1, \dots, 7$. Each product can be produced on only one type of machine which is assigned to this product only. The existing production floor space is $A = 50$ m^2, in which each machine type requires m_i ($m = [3, 6, 5, 3, 7, 8, 9]$) in m^2 per machine of type i. The cost of operating each machine involves two types of costs: fixed cost f_i ($f = [40, 75, 62, 39, 53, 19, 38]$) and variable production cost c_i. The total production budget is $B = \$670$. The variable production cost components $c_{1,\dots,7}$ are uncertain, taking different values in each world $w_{1,\dots,4}$ (see Table 6). The produced amount of each product depends on the number of machines used, x_i, and the uncertain machine production rate, r_i, is also given in Table 6. Table 6 shows two more uncertain problem parameters: demand d_i and selling price p_i.

Under these uncertainties, a realistic objective is to determine the most reliable plan (i.e. how many machines to purchase of each type) that maximizes our chances of meeting our demand constraints as much as possible, while achieving a specified target profit of $T = \$40$, not exceeding our budget B, and meeting all space and production constraints. It is assumed that meeting customer demands and the profit target are equally important events.

In specific solution/plans, depending on unfolding of uncertainties, the budget constraint may hold, as well as demand and target profit objectives. It should be noted that it is not gen-

Table 6 Problem data

w	Pr	Production cost							Demand						
		c_1	c_2	c_3	c_4	c_5	c_6	c_7	d_1	d_2	d_3	d_4	d_5	d_6	d_7
1	0.16	3	6	1	1	6	10	2	4	7	2	8	3	5	2
2	0.19	4	4	7	2	4	7	7	7	9	9	9	4	7	4
3	0.38	5	3	5	8	7	6	10	9	11	12	10	7	8	7
4	0.27	5	6	8	5	5	3	6	11	13	17	11	13	16	13

w	Pr	Selling price							Production rate						
		p_1	p_2	p_3	p_4	p_5	p_6	p_7	r_1	r_2	r_3	r_4	r_5	r_6	r_7
1	0.16	8	14	4	16	14	10	4	2	3	2	1	2	1	2
2	0.19	10	16	18	18	10	14	14	4	4	5	2	4	3	6
3	0.38	18	22	14	18	14	16	24	5	5	6	3	5	5	4
4	0.27	22	26	26	22	16	24	18	9	6	8	4	7	7	7

Maximize:

$$\frac{1}{2n}\sum_{i=1}^{n} E\{e_i : \min(r_i x_i, d_i) = d_i\}$$
$$+\ \frac{1}{2}E\{\bar{e} : \textstyle\sum_{i=1}^{n} p_i \min(r_i x_i, d_i) - f_i x_i - c_i r_i x_i \geq T\}$$

Given that:

$$\textsc{Dependency}(e_j, \textstyle\sum_{i=1}^{n}(f_i + c_i r_i)x_i \leq B, \text{True}),\ \forall j \in \{1, \dots, n\}$$
$$\textsc{Dependency}(\bar{e}, \textstyle\sum_{i=1}^{n}(f_i + c_i r_i)x_i \leq B, \text{True})$$

Subject to:

$$\sum_{i=1}^{n} m_i x_i \leq A$$
$$x_i \in \mathbb{Z}^{0,+}$$

Fig. 6 An EDP-CP model for production planning/capital budgeting

erally possible to find a solution which always satisfies all the constraints. For that reason, the problem addressed here is very different from the well-established techniques dealing with uncertainty.

An EDP-CP model of the production planning/capital budgeting problem is shown in Fig. 6, where $r_i x_i$ and $\min(r_i x_i, d_i)$ denote the amount produced and sold, respectively, of product type $i \in \{1, \dots, n\}$, and x_i denotes the number of machine used in the production of type i product. There is only one pre-requisite constraint (the budget constraint) and no condition constraint.

The optimal solution found is $x^* = [2, 0, 2, 0, 0, 2, 2]$. This production plan gives $E\{e_1\} = 100\%$, $E\{e_2\} = 0$, $E\{e_3\} = 73\%$, $E\{e_4\} = 0$, $E\{e_5\} = 0$, $E\{e_6\} = 38\%$, $E\{e_7\} = 100\%$, where event constraint e_i denotes the complete satisfaction of demand for product type i. In this plan the profit target is achieved $E\{\bar{e}\} = 65\%$ of the time.

Table 7 Expected value problem data

Product type	1	2	3	4	5	6	7
Production cost	4.49	4.48	5.55	4.93	5.73	6.02	7.07
Demand	8.36	10.52	11.18	9.76	7.41	9.49	7.25
Selling price	15.96	20.66	16.40	18.76	13.78	16.82	17.28
Production rate	5.41	4.76	5.71	2.76	4.87	4.52	4.87

We also solved the *expected value problem* in which the random variables production cost, demand, selling price and production rate are replaced with their expected values. The expected value data used in the deterministic problem are given in Table 7.

The solution to this resultant deterministic problem is $x^* = [2, 0, 0, 4, 2, 0, 2]$. We used this plan in the original probabilistic setting to evaluate the quality of the expected value solution. The expected value solution gives $E\{e_1\} = 35\%$, $E\{e_2\} = 0$, $E\{e_3\} = 0\%$, $E\{e_4\} = 0$, $E\{e_5\} = 35$, $E\{e_6\} = 0\%$, $E\{e_7\} = 35\%$, where event constraint e_i denotes the complete satisfaction of demand for product type i. It is not possible to achieve the profit target under any scenario using this plan; in other words, $E\{\bar{e}\} = 0\%$.

Also in this case the expected value approach to probabilistic problems produces a solution which is far from being close to the optimal one.

7 Related works

The EDP-CP framework we present is a generalization of the work of Liu and Iwamura (1997) on dependent-chance programming. Firstly, our notion of constraint dependency introduces condition constraints in addition to the event and pre-requisite constraints. It should be noted that constraint dependency without condition constraints does not guarantee optimal plans since in certain instances common variables may take values which break the link between two dependent constraints. Secondly, while a feasible solution in Liu's framework satisfies all event constraints, in our framework such a requirement is relaxed, and this gives the decision-maker more flexibility in modeling. Finally, while Liu's work only considers Monte Carlo-based simulation methods, we propose a complete solution method.

EDP-CP is also related to the probabilistic CSP framework (Fargier et al. 1995). However, probabilistic CSP treats all probabilistic constraints uniformly, whereas EDP-CP distinguishes between event, pre-requisite, condition, and hard constraints. For instance, in probabilistic CSP, all customer and demand constraints will be treated in the same way. In a given world, either all constraints are satisfied or the problem is over-constrained. While finding a plan that has the highest probability of success is an interesting objective, our approach answers different questions and achieves different objectives.

It should also be noted that, when all the constraints are deterministic, our EDP-CP framework is closely related to Partial CSPs (Freuder and Richard 1992). Partial CSPs can be divided into two main categories: The Minimal Violation Problem and the Maximal Utility Problem. In the first case the goal is to find a solution which satisfies as many constraints as possible (e.g. Soft CSPs Bistarelli et al. 2002) or equivalently to minimise the number of violated constraints. In the Maximal Utility Problem the objective is to find a partial solution, which violates none of the constraints where a partial solution is an assignment in which not all variables are assigned a value. In our approach we also find partial solutions, but instead of treating all constraints equally we have shown that we can obtain partial assignments that satisfy as many event constraints as possible according to the given probability distributions for the random variables and to the dependencies that have been modeled. Partial CSPs do not explicitly model high level concepts such as probability distributions, event, pre-requisite, condition, and hard constraints.

Another technique addressing constraint problems under uncertainty is Stochastic Constraint Programming (SCP) (Tarim et al. 2003). The SCP approach assumes that the constraints are stochastically independent (i.e., there are no DEPENDENCY constraints among them). Thus SCP addresses a completely different class of stochastic problems.

8 Conclusion

In this paper we propose EDP-CP as a novel modeling framework that helps decision makers in uncertain environments to realistically model their problems and find reliable solutions. The characteristic features of our modeling framework can be summarized as follows:

- To better model the uncertainties in real-world problems, we allow the set of constraints to be either deterministic or probabilistic;
- We move away from classical approaches that treat all constraints uniformly to one that distinguishes between event, pre-requisite, condition, and hard constraints;
- We introduce the DEPENDENCY meta-constraint that allows the modeler to state a problem by explicitly specifying dependency relationships between event, pre-requisite, and condition constraints;
- In an uncertain environment, it is quite unrealistic to assume that a solution is valid irrespective of the unfolding of the uncertain parameters. In fact, there is a certain degree of fuzziness associated with each candidate solution. Therefore, in our framework, we view the set of feasible solutions as probabilistic due to the inherent uncertainties;
- We introduce an event realization measure, which can be used by the modeler to define solution reliability.

Our future work will extend the proposed framework in various directions, and provide efficient and effective solving methods. Our first steps will be:

- The development of specialized solution methods for EDC-CP. For instance a specialized global constraint for the DEPENDENCY meta-constraint can be designed;
- In large-scale uncertain problems, the number of worlds can be prohibitively large. We proposed a well-known scenario reduction technique that may help to reduce the number of scenarios considered. However we will investigate further ways of reducing the number of world as well as employing effective decomposition techniques;
- We will look at ways of extending EDP-CP to deal with recourse actions.

Acknowledgements S. Armagan Tarim and Brahim Hnich are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027. Roberto Rossi is supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain-Driven Research (CTVR) and Grant No. 05/IN/I886.

References

Beck, J. C., & Wilson, N. (2007). Proactive algorithms for job shop scheduling with probabilistic durations. *Journal of Artificial Intelligence Research*, *28*, 183–232.

Bidot, J. (2005). *A general framework integrating techniques for scheduling under uncertainty*. Ph.D. thesis, Ecole Nationale d'Ingèieurs de Tarbes.

Birge, J. R., & Louveaux, F. (1997). *Introduction to Stochastic Programming*. New York: Springer.

Bistarelli, S., Montanari, U., & Rossi, F. (2002). Soft constraint logic programming and generalized shortest path problems. *Journal of Heuristics*, *8*(1), 25–41.

Charnes, A., & Cooper, W. W. (1959). Chance-constrainted programming. *Management Science*, *6*(1), 73–79.

Davenport, A., & Beck, J. C. (2000). *A survey of techiniques for scheduling with uncertainty* (Technical Report). Available at: http://www.tidel.mie.utoronto.ca/publications.php.

de Kok, A. G., & Graves, S. C. (2003). *Handbooks in operations research and management science: supply chain management: design, coordination and operation* (Vol. 11). Amsterdam: Elsevier.

Fargier, H., Martin-Clouaire, R., Lang, J., & Schiex, T. (1995). A constraint satisfaction framework for decision under uncertainty. In *Proceedings of the eleventh international conference on uncertainty in artificial intelligence*, Montreal, Canada.

Freuder, E. C., & Richard, J. W. (1992). Partial constraint satisfaction. *Artificial Intelligence*.

Herroelen, W., & Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, *165*, 289–306.

Hooker, J. N., Ottosson, G., Thorsteinsson, E. S., & Kim, H. J. (1999). On integrating constraint propagation and linear programming for combinatorial optimization. In *Proceedings of the sixteenth national conference on artificial intelligence* (pp. 136–141). Menlo Park/Cambridge: AAAI Press/MIT Press.

Jain, V., & Grossmann, I. E. (2001). Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS Journal on Computing*, *13*, 258–276.

Jeffreys, H. (1961). *Theory of probability*. Oxford: Clarendon.

Kall, P., & Wallace, S. W. (1994). *Stochastic programming*. New York: Wiley.

Kingsman, B. G. (1985). *Raw materials purchasing: an operational research approach*. Elmsford: Pergamon.

Liu, B. (1995a). *Dependent-chance goal programming: a class of stochastic programming* (Technical Report). Institute of System Science, Chinese Academy of Sciences.

Liu, B. (1995b). *Dependent-chance goal programming and its genetic algorithm approach* (Technical Report). Institute of System Science, Chinese Academy of Sciences.

Liu, B. (1997). Dependent-chance programming: A class of stochastic optimization. *Computers & Mathematics with Applications*, *34*, 89–104.

Liu, B. (1999). *Uncertain programming*. New York: Wiley.

Liu, B., & Iwamura, K. (1997). Modelling stochastic decision systems using dependent-chance programming. *European Journal of Operational Research*, *101*, 193–203.

Liu, B., & Ku, C. (1993). Dependent-chance goal programming and an application. *Journal of Systems Engineering & Electronics*, *4*, 40–47.

McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, *21*, 239–245.

Porteus, E. L. (2002). *Foundations of stochastic inventory theory*. Stanford: Stanford University Press.

Sengupta, J. K. (1972). *Stochastic programming: methods and applications*. Amsterdam: North-Holland.

Tarim, S. A., Manandhar, S., & Walsh, T. (2003). Scenario-based stochastic constraint programming. In *Proceedings of the eighteenth international joint conference on artificial intelligence*, Acapulco, Mexico (pp. 257–262).

Tarim, S. A., Manandhar, S., & Walsh, T. (2006). Stochastic constraint programming: A scenario-based approach. *Constraints*, *11*, 53–80.

Vajda, S. (1972). *Probabilistic programming*. San Diego: Academic Press.

Wu, Y., Zhou, J., & Yang, J. (2005). Dependent-chance programming model for stochastic network bottleneck capacity expansion based on neural network and genetic algorithm. In *Lecture notes in computer science: Vol. 3612. Advances in Natural Computation* (pp. 120–128). Berlin: Springer.

European Journal of Operational Research 198 (2009) 983–986

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Short Communication

A note on Liu–Iwamura's dependent-chance programming

Roberto Rossi [a,d,*], S. Armagan Tarim [b], Brahim Hnich [c], Steven Prestwich [d], Cahit Guran [e]

[a] Centre for Telecommunication, Value-Chain Research, Ireland
[b] Department of Management, Hacettepe University, Ankara, Turkey
[c] Faculty of Computer Science, Izmir University of Economics, Izmir, Turkey
[d] Cork Constraint Computation Centre, University College, 14th Washington Street West, Cork, Ireland
[e] Department of Public Finance, Hacettepe University, Ankara, Turkey

ARTICLE INFO

Article history:
Received 5 February 2008
Accepted 4 November 2008
Available online 14 November 2008

Keywords:
Inventory control
Integer programming
Constraint programming
Demand uncertainty

ABSTRACT

Sometimes a complex stochastic decision system undertakes multiple tasks called events, and the decision-maker wishes to maximize the chance functions which are defined as the probabilities of satisfying these events. Originally introduced by Liu and Iwamura [B. Liu, K. Iwamura, Modelling stochastic decision systems using dependent-chance programming, European Journal of Operational Research 101 (1997) 193–203], dependent-chance programming is aimed at maximizing some chance functions of events in an uncertain environment. In this work, we show that the original dependent chance-programming framework needs to be extended in order to capture an exact reliability measure for a given plan.

1. Introduction

Chance-constrained programming, pioneered by Charnes and Cooper [1], provides a means of handling uncertainty by specifying a confidence level at which it is desired that the stochastic constraint holds. Chance-constrained programming models can be converted into deterministic equivalents only for some special cases, and then solved by using solution methods of deterministic mathematical programming. In order to overcome this difficulty, Liu [4] provided a new stochastic programming framework, called *dependent-chance programming*, in which a complex stochastic decision system undertakes multiple tasks called events, and the decision-maker wishes to maximize the chance functions which are defined as the probabilities of satisfying these events. Liu and Iwamura [6] proposed a stochastic simulation-based genetic algorithm for solving general chance-constrained programming as well as chance-constrained multi-objective programming, and chance-constrained goal programming (for a more detailed discussion see [5]).

Roughly speaking, dependent-chance programming is aimed at maximizing some chance functions of events in an uncertain environment. In deterministic mathematical programming the feasible set is essentially assumed to be deterministic and the optimal solution can always be implemented. However when uncertainty is taken into account the given solution may be infeasible if the realization of uncertain parameters is unfavorable. In other words, the feasible set of dependent chance-programming is described by a so-called *uncertain environment*. Although a deterministic solution is given by the dependent chance-programming model, this solution needs to be as flexible as possible with respect to the uncertain environment. This special feature of dependent chance-programming is very different from other existing stochastic programming frameworks. However, such problems do exist in the real world. Some applications of dependent chance programming have been presented by Liu and Ku [7], Liu [2,3], Liu and Iwamura [6], and more recently by Wu et al. [8].

In this note, we argue that the original dependent chance-programming framework proposed by Liu and Iwamura needs to be extended in order to capture an exact notion of reliability and we show that the way Liu and Iwamura express constraint dependencies, without taking into account the values assigned to decision variables, does not guarantee optimal plans since in certain instances common variables may take values which break the link between dependent constraints.

This paper is organized as follows: In Section 2, we recall the dependent-chance programming framework proposed by Liu and Iwamura. In Section 3, we describe a motivational water supply-allocation problem originally proposed in [4] and we analyze the reliability of different distribution plans according to their framework. In Section 4, we propose an exact notion of reliability obtained by expressing constraint dependencies taking into account

[*] Corresponding author. Address: Cork Constraint Computation Centre, University College, 14th Washington Street West, Cork, Ireland. Tel.: +353 (0)85 122 3582; fax: +353 (0)21 425 5424.

E-mail addresses: rrossi@4c.ucc.ie (R. Rossi), armagan.tarim@hacettepe.edu.tr (S. Armagan Tarim), brahim.hnich@ieu.edu.tr (B. Hnich), s.prestwich@4c.ucc.ie (S. Prestwich), guran@hacettepe.edu.tr (C. Guran).

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.11.004

the values assigned to decision variables. An exact reliability measure is then proposed for the distribution plans being analyzed. In Section 5, we draw conclusions.

2. Formal background

This section presents a summary of dependent-chance programming of Liu [2,3] and underlying concepts.

If Ω is a collection of objects denoted generally by x, then the *stochastic set A* in Ω is defined as a set of ordered pairs:

$$A = \{(x, \mu_A(x)) | x \in \Omega\},$$

where $\mu_A(x)$ is called the probability function of x in A. In uncertain environments, the feasible set, represented by a series of stochastic constraints, may be described by a stochastic set. In contrast to the deterministic case, we cannot say a point is feasible or not when our problem is defined on a stochastic set. We have to say a point x^* is feasible with probability α, where α is the value of probability function $\mu_A(x^*)$.

Usually, a solution x is a vector composed of n components, $x_1, x_2, \ldots, x_n$. We will suppose that we know the following relationship among the decision components.

Stochastic relationship: there is a known partition of n components of a decision vector into k groups such that these k groups are mutually stochastically independent and in each group any elements are stochastically dependent and have the same chance to appear if they require to be realized simultaneously.

Thus, in stochastic decision systems, the feasible set of decision vectors is represented by a stochastic set, say S, whose probability function is $\mu_S(x)$.

Next we consider the purpose of our system. Usually there are multiple purposes, functions or tasks of a complex system. Liu denotes the actions meeting the purposes or performing the tasks as *events*. Each event is represented by a set E which is composed of all the possible decisions meeting certain conditions. Let $V(E)$ denote the set of all components of x which are necessary to the event E and $D(E)$ be the set of all components which are stochastically dependent of any elements in $V(E)$. It is clear that $V(E) \subset D(E)$.

For each element of an event E, we have to give an evaluation, i.e. criterion function, of a decision vector. In view of the uncertainty of the stochastic decision system, we are not certain whether a decision is feasible before knowing the realization of stochastic parameters, so we employ *chance functions* as objective functions to evaluate some of the events. Generally, the chance function, denoted by $f(x)$, is the probability function on the event E.

Thus, for single event case, the dependent-chance programming (DCP) is given as follows:

$$\max_{x \in S} f(x), \tag{1}$$

where x is an n-dimensional decision vector, S is a stochastic set on $\mathbb{R}^n$ with probability function $\mu_S(x)$, $f(x)$ is a chance function of a certain event, borrowing the symbol $\in$ from classical set theory, $x \in S$ means x is feasible with probability $\mu_S(x)$. A point $x^* \in S$ is called an optimal solution of the problem in Eq. (1) if $f(x^*) \geqslant f(x)$ for any $x \in S$.

As an extension, the dependent-chance multiobjective programming (DCMOP) for multiple events case is given as follows,

$$\max_{x \in S} f(x) = [f_1(x), f_2(x), \ldots, f_m(x)], \tag{2}$$

where $f(x)$ is a vector of real-valued functions f_i which are chance or deterministic functions.

In Liu and Iwamura [6] the authors highlight that the key aspect of algorithm for solving DCP, DCMOP and DCGP (i.e. dependent-chance goal programs, for a detailed discussion refer to [6]) consists in constructing the relationship between the decision vectors and chance functions. They consider a set of t objectives $f_i(x), i = 1, 2, \ldots, t$. They assume that every $f_i(x)$ is a chance function that represents a probability of a certain event which is represented by E_i. Then they define

$$E = E_1 \cap E_2 \cap \cdots \cap E_t,$$

and

$$V(E) = V(E_1) \cup V(E_2) \cup \cdots \cup V(E_t).$$

In order to realize each event E_i, as far as possible without sacrificing the chances of other events, they treat all elements in the stochastically dependent set $D(E_i)$ of $V(E_i)$ at an equitable level, i.e., these elements would have the same chance to be realized. On the other hand they disregard elements out of $V(E)$ because they do not make any contribution to the events that have to be realized. Thus the authors consider all the elements in and only in $D(E_i) \cap V(E)$ simultaneously for the event E_i. From the stochastic relationship it follows that all the elements in $D(E_i) \cap V(E)$ are independent of any other elements in $V(E)$, therefore we can perform the elements in $D(E_i) \cap V(E)$ as far as possible.

It has to be noted that the relationship between the decision vectors and chance functions is defined by the authors in [6] without taking into account the values assigned to decision variables. For this reason we shall see that their definition does not guarantee optimal plans, since in certain instances common variables may take values which break the link between two dependent constraints. In order to show this, in the following section we recall the water supply-allocation problem presented in Liu and Iwamura [6] to demonstrate the subtleties inherent in dependent-chance programming.

3. A dependent-chance programming example

Fig. 1 depicts a water supply system with three suppliers S_1, S_2, S_3 with their given *probabilistic* supply capacities and three different customers, denoted by C_1, C_2, C_3, with known demands. The scopes of the suppliers are $S_1 \rightsquigarrow \{C_1, C_2\}, S_2 \rightsquigarrow \{C_1, C_2, C_3\}, S_3 \rightsquigarrow \{C_2, C_3\}$.

The deterministic customer demands are [8,7,4]. The suppliers' probabilistic capacities are expressed as discrete probability density functions:

$$\phi_{S_1} = \{3(0.3), 7(0.5), 12(0.2)\},$$
$$\phi_{S_2} = \{6(0.4), 7(0.2), 10(0.4)\},$$
$$\phi_{S_3} = \{3(0.3), 8(0.7)\},$$

where values in parentheses represent probabilities. We must answer the following two types of question.

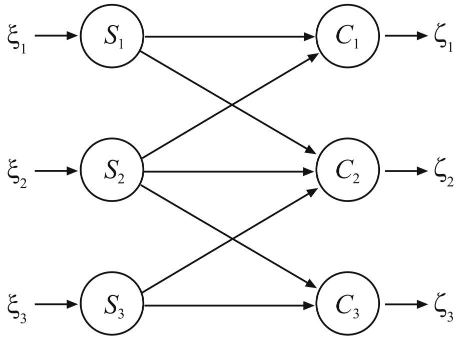


Fig. 1. Water supply-allocation problem.

- Supply problems. In order to achieve certain objectives in the future, decisions must be made concerning present actions to be taken. That is, we must determine the optimal combination of *inputs*, for example to determine the quantities ordered from the 3 inputs.
- Allocation problems. One of the basic allocation problems is the optimal allocation of the resources. Here the task is to determine the *outputs* that result from various combinations of resources such that certain objectives are achieved.

Certainly, in this system supply and allocation decisions should not be separate.

Let $\mathbf{S}$ be the set of suppliers and $\mathbf{C}$ the set of customers. Define decision variables $x_{s,c} \in \mathbb{Z}^+ \bigcup\{0\}$ denoting the planned non-negative supply from supplier s to customer c. Also define random variables ξ_s, with probability density function ϕ_s, denoting the uncertain supply available to supplier s. First we have the following constraints

$$\sum_{c\in\mathbf{C}_s} x_{s,c} \leqslant \xi_s, \quad \forall s \in \mathbf{S},$$

where $\mathbf{C}_s$ is the set of customers for supplier s. A constant ζ_c denotes the deterministic demand of customer c. Event E_c is defined as follows:

$$E_c : \sum_{s\in\mathbf{S}_c} x_{s,c} = \zeta_c,$$

where $\mathbf{S}_c$ is the set of suppliers for customer c. Event E_c means that the decision should satisfy the demand of customer c. In view of the uncertainty of this system, we are not sure whether a decision is feasible before knowing the realization of stochastic variables, so we employ chance functions to evaluate these events. Let

$$f_c(x) = \Pr\left\{E_c : \sum_{s\in\mathbf{S}_c} x_{s,c} = \zeta_c\right\},$$

where Pr denotes the probability of the event in $\{\cdot\}$. Usually we hope to maximize all the chance functions, i.e. increase the reliability levels of all the events as much as possible.

Without loss of generality we will now assume that all the events have the same priority and we will formulate the problem as DCGP. The model is therefore

$$\max \quad \sum_{c\in\mathbf{C}} f_c(x) \tag{3}$$

$$\text{subject to,} \tag{4}$$

$$\sum_{c\in\mathbf{C}_s} x_{s,c} \leqslant \xi_s \quad s \in \mathbf{S}, \tag{5}$$

$$x_{s,c} \in \mathbb{Z}^+ \bigcup\{0\} \quad s \in \mathbf{S},\ c \in \mathbf{C}. \tag{6}$$

The stochastic feasible set S will be defined by a probability function

$$\mu_S(x) = \Pr\left\{\sum_{c\in\mathbf{C}_s} x_{s,c} \leqslant \xi_s, \quad \forall s \in \mathbf{S}\right\}. \tag{7}$$

The authors in Liu and Iwamura [6] divide the decision components into three groups $\{x_{s,c}|s = S_1\}$, $\{x_{s,c}|s = S_2\}$ and $\{x_{s,c}|s = S_3\}$ which are mutually stochastically independent and in each group any element has the same probability of occurring. From the water supply-allocation problem definition it follows that

$$V(E_1) = \{x_{S_1,C_1}, x_{S_2,C_1}\}, \tag{8}$$
$$V(E_2) = \{x_{S_1,C_2}, x_{S_2,C_2}, x_{S_3,C_2}\}, \tag{9}$$
$$V(E_3) = \{x_{S_2,C_3}, x_{S_3,C_3}\}, \tag{10}$$

and

$$D(E_1) = \{x_{S_1,C_1}, x_{S_1,C_2}, x_{S_2,C_1}, x_{S_2,C_2}, x_{S_2,C_3}\}, \tag{11}$$
$$D(E_2) = \{x_{S_1,C_1}, x_{S_1,C_2}, x_{S_2,C_1}, x_{S_2,C_2}, x_{S_2,C_3}, x_{S_3,C_2}, x_{S_3,C_3}\}, \tag{12}$$
$$D(E_3) = \{x_{S_2,C_1}, x_{S_2,C_2}, x_{S_2,C_3}, x_{S_3,C_2}, x_{S_3,C_3}\}, \tag{13}$$

therefore the induced constraint on $D(E_1) \cap V(E)$ is then, according to Liu and Iwamura, $\{x_{S_1,C_1} + x_{S_1,C_2} \leqslant \xi_{S_1}, x_{S_2,C_1} + x_{S_2,C_2} + x_{S_2,C_3} \leqslant \xi_{S_2}\}$; on $D(E_2) \cap V(E)$ it is $\{x_{S_1,C_1} + x_{S_1,C_2} \leqslant \xi_{S_1}, x_{S_2,C_1} + x_{S_2,C_2} + x_{S_2,C_3} \leqslant \xi_{S_2}, x_{S_3,C_2} + x_{S_3,C_3} \leqslant \xi_{S_3}\}$; and finally on $D(E_3) \cap V(E)$ it is $\{x_{S_2,C_1} + x_{S_2,C_2} + x_{S_2,C_3} \leqslant \xi_{S_2}, x_{S_3,C_2} + x_{S_3,C_3} \leqslant \xi_{S_3}\}$. Hence

$$f_{C_1}(x) = \Pr\{(\xi_{S_1}, \xi_{S_2})|x_{S_1,C_1} + x_{S_1,C_2} \leqslant \xi_{S_1}, x_{S_2,C_1} + x_{S_2,C_2} + x_{S_2,C_3} \leqslant \xi_{S_2}\}, \tag{14}$$

$$f_{C_2}(x) = \Pr\{(\xi_{S_1}, \xi_{S_2}, \xi_{S_3})|x_{S_1,C_1} + x_{S_1,C_2} \leqslant \xi_{S_1}, x_{S_2,C_1} + x_{S_2,C_2} + x_{S_2,C_3} \leqslant \xi_{S_2}, x_{S_3,C_2} + x_{S_3,C_3} \leqslant \xi_{S_3}\}, \tag{15}$$

$$f_{C_3}(x) = \Pr\{(\xi_{S_2}, \xi_{S_3})|x_{S_2,C_1} + x_{S_2,C_2} + x_{S_2,C_3} \leqslant \xi_{S_2}, x_{S_3,C_2} + x_{S_3,C_3} \leqslant \xi_{S_3}\}. \tag{16}$$

Table 1 presents some representative distribution plans (columns 2–8) and their corresponding reliability measures according to Liu and Iwamura (column under heading "Liu–Iwamura").

4. Decision variable value based dependency

Liu–Iwamura's framework ignores the important dependency between constraints and values of decision variables.

Consider a plan in which $x_{S_1,C_1} = 0$ so that C_1 must receive all supplies from S_2. The reliability of the satisfaction of C_1 (event E_1) should now be independent of the ability of S_1 to meet its demand. But the dependent-chance programming, in its current form which does not take variable assignments into account, always relates the demand satisfaction of C_1 to S_1 and S_2 (Eq. (14)), which is not necessarily correct. Therefore, one should refine the objectives (Eqs. (14)–(16)) via further logical connectives between constraints:

Table 1
Representative distribution plans.

Plan no.	Planned delivery $S_i \rightsquigarrow D_j : (i,j)$							Reliability measures	
	(1,1)	(1,2)	(2,1)	(2,2)	(2,3)	(3,2)	(3,3)	Liu–Iwamura	New
1	3	5	5	1	1	1	3	0.624	0.624
2	4	7	4	0	4	0	0	0.560	0.680
3	6	2	2	5	0	0	4	0.624	0.940
4	5	0	3	3	4	4	0	0.756	0.960
5	7	5	1	1	1	1	3	1.040	1.040
6	2	5	6	0	4	2	0	0.960	1.380
7	8	2	0	2	4	3	0	1.400	1.400
8	0	7	8	0	0	0	4	0.756	1.800
9	5	0	3	3	0	4	4	1.890	2.100
10	6	0	2	0	4	7	0	1.890	2.400

$$f_c(x) = \Pr\left\{x_{s,c} \neq 0 \rightarrow \sum_{c' \in \mathbf{C}_s} x_{s,c'} \leqslant \xi_s, \quad \forall s \in \mathbf{S}_c\right\}, \tag{17}$$

where → denotes logical implication: $\mathfrak{C} \rightarrow \mathfrak{C}'$ is the sum of the probabilities of the scenarios in which either $\mathfrak{C}$ is violated or $\mathfrak{C}'$ is satisfied, or both. Because of this modification, under a decision in which $x_{S_1,C_1} = 0$ there is no longer a penalty if

$$\sum_{c' \in \mathbf{C}_1} x_{s,c'} \leqslant \xi_1$$

is violated.

The new reliability measures calculated using Eq. (17) are listed in the last column in Table 1.

To gain more insight into this problem class we examine allocation plans given in Table 1 in three categories: Plans {1,5}, Plans {2,3,4,6,8,9,10}, and Plan {7}.

In the first category (Plans 1 and 5) the plans have non-zero values assigned to all decision variables and, therefore, as expected the results of Liu–Iwamura and those produced by the extended model proposed here are the same (in Eq. (17), $x_{s,c} \neq 0$ becomes redundant). In the second group, however, since certain variables have zero assignments now a discrepancy between the Liu–Iwamura model and the extended model proposed here is observed. As explained above, this difference in probabilistic measure values is due to the broken constraint dependencies that arise when decision variables are assigned value zero. In the third group, we have only one plan (Plan 7). In this case, although two decision variables are assigned zero values the two frameworks produce the same result. To understand the reason behind this observation we need to look at the amounts committed by suppliers S_2 and S_3 according to Plan 7. Supplier S_2 (S_3) is expected to provide in total $x_{S_2,C_1} + x_{S_2,C_2} + x_{S_2,C_3} = 6$ ($x_{S_3,C_2} + x_{S_3,C_3} = 3$) units. When we look at the uncertain supply capacities for suppliers S_2 and S_3, it is clear that these units can be provided in full even under the worst-case scenarios. In other words, the zero assignment does not make any difference in Plan 7, because breaking the dependency is important only if there is a chance of failure in complying with supply commitments.

5. Conclusion

We showed how to extend Liu and Iwamura's original dependent-chance programming framework in order to obtain an exact reliability measure. Our experiments show that in most cases expressing constraint dependency without taking into account the values assigned to decision variables does not guarantee optimal plans, in fact in certain instances common variables may take values which break the link between two dependent constraints.

Acknowledgements

Roberto Rossi is supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain-Driven Research (CTVR) and Grant No. 05/IN/I886. S. Armagan Tarim and Brahim Hnich are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027.

References

[1] A. Charnes, W.W. Cooper, Chance-constrained programming, Management Science 6 (1) (1959) 73–79.

[2] B. Liu, Dependent-chance goal programming: A class of stochastic programming, Technical report, Institute of System Science, Chinese Academy of Sciences, 1995.

[3] B. Liu, Dependent-chance goal programming and its genetic algorithm approach, Technical report, Institute of System Science, Chinese Academy of Sciences, 1995.

[4] B. Liu, Dependent-chance programming: A class of stochastic optimization, Computers & Mathematics with Applications 34 (1997) 89–104.

[5] B. Liu, Uncertain Programming, John Wiley & Sons, New York, 1999.

[6] B. Liu, K. Iwamura, Modelling stochastic decision systems using dependent-chance programming, European Journal of Operational Research 101 (1997) 193–203.

[7] B. Liu, C. Ku, Dependent-chance goal programming and an application, Journal of Systems Engineering & Electronics 4 (1993) 40–47.

[8] Y. Wu, J. Zhou, J. Yang, Dependent-chance programming model for stochastic network bottleneck capacity expansion based on neural network and genetic algorithm, in: Advances in Natural Computation, Lecture Notes in Computer Science, vol. 3612, Springer-Verlag, 2005, pp. 120–128.

Contents lists available at ScienceDirect

Int. J. Production Economics

journal homepage: www.elsevier.com/locate/ijpe

A multi-objective stochastic programming approach for supply chain design considering risk

A. Azaron [a,b,*], K.N. Brown [a,c], S.A. Tarim [a,d], M. Modarres [e]

[a] *Center for Telecommunications Value-chain Research, Ireland*
[b] *Department of Industrial Engineering, Dalhousie University, Halifax, Nova Scotia, Canada*
[c] *Cork Constraint Computation Centre, Department of Computer Science, University College Cork, Cork, Ireland*
[d] *Department of Management, Hacettepe University, Ankara, Turkey*
[e] *Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran*

ARTICLE INFO

Article history:
Received 7 February 2007
Accepted 10 August 2008
Available online 19 August 2008

Keywords:
Supply chain management
Stochastic programming
Multiple-objective programming
Risk management

ABSTRACT

In this paper, we develop a multi-objective stochastic programming approach for supply chain design under uncertainty. Demands, supplies, processing, transportation, shortage and capacity expansion costs are all considered as the uncertain parameters. To develop a robust model, two additional objective functions are added into the traditional comprehensive supply chain design problem. So, our multi-objective model includes (i) the minimization of the sum of current investment costs and the expected future processing, transportation, shortage and capacity expansion costs, (ii) the minimization of the variance of the total cost and (iii) the minimization of the financial risk or the probability of not meeting a certain budget. The ideas of unreliable suppliers and capacity expansion, after the realization of uncertain parameters, are also incorporated into the model. Finally, we use the goal attainment technique to obtain the Pareto-optimal solutions that can be used for decision-making.

1. Introduction

A supply chain (SC) is a network of suppliers, manufacturing plants, warehouses and distribution channels organized to acquire raw materials, convert these raw materials to finished products and distribute these products to customers. The concept of SC management, which appeared in the early 1990s, has recently raised a lot of interest since the opportunity of an integrated management of the SC can reduce the propagation of unexpected/undesirable events through the network and can affect decisively the profitability of all the members.

A crucial component of the planning activities of a manufacturing firm is the efficient design and operation of its SC. Strategic-level SC planning involves deciding the configuration of the network, i.e., the number, location, capacity and technology of the facilities. The tactical-level planning of SC operations involves deciding the aggregate quantities and material flows for purchasing, processing and distributing of products. The strategic configuration of the SC is a key factor influencing efficient tactical operations, and therefore has a long-lasting impact on the firm. Furthermore, the fact that the SC configuration involves the commitment of substantial capital resources over long periods of time makes the SC design problem an extremely important one.

Many attempts have been made to model and optimize SC design, most of which are based on deterministic approaches, see for example Bok et al. (2000), Timpe and Kallrath (2000), Gjerdrum et al. (2000) and many others. However, most real SC design problems are characterized by numerous sources of technical and commercial uncertainty, and so the assumption that all model

[*] Corresponding author at: Department of Industrial Engineering, Dalhousie University, Halifax, Nova Scotia, Canada.
E-mail addresses: a.azaron@dal.ca, aazaron@gmail.com (A. Azaron).

0925-5273/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2008.08.002

parameters, such as cost coefficients, supplies, demand, etc., are known with certainty is not realistic.

In order to take into account the effects of the uncertainty in the production scenario, a two-stage stochastic model is proposed in this paper. Decision variables, which characterize the network configuration, namely those binary variables that represent the existence and the location of plants and warehouses of the SC, are considered as first-stage variables—it is assumed that they have to be taken at the design stage before the realization of the uncertainty. On the other hand, decision variables related to the amount of products to be produced and stored in the nodes of the SC and the flows of materials transported among the entities of the network are considered as second-stage variables, corresponding to decisions taken after the uncertain parameters have been revealed.

In traditional stochastic programming approaches, the objective function consists of the sum of the first-stage performance measure and the expected second-stage performance, and most commonly, the dominant uncertain parameters are the product demands. Approaches differ primarily in the selection of the decision variables and the way in which the expected value term, which in principle involves a multidimensional integral involving the joint probability distribution of the uncertain parameters, is computed.

There are a few research works addressing comprehensive (strategic and tactical issues simultaneously) design of SC networks using two-stage stochastic models. MirHassani et al. (2000) considered a two-stage model for multi-period capacity planning of SC networks. The authors used Benders decomposition to solve the resulting stochastic integer program. Tsiakis et al. (2001) also considered a two-stage stochastic programming model for SC network design under demand uncertainty. The authors developed a large-scale mixed-integer linear programming model for this problem. Alonso-Ayuso et al. (2003) proposed a branch-and-fix heuristic for solving two-stage stochastic SC design problems. Santoso et al. (2005) integrated a sampling strategy with an accelerated Benders decomposition to solve SC design problems with continuous distributions for the uncertain parameters. There are also some other papers in SC planning under uncertainty. Petkov and Maranas (1997), Gupta and Maranas (2003) and Gupta et al. (2000) incorporated the uncertain demands as multivariate-normal distributions. Then, they converted stochastic features of the problem into a chance-constraint programming problem. Goh et al. (2007) developed a stochastic model of the multi-stage global SC network problem, considering supply, demand, exchange and disruption as the uncertain parameters. However, the robustness of decision to uncertain parameters is not considered in above studies.

Although stochastic programming has been studied for four decades, conventional stochastic programming models are severely limited owing to its inability to handle risk aversion or decision-makers' preferences in a direct manner, subsequently excluding many important domains of application. Mulvey et al. (1995) presented an improved stochastic programming called robust programming capable of tackling the decision-makers' favored risk aversion. In this method, the variance term is simply added into the main objective function with an associated weighting parameter that represents the risk tolerance of the modeler, see for example Yu and Li (2000) and Lai and Ng (2005). This idea has also been used in some other areas, which are not directly related to the SC design problem. For example, Ahmed and Sahinidis (1998) used this construct to develop a linear programming recourse formulation for production planning in the presence of scenarios. Bok et al. (1998) also employed the penalized variance term and introduced an additional penalized term reflecting the underutilization of capacity.

The main disadvantages of traditional stochastic SC design approaches are as follows:

1. Minimizing cost or maximizing profit as a single objective is often the optimization focus (Cohen and Lee, 1989; Tsiakis et al., 2001).
2. Most multi-objective SC approaches are either deterministic (Chen et al., 2003) or only demand is considered as the source of uncertainty (Guillen et al., 2005).
3. Minimizing the risk reflected by the variance of the total cost and the financial risk has not been considered in existing comprehensive SC design models.
4. Reliability issues have not been considered during the strategic planning phase.

To overcome these disadvantages, we develop a robust stochastic programming approach for designing SCs under uncertainty. In our approach, not only demands, but also supplies, processing, transportation, shortage and capacity expansion costs are all considered as the uncertain parameters. Moreover, we assume that suppliers are unreliable and may lose their abilities to supply, like oil suppliers in the Middle East. The reliabilities of suppliers are known in advance, but the actual situations of suppliers become clear after building the facilities.

The first objective function of our proposed model is the minimization of the sum of first-stage investment costs and the expected second-stage processing, transportation, shortage and capacity expansion costs. To develop a robust model, two additional objective functions are added into the final model. The first additional objective function is the minimization of the variance of the total cost. The variance of the total cost should be considered in the model, because when we focus only on the expected total cost, the design scheme may be suboptimal if the total cost substantially varies because of randomness. Practically, the variance of the total cost is difficult to interpret. Therefore, it is necessary to introduce a new objective function to clearly capture the notion of risk. This objective function is the minimization of the financial risk. The financial risk associated with a design project under uncertainty is defined as the probability of not meeting a certain cost level or budget.

Although the ideas of variance and financial risk have been considered in other areas, but to the best of our knowledge, it is the first time they are considered all together in a multi-objective scheme to design robust SCs under uncertainty and unreliable suppliers. Moreover, the idea of capacity expansion at the second stage, after the realization of uncertain parameters, is also incorporated into the model. Using this idea, we have the option of expanding the capacities of plants and warehouses, if we face favourable economic conditions with large demands.

Since the expected total cost, the variance of the total cost and the financial risk are in conflict with each other, it is proposed to set up a multi-objective design problem whose solution will be a set of Pareto-optimal possible design alternatives representing the trade-off among different objectives rather than a unique solution. To the best of our knowledge, only ε−constraint method (Guillen et al., 2005) and fuzzy optimization (Chen and Lee, 2004) have been used to solve multi-objective SC design models. We use the goal attainment technique, see Hwang and Masud (1979) for details, to solve the resulting multi-objective problem.

The present work formulates the SC design problem as a multi-objective stochastic mixed-integer nonlinear programming problem, which is solved by using the goal attainment technique. This formulation takes into account not only SC expected total cost, but also the risk reflected by the variance of the total cost and the financial risk. The result of the model provides a set of Pareto-optimal solutions to be used by the decision-maker in order to find the best SC configuration according to his/her preferences.

This paper is organized as follows. In Section 2, we describe the SC design problem. Section 3 presents the multi-objective SC design problem considering risk. In Section 4, we explain about the goal attainment technique to solve the multi-objective problem. Section 5 presents the computational experiments. Finally, we draw the conclusion of the paper in Section 6.

2. Problem description

We first describe a deterministic mathematical formulation for the SC design problem. Consider an SC network $G = (N, A)$, where N is the set of nodes and A is the set of arcs. The set N consists of the set of suppliers S, the set of possible processing facilities P and the set of customer centers C, i.e., $N = S \cup P \cup C$. The processing facilities include manufacturing centers M and warehouses W, i.e., $P = M \cup W$. Let K be the set of products flowing through the SC.

The SC configuration decisions consist of deciding which of the processing centers to build. We associate a binary variable y_i to these decisions: $y_i = 1$ if processing facility i is built, and 0 otherwise. The tactical decisions consist of routing the flow of each product $k \in K$ from the suppliers to the customers. We let x_{ij}^k denote the flow of product k from a node i to a node j of the network where $(ij) \in A$, and z_j^k denote shortfall of product k at customer center j, when it is impossible to meet demand. A deterministic mathematical model for this SC design problem is formulated as follows (see Santoso et al. (2005) for more details):

$$\text{Min} \quad \sum_{i \in P} c_i y_i + \sum_{k \in K} \sum_{(ij) \in A} q_{ij}^k x_{ij}^k + \sum_{k \in K} \sum_{j \in C} h_j^k z_j^k$$

$$\text{s.t.} \tag{1.1}$$

$$y \in Y \subseteq \{0,1\}^{|P|} \tag{1.2}$$

$$\sum_{i \in N} x_{ij}^k - \sum_{l \in N} x_{jl}^k = 0 \quad \forall j \in P, \quad \forall k \in K \tag{1.3}$$

$$\sum_{i \in N} x_{ij}^k + z_j^k \geqslant d_j^k \quad \forall j \in C, \quad \forall k \in K \tag{1.4}$$

$$\sum_{j \in N} x_{ij}^k \leqslant s_i^k \quad \forall i \in S, \quad \forall k \in K \tag{1.5}$$

$$\sum_{k \in K} r_j^k \left(\sum_{i \in N} x_{ij}^k \right) \leqslant m_j y_j \quad \forall j \in P \tag{1.6}$$

$$x_{ij}^k \geqslant 0 \quad \forall (ij) \in A, \quad \forall k \in K \tag{1.7}$$

$$z_j^k \geqslant 0 \quad \forall j \in C, \quad \forall k \in K \tag{1.8}$$

In the above model, c_i denotes the investment cost for building facility i, q_{ij}^k denotes the per-unit cost of processing product k at facility i and/or transporting product k on arc (ij), and h_j^k denotes the per-unit penalty incurred for failing to meet demand of product k at customer center j. The objective function (1.1) consists of minimizing total investment, tactical and shortage costs. Constraint (1.2) enforces the binary nature of the configuration decisions for the processing facilities. Constraint (1.3) enforces the flow conservation of product k across each processing node j. Constraint (1.4) requires that the total flow of product k to a customer node j plus shortfall should exceed the demand d_j^k at that node. Constraint (1.5) requires that the total flow of product k from a supplier node i should be less than the supply s_i^k at that node. Constraint (1.6) enforces capacity constraints of the processing nodes. Here, r_j^k denotes per-unit processing requirement for product k at node j. The capacity constraint then requires that the total processing requirement of all products flowing into a processing node j should be smaller than the capacity m_j of facility j if it is built ($y_j = 1$). If facility j is not built ($y_j = 0$), the constraint will force all flow variables $x_{ij}^k = 0$ for all $i \in N$. Finally, the last set of constraints enforces the non-negativity of the flow variables and shortfalls.

It will be convenient to work with the following compact notation for models (1.1)–(1.8):

$$\text{Min} \quad c^T y + q^T x + h^T z$$

$$\text{s.t.} \tag{2.1}$$

$$y \in Y \subseteq \{0,1\}^{|P|} \tag{2.2}$$

$$Bx = 0 \tag{2.3}$$

$$Dx + z \geqslant d \tag{2.4}$$

$$Sx \leqslant s \tag{2.5}$$

$$Rx \leqslant My \tag{2.6}$$

$$x \in R_+^{|A|\times|K|}, \quad z \in R_+^{|C|\times|K|} \tag{2.7}$$

Above vectors c, q, h, d and s correspond to investment costs, processing/transportation costs, shortfall costs, demands and supplies, respectively. The matrices B, D and S are appropriate matrices corresponding to the summations on the left-hand side of the expressions (1.3)–(1.5), respectively. The notation R corresponds to a matrix of r_j^k, and the notation M corresponds to a matrix with m_j along the diagonal.

We now propose a stochastic programming approach based on a recourse model with two stages to incorporate the uncertainty associated with demands, supplies, processing costs, transportation costs, shortage costs and capacity expansion costs.

In a two-stage stochastic optimization approach, the uncertain parameters are considered as random variables with an associated probability distribution and the decision variables are classified into two stages. The first-stage variables correspond to those decisions that need to be made here-and-now, prior to the realization of the uncertainty. The second-stage or recourse variables correspond to those decisions made after the uncertainty is unveiled and are usually referred to as wait-and-see decisions. After the first-stage decisions are taken and the random events realized, the second-stage decisions are subjected to the restrictions imposed by the second-stage problem. Due to the stochastic nature of the performance associated with the second-stage decisions, the objective function, traditionally, consists of the sum of the first-stage performance measure and the expected second-stage performance; refer to Birge and Louveaux (1997) for more details.

It is assumed that we have the option of expanding the capacities of plants and warehouses after the realization of uncertain parameters. Clearly, when we face favourable economic conditions with high demands, at the second-stage, it may be reasonable to expand the capacities of sites, even if unit expansion costs are relatively high.

Considering vectors e, f, O and $\xi = (q, h, f, d, s)$ as capacity expansions, per-unit expansion costs, expansion limits and random data, respectively, the two-stage stochastic model is formulated as follows:

$$\text{Min} \quad c^T y + E[G(y,\xi)]$$
$$\text{s.t.} \tag{3.1}$$

$$y \in Y \subseteq \{0,1\}^{|P|} \tag{3.2}$$

where $G(y,\xi)$ is the optimal value of the following problem:

$$\text{Min} \quad q^T x + h^T z + f^T \mathrm{e}$$
$$\text{s.t.} \tag{3.3}$$

$$Bx = 0 \tag{3.4}$$

$$Dx + z \geqslant d \tag{3.5}$$

$$Sx \leqslant s \tag{3.6}$$

$$Rx \leqslant My + \mathrm{e} \tag{3.7}$$

$$e \leqslant Oy \tag{3.8}$$

$$x \in R_+^{|A|\times|K|}, \quad z \in R_+^{|C|\times|K|}, \quad \mathrm{e} \in R_+^{|P|} \tag{3.9}$$

Note that the optimal value $G(y,\xi)$ of the second-stage problem (3.3)–(3.9) is a function of the first-stage decision variable y and a realization $\xi = (q, h, f, d, s)$ of the uncertain parameters. The expectation in (3.1) is taken with respect to the joint probability distribution of uncertain parameters.

In the above problem, decision variables, which represent the existence of the different nodes of the SC, are considered as first-stage variables as it is assumed that they have to be taken at the design stage before the uncertain parameters are unveiled. On the other hand, decision variables related to the amount of products to be produced and stored in the nodes of the SC, the flows of materials transported among the entities of the network, shortfalls at the customer centers and the amount of expansion of the capacities of sites are considered as second-stage variables.

It should be mentioned that stochastic programming is generally difficult to handle and implement. The readers may refer to van Delft and Vial (2004), which describes a powerful tool for practical implementation of stochastic programming in an SC problem.

In this paper, the uncertainty associated with demands, supplies, processing, transportation, shortage and capacity expansion costs is represented by a set of discrete scenarios with given probability of occurrence. Such scenarios together with their associated probabilities, and also the reliabilities of suppliers are provided as input data into the model. The difficulty of continuous distributions is avoided by introducing discrete scenarios, or combinations of discrete samples of all the uncertain parameters using Monte Carlo simulation. This approach is explained in detail at the end of Section 4.

3. Multi-objective supply chain design problem

As explained, to develop a robust model, two additional objective functions are added into the traditional SC design problem. The first is the minimization of the variance of the total cost, and the second is the minimization of the probability of not meeting a certain budget. However, by considering the variance of the total cost as an objective function, we actually introduce nonlinearity into the proposed model, but that is the only nonlinear term in the final mathematical program.

We also assume that some suppliers may lose their abilities to supply, like oil suppliers in the Middle East. The reliabilities of suppliers are known in advance. But the situations of suppliers will actually clear after building the facilities. So, we will have $2^{|S|}$ scenarios for the situations of suppliers, in some of them one or more suppliers are unable to supply. If in each scenario, some suppliers are unable to supply, those suppliers with their external links can be easily dropped off from further consideration. So, it will affect the topology of network and decrease the number of decision variables and constraints of the final mathematical model. An alternative method to deal with unreliable suppliers is to set the supply values of the unreliable suppliers in the corresponding scenarios to zero.

Let T be the set of scenarios with given probability of occurrence associated with demands, supplies, processing costs, transportation costs, shortage costs and capacity expansion costs. Such scenarios together with their associated probabilities must be provided as input data into the model.

A different value for the sum of the first-stage and the second-stage costs is obtained for each particular realization of uncertain parameters. The proposed model accounts for the minimization of the sum of first-stage and the expected second-stage costs, minimization of the variance of second-stage costs and the minimization of financial risk or the probability of not meeting a certain budget.

The financial risk associated with a design project under uncertainty is defined as the probability of not meeting a certain target cost level. For the two-stage stochastic problem, the financial risk associated with a certain budget Ω can be rewritten with the help of binary variables as follows:

$$\text{Risk} = \sum_{l=1}^{L} p_l u_l \qquad (4)$$

where p_l denotes the occurrence probability of the lth scenario, $L = |T| \times 2^{|S|}$ denotes the total number of scenarios including those related to the reliabilities of suppliers and u_l is a new binary variable defined for each scenario as follows:

$$u_l = \begin{cases} 1 & \text{if } Cost_l > \Omega, \\ 0 & \text{otherwise} \end{cases} \qquad (5)$$

where $Cost_l$ is the total cost when the lth scenario is realized.

Considering V as a very large constant value (approaching infinity), the proper multi-objective stochastic model for our SC design problem would be

$$\text{Min} \quad c^T y + \sum_{l=1}^{L} p_l (q_l^T x_l + h_l^T z_l + f_l^T \mathrm{e}_l) \qquad (6.1)$$

$$\text{Min} \quad \sum_{l=1}^{L} p_l \left(q_l^T x_l + h_l^T z_l + f_l^T \mathrm{e}_l - \sum_{l=1}^{L} p_l (q_l^T x_l + h_l^T z_l + f_l^T \mathrm{e}_l) \right)^2 \qquad (6.2)$$

$$\text{Min} \quad \sum_{l=1}^{L} p_l u_l$$

s.t. (6.3)

$$Bx_l = 0 \quad l = 1, \ldots, L \qquad (6.4)$$

$$Dx_l + z_l \geqslant d_l \quad l = 1, \ldots, L \qquad (6.5)$$

$$Sx_l \leqslant s_l \quad l = 1, \ldots, L \qquad (6.6)$$

$$Rx_l \leqslant My + e_l \quad l = 1, \ldots, L \qquad (6.7)$$

$$e_l \leqslant Oy \quad l = 1, \ldots, L \qquad (6.8)$$

$$c^T y + q_l^T x_l + h_l^T z_l + f_l^T e_l - \Omega \leqslant V u_l \quad l = 1, \ldots, L \qquad (6.9)$$

$$y \in Y \subseteq \{0,1\}^{|P|}, \quad u \in U \subseteq \{0,1\}^{L} \qquad (6.10)$$

$$x \in R_+^{|A| \times |K| \times L}, \quad z \in R_+^{|C| \times |K| \times L}, \quad \mathrm{e} \in R_+^{|P| \times L} \qquad (6.11)$$

Objective function (6.1) is related to the expected total cost or the sum of the first-stage and the expected second-stage costs. Objective function (6.2) is related to the variance of second-stage costs or the variance of total cost. Objective function (6.3) is related to the financial risk. Constraint (6.8) enforces the capacity expansion limit for each processing facility, if it is built. According to constraint (6.9), if the total cost for a scenario is greater than a certain budget Ω, then the binary variable associated with that particular scenario will be equal to 1, which increases the financial risk (6.3) by the corresponding probability. Otherwise, if the total cost for a scenario is smaller than Ω, then the binary variable associated with this scenario will be equal to 0, because we intend to minimize (6.3). Therefore, this situation will not change the value of financial risk.

Using a multi-objective model in an SC context is not an artificial one, as we know from “portfolio optimization” that it is not possible to give any monetary value to risk, which leads to the concept of “efficient frontier” defined by Markowitz (1952, 1959). We may treat volatility and expected return as proxies for risk and reward. Out of the entire universe of possible portfolios, certain ones will optimally balance risk and reward. These comprise what Markowitz called an efficient frontier of portfolios. This frontier is a curve in the “risk vs expected return” space. If it was possible to replace risk by reward/loss, then we would have had only a single dot in this space representing the optimal portfolio. But we know from finance theory that it is not the case; having said that it is possible only if one can clearly define his/her utility function (a relation between risk and return).

Now, the question is how to define the firm’s utility function in an SC. We will show in one SC problem (Section 5) that the relationship between cost, variance and risk is not clear, and that decision-makers need support in determining what that utility function should be—they need to see the effect of weighting the criteria differently before they can make their decision, and obviously, setting a single function does not guarantee us a Pareto-optimal solution. That is why we are adopting a multi-objective approach to this SC problem.

4. Goal attainment technique

We use the goal attainment technique, which is a variation of goal programming technique, to solve the multi-objective problem. Goal attainment method is one of the multi-objective techniques with priori articulation of preference information given. In this method, the preferred solution is sensitive to the goal vector and the weighting vector given by the decision-maker; the same as the goal programming technique.

Goal attainment method has fewer variables to work with and is a one-stage method, unlike interactive multi-objective techniques, so it will be computationally faster. Therefore, in terms of computational time, it is one of the best techniques to solve our SC problem, whose deterministic equivalent form is a large-scale mixed-integer nonlinear program. We successfully applied the goal attainment technique in solving a number of real-world multi-objective problems arising in reliability optimization (Azaron et al., 2007a), project management (Azaron et al., 2007b) and production systems (Azaron et al., 2006), and it is for the first time that we use this technique to solve a multi-objective SC design problem and to generate its Pareto-optimal solutions.

This method requires setting up a goal and weight, b_j and g_j ($g_j \geqslant 0$) for $j = 1, 2, 3$, for the three mentioned objective functions. The g_j relates the relative under-attainment of the b_j. For under-attainment of the goals, a smaller g_j is associated with the more important objectives. When g_j approaches 0, then the associated objective function should be fully satisfied or the corresponding objective function value should be less than or equal to its goal b_j. g_j, $j = 1, 2, 3$, are generally normalized so that $\sum_{j=1}^{3} g_j = 1$. The proper goal attainment formulation for our problem is

$$\text{Min} \quad w$$

$$\text{s.t.} \tag{7.1}$$

$$c^T y + \sum_{l=1}^{L} p_l (q_l^T x_l + h_l^T z_l + f_l^T \mathrm{e}_l) - g_1 w \leqslant b_1 \tag{7.2}$$

$$\sum_{l=1}^{L} p_l \left(q_l^T x_l + h_l^T z_l + f_l^T \mathrm{e}_l - \sum_{l=1}^{L} p_l (q_l^T x_l + h_l^T z_l + f_l^T \mathrm{e}_l) \right)^2 - g_2 w \leqslant b_2 \tag{7.3}$$

$$\sum_{l=1}^{L} p_l u_l - g_3 w \leqslant b_3 \tag{7.4}$$

$$Bx_l = 0 \quad l = 1, \ldots, L \tag{7.5}$$

$$Dx_l + z_l \geqslant d_l \quad l = 1, \ldots, L \tag{7.6}$$

$$Sx_l \leqslant s_l \quad l = 1, \ldots, L \tag{7.7}$$

$$Rx_l \leqslant My + e_l \quad l = 1, \ldots, L \tag{7.8}$$

$$e_l \leqslant Oy \quad l = 1, \ldots, L \tag{7.9}$$

$$c^T y + q_l^T x_l + h_l^T z_l + f_l^T e_l - \Omega \leqslant V u_l \quad l = 1, \ldots, L \tag{7.10}$$

$$y \in Y \subseteq \{0, 1\}^{|P|}, \quad u \in U \subseteq \{0, 1\}^L \tag{7.11}$$

$$x \in R_+^{|A| \times |K| \times L}, \quad z \in R_+^{|C| \times |K| \times L}, \quad \mathrm{e} \in R_+^{|P| \times L} \tag{7.12}$$

Lemma 1. If (y^*, u^*, x^*, z^*, e^*) is Pareto-optimal, then there exists a b, g pair such that (y^*, u^*, x^*, z^*, e^*) is an optimal solution to the optimization problem (7).

The optimal solution using this formulation is sensitive to b and g. Depending on the values for b, it is possible that g does not appreciably influence the optimal solution. Instead, the optimal solution can be determined by the nearest Pareto-optimal solution from b. This might require that g be varied parametrically to generate a set of Pareto-optimal solutions. In the next section, we consider several pairs of b and g to generate different Pareto-optimal solutions.

The mixed-integer nonlinear programming problem (7) has $(|A| \times |K| + |C| \times |K| + |P|) \times L + 1$ continuous decision variables, excluding slack variables, $|P| + L$ binary variables and $(|A| \times |K| + 2 \times |C| \times |K| + |P| \times |K| + |S| \times |K| + 3 \times |P| + 1) \times L + 3$ constraints.

In case the random data vector $\xi = (q, h, f, d, s)$ follows a known continuous joint distribution, one should resort to a sampling procedure, for example Santoso et al. (2005), to solve the proposed model. In the sampling strategy, a random sample $\xi^1, \xi^2, \ldots, \xi^Q$ of Q realizations of the random vector ξ is generated. Then, considering $L = Q \times 2^{|S|}$ in this case, the proper goal attainment formulation can be approximated as

$$\text{Min} \quad w$$

$$\text{s.t.} \tag{8.1}$$

$$c^T y + \frac{1}{L} \sum_{l=1}^{L} (q_l^T x_l + h_l^T z_l + f_l^T \mathrm{e}_l) - g_1 w \leqslant b_1 \tag{8.2}$$

$$\frac{1}{L-1} \sum_{l=1}^{L} \left(q_l^T x_l + h_l^T z_l + f_l^T \mathrm{e}_l - \frac{1}{L} \sum_{l=1}^{L} (q_l^T x_l + h_l^T z_l + f_l^T \mathrm{e}_l) \right)^2 - g_2 w \leqslant b_2 \tag{8.3}$$

$$\frac{1}{L} \sum_{l=1}^{L} u_l - g_3 w \leqslant b_3 \tag{8.4}$$

$$Bx_l = 0 \quad l = 1, \ldots, L \tag{8.5}$$

$$Dx_l + z_l \leqslant d_l \quad l = 1, \ldots, L \tag{8.6}$$

$$Sx_l \leqslant s_l \quad l = 1, \ldots, L \tag{8.7}$$

$$Rx_l \leqslant My + \mathrm{e}_l \quad l = 1, \ldots, L \tag{8.8}$$

$$e_l \leqslant Oy \quad l = 1, \ldots, L \tag{8.9}$$

$$c^T y + q_l^T x_l + h_l^T z_l + f_l^T e_l - \Omega \leqslant V u_l \quad l = 1, \ldots, L \tag{8.10}$$

$$y \in Y \subseteq \{0, 1\}^{|P|}, \; u \in U \subseteq \{0, 1\}^L \tag{8.11}$$

$$x \in R_+^{|A| \times |K| \times L}, \quad z \in R_+^{|C| \times |K| \times L}, \quad \mathrm{e} \in R_+^{|P| \times L} \tag{8.12}$$

where the expected total cost, the variance of the total cost and financial risk are approximated by (8.2), (8.3) and (8.4), respectively.

Let v_Q and $\hat{y}_Q$ be the optimal value and the optimal solution vector, respectively, of the approximated problem (8). Clearly, for a particular realization ξ^1, ξ^2, …, ξ^Q of the random vector, problem (8) is deterministic. It is possible to show that under mild regularity conditions, as the sample size Q increases, v_Q and $\hat{y}_Q$ converge with probability one to their true counterparts, see Kleywegt et al. (2001). The performance of the sampling strategy is beyond the scope of this paper and can be considered as a direction for future research in this area.

5. Numerical experiments

Consider the SC network design problem depicted in Fig. 1 (modified from Yu and Li, 2000). A wine company is willing to design its SC. This company owns three customer centers located in three different cities L, M and N. Uniform-quality wine in bulk (raw material) is supplied from four wineries located in A, B, C and D. There are four possible locations E, F, G and H for building the bottling plants.

For simplicity, without considering other market behaviors (e.g. novel promotion, marketing strategies of competitors and market-share effect in different markets), each market demand merely depends on the local economic conditions. Assume that the future economy is either boom, good, fair or poor, i.e. four situations with associated probabilities of .13, .25, .45 or .17, respectively. The unit production costs and market demands under each scenario are shown in Table 1.

The supplies, transportation costs and shortage costs are considered as deterministic parameters. In all, 475,000, 425,000, 500,000 and 450,000 are investment costs for building each bottling plant E, F, G and H, respectively. In all, 65.6, 155.5, 64.3, 175.3, 62, 150.5, 59.1, 175.2, 84, 174.5, 87.5, 208.9, 110.5, 100.5, 109, 97.8 are the unit costs of transporting bulk wine from each winery A, B, C and D to each bottling plant E, F, G and H, respectively. The unit costs of transporting bottled wine from each bottling plant E, F, G and H to each distribution center L, M, and N, respectively, are 200.5, 300.5, 699.5, 693, 533, 362, 163.8, 307, 594.8, 625, 613.6, 335.5. The unit shortage costs at each distribution center L, M and N are 10,000, 13,000 and 12,000, respectively. In all, 375, 187, 250 and 150 are the maximum amount of bulk wine that can be shipped from each winery A, B, C and D, respectively, if it is reliable. In all, 315, 260, 340 and 280 are the capacities of each bottling plant E, F, G and H, respectively, if it is built.

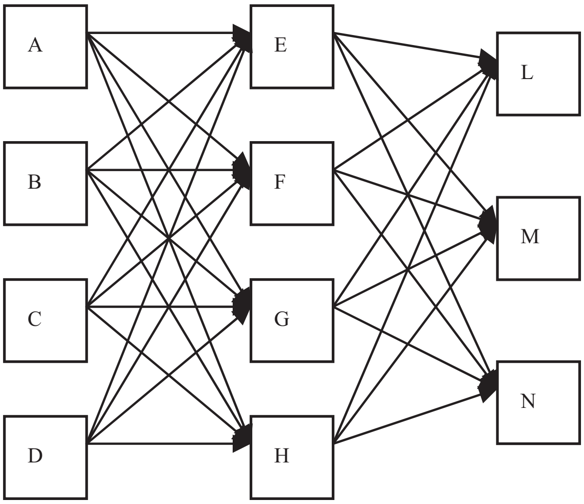


Fig. 1. The supply chain design problem of the wine company.

We also have the option of expanding the capacity of bottling plant F, if it is built. In all, 100, 80, 60 and 50 are the unit capacity expansion costs, when the future economy is boom, good, fair or poor, respectively. In addition, we cannot expand the capacity of this plant more than 40 units in any situation. Moreover, winery D is an unreliable supplier and may lose its ability to supply the bottling plants. The reliability of this winery is estimated as .9. So, the total number of scenarios for this SC design problem is equal to $4 \times 2 = 8$.

Clearly, this system produces one type of product and the processing facilities include only manufacturing centers M. So, the per-unit processing requirements r_j^k are all equal to 1 and $W = \phi$.

This problem attempts to minimize the expected total cost, the variance of the total cost and the financial risk in a multi-objective scheme while making the following determinations:

(a) Which of the bottling plants to build (first-stage variables)?
(b) Amount of bulk wine to be bottled in each bottling plant, amount of bulk wine and bottled wine to be transported among the entities of the network, amount of shortfall at each customer center and finally amount of expansion of the capacity of bottling plant F, if it is built (second-stage variables)?

We use goal attainment formulation (7) to solve this multi-objective SC design problem. The mathematical model has 12 binary variables, 257 continuous decision variables and 407 constraints.

Table 1
Characteristics of the problem

Future economy	Demands			Unit production costs				Probabilities
	L	M	N	E	F	G	H	
Boom	400	188	200	755	650	700	800	.13
Good	350	161	185	700	600	650	750	.25
Fair	280	150	160	675	580	620	720	.45
Poor	240	143	130	650	570	600	700	.17

Table 2
Pareto-optimal solutions

No.	g_1	g_2	g_3	b_1	b_2	b_3	Ω	E	F	G	H	Mean	Variance	Risk	Time
1	E−6	.99999	E−8	185E4	E9	.1	218E4	1	1	1	0	2,007,034	109,871E5	.13	2:21
2	E−4	.99989	E−8	185E4	E8	.1	218E4	1	1	1	0	2,086,941	246,917E4	.13	2:49
3	.01	.98999	E−8	185E4	E8	.1	218E4	1	1	1	0	2,184,688	133,134E3	.397	2:52
4	.01	.98999	E−9	185E4	E8	.1	218E4	1	1	1	0	2,184,467	134,974E3	.13	3:11
5	.1	.89999	E−9	185E4	E9	.1	221E4	1	1	0	1	2,221,661	912,376E3	.13	4:15
6	.1	.89999	E−8	185E4	E9	.1	221E4	1	1	1	0	2,150,000	715,080E3	.13	0:46
7	.1	.89999	E−8	185E4	E8	.1	221E4	1	1	1	0	2,188,285	103,045E3	.13	0:48
8	.1	.89999	E−8	185E4	E8	.1	218E4	1	1	1	0	2,186,120	127,592E3	.4	2:21
9	.1	.89999	E−9	185E4	E8	.1	218E4	1	1	1	0	2,184,467	134,974E3	.13	3:51
10	.1	.89999	E−7	185E4	E8	.1	218E4	1	1	1	0	2,192,098	105,670E3	.73	1:01
11	.1	.89999	E−7	185E4	E9	.1	218E4	1	1	1	0	2,132,511	100,254E4	.13	5:12
12	.25	.74999	E−8	185E4	E8	.1	218E4	1	1	1	0	2,186,493	125,924E3	.442	2:30
13	.25	.74999	E−9	185E4	E8	.1	218E4	1	1	1	0	2,184,503	137,060E3	.13	4:01
14	.25	.74999	E−8	185E4	E9	.1	218E4	1	1	1	0	2,184,596	146,415E3	.13	3:51
15	.5	.49999	E−8	185E4	E8	.1	218E4	1	1	1	0	2,187,987	121,750E3	.535	2:26
16	.5	.49999	E−9	185E4	E8	.1	218E4	1	1	1	0	2,184,582	134,462E3	.155	0:41
17	.75	.24999	E−8	185E4	E8	.1	218E4	1	1	1	0	2,188,794	115,646E3	.622	1:48
18	.75	.24999	E−9	185E4	E8	.1	218E4	1	1	1	0	2,184,893	133,072E3	.217	1:59
19	.9	.09999	E−8	185E4	E8	.1	218E4	1	1	1	0	2,192,212	106,300E3	.73	1:54
20	.9	.09999	E−9	185E4	E8	.1	218E4	1	1	1	0	2,185,957	128,321E3	.38	5:21
21	.9	.09999	E−9	185E4	E9	.1	218E4	1	1	1	0	2,184,470	135,165E3	.13	2:48
22	.9	.09999	E−9	185E4	E10	.1	218E4	0	1	1	1	2,192,825	110,321E4	.13	3:55
23	.99	.00999	E−8	185E4	E8	.1	218E4	1	1	1	0	2,194,198	100,677E3	.777	1:23
24	.99	.00999	E−8	185E4	E9	.1	218E4	1	1	1	0	2,184,469	135,041E3	.13	1:10
25	.9999	E−4	E−8	185E4	E8	.1	218E4	1	1	1	0	2,192,571	100,007E3	.777	2:51
26	.99999	E−6	E−9	220E4	E2	.1	222E4	1	0	0	0	1022E5	0	1	1:16
27	9E−4	.9991	E−8	185E4	E8	.1	220E4	1	1	1	0	2,159,937	444,064E3	.13	2:03
28	E−5	.99999	E−9	185E4	E8	.1	220E4	1	1	1	0	2,007,034	109,871E5	.13	1:56
29	E−8	.99999	E−8	185E4	E9	.1	220E4	0	1	1	0	1,853,385	310,218E6	.13	2:37
30	9E−6	.99999	E−6	200E4	E8	.1	222E4	1	1	1	0	2,046,929	531,427E4	.013	0:55
31	9E−6	.99999	E−6	185E4	E9	.1	222E4	1	1	1	0	2,007,034	109,871E5	.013	2:32
32	E−6	.99999	E−6	185E4	E10	.1	221E4	1	1	1	0	2,007,034	109,871E5	.13	2:16
33	E−7	.99999	E−7	185E4	E10	.1	221E4	0	1	1	0	1,878,088	290,880E6	.13	2:18
34	E−7	.99999	E−9	185E4	E10	.1	210E4	1	1	1	0	2,104,936	232,243E4	.13	1:48
35	.099	.9	.001	185E4	E7	.1	220E4	1	1	1	0	2,205,472	132,316E2	1	1:18
36	.099	.9	.001	185E4	E9	.1	220E4	1	1	1	0	2,132,510	100,257E4	.13	3:10
37	.08999	.91	3E−7	185E4	E7	.1	220E4	1	1	1	0	2,205,337	135,929E2	1	1:57
38	.09	.90999	2E−7	185E4	E7	.1	220E4	1	1	1	0	2,206,307	136,992E2	.913	2:14
39	.005	.99499	E−6	185E4	E8	.1	220E4	1	1	1	0	2,181,185	165,906E3	.13	3:02
40	.009	.99	.001	185E4	E8	.1	220E4	1	1	1	0	2,184,267	136,769E3	.13	2:48
41	.009	.99	.001	185E4	E9	.1	220E4	1	1	1	0	2,131,358	103,095E4	.13	2:12
42	.49999	.5	5E−7	185E4	E7	.1	220E4	1	1	1	0	2,209,452	108,300E2	.93	1:49
43	.89999	.1	5E−7	185E4	E7	.1	220E4	1	1	1	0	2,222,635	970,609E1	.983	6:06
44	.89999	.1	E−9	185E4	E10	.1	221E4	0	1	1	1	2,215,469	851,956E3	.13	2:30
45	.79999	.2	2E−6	185E4	E7	.1	220E4	1	1	1	0	2,209,999	930,143E1	1	3:51
46	.94999	.05	5E−6	185E4	E6	.1	220E4	1	1	1	0	2,215,476	101,924E1	1	4:23
47	.98999	.01	E−5	185E4	E6	.1	220E4	1	1	1	0	2,215,543	100,369E1	1	1:26
48	.99899	.001	E−5	215E4	E6	.1	220E4	1	1	1	0	2,239,909	251,188	1	1:46
49	.9989	.001	E−4	215E4	E5	.1	220E4	1	1	1	0	2,221,516	100,072	1	3:26
50	.99989	E−4	E−6	215E4	E5	.1	220E4	1	0	1	1	2,279,960	61,998	1	3:47
51	.99998	E−6	E−5	220E4	E4	.1	220E4	1	0	1	1	2,278,040	2229	1	4:31
52	.99999	E−6	E−6	220E4	E2	.1	220E4	1	1	1	1	2,689,734	0	1	3:24
53	.99998	E−6	E−5	220E4	E4	.1	222E4	1	1	1	0	2,225,870	4104	1	3:48
54	.99999	E−6	E−6	220E4	E2	.1	222E4	1	1	1	0	2,224,348	57	1	4:39
55	.99989	E−4	E−6	200E4	E6	.1	225E4	1	1	1	0	2,215,559	10,0002E1	0	1:29

Then, we use LINGO 10 to solve the problem on a PC Pentium IV 2.1-GHz processor and to generate different Pareto-optimal solutions. Table 2 shows 55 generated Pareto-optimal configurations (1 means the bottling plant is built and 0 otherwise), the values of the expected total cost, the variance of the total cost, the financial risk and the computational times (mm:ss).

To generate the Pareto-optimal solutions, b, g and Ω are varied manually. When one of the parameters is varied and the others are fixed, changing the output shows its sensitivity with respect to that parameter. According to the obtained absolute minimum values for the expected total cost, the variance of the total cost and the financial risk, by solving the associated single objective problems, b_3 is fixed at .1, b_2 is varied from 100 to 10,000,000,000, b_1 is varied from 1,850,000 (close to the absolute minimum expected total cost) to 2,200,000 (close to the absolute minimum expected total cost plus three times of the maximum goal for the standard deviation of the total cost), g_1 is varied from .00000001 to .99999, g_2 is varied from .000001 to .99999, g_3 is varied from .000000001 to .001 and Ω is varied from 2,100,000 to 2,250,000.

As mentioned, the weights relate the relative under-attainment of the goals and a smaller g_j is associated with the more important objectives. For each goal vector b, the corresponding weight vector g can be obtained using Saaty's method of pairwise comparisons (Hwang and Yoon, 1981). For each pair of b and g, the solution is Pareto-optimal. If we are not satisfied with any Pareto-optimal solution or there are much differences between some of the obtained objective function values and the corresponding goals, the g vector should be modified. For example, if the obtained financial risk value is much greater than .1, g_3 should be decreased (e.g. 10 times) and both g_1 and g_2 should be increased from their earlier values, appropriately, in which the summation of g_j remains unchanged. This process continues with several different pairs of b and g, and several Pareto-optimal solutions are generated that can be used for decision-making.

For example, the first set of g in Table 2 (instance 1) implies that one dollar deviation of the expected total cost from 1,850,000 is about 1,000,000 times as important as one unit deviation of the variance of total cost from 1,000,000,000 and the same important as .01 deviation of the financial risk from .1. In this instance, the goal and weight for the expected total cost and weight for financial risk are relatively low, which causes the solution to have low expected total cost and risk values. In instance 26, we have a high budget Ω, and a low financial risk weight g_3, which seems to indicate that the solution should have a low risk value, but the goal for variance, b_2, is very low, with a low weight g_2, and in the solution this overrides risk, and the variance is minimized.

The lowest expected cost is from instance 29, with a value of 1,853,385. This has relatively low risk, but high variance. The optimal variance is obtained in two separate instances (26 and 52), but these give widely different values for cost (2.69E6 or 1.02E8). The optimal financial risk is obtained in instance 55, with a cost of 2.22E6, a variance of 1.00E6 and a budget of 2.25E6. So, the expected cost ranges from 1.85E6 to 1.02E8. If we aim for a financial risk of .13, within the budget of 2.20E6, then the variance can still range from 1.37E8 to 3.10E11, and the cost ranges from 1.85E6 to 2.18E6.

In order to make sense of this, and to arrive at an appropriate solution, the decision-maker needs to see this range of outcomes, to be able to trade-off one criteria against the other in terms of the results. So, by solving this SC design problem, it is concluded that the relationship between cost, variance and risk is not clear and it is not possible to easily define a utility function (a relation between risk and return). That is why we are adopting a multi-objective approach to this SC problem.

According to the numerical results, increasing goal for the variance and also decreasing weight for the expected total cost cause the financial risk to be decreased. Moreover, increasing goal for the variance causes both the expected total cost and the financial risk to be decreased. Also, increasing goal for the expected total cost causes the variance of the total cost to be decreased. So, it seems by increasing goal for any of the objectives, we give more space for other objectives to be improved. It is also concluded that there are some positive correlations between the expected cost and the financial risk.

It is also seen that in most instances we have to build the bottling plant in F, which is expandable, and then expand it when we either face boom economy or reliable suppliers. For example, in instance 5, where the bottling plants are built in E, F and H, the capacity of F should be expanded 40, 31.4 and 40 units, when the economy is boom and D is reliable (scenario 1), the economy is fair and D is reliable (scenario 3) and the economy is boom and D is not reliable (scenario 5), respectively.

In order to evaluate the performance of the proposed method in solving larger cases, we consider another problem with 10 suppliers, 10 plants and 10 customer centers with the same number of unreliable suppliers, expandable plants and scenarios as the earlier case. In this case, the unit production costs, market demands and capacity expansion costs are uncertain, while the other parameters are supposed to be certain following the similar pattern of the earlier case. Then, it is solved on the same computer and 10 new Pareto-optimal solutions are generated. The mean computational time for this medium size case is equal to 14:18, while the mean computational time in 55 generated Pareto-optimal solutions of the earlier small size case was equal to 2:41.

In order to show the sensitivity of the numerical solution with respect to the number of scenarios, we also conduct two more experiments with 4 and 16 scenarios. In the smaller case with 4 scenarios, it is assumed that all suppliers are reliable. In the bigger case with 16 scenarios, it is assumed that the future economy will have eight situations, instead of four in the original problem, and one of the suppliers is unreliable. In both cases, the uncertain parameters are the unit production costs, capacity expansion costs and market demands, but with different values for each scenario, while the other parameters are all certain following the same pattern of the original problem. Then, 10 new Pareto-optimal solutions for each of the new problems are generated. The mean computational time for the smallest size case with 4 scenarios and the largest size case with 16 scenarios are equal to 0:57 and 6:46, respectively, comparable to 2:41 in the original case with 8 scenarios. So, it seems the proposed model can at least solve the medium size cases with limited number of scenarios in acceptable CPU time.

6. Conclusion

Determining the optimal SC configuration is a difficult problem since a lot of factors and objectives must be taken into account when designing the network under uncertainty. The proposed model in this paper accounts for the minimization of the expected total cost, the variance of the total cost and the financial risk in a multi-objective scheme to design a robust SC network. Therefore, this approach seems to be a good way of capturing the high complexity of the problem.

According to the numerical experiments, considering risk directly affects the design of the SC networks under uncertainty. By using this methodology, the trade-off between the expected total cost and risk terms can be obtained. The interaction between the design objectives has been shown. This way of generating different possible configurations will help the decision-maker determine the best design among all generated Pareto-optimal solutions based on his/her preferences.

We used the goal attainment technique, which is a variation of the goal programming technique, to solve the multi-objective SC design problem and to generate the Pareto-optimal solutions. Goal attainment method is one of the multi-objective techniques with priori articulation of preference information given. This method has the same disadvantages as those of goal programming, namely, the preferred solution is sensitive to the goal vector and the weighting vector given by the decision-maker. However, the goal attainment method has fewer variables to work with, and therefore is one of the best methods to solve this large-scale mixed-integer nonlinear programming problem, in terms of computational time. In this regard, using a meta-heuristic approach such as genetic algorithm or simulated annealing in solving large-scale cases would be suitable.

An interactive multi-objective technique such as SWT or STEM can also be used to solve the multi-objective problem (6). The main disadvantage of the interactive approaches is that the number of variables and also the number of stages which we need to solve the associated single-objective optimization problems to get the final solution are much more than the goal attainment technique. So, in terms of computational time, the goal attainment technique is much better than any interactive multi-objective technique for solving the SC design problem proposed in this paper.

The proposed model can also be extended to the multi-period case considering the associated production, transportation and especially inventory-holding costs at different time intervals. In this case, the suppliers' lifetimes can also be considered as independent random variables with time-dependent continuous or discrete distributions such as exponential or geometric. Then, we will need to develop a proper stochastic optimal control model to solve the resulting problem.

Acknowledgement

This work received support from Science Foundation Ireland (Grant 03/CE3/I405), Killam Trusts (Dalhousie University), Scientific and Technological Research Council of Turkey (Grant SOBAG-108K027) and Iran National Science Foundation (INSF).

References

Ahmed, S., Sahinidis, N.V., 1998. Robust process planning under uncertainty. Industrial and Engineering Chemistry Research 37, 1883–1892.

Alonso-Ayuso, A., Escudero, L.F., Garin, A., Ortuno, M.T., Perez, G., 2003. An approach for strategic supply chain planning under uncertainty based on stochastic 0-1 programming. Journal of Global Optimization 26, 97–124.

Azaron, A., Katagiri, H., Kato, K., Sakawa, M., 2006. Modelling complex assemblies as a queueing network for lead time control. European Journal of Operational Research 174, 150–168.

Azaron, A., Katagiri, H., Kato, K., Sakawa, M., 2007. A multi-objective discrete reliability optimization problem for dissimilar-unit standby systems. OR Spectrum 29, 235–257.

Azaron, A., Katagiri, H., Sakawa, M., 2007. Time-cost trade-off via optimal control theory in Markov PERT networks. Annals of Operations Research 150, 47–64.

Birge, J.R., Louveaux, F., 1997. Introduction to Stochastic Programming. Springer, New York.

Bok, J.K., Lee, H., Park, S., 1998. Robust investment model for long range capacity expansion of chemical processing networks under uncertain demand forecast scenarios. Computers and Chemical Engineering 22, 1037–1050.

Bok, J.K., Grossmann, I.E., Park, S., 2000. Supply chain optimization in continuous flexible process networks. Industrial and Engineering Chemistry Research 39, 1279–1290.

Chen, C.L., Wang, B.W., Lee, W.C., 2003. Multi-objective optimization for a multi-enterprise supply chain network. Industrial and Engineering Chemistry Research 42, 1879–1889.

Chen, C.L., Lee, W.C., 2004. Multi-objective optimization of multi-echelon supply chain networks with uncertain demands and prices. Computers and Chemical Engineering 28, 1131–1144.

Cohen, M.A., Lee, H.L., 1989. Resource deployment analysis of global manufacturing and distribution networks. Journal of Manufacturing and Operations Management 2, 81–104.

Gjerdrum, J., Shah, N., Papageorgiou, L.G., 2000. A combined optimisation and agent-based approach for supply chain modelling and performance assessment. Production Planning and Control 12, 81–88.

Goh, M., Lim, J.Y.S., Meng, F., 2007. A stochastic model for risk management in global chain networks. European Journal of Operational Research 182 (1), 164–173.

Guillen, G., Mele, F.D., Bagajewicz, M.J., Espuna, A., Puigjaner, L., 2005. Multiobjective supply chain design under uncertainty. Chemical Engineering Science 60, 1535–1553.

Gupta, A., Maranas, C.D., 2003. Managing demand uncertainty in supply chain planning. Computer and Chemical Engineering 27, 1219–1227.

Gupta, A., Maranas, C.D., McDonald, C.M., 2000. Mid-term supply chain planning under demand uncertainty: Customer demand satisfaction and inventory management. Computers and Chemical Engineering 24, 2613–2621.

Hwang, C.L., Masud, A.S.M., 1979. Multiple Objective Decision Making. Springer, Berlin.

Hwang, C.L., Yoon, K., 1981. Multiple Attribute Decision Making. Springer, Berlin.

Kleywegt, A.J., Shapiro, A., Homem-De-Mello, T., 2001. The sample average approximation method for stochastic discrete optimization. SIAM Journal of Optimization 12, 479–502.

Lai, K.K., Ng, W.L., 2005. A stochastic approach to hotel revenue optimization. Computers and Operations Research 32, 1059–1072.

Markowitz, H.M., 1952. Portfolio selection. Journal of Finance 7, 77–91.

Markowitz, H.M., 1959. Portfolio Selection: Efficient Diversification of Investments. Wiley, New Jersey.

MirHassani, S.A., Lucas, C., Mitra, G., Messina, E., Poojari, C.A., 2000. Computational solution of capacity planning models under uncertainty. Parallel Computing 26, 511–538.

Mulvey, J.M., Vanderbei, R.J., Zenios, S.A., 1995. Robust optimization of large-scale systems. Operations Research 43, 264–281.

Petkov, S.B., Maranas, C.D., 1997. Multiperiod planning and scheduling of multipurpose batch plants under demand uncertainty. Industrial and Engineering Chemistry Research 36, 4864–4881.

Santoso, T., Ahmed, S., Goetschalckx, M., Shapiro, A., 2005. A stochastic programming approach for supply chain network design under uncertainty. European Journal of Operational Research 167, 96–115.

Timpe, C.H., Kallrath, J., 2000. Optimal planning in large multi-site production networks. European Journal of Operational Research 126, 422–435.

Tsiakis, P., Shah, N., Pantelides, C.C., 2001. Design of multiechelon supply chain networks under demand uncertainty. Industrial and Engineering Chemistry Research 40, 3585–3604.

van Delft, Ch., Vial, J.-Ph., 2004. A practical implementation of stochastic programming: An application to the evaluation of option contracts in supply chains. Automatica 40, 743–756.

Yu, C., Li, H., 2000. A robust optimization model for stochastic logistic problems. International Journal of Production Economics 64, 385–397.

J Comb Optim
DOI 10.1007/s10878-009-9207-z

Scheduling internal audit activities: a stochastic combinatorial optimization problem

Roberto Rossi · S. Armagan Tarim · Brahim Hnich · Steven Prestwich · Semra Karacaer

Abstract The problem of finding the optimal timing of audit activities within an organisation has been addressed by many researchers. We propose a stochastic programming formulation with Mixed Integer Linear Programming (MILP) and Constraint Programming (CP) certainty-equivalent models. In experiments neither approach dominates the other. However, the CP approach is orders of magnitude faster for large audit times, and almost as fast as the MILP approach for small audit times.

Roberto Rossi is supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain-Driven Research (CTVR) and Grant No. 05/IN/I886. S. Armagan Tarim and Brahim Hnich are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027.

R. Rossi (✉) · S. Prestwich
Cork Constraint Computation Centre, University College, 14 Washington St. West, Cork, Ireland
e-mail: rrossi@4c.ucc.ie

S. Prestwich
e-mail: s.prestwich@4c.ucc.ie

R. Rossi
Centre for Telecommunication Value-Chain Driven Research, University College, Dublin, Ireland
e-mail: robros@gmail.com

S.A. Tarim · S. Karacaer
Department of Management, Hacettepe University, Ankara, Turkey

S.A. Tarim
e-mail: armagan.tarim@hacettepe.edu.tr

S. Karacaer
e-mail: semra@hacettepe.edu.tr

B. Hnich
Faculty of Computer Science, Izmir University of Economics, Izmir, Turkey
e-mail: brahim.hnich@ieu.edu.tr

Published online: 21 January 2009

This work generalises a previous approach by relaxing the assumption of instantaneous audits, and by prohibiting concurrent auditing.

Keywords Uncertainty · Audit scheduling · Combinatorial optimization · Mathematical programming · Constraint programming

1 Introduction

Based on costs and benefits that change over time, the focus of the internal audit scheduling problem is how often to conduct an internal audit on an auditable unit. Auditable units are the units upon which internal control procedures are applied, in order to safeguard assets and assure the reliability of information flows. The scope of auditable units depends on organizational characteristics: they could be organizational units (finance, accounting department), geographic regions (branches, cities) or activities (budgeting, purchasing, etc.) (Boritz and Broca 1986).

The problem of finding the optimal timing of audit activities within an organisation has been addressed by many researchers including Wilson and Ranson (1971), Hughes (1977), Boritz and Broca (1986), and Knechel and Benson (1991). The first study of audit scheduling was by Wilson and Ranson (1971) who found the audit frequency that minimizes the discounted present value of losses and audit costs. Audit costs are assumed to be incurred at a uniform rate, while losses in the absence of auditing are assumed to rise exponentially from zero to an asymptotic level. After an audit is conducted, the losses drop to zero but start to accrue until the next audit. In Hughes (1977) the audits are chosen at the beginning of each of an infinite number of periods, conditional upon available information concerning the state of internal control system, and a model is proposed for determining the optimal timing of internal audits. The model proposed by Boritz and Broca (1986) determines the optimal audit interval, assuming that expected losses accrue if a unit remains unaudited and audit cost is incurred each time the decision to audit is made. In order to use their formulation, each audit unit is assessed using an index of loss riskiness called the Audit Unit Priority Score (AUPS). The parameters of the model (i.e. the shape of the loss function and the rate of increase in losses) are determined through auditors' judgmental process, which gives auditors flexibility in the scheduling of audit activities.

In a recent paper by Tarim et al. (2008) a stochastic version of the internal audit scheduling problem is formulated under relatively relaxed assumptions. Unlike previous models that determine optimal timing for one audit unit, their model determines the optimal timing of audit activities for multiple audit units. This is important because many firms have more than one auditable unit to which audit resources must be allocated. Moreover, their formulation takes into account uncertainty in the losses accrued in the absence of auditing, and employs a chance-constraint to keep the expected losses below a certain level with a given probability. However, in Tarim et al. (2008) it is assumed that audit activities are instantaneous, i.e. that conducting an audit does not take any time. The computational issues are not addressed by Tarim et al.

Mixed Integer Linear Programming (MILP) (Nemhauser and Wolsey 1988) and Constraint Programming (CP) (Hanus 2001) are two orthogonal approaches used to address combinatorial problems. MILP-based methods are rooted at the area of Operations Research (Nemhauser and Wolsey 1988), whereas CP-based methods are the result of research by the Artificial Intelligence community in the areas of Logic Programming and Constraint Satisfaction (Colmerauer 1985; Van Hentenryck 1989; Tsang 1993). MILP and CP methods have both been successfully applied to solve diverse problems such as network synthesis, crew scheduling, planning, and capital budgeting. Determining which type of problems or instances are best solved by which method is an active research area. This is the line of research pursued in this paper. We relax the assumption of instantaneous audits and propose a stochastic programming formulation for this important class of combinatorial problems involving uncertainty. To solve this stochastic program we develop two alternatives deterministic equivalent models: a MILP model and a CP model. The two approaches are complementary in the sense that for some instances MILP is superior and for other instances CP is superior. Our numerical experiments show that the certainty-equivalent MILP formulation is efficient when the time to perform an audit is relatively short. However, as the audit time gets longer our CP model proved to be significantly much more effective than the MILP model.

The paper is organised as follows. Section 2 describes the problem, Sect. 3 provides a stochastic programming formulation of the problem, Sect. 4 surveys possible solution methods, Sect. 5 reports experimental results comparing methods, and Sect. 6 concludes the paper. Finally, in the Appendix we provide a complete list of the notation adopted in the paper.

2 Problem statement

We consider a planning horizon comprising N time periods. We are given a set of M audit units over which random losses may accrue over time. In particular, l_t^m corresponds to the losses that accrue in audit unit m during period t. l_t^m is a random variable with a known probability density function $g_{l_t^m}(l_t^m)$. For convenience, losses in each period are assumed to be normally distributed with a constant coefficient of variation: $\rho = \sigma_t^m / \mu_t^m$ in this problem, but this assumption may be relaxed without loss of generality. The distribution of losses may vary from period to period, i.e. it is non-stationary. Losses in different time periods are assumed to be independent. Figure 1 illustrates expected losses on a single auditable unit.

Without loss of generality, we consider the case in which a single audit team has to be employed to keep losses under control. Auditing is a time-consuming task, and we

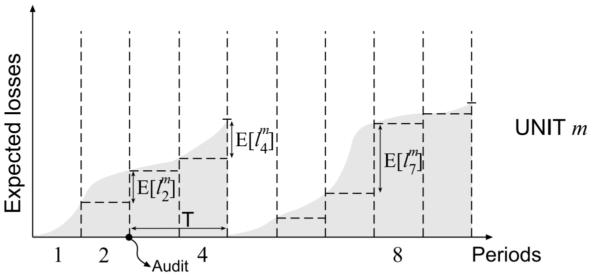
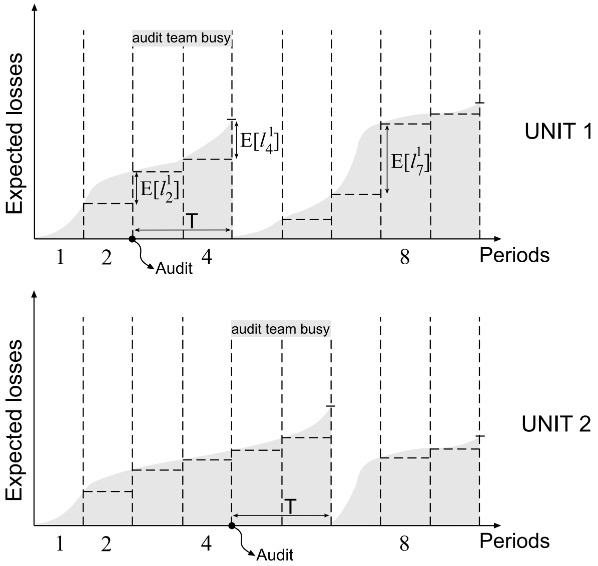


Fig. 1 Expected losses; $E[l_t^m]$ denotes the expected value of l_t^m

Fig. 2 Multiple units



assume that the team is given a strict deadline for performing an audit. Specifically, an audit must be completed in T time periods ($T > 0$). Therefore after T periods the accrued losses will drop to zero. If a team has already started auditing a unit at a given period, then no other audit can be initiated during this period for the given audit team. Figure 2 depicts such a situation in which an audit duration of 2 periods is assumed. An audit scheduled for unit 1 at the beginning of period 3 rules out any following audit for unit 2 until period 5.

Note that the timing of audits are fixed once and for all at the beginning of the planning horizon, and cannot be changed thereafter even if it is suspected that certain auditable units have accrued unexpected losses. The objective is to find the optimal audit schedule while respecting the maximum loss level criterion. That is, the invariant audit costs (i.e. fixed audit costs incurred each time an audit is conducted) and expected total discounted audit losses (i.e. cumulative losses accrued at the end of each period) are minimized by satisfying a maximum loss level constraint, which in this problem is defined by specifying a minimum probability α that the losses will not exceed a predetermined level $\bar{L}$ (allowed maximum loss) in any given audit period for any auditable unit.

Example 1 In what follows we will employ a running example to better exemplify the above concepts. We consider the following simple instance:

M: the total number of audit units, equal to 2

N: the number of periods in the planning horizon, equal to 6

T: the duration of an audit in time periods, equal to 2

a: the fixed cost incurred each time an audit is conducted, equal to 100

h: the loss discount factor measuring the opportunity cost associated to a given loss level, equal to 1

$\bar{L}$: a threshold indicating the maximum allowed loss level in each period, equal to 200

α: the probability of not exceeding the loss threshold $\bar{L}$, equal to 0.95.

We assume the losses accrued in each period to be normally distributed with a constant coefficient of variation $\rho = 0.2$, where $\rho = \sigma_t^m / \mu_t^m$. The expected value, μ_t^m, for the losses in each period t and for each audit unit m is respectively $\{50, 30, 50, 30, 50, 30\}$ in each period $t = 1, \dots, 6$ for audit unit 1, and $\{10, 20, 30, 40, 50, 60\}$ in each period for audit unit 2.

3 Stochastic programming formulation

Stochastic programming (Birge and Louveaux 1997) is a well known modeling technique that deals with problems where uncertainty comes into play. Problems of optimization under uncertainty are characterized by the necessity of making decisions without knowing what their full effect will be. Stochastic programming needs to represent uncertain elements of the problem. Typically random variables are employed to model this uncertainty to which probability theory (Ventsel 1979; Jeffreys 1961) can be applied. For this purpose such uncertain elements must have a known probability distribution. The typical requirement in stochastic programs is to maintain certain constraints, called *chance constraints* (Charnes and Cooper 1959), satisfied at a prescribed level of probability. The objective is typically related to the minimization/maximization of some expectation on the problem costs.

The stochastic programming model we propose balances the discounted cost of losses accrued due to lack of audits with the cost of conducting audits. Taking the discounted cost of losses into account is particularly relevant when the cost of money must be considered. Consider, for instance, the situation in which some losses are due to a specific reason in a given period. Then in the following periods this loss will affect company assets until the originating factor is discovered by an audit and cleared. Obviously these effects will have a higher impact the longer it takes to clear such an originating factor. Consider, for instance, the case in which a company's tax liabilities are overestimated. The capital tied into tax liabilities could be invested in a more profitable way if the accounts were not flawed. In this case the discount factor would reflect the opportunity cost associated with the fact that capitals may be invested in a more profitable way if an audit were scheduled.

We employ the expected value criterion to minimize the sum of the expected discounted period losses and audit costs over an N period planning horizon. Let us consider, without loss of generality, an initial loss levels L_1^m set to any non-negative values for each audit unit $m = 1, \dots, M$. Let

L_t^m: the loss level in audit unit m at the beginning of period t.

The objective function below and the following constraints give the optimum audit timing for each audit unit by minimizing $E[TC]$, that is the sum of expected audit costs and discounted period losses that are expected to accrue in the absence of auditing.

$$\min E[TC] = \sum_{m=1}^{M} \int_{l_1^m} \int_{l_2^m} \cdots \int_{l_N^m} \sum_{t=1}^{N} (aK_t^m + h(L_t^m + l_t^m))$$
$$\times\, g_{l_1^m}(l_1^m) g_{l_2^m}(l_2^m) \cdots g_{l_N^m}(l_N^m) \mathrm{d}(l_1^m) d(l_2^m) \ldots d(l_N^m) \qquad (1)$$

where

M: the total number of audit units

N: the number of periods in the planning horizon

a: the amount of cost incurred each time an audit is conducted

h: the loss discount factor measuring the opportunity cost associated to a given loss level

K_t^m: a variable that takes the value of 1 if an internal audit (lasting T periods) is started for audit unit m in period t, otherwise 0.

The above objective function is subject to several constraints. If ($K_t^m = 1$) an internal audit is conducted at the beginning of period t (i.e. at the end of period $t - 1$), then the loss level at the beginning of period $t + T$ should be 0. Yet, if an internal audit is not conducted, the loss level at the beginning of period $t + T$ will be equal to the loss level at the beginning of the preceding period plus the loss accrued during the preceding period. This can be expressed as

$$L_{t+T}^m \geq L_{t+T-1}^m + l_{t+T-1}^m - \mathfrak{M} K_t^m, \quad m = 1, \ldots, M,\ t = 1, \ldots, N - T, \qquad (2)$$

where $\mathfrak{M}$ is some very large number. Obviously in the first T periods no audit can be completed, therefore

$$L_t^m \geq L_{t-1}^m + l_{t-1}^m, \quad m = 1, \ldots, M,\ t = 1, \ldots, T. \qquad (3)$$

For convenience we consider $l_t^m = L_t^m = 0$, for $\{t \mid t < 1\}$.

Now consider a plan for audit unit m, which schedules r audits over the N period planning horizon with audits conducted at $\{A_1^m, \ldots, A_r^m\}$, where $A_j^m > A_{j-1}^m$, $A_r^m \leq N - T$. For convenience $A_1^m = 1 - T$, because the initial loss level is set to 0; $A_{r+1}^m = N - T + 1$ is defined as the earliest period for which an associated audit

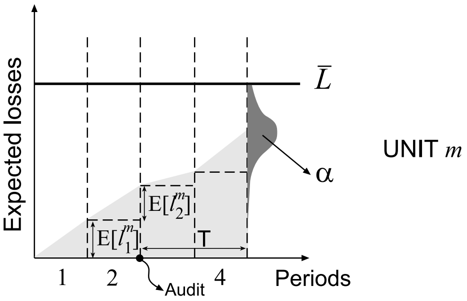


Fig. 3 Chance-constraint on the maximum loss level. Assuming losses to be normally distributed: α is the desired minimum probability (*area marked* in the figure) that the loss level in any time period will not exceed a subjectively determined level, $\bar{L}$

would be completed only after the end of the horizon. The associated audits will take place at the beginning of periods A_i^m, $i = 1$ to r. In the considered plan there are clearly no audits scheduled for audit unit m except at periods $A_1^m, \dots, A_r^m$. The accumulated loss level L_{t+1}^m carried over from period t to period $t+1$ is the loss to date since the last completed audit. This can be written as

$$L_{t+1}^m = \sum_{k=A_i^m+T}^{t} l_k^m, \quad A_i^m + T \le t < A_{i+1}^m + T, \quad i = 1, \dots, r. \tag{4}$$

As defined above, α is the desired minimum probability that the loss level in any time period will not exceed a subjectively determined level, $\bar{L}$ (Fig. 3). In this regard the chance constraint becomes

$$\Pr\{L_t^m + l_t^m \le \bar{L}\} \ge \alpha, \quad t = 1 + T, \dots, N. \tag{5}$$

Using (4), this can be written alternatively as, for $t \ge 1 + T$

$$\Pr\left\{ \sum_{k=A_i^m+T}^{t} l_k^m \le \bar{L} \right\} \ge \alpha, \quad A_i^m + T \le t < A_{i+1}^m + T, \quad i = 1, \dots, r, \tag{6}$$

which implies

$$G_{l^m_{A_i^m+T} + l^m_{A_i^m+T+1} + \dots + l_t^m}(\bar{L}) \ge \alpha, \quad A_i^m + T \le t < A_{i+1}^m + T, \tag{7}$$

where $G_{l_t^m}(x) = \int_{-\infty}^{x} g_{l_t^m}(\tau)\mathrm{d}\tau$ is the cumulative distribution function of l_t^m. By assuming $G_{l_t^m}(x)$ to be strictly increasing, thus invertible, (7) can then be rewritten as

$$\bar{L} \ge G^{-1}_{l^m_{A_i^m+T} + l^m_{A_i^m+T+1} + \dots + l_t^m}(\alpha), \quad A_i^m + T \le t < A_{i+1}^m + T, \tag{8}$$

where $G^{-1}_{l_t^m}(\alpha)$ is the inverse cumulative distribution function (or α-quantile) of l_t^m.

Since the problem has a finite planning horizon of N periods, for all the relevant cases the right-hand side of (8), $G^{-1}_{l^m_{A_i^m+T} + l^m_{A_i^m+T+1} + \dots + l_t^m}(\alpha)$, can be computed or possibly read from a table, once the form of $g_{l_t^m}(.)$ is decided. If the binary variable $P_{t,j}^m$ is defined as taking a value of 1 if the most recent audit prior to period t was in period j and zero elsewhere for a given audit unit m, then (8) can be written as

$$\bar{L} \ge \sum_{j=1}^{t} \big(G^{-1}_{l_j^m + l_{j+1}^m + \dots + l_t^m}(\alpha)\ P_{t,j-T}^m \big). \tag{9}$$

There can be at most only one most recent audit prior to period t. Thus $P_{t,j}^m$ must satisfy

$$\sum_{j=1-T}^{t} P_{t,j}^m = 1, \quad m = 1, \dots, M, \ t = 1, \dots, N. \tag{10}$$

Note that losses in the first T periods cannot be controlled with respect to the threshold $\bar{L}$ and the probability α. In fact benefits from the first possible audit appear only in period $T+1$. Therefore we assume, for all $m=1,\dots,M$, $K^m_{1-T}=1$ (according to the fact that initial losses should be equal to 0) and $K^m_{i-T}=0$, $i=2,\dots,T$.

The following equation (11) is necessary to identify uniquely the period in which the most recent audit prior to any period t took place. For each audit unit $m=1,\dots,M$

$$P^m_{t,j} \geq K^m_j - \sum_{k=j+1}^{t-T} K^m_k, \quad t=1,\dots,N,\ j=1-T,\dots,t-T. \tag{11}$$

It is not common practice for internal audit teams to conduct multiple audits simultaneously. To have a modicum of resemblance to reality, as already stated, here it is assumed that a team can conduct an internal audit only for one audit unit at a given time period. In our model we shall consider the following capacity constraint

$$\sum_{k=1}^{m} K^m_t \leq C, \quad m=1,\dots,M,\ t=1,\dots,N-T, \tag{12}$$

which states that the firm can assign at most C audit teams to conduct audits in any given time period. For simplicity, in what follows we will assume $C=1$.

Example 2 For the running example introduced in Sect. 2, in Table 1 we show the values of $G^{-1}_{l^m_j+l^m_{j+1}+\cdots+l^m_t}(\alpha)$ for audit unit $m=1,2$. Values in italic are those that satisfy constraint (9), that is values that stay below the loss threshold $\bar{L}=200$. Underlined values identify audit cycles[1] in the optimal solution. The optimal audit plan is also presented graphically in Fig. 4. In this plan, in order to keep accrued losses under control, one single audit for unit 1 is scheduled at the beginning of period 2, and this audit terminates at the end of period 3. Similarly, for unit 2 a single audit is scheduled at period 4 and this audit terminates at the end of period 5. The expected total cost for this plan is 1090.

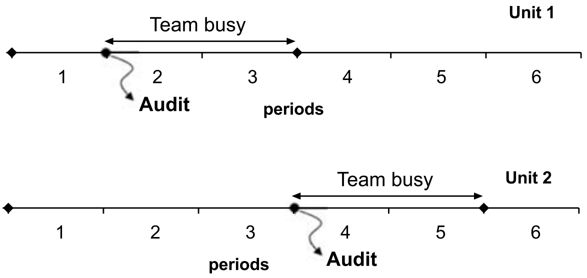


Fig. 4 Optimal audit plan for the numerical example

[1] An audit cycle is a set of periods $\{j,\dots,t\}$, $j\leq t$, where j is the first period after the completion of an audit, and no other audit is completed by period t.

Table 1 Values of $G^{-1}_{l^m_j+l^m_{j+1}+\cdots+l^m_t}(\alpha)$ for audit unit $m=1,2$. Values in italic are those that satisfy constraint (9). That is, values that stay below the loss threshold $\bar{L}=200$. Underlined values identify audit cycles in the optimal solution

	t					
j	1	2	3	4	5	6
Unit 1						
1	*66.5*	*99.2*	$\underline{\textit{155.3}}$	*187.1*	241.7	273.2
2		*39.9*	*99.2*	*131.6*	*187.1*	218.9
3			*66.5*	*99.2*	*155.3*	*187.1*
4				*39.9*	*99.2*	$\underline{\textit{131.6}}$
5					*66.5*	*99.2*
6						*39.9*
Unit 2						
1	*13.3*	*37.4*	*72.3*	*118.0*	$\underline{\textit{174.4}}$	241.4
2		*26.6*	*61.9*	*107.7*	*164.2*	231.2
3			*39.9*	*86.5*	*143.3*	210.5
4				*53.2*	*111.1*	*178.9*
5					*66.5*	*135.7*
6						$\underline{\textit{79.7}}$

We now show in detail how the values in Table 1 are computed. Let G_N be the standard normal distribution function. This function is strictly increasing, therefore G_N^{-1} is uniquely defined.[2] $G_N^{-1}(0.95)=1.645$ corresponds to the 0.95-quantile. Therefore, since all the random variables l_t^m, $t=1,\dots,N$, $m=1,\dots,M$, are independent and normally distributed, $G^{-1}_{l^1_2+l^1_3+l^1_4}(0.95)=1.645\cdot 0.2\cdot\sqrt{30^2+50^2+30^2}=131.6$. This value can be found in the first matrix presented in Table 1 at position (2, 4). It corresponds to a sequence of periods starting in period 2 and ending in period 4, where no audit is completed and where the last audit performed has been completed by the end of period 1 (therefore losses at the beginning of period 2 are null). Since $131.6<200$ it follows that this set of periods constitutes a feasible audit cycle.

4 Solution methods

In this section we present two alternative certainty equivalent (Birge and Louveaux 1997) models for the stochastic programming model presented in the former section: a Mixed Integer Linear Programming model and a Constraint Programming model.

4.1 A certainty equivalent MILP model

In Linear Programming (Dantzig 1963; Chvtal 1983; Schrijver 1986) a model, called a "program", consists of continuous variables and linear constraints (inequalities or

[2] Tables are available for obtaining values of the inverse normal cumulative distribution function (also known as *α-quantile*, Ventsel 1979).

equalities), and the aim is to optimize(minimize or maximize) a linear cost function. In matrix notation the standard form of a linear program is $\min\{c^{\mathrm{T}}x \mid Ax = b, x \geq 0\}$, where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$. Here c represents the cost vector and x is the vector of variables. Linear programs are usually solved by the *simplex method* (Dantzig 1951) which is very fast in practice, though it has exponential worst-case time complexity. A Mixed Integer Linear Program (MILP) is a Linear Program (LP) plus integrality requirements on some decision variables. Discrete variables in MILP are often 0–1 variables representing yes/no, on/off or true/false choices. Logical conditions between 0–1 variables such as $\vee$ (or), $\wedge$ (and), $\neg$ (not), $\Rightarrow$ (if ... then), and $\Leftrightarrow$ (if and only if) can be imposed using appropriate linear constraints (Williams 1994). Solution methods for MILP typically employ tree search in which internal nodes correspond to partial solutions, branches are choices partitioning the search space, and leaf nodes are solutions. Branching is intertwined with a relaxation to eliminate the exploration of nodes for which the relaxation is either infeasible or worse than the best solution found so far. Each node represents a partial assignment of the discrete variables, and at each node a relaxation is formed by turning the integrality requirements into bounds, thus transforming the subproblem into an LP. This LP is solved, and if the solution is not suboptimal then descendant nodes are formed by branching on the fractional relaxation value of a discrete variable. The historical popularity of MILP derives from Dantzig's discovery that the vocabulary of LP is surprisingly versatile in many applications. MILP-based methods have been developed over the last four decades by the Operations Research community (Nemhauser and Wolsey 1988).

The mathematical programming model of the previous section, as presented, is a stochastic nonlinear combinatorial optimization model, which is extremely complex to solve. In this section we adopt the static-dynamic uncertainty strategy, proposed by Bookbinder and Tan (1988) to solve their stochastic inventory lot-sizing problem, and apply it to the mathematical programming model of Sect. 3. The model can be expressed as minimizing the objective function given in (1) subject to the constraints (2–4), (9–12), and non-negativity and 0/1 integrality conditions for L_t^m, K_t^m and $P_{i,j}^m$.

In our internal audit scheduling problem the analysis is completed at the beginning of the planning horizon by taking expectations (see Bookbinder and Tan 1988). Hence the deterministic equivalent model for the original chance-constrained stochastic programming model is obtained. The resultant model is in the form of a mixed-integer program, given below, in which the expected value operator is denoted by $E[.]$.

$$\min \quad \sum_{m=1}^{M}\left(\sum_{t=1}^{N} aK_t^m + \sum_{t=1}^{N} hE[L_t^m + l_t^m]\right) \tag{13}$$

$$\text{subject to, for } m = 1, \ldots, M, \tag{14}$$

$$E[L_1^m] = 0, \tag{15}$$

$$E[L_{t+1}^m] \geq E[L_t^m] + E[l_t^m], \quad t = 1, \ldots, T, \tag{16}$$

$$E[L_{t+T}^m] \geq E[L_{t-1+T}^m] + E[l_{t-1+T}^m] - \mathfrak{M}K_t^m, \quad t = 1, \ldots, N-T, \tag{17}$$

$$\sum_{k=1}^{M} \sum_{h=1}^{\min(t+T-1,N)} K_h^k \leq 1, \quad t = 1, \ldots, N, \tag{18}$$

$$\bar{L} \geq \sum_{j=1}^{t} G^{-1}_{l_j^m + l_{j+1}^m + \cdots + l_t^m}(\alpha) \cdot P^m_{t,j-T}, \quad t = 1, \ldots, N, \tag{19}$$

$$\sum_{j=1}^{t} P^m_{t,j} = 1, \quad t = 1 - T, \ldots, N, \tag{20}$$

$$P_{m,t,j} \geq K_j^m - \sum_{k=j+1}^{t-T} K_k^m, \quad j = 1 - T, \ldots, t - T, \tag{21}$$

$$K^m_{1-T} = 1, \tag{22}$$

$$K^m_{t-T} = 0, \quad t = 2, \ldots, T, \tag{23}$$

$$E[L_t^m] \geq 0, \tag{24}$$

$$K_t^m, P^m_{t,j} \in \{0, 1\}, \quad t = 1, \ldots, N, \ j = 1 - T, \ldots, t. \tag{25}$$

This model thus determines the optimal audit schedule by balancing the fixed audit costs and discounted expected period losses that accrue in the absence of auditing. The problem is to determine the values of the $0/1$ integer variables, K_t^m for $m = 1, \ldots, M$, $t = 1, \ldots, N$, and $P^m_{t,j}$ for $t = 1, \ldots, N$, $j = 1 - T, \ldots, t$, and the non-negative continuous variable $E[L_t^m]$ for $t = 1, \ldots, N$, that minimize the objective function. The times of the audit reviews in each audit unit m are given by the values of t such that $K_t^m = 1$. Constraint (15) states that the initial losses are equal to 0. Constraint (16) lets expected losses accumulate in the first T periods for each audit unit, since no audit can be terminated before period $T + 1$. Constraint (17) states that if an audit is planned in period t, then expected losses must drop to zero in period $t + T$, as soon as the audit terminates, while if no audit is planned in period t, then expected losses in period $t + T$ must be equal to the expected losses accumulate till the beginning of the previous period $(t + T - 1)$ plus the expected losses accrued in such a period. Constraint (18) prevents multiple audits in any given period. If an audit team starts an audit in period t on a given unit, this means that no other audit can be performed on any unit before period $t + T$. Constraints (19–21) implement (8) and therefore they identify feasible audit schedules, that is those for which losses never exceed the given threshold $\bar{L}$ more than α percent of the times.

Example 3 By employing the mathematical programming model presented in this section we can solve the running example originally presented in Sect. 2. The optimal plan, which we already described in Sect. 3, is shown in Fig. 5. In this picture, we also show the expected losses accumulated in each period and computed by the MILP model. The plan schedules one single audit for unit 1 at period 2, and for unit 2 at period 4. The expected total cost for this plan is 1090. Note that for each audit, losses drop to 0 only after $T = 2$ periods, which is in fact the time required to perform an audit. It is also clear from the plan shown that the audit team cannot perform multiple audits at any given time period.

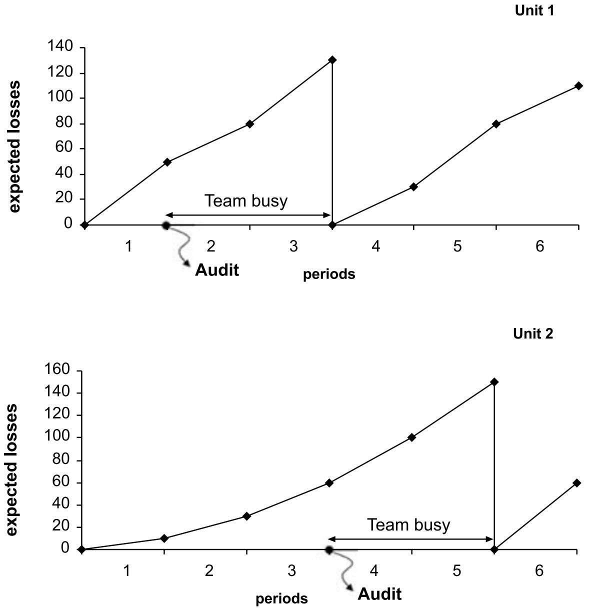


Fig. 5 Optimal audit plan for the numerical example

4.2 A certainty equivalent CP model

A *Constraint Satisfaction Problem* (CSP) (Apt 2003; Brailsford et al. 1999; Lustig and Puget 2001) is a triple $\langle V, C, D\rangle$ where V is a set of decision variables, D is a function mapping each element of V to a domain of potential values, and C is a set of constraints stating allowed combinations of values for subsets of variables in V. A *solution* to a CSP is simply a set of values of the variables such that the values are in the domains of the variables and all the constraints are satisfied. We may also be interested in finding a feasible solution that minimizes or maximizes the value of a given objective function over a subset of the variables.

We now recall some key concepts in *Constraint Programming* (CP): constraint filtering algorithm, constraint propagation and arc-consistency (Regin 2003). A *filtering algorithm* is typically associated with a constraint, and removes values from the domains of its variables that cannot belong to any solution of the CSP. These algorithms are repeatedly called until no new deduction can be made, a process called *propagation*. In conjunction with this process CP uses a search procedure (typically a backtracking algorithm) in which filtering is systematically applied whenever the domain of a variable is modified. One of the most interesting properties of a filtering algorithm is *arc-consistency*: we say that a filtering algorithm associated with a constraint establishes arc-consistency if it removes all the values from the domains of the variables involved in the constraint that are not consistent with the constraint. Studies on arc-consistency are often limited to binary constraints, but modeling problems by means of binary constraints has drawbacks: they are not very expressive, and their domain reduction is typically weak. To overcome both these drawbacks, constraints that capture a relation among a non-fixed number of variables were introduced. These constraints are not only more expressive than the equivalent aggregation of simple constraints, but they can be associated with more powerful filtering algorithms that take into account the simultaneous presence of several simple constraints to further reduce the domains of the variables. These more powerful constraints are called *global constraints*. One of the best-known examples is the `alldiff` constraint (Regin 1994), both because of its expressiveness and its efficiency in establishing arc-consistency. A comprehensive and up-to-date survey of the state of knowledge regarding CP is Apt (2003), while a general account of the CP–MP relationship is given by Brailsford et al. (1999) and Lustig and Puget (2001).

In this section we propose a CP reformulation for the mathematical programming model in Sect. 4.1. This reformulation follows the approach originally proposed in Tarim and Smith (2008), it exploits non-binary (global) constraints and other features of CP. The CP model, similarly to the mathematical programming one, can be expressed as minimizing the objective function given in (1). But, as we shall see, in the CP model constraints (2–4) and (9–12) are now reformulated and expressed in a more compact and readable way. Furthermore, as we will see, the number of decision variables employed is dramatically reduced as we do not employ anymore the binary decision variables $P_{i,j}^m$. The number of constraints is also significantly reduced. The CP model is as follows.

$$\min \quad \sum_{m=1}^{M}\left(\sum_{t=1}^{N} aK_t^m + \sum_{t=1}^{N} hE[L_t^m + l_t^m]\right) \tag{26}$$

$$\text{subject to, for } m = 1, \ldots, M, \tag{27}$$

$$E[L_1^m] = 0, \tag{28}$$

$$E[L_{t+1}^m] \geq E[L_t^m] + E[l_t^m], \quad t = 1, \ldots, T, \tag{29}$$

$$K_t^m = 1 \to E[L_{t+T}^m] = 0, \quad t = 1, \ldots, N-T, \tag{30}$$

$$K_t^m = 0 \to E[L_{t+T}^m] = E[L_{t-1+T}^m] + E[l_{t-1+T}^m], \quad t = 1, \ldots, N-T, \tag{31}$$

$$\sum_{k=1}^{M} \sum_{h=1}^{\min(t+T-1,N)} K_h^k \leq 1, \quad t = 1, \ldots, N, \tag{32}$$

$$\Phi\Big[m, t+T, \max\Big(1, \max_{j=1,\ldots,t} (j+T) \cdot K_j^m\Big)\Big] \geq 0, \quad t = 1-T, \ldots, N-T, \tag{33}$$

where

$$\Phi[m, t, j] = \bar{L} - G^{-1}_{l_j^m + l_{j+1}^m + \cdots + l_t^m}(\alpha), \quad t = 1, \ldots, N, \; j = 1, \ldots, N.$$

The objective function, as in the mathematical programming model, balances the fixed audit costs and discounted expected period losses that accrue in the absence of auditing. Constraint (28) states that the initial losses are equal to 0. Constraint (29) lets losses accumulate in the first T periods for each audit unit, since no audit can be terminated before period $T+1$. Constraint (30) states that if an audit is planned in period t, then losses must drop to zero in period $t+T$, as soon as the audit terminates. Conversely, constraint (31) states that if no audit is planned in period t, then losses in period $t+T$ must be equal to the losses accumulated till the beginning of the previous period $(t+T-1)$ plus the losses accrued in such a period. These two non-linear constraints are equivalent to constraint (17) in the mathematical programming formulation. Constraint (32), similarly to constraint (18), prevents multiple audits in any given period. If an audit team starts an audit in period t on a given unit, this means that no other audit can be performed on any unit before period $t+T$. Constraint (33) identifies feasible audit schedules, that is those for which losses never exceed the given threshold $\bar{L}$ more than α percent of the time. This constraint replaces the set of constraints (19–21). The model given in (26–33) can be directly implemented using the OPL optimization programming language (Van Hentenryck 1999). It should be noted that, in OPL, constraint (33) is implemented using the `element`(I, A, J) constraint (Van Hentenryck and Carillon 1988). The `element` constraint holds iff $A[I]=J$, where I and J are decision variables, and A is an array of decision variables.

5 Experiments

In this section, we compare the computational performance of the MILP formulation, presented in Sect. 4.1, versus the CP equivalent formulation, presented in Sect. 4.2, on a number of test problems.

Computational tests are performed on a 1.5 GHz, 2 GB RAM, Centrino machine using ILOG Cplex 9.0 (Ilog 2007a) in OPL Studio 3.7 (Ilog 2007b). The packages are used with their default settings.

In the MILP model, the $\mathfrak{M}$ in constraint (17) must have a numerical value. It is well known that the computational performance of the MILP model can be improved by choosing $\mathfrak{M}$ as small as possible, without ruling out any possible solution. It is also clear that in different time periods, the corresponding $\mathfrak{M}$ may have been assigned different numerical values. One way of generating such $\mathfrak{M}$ is by observing that, by assuming a reasonably high service level (that is $\alpha > 0.5$) the loss level will never exceed $\bar{L}$. Hence, $\mathfrak{M}=\bar{L}$.

It should be also noted that, while the integer program is treated in its matrix form, and different heuristics are used to choose the variable to branch on based on the solution of the LP relaxation that is solved at each node, in a CP approach the user specifies the branching strategy in terms of the formulation of the problem. The following search strategy is employed in solving the CP model proposed: $K_t^m = 0$ and 1 are tried in order, for all $m \in \{1,\dots,M\}$, for all $t \in \{1,\dots,N\}$.

5.1 Experimental settings

The design of the test problems is as follows. We consider the following inputs:

M: the total number of audit units, equal to 5

N: the number of periods in the planning horizon, taking values in $\{20, 30, 40\}$

T: the duration of an audit in time periods, taking values in $\{1,\dots,6\}$

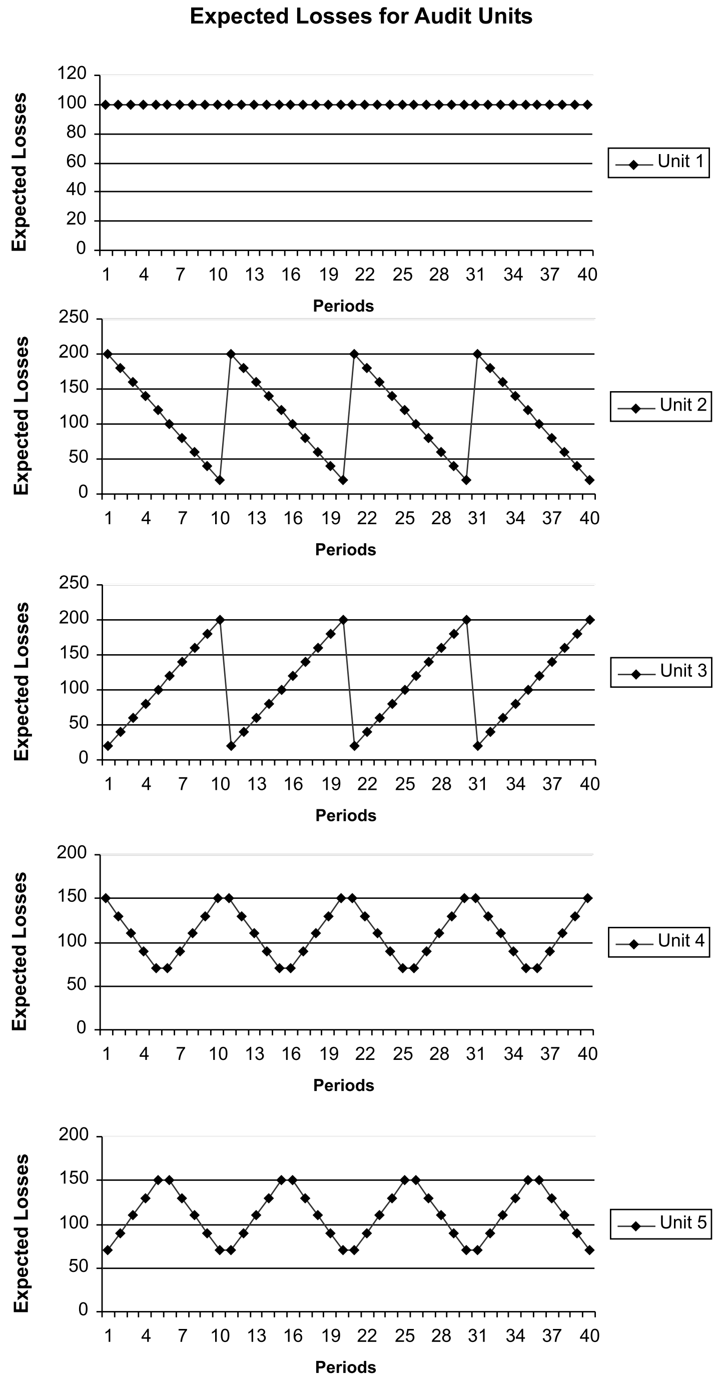


Fig. 6 Expected value μ_t^m for the losses in each period t and for each audit unit m

- a: the amount of cost incurred each time an audit is conducted, taking values in $\{500, 750, 1000\}$
- h: the loss discount factor measuring the opportunity cost associated to a given loss level, equal to 1
- $\bar{L}$: a threshold indicating the maximum allowed loss level in each period, taking values in $\{1500, 2500, 3500\}$
- α: the probability of not exceeding the loss threshold $\bar{L}$, equal to 0.95.

We assume the losses accrued in each period to be normally distributed with a constant coefficient of variation $\rho \in \{0.15, 0.3\}$, where $\rho = \sigma_t^m/\mu_t^m$. The expected value μ_t^m for the losses in each period t and for each audit unit m is shown in Fig. 6.

The total number of test problems generated is 108. We further partition our set of problem instances into two classes as follows:

- The instances where the audit time is 1, i.e., $T = 1$ (18 instances).
- The instances where the audit time is greater than 1, i.e., $T \in \{2, \dots, 6\}$ (90 instances).

We now analyze each set separately.

5.1.1 Instances for which $T = 1$

For each test problem the solution time (in seconds), for both the MILP and the CP approach, is given in Table 2. In this table italic figures highlight the approach that produced the best run time. In Table 3 instead we reported for the MILP approach and for the CP approach, respectively, the simplex iterations performed and the nodes explored. In this first set of 18 instances, where $T = 1$, the MILP approach always dominates the CP approach in terms of run time. Nevertheless the discrepancy between the two approaches reaches only one order of magnitude in the worst case. In the average case MILP is faster than CP by a factor of 7.7.

5.1.2 Instances for which $T > 1$

For each test problem the solution time (in seconds), for both the MILP and the CP approach, is given in Table 4. In this table, again italic figures highlight the approach that produced the best run time, while those instances for which the figures are underlined are infeasible. In Table 5 we reported for the MILP approach and for the CP approach, respectively, the simplex iterations performed and the nodes explored.

In contrast to what we observed in the first set of 18 instances, where $T = 1$, in this second set of 90 instances, where $T > 1$, the CP approach always dominates the MILP approach in terms of run time. When $T > 1$ the MILP approach does not scale particularly well with respect to N, $\bar{L}$ and ρ. Instances with a large N, T, $\bar{L}$ and ρ require, in fact, up to more than 30000 seconds to be solved. For these instances CP is able to quickly prove optimality or efficiently detect infeasibility. In contrast, CPLEX requires several simplex iterations and a long time to prove infeasibility. The discrepancy between the two approaches for infeasible problems reaches a factor of

Table 2 Computational times (in sec) for the MILP approach (MILP) and for the CP approach (CP). Italic figures in the table highlight the approach that produced the best run time

$\bar{L}$	1500			2500			3500			T	N
a	500	750	1000	500	750	1000	500	750	1000		
$\rho = 0.15$											
MILP	*1.1*	*0.98*	*0.7*	*1.4*	*1.5*	*0.56*	*0.72*	*1.4*	*0.59*	1	10
CP	6.2	7.6	8.2	6.7	8.1	8.2	7.2	9.2	8.8	1	10
$\rho = 0.3$											
MILP	*1.1*	*0.96*	*0.63*	*1.3*	*1.1*	*0.57*	*1.2*	*1.0*	*0.59*	1	10
CP	5.9	6.8	7.5	6.5	7.5	7.8	6.5	7.7	8.05	1	10

Table 3 Simplex iterations performed by the MILP approach (MILP-SI) and nodes explored by the CP approach (CP-Nod). Italic figures in the table highlight the approach that produced the best run time

$\bar{L}$	1500			2500			3500			T	N
a	500	750	1000	500	750	1000	500	750	1000		
$\rho = 0.15$											
MILP-SI	*14122*	*13383*	*8071*	*15868*	*13483*	*6666*	*15350*	*13854*	*7083*	1	10
CP-Nod	83480	103442	108266	83480	103442	108266	83480	103372	108266	1	10
$\rho = 0.3$											
MILP-SI	*16084*	*13460*	*8381*	*16937*	*13649*	*6683*	*15357*	*13854*	*7083*	1	10
CP-Nod 1	83480	103442	108266	83480	103442	108266	83480	103442	108266	1	10

3900, that is three orders of magnitude. Although MILP performs better at proving optimality, its performances are still far from those achieved by the CP approach. In fact the discrepancy between the two approaches with respect to feasible problems reaches a factor of 88: almost two orders of magnitude.

5.2 Discussion of results

The results presented indicate that the CP approach is in general more tractable than the mathematical programming one for this class of scheduling problems. The average solution time over all the instances considered is 950 seconds for the MILP approach and 24 seconds for the CP approach. This shows that, on average, CP is about one order of magnitude faster than MILP for the test bed analyzed. A comparison of solution times for the test problems reveals that, as the value of T increases (Fig. 7), CP is orders of magnitude faster than MILP, irrespectively of N, $\bar{L}$ and ρ. It should be noted that CP, as a consequence of constraint propagation, is extremely good at proving infeasibility, while this is the class of problems for which MILP requires significant computational efforts. CP also shows a more stable behavior and scalable performances as T, N, $\bar{L}$ and ρ increase. MILP performs poorly for the largest instances considered both in proving optimality and detecting infeasibility.

Table 4 Computational times (in sec) for the MILP approach (MILP) and for the CP approach (CP). Italic figures in the table highlight the approach that produced the best run time. Underlined figures are infeasible instances. +30000 means that the search has been stopped before infeasibility could be proved, after 30000 sec (8,3 hour)

$\bar{L}$	1500			2500			3500			T	N
a	500	750	1000	500	750	1000	500	750	1000		
$\rho = 0.15$											
MILP	150	120	98	240	240	200	310	420	310	2	20
CP	*9.4*	*9.6*	*9.6*	*92*	*110*	*130*	*110*	*120*	*170*	2	20
MILP	22	17	41	37	25	26	36	30	48	3	20
CP	*0.12*	*0.11*	*0.10*	*7.6*	*7.3*	*7.4*	*6.4*	*6.8*	*7.6*	3	20
MILP	20	110	80	500	320	400	610	540	610	4	30
CP	*0.07*	*0.07*	*0.07*	*8.3*	*7.9*	*8.4*	*100*	*100*	*100*	4	30
MILP	*0*	*0*	*0*	700	1500	810	250	140	420	5	30
CP	0.06	0.07	0.06	*1.0*	*1.0*	*1.0*	*20*	*20*	*21*	5	30
MILP	*0*	*0*	*0*	2300	1500	1900	660	1200	1500	6	40
CP	0.11	0.11	0.10	*0.45*	*0.39*	*0.41*	*17*	*20*	*17*	6	40
$\rho = 0.3$											
MILP	70	100	87	190	420	200	340	400	260	2	20
CP	*4.3*	*4.12*	*4.3*	*87*	120	130	*100*	*140*	*130*	2	20
MILP	17	12	36	22	31	26	36	34	33	3	20
CP	*0.08*	*0.08*	*0.08*	*6.2*	*7.2*	*6.9*	*6.5*	*6.8*	*7.4*	3	20
MILP	39	20	23	160	370	550	550	580	620	4	30
CP	*0.06*	*0.06*	*0.06*	*5.0*	*4.9*	*5.2*	*110*	*110*	*120*	4	30
MILP	*0*	*0*	*0*	400	310	320	300	130	440	5	30
CP	0.07	0.07	0.07	*0.77*	*0.77*	*0.77*	*20*	*20*	*20*	5	30
MILP	*0*	*0*	*0*	1600	1500	700	20000	24000	+30000	6	40
CP	0.1	0.1	0.1	*0.35*	*0.39*	*0.47*	*7.7*	*7.7*	*7.7*	6	40

The performance of CP-based and MILP-based approaches for solving a number of combinatorial optimization problems has been the scope of many recent studies (e.g., the modified generalized assignment problem, Darby-Dowman et al. 1997; the template design problem, Proll and Smith 1998; the progressive party problem, Smith et al. 1995). There has been effort to characterise the properties of different problems by their effect on the performance of CP and MILP approaches (Darby-Dowman et al. 1997; Jain and Grossmann 2001). The key result of that work is that MILP is very efficient when the relaxation is tight and the models have a structure that can be effectively exploited, while CP seems to work better for highly constrained discrete optimization problems in which the expressiveness of MILP is a major limitation. Our results confirm that the best model of choice depends on the characteristics of the instances rather than of the structure exposed at the problem level. Our experiments suggest that when the audit time is small the relaxation is tight, hence MILP performs well; when the audit time gets longer the problem becomes more constrained, and CP seems to scale up much better than MILP.

Table 5 Simplex iterations performed by the MILP approach (MILP-SI) and Nodes explored by the CP approach (CP-Nod). Italic figures in the table highlight the approach that produced the best run time. Underlined figures are infeasible instances. +59652116 means that the search has been stopped before infeasibility could be proved, after 30000 sec (8,3 hour)

$\bar{L}$	1500			2500			3500			T	N
a	500	750	1000	500	750	1000	500	750	1000		
$\rho = 0.15$											
MILP-SI	737530	534238	444055	1115993	1102108	952999	1548908	2182999	1346656	2	20
CP-Nod	*70246*	*75317*	*80386*	*636450*	*764147*	*911016*	*636450*	*764147*	*911134*	2	20
MILP-SI	115666	107912	227602	191912	133002	137485	179505	150434	204205	3	20
CP-Nod	*511*	*511*	*511*	*48025*	*51003*	*54077*	*48025*	*51003*	*54077*	3	20
MILP-SI	23915	176870	122716	1745261	1056964	1285394	1842453	1580697	1565614	4	30
CP-Nod	*90*	*90*	*90*	*56456*	*56526*	*56620*	*517734*	*538358*	*559576*	4	30
MILP-SI	*p.*	*p.*	*p.*	1889563	4362836	2365598	889056	484259	1493608	5	30
CP-Nod	26	26	26	*6017*	*6017*	*6017*	*104956*	*107308*	*109671*	5	30
MILP-SI	*p.*	*p.*	*p.*	3068784	2184342	2979721	1440912	2280890	3245663	6	40
CP-Nod	17	17	17	*1358*	*1358*	*1358*	*94461*	*94465*	*94467*	6	40
$\rho = 0.3$											
MILP-SI	358921	455248	425701	902365	1946679	965218	1477861	2142193	1347372	2	20
CP-Nod	*38004*	*39394*	*40679*	*636450*	764147	911134	*636450*	*764147*	*911134*	2	20
MILP-SI	105962	80243	210615	117851	174027	129448	181685	165411	177510	3	20
CP-Nod	*511*	*511*	*511*	*48025*	*51003*	*54077*	*48025*	*51003*	*54077*	3	20
MILP-SI	62566	29487	30184	492571	1145508	1771707	1470525	1691075	1694070	4	30
CP-Nod	*90*	*90*	*90*	*33660*	*33622*	*33622*	*511950*	*532192*	*553003*	4	30
MILP-SI	*p.*	*p.*	*p.*	1317658	951745	1052849	1226532	444214	1482423	5	30
CP-Nod	26	26	26	*4885*	*4885*	*4885*	*97399*	*99383*	*101332*	5	30
MILP-SI	*p.*	*p.*	*p.*	2267185	2616084	884031	34210413	46569011	+59652116	6	40
CP-Nod	17	17	17	*1198*	*1198*	*1198*	*35354*	*35354*	*35354*	6	40

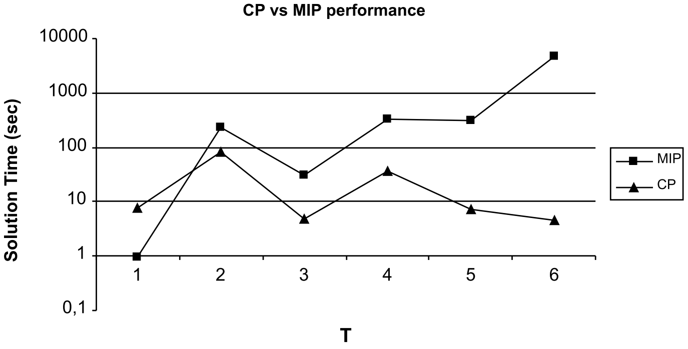


Fig. 7 Comparison of the average solution time for the CP approach and for the MILP approach as a function of the audit time T

6 Conclusions

This paper addresses the stochastic combinatorial optimization problem of scheduling internal audit activities. In Tarim et al. (2008) a related problem has been tackled by means of a similar MILP approach, but the authors assume that audit activities are instantaneous (conducting an audit does not take any time). Our work is more general and more realistic since we consider non-instantaneous audit activities, and we schedule the audit team in such a way as to prevent concurrent auditing.

We proposed a stochastic programming formulation and we developed two alternative certainty equivalent approaches to solve this model: an MILP model and a CP model. Our computational experience shows that MILP proved to be effective when the time required to perform an audit is short ($T \leq 1$). In contrast, our CP approach proved to be very effective when the audit time T is greater than one period. The CP approach proved extremely effective both in proving optimality and detecting infeasibility for most of the instances considered. For instances where the audit time T is greater than one, the CP approach proves optimality or detect infeasibility in a time that is typically orders-of-magnitude less than the one required by the MILP approach. Nevertheless the performance of the CP approach when the audit time is short still remains acceptable and close to that achieved by the MILP approach.

Finally, we believe that introducing additional complexity in the model may constitute an interesting direction for future research. For instance, heterogeneous audit teams may be considered, which may take different times to perform audits; alternatively, random audit durations—rather than a fixed and deterministic duration T—may be incorporated in the stochastic programming model.

Appendix

In this Appendix a complete list of the notation adopted in the paper is given.

N:	(constant) number of time periods in the planning horizon
i, j, t:	(index) a time period
M:	(constant) number of audit units
m:	(index) an audit unit
l_t^m:	(random variable) a normally distributed random variable representing losses that accrue in audit unit m during period t
$g_{l_t^m}(l_t^m)$:	(function) probability density function of l_t^m
$E[.]$:	(function) expected value operator
μ_t^m:	(constant) expected value of l_t^m, sometimes expressed as $\mathrm{E}[l_t^m]$
σ_t^m:	(constant) standard deviation of l_t^m
ρ:	(constant) coefficient of variation of l_t^m, $\rho = \sigma_t^m / \mu_t^m$

T: (constant) number of time periods required by the audit team to complete an audit

a: (constant) the fixed cost incurred each time an audit is conducted

h: (constant) the loss discount factor measuring the opportunity cost associated with a given loss level

$\bar{L}$: (constant) a threshold indicating the maximum allowed loss level in each period

α: (constant) the probability of not exceeding the loss threshold $\bar{L}$

L_t^m: (decision variable) the loss level in audit unit m at the beginning of period t

$E[TC]$: (objective function) the sum of expected audit costs and discounted period losses that are expected to accrue in the absence of auditing

K_t^m: (decision variable) a variable that takes the value of 1 if an internal audit (lasting T periods) is started for audit unit m in period t, otherwise 0

$\mathfrak{M}$: (constant) some very large number

A_r^m: (index) time period in which the rth audit is performed on unit m

$G_{l_t^m}(x)$: (function) $G_{l_t^m}(x) = \int_{-\infty}^{x} g_{l_t^m}(\tau)\mathrm{d}\tau$ is the cumulative distribution function of l_t^m

$G_{l_t^m}^{-1}(\alpha)$: (function) the inverse cumulative distribution function (or α-quantile) of l_t^m

$G_N(.)$: (function) the standard normal distribution function

$G_N^{-1}(.)$: (function) the inverse of $G_N(.)$

$P_{t,j}^m$: (decision variable) a binary variable that takes a value of 1 if the most recent audit prior to period t was in period j and zero elsewhere for a given audit unit m

C: (constant) maximum number of audit teams that the firm can assign to conduct audits in any given time period

$\Phi[m,t,j]$: (constant table) a 3-dimensional table whose elements are defined as $\Phi[m,t,j] = \bar{L} - G^{-1}_{l_j^m + l_{j+1}^m + \cdots + l_t^m}(\alpha), t = 1, \ldots, N, j = 1, \ldots, N$

References

Apt K (2003) Principles of constraint programming. Cambridge University Press, New York

Birge JR, Louveaux F (1997) Introduction to stochastic programming. Springer, New York

Bookbinder JH, Tan JY (1988) Strategies for the probabilistic lot-sizing problem with service-level constraints. Manag Sci 34:1096–1108

Boritz JE, Broca DS (1986) Scheduling internal audit activities. Auditing A J Pract Theory 6:1–19

Brailsford SC, Potts CN, Smith BM (1999) Constraint satisfaction problems: Algorithms and applications. Eur J Oper Res 119:557–581

Charnes A, Cooper WW (1959) Chance-constrained programming. Manag Sci 6(1):73–79

Chvtal V (1983) Linear programming. Freeman, New York

Colmerauer A (1985) Prolog in 10 figures. Commun ACM 28(12):1296–1310

Dantzig GB (1951) Maximization of a linear function of variables subject to linear inequalities. In: Activity analysis of production and allocation. Wiley, New York (Chap XXI)

Dantzig GB (1963) Linear programming and extensions. Princeton University Press, Princeton

Darby-Dowman K, Little J, Mitra G, Zaffalon M (1997) Constraint logic programming and integer programming approaches and their collaboration in solving an assignment scheduling problem. Constraints 1(3):245–264

Hanus M (1998) Programming with constraints: An introduction by K Marriott and PJ Stuckey. MIT Press, Cambridge. See also J Funct Program 11(2):253–262 (2001)

Hughes J (1977) Optimal internal audit timing. Account Rev 52:56–68

Ilog (2007a) CPLEX 9.0 Users Manual. ILOG Inc, Incline Village, NV

Ilog (2007b) OPL Studio 3.7 Users Manual. ILOG Inc, Incline Village, NV

Jain V, Grossmann IE (2001) Algorithms for hybrid milp/cp models for a class of optimization problems. INFORMS J Comput 13:258–276

Jeffreys H (1961) Theory of probability. Clarendon, Oxford

Knechel WR, Benson HP (1991) The optimization approach for scheduling internal audits of division. Decis Sci 22:391–405

Lustig IJ, Puget JF (2001) Program does not equal program: Constraint programming and its relationship to mathematical programming. Interfaces 31:29–53

Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley-Interscience, New York

Proll L, Smith B (1998) Integer linear programming and constraint logic programming approaches to a template design problem. INFORMS J Comput 10:265–275

Regin J-C (1994) A filtering algorithm for constraints of difference in CSPS. In: Proceedings of the national conference on artificial intelligence (AAAI-94), Seattle, WA, USA, pp 362–367

Regin J-C (2003) Global constraints and filtering algorithms. In: Milano M (ed) Constraints and integer programming combined. Kluwer, Dordrecht

Schrijver A (1986) Theory of linear and integer programming. Wiley, New York

Smith B, Brailsford S, Hubbard P, Williams HP (1995) The progressive party problem: integer linear programming and constraint programming compared. In: CP95: Proceedings of the 1st international conference on principles and practice of constraint programming, Marseilles

Tarim SA, Smith B (2008) Constraint programming for computing non-stationary (R,S) inventory policies. Eur J Oper Res 189(3):1004–1021

Tarim SA, Tayfur O, Karacaer S (2008) A mip model for scheduling multi-location internal audit activities. Technical report, Hacettepe University, Dept of Management

Tsang EPK (1993) Foundations of constraint satisfaction. Academic Press, London/San Diego

Van Hentenryck P, Carillon JP (1988) Generality vs. specificity: an experience with AI and or techniques. In: Proceedings of the national conference on artificial intelligence (AAAI-88)

Van Hentenryck P (1989) Constraint satisfaction in logic programming. MIT Press, Cambridge

Van Hentenryck P (1999) The OPL optimization programming language. MIT Press, Cambridge

Ventsel ES (1979) Theory of probability. Nauka, Moscow (in Russian)

Williams HP (1994) Model building in mathematical programming, 3rd edn. Wiley, New York

Wilson D, Ranson R (1971) Internal audit scheduling—a mathematical model. The Internal Auditor, No. July–August, pp. 42–50

Contents lists available at ScienceDirect

Int. J. Production Economics

journal homepage: www.elsevier.com/locate/ijpe

A state space augmentation algorithm for the replenishment cycle inventory policy

Roberto Rossi [a,*], S. Armagan Tarim [b], Brahim Hnich [c], Steven Prestwich [d]

[a] Logistics, Decision and Information Sciences, Wageningen UR, Hollandseweg 1, 6706 KN Wageningen, The Netherlands
[b] Department of Management, Hacettepe University, Turkey
[c] Faculty of Computer Science, Izmir University of Economics, Izmir, Turkey
[d] Cork Constraint Computation Centre, University College, Cork, Ireland

ARTICLE INFO

Keywords:
Inventory control
Non-stationary stochastic demand
Replenishment cycle policy
Dynamic programming
State space relaxation
State space filtering
State space augmentation

ABSTRACT

In this work we propose an efficient dynamic programming approach for computing replenishment cycle policy parameters under non-stationary stochastic demand and service level constraints. The replenishment cycle policy is a popular inventory control policy typically employed for dampening planning instability. The approach proposed in this work achieves a significant computational efficiency and it can solve any relevant size instance in trivial time. Our method exploits the well known concept of state space relaxation. A filtering procedure and an augmenting procedure for the state space graph are proposed. Starting from a relaxed state space graph our method tries to remove provably suboptimal arcs and states (filtering) and then it tries to efficiently build up (augmenting) a reduced state space graph representing the original problem. Our experimental results show that the filtering procedure and the augmenting procedure often generate a small filtered state space graph, which can be easily processed using dynamic programming in order to produce a solution for the original problem.

1. Introduction

Q1 Inventory theory provides methods for managing and controlling inventories under different constraints and environments. An interesting class of production/inventory control problems is the one that considers the single-location, single-product case under non-stationary stochastic demand and service level constraints. Such a problem has been widely studied because of its key role in practice.

Different inventory control policies can be adopted for the above mentioned problem. For a discussion of inventory control policies, see Silver et al. (1998). One of the possible policies that can be adopted is the replenishment cycle policy, (*R*,*S*). A detailed discussion on the characteristics of (*R*,*S*) can be found in de Kok (1991). In this policy an order is placed every *R* periods to raise the inventory level to the order-up-to-level *S*. This provides an effective means of dampening planning instability (deviations in planned orders, also known as *nervousness* (de Kok and Inderfurth, 1997; Heisig, 2002) and coping with demand uncertainty. As pointed out by Silver et al. (1998, pp. 236–237), (*R*,*S*) is particularly appealing when items are ordered from the same supplier or require resource sharing. In these cases all items in a coordinated group can be given the same replenishment period. Periodic review also allows a reasonable prediction of the level of the workload on the staff involved, and is particularly suitable for advanced planning environments and risk management (Tang, 2006).

Under the non-stationary demand assumption the replenishment cycle policy takes the form (R^n,S^n) where R^n denotes the length of the *n*th replenishment cycle and S^n the respective order-up-to-level. In this policy, the actual order quantity for replenishment cycle *n* is determined after the demand in previous periods has been observed. The order quantity is computed as the amount of stock required to raise the closing inventory level of replenishment cycle $n-1$ up to level S^n. In order to provide a solution for our problem under the (R^n,S^n) policy we must populate both the sets $\{R^n|n=1,\ldots,M\}$ and $\{S^n|n=\{1,\ldots,M\}$, where *M* denotes the number of replenishment cycles scheduled over a finite planning horizon of *N* periods.

The problem of populating these sets has been solved to optimality only recently, due to the complexity involved in the modeling of uncertainty and of the policy-of-response. As Silver points out, computing replenishment cycle policy parameters under non-stationary stochastic demand is a computationally hard task (Silver, 1978). Early works in this area adopted heuristic strategies such as those proposed by Silver (1978), Askin (1981), and Bookbinder and Tan (1988). Under some mild assumptions, the first complete solution method for this problem was introduced by Tarim and Kingsman (2004), who proposed a

[*] Corresponding author. Tel.: +31 317 482321; fax: +31 317 485646.
E-mail addresses: roberto.rossi@wur.nl, robros@gmail.com (R. Rossi), armtar@yahoo.com (S.A. Tarim), brahim.hnich@ieu.edu.tr (B. Hnich), s.prestwich@4c.ucc.ie (S. Prestwich).

0925-5273/$ - see front matter © 2010 Published by Elsevier B.V.
doi:10.1016/j.ijpe.2010.04.017

Please cite this article as: Rossi, R., et al., A state space augmentation algorithm for the replenishment cycle inventory policy. International Journal of Production Economics (2010), doi:10.1016/j.ijpe.2010.04.017

deterministic equivalent mixed integer programming (MIP) formulation for computing (R^n,S^n) policy parameters. Tempelmeier (2007) extended Tarim and Kingsman's MIP formulation in order to consider different service level measures, such as the "fill rate". Nevertheless, empirical results showed that Tarim and Kingsman's model is unable to solve large instances. Tarim and Smith (2008) therefore introduced a more compact and efficient constraint programming formulation of the same problem that showed a significant computational improvement over the MIP formulation. The constraint programming formulation has been further enhanced by means of dedicated cost-based filtering algorithms developed by Tarim et al. (2009). A stochastic constraint programming (Tarim et al., 2006) approach for computing optimal (R^n,S^n) policy parameters is proposed in Rossi et al. (2008). In this work the authors drop the mild assumptions originally introduced by Tarim and Kingsman and compute optimal (R^n,S^n) policy parameters. Of course, there is a price to pay for dropping Tarim and Kingsman's assumptions, in fact this latter approach is less efficient than the one in Tarim and Smith (2008). Finally, Pujawan and Silver (2008) recently proposed a novel and effective heuristic approach.

In this paper, we build on Tarim and Kingsman's modeling assumptions and we develop a state-of-the-art algorithm for computing optimal (R^n,S^n) policy parameters. Two existing techniques—dynamic programming and state space relaxation—are combined in order to obtain an effective approach for computing (R^n, S^n) policy parameters. Dynamic programming (DP) is an optimization procedure that solves optimization problems by decomposing them into a nested family of subproblems. DP is based on the *principle of optimality* (Bellman, 1957; Dreyfus and Law, 1989) and it has been applied to solve a wide variety of combinatorial optimization problems, as well as optimal control problems. State space relaxation (SSR) considers the DP formulation of a combinatorial optimization problem, and modifies this formulation to obtain a different—and possibly more compact—DP formulation whose optimal solution is a lower bound for the original problem. Christofides et al. (1981) proposed that SSR has been successfully applied to constrained variants of routing problems (see, e.g. Mingozzi et al., 1997; Focacci and Milano, 2001). Roughly speaking, SSR maps the original state space graph to a new state space graph having a smaller number of vertices, and whose shortest path represents a lower bound for the cost of the shortest path in the original state space graph.

In this work, we enhance these known approaches with a novel strategy: we introduce a filtering procedure for the state space graph and an augmenting procedure that is able to build a reduced state space graph for the original problem starting from a filtered state space graph for the relaxed problem. The concept of state space augmentation (Boland et al., 2006) is known in the operations research literature. A dual approach to state space augmentation also exists and is known as decremental SSR (Righini and Salani, 2008). Nevertheless, the idea of filtering a relaxed state space graph is, to the best of our knowledge, a novel contribution. Our experimental results prove the effectiveness of such an approach for computing optimal (R^n, S^n) policy parameters.

The paper is structured as follows. In Section 2 we introduce the problem definition and the modeling assumptions adopted in this work. In Section 3 we describe a DP reformulation for Tarim and Kingsman's model. An SSR for this reformulation is presented in Section 4. A procedure for filtering the relaxed state space graph is presented in Section 5. An augmenting procedure for the relaxed state space graph is described in Section 6. An example that demonstrates the algorithm proposed is given in Section 7. Our computational experience and a comparison with the state-of-the-art approaches for computing replenishment cycle policy parameters are discussed in Section 8. In Section 9 we draw conclusions.

2. Problem definition and modeling assumptions

The single-location, single-product production/inventory control problem under non-stationary stochastic demand and service level constraints are formulated in this paper by using the following inputs and assumptions.

We consider a planning horizon of N periods and a demand d_t for each period $t\in\{1,\ldots,N\}$, which is a non-negative random variable with known probability density function and expected value $\tilde{d}_t$. We assume that the demand occurs instantaneously at the beginning of each time period. The demand is non-stationary, that is it can vary from period to period, demands in different periods are assumed to be independent. Demands occurring when the system is out of stock are assumed to be back-ordered and satisfied as soon as the next replenishment order arrives. The sell-back of excess stock is not allowed, if the actual stock exceeds the order-up-to-level for a given review, this excess stock is carried forward and it is not returned to the supply source. However, as in Bookbinder and Tan (1988), Tarim and Kingsman (2004), Tarim and Smith (2008), and Tempelmeier (2007) such occurrences are regarded as rare events and accordingly the cost of carrying this excess stock and its effect on the service levels of subsequent periods are ignored.

A fixed delivery cost a is incurred for each order. A linear holding cost h is incurred for each unit of product carried in stock from one period to the next. Our aim is to find a replenishment plan that minimizes the expected total cost, which is composed of ordering costs and holding costs, over the N-period planning horizon, satisfying the service level constraints. As a service level constraint we require that, with a probability of at least a given value α, at the end of each period the net inventory will be non-negative. As pointed out in Tempelmeier (2007), since period demands are random, the net inventory may become negative. However, the number of stock-outs is restricted by the service level constraints enforced. While computing holding costs, we will assume, as in Bookbinder and Tan (1988), Tarim and Kingsman (2004), Tarim and Smith (2008), and Tempelmeier (2007), that the service level is set large enough to ensure that the net inventory will be a good approximation of the inventory on hand.

3. A DP formulation for the deterministic equivalent problem

We hereby introduce a deterministic equivalent DP formulation for computing optimal (R^n,S^n) policy parameters.

Definition. A replenishment cycle, $T(i,j)$, is the time span between two consecutive orders/productions occurring in periods i and $j+1$, $j\geq i$.

Definition. The cycle buffer stock, $b(i,j)$, denotes the minimum expected buffer stock level required to satisfy the required non-stock-out probability during $T(i,j)$.

We define $b(i,j)$, $i=1,\ldots,N$, $j=\text{i},\ldots,N$, as

$$b(i,j)=G^{-1}_{d_i+d_{i+1}+\cdots+d_j}(\alpha)-\sum_{k=i}^{j}\tilde{d}_k, \qquad (1)$$

where $G_{d_i+d_{i+1}+\cdots+d_j}$ is the cumulative probability distribution function of $d_i+d_{i+1}+\cdots+d_j$. It is assumed that G is strictly increasing, hence G^{-1} is uniquely defined. It should be noted that it is possible to consider different service level measures—for instance the "fill rate"—simply by introducing a different definition for the cycle buffer stock (see also Tempelmeier, 2007).

Since N is the number of periods in our planning horizon, this will also be the number of steps in the system. A state s_k at step k represents a possible expected closing-inventory-level, $\tilde{I}_k$, at the end of period k. The decision x_k to be taken at step k is to place an

Please cite this article as: Rossi, R., et al., A state space augmentation algorithm for the replenishment cycle inventory policy. International Journal of Production Economics (2010), doi:10.1016/j.ijpe.2010.04.017

order in such a period or not; if an order is placed, x_k also indicates how many subsequent periods this order should cover.

Let $X_k(s_{k-1})$ denote the set of possible feasible decisions x_k at period k, when the expected closing inventory level at period $k-1$ is s_{k-1}. This set may comprise: the decision of not placing an order ($x_k=0$), the decision of covering one period with the order placed ($x_k=k$), the decision of covering two periods with the order placed ($x_k=k+1$),…, and the decision of covering $N-k+1$ periods with the order placed ($x_k=N$). In other words, if $x_k=0$, no order is placed in period k; if $k \le x_k \le N$, x_k schedules a replenishment cycle $T(k,x_k)$. However, one should note that the decision $x_k=0$ is only allowed if

$$b(v,k) \le s_{k-1} - \tilde{d}_k,$$

where $v = \max\{t | 1 \le t \le k, x_t > 0\}$. Intuitively, we can decide not to place an order at the beginning of period k if and only if we have sufficient stocks to guarantee the required service level at least for this period.

Given a pair $\langle s_k, x_k \rangle$ the cost function $p_k(s_k,x_k)$ is clearly given by the sum of the fixed ordering cost a, which is charged if x_k states that an order should be placed, and of the inventory holding cost at the end of the period, which is equal to the expected closing-inventory-level s_k, multiplied by the per-unit holding cost h. A per-unit purchase/production cost may also be considered, this will be briefly discussed in Section 6.

The state transition function, $s_k = t_k(s_{k-1}, x_k)$, is as follows:

$$s_k = \begin{cases} s_{k-1} - \tilde{d}_k & \text{if } x_k = 0, \\ \max(s_{k-1} - \tilde{d}_k, & b(k,x_k) + \sum_{i=k+1}^{x_k} \tilde{d}_i) \quad \text{if } k \le x_k \le N. \end{cases} \tag{2}$$

S_k, the set of feasible expected closing-inventory-levels at the end of period k, is obtained recursively from the state transition functions $t_1, t_2, \ldots, t_k$, by assuming $s_0 = 0$ and, therefore, that an order should be always placed at period 1 in order to cover one or more following periods. In other words, $X_1(s_0)$ does not include the option of not placing an order.

The objective function is

$$z = \min\left\{ \sum_{k=1}^{N} p_k(s_k, x_k) \right\}. \tag{3}$$

To determine the value of z, DP solves a set of problems $i = 1, \ldots, N$, each corresponding to a system composed by i steps and characterized by the state s_i at the end of step i. The recursive formulation of the cost function at step i is

$$f_i(s_i) = \min_{x_i \in X_i(s_{i-1})} \{ f_{i-1}(s_{i-1}) + p_i(s_i, x_i) \}, \tag{4}$$

where $s_i = t_i(s_{i-1}, x_i)$. In addition, we have the following boundary condition:

$$f_1(s_1) = \min_{x_1 \in X_1(s_0)} \{ p_1(s_1, x_1) \}, \tag{5}$$

where $s_1 = t_1(s_0, x_1)$.

Clearly, a mere recursive approach would immediately generate a very large state space graph that would certainly be unmanageable. For this reason, in the following sections we will propose an effective strategy for limiting the size of the state space graph.

4. A state space relaxation for the deterministic equivalent problem

Intuitively, the first way of keeping the state space graph compact consists in employing a relaxation that clusters states together. More specifically, in order to do so we will employ a relaxation proposed by Tarim (1996).

The core observation in Tarim's relaxation lies in the fact that, if we relax the constraint which enforces non-negative order quantities—i.e. we give the opportunity to sell back items in excess to the supplier at the beginning of a given replenishment cycle—then the model proposed can be reduced to a shortest path problem on a state space graph having a number of nodes and arcs polynomial in the number N of periods.

In this relaxation, since the inventory conservation constraint is relaxed between replenishment cycles, each replenishment cycle can be treated independently and its expected total cost can be computed *a priori*. In fact, given a replenishment cycle $T(i,j)$, we recall that $b(i,j)$, as defined above, denotes the minimum expected buffer stock level required to satisfy a given service level constraint during the replenishment cycle $T(i,j)$. It directly follows that $\tilde{I}_j = b(i,j)$. Furthermore for each period $t \in \{i, \ldots, j-1\}$ the expected closing-inventory-level is $\tilde{I}_t = b(i,j) + \sum_{k=t+1}^{j} \tilde{d}_k$. Since all the $\tilde{I}_t$ for $t \in \{i, \ldots, j\}$ are known it is easy to compute the expected total cost for $T(i,j)$, which is by definition the sum of the ordering cost and of the holding cost components, $a + h\sum_{t=i}^{j} \tilde{I}_t$.

We now have a set $\mathcal{S}$ of $N(N+1)/2$ possible different replenishment cycles and their respective costs. Our new problem is to find an optimal set $\mathcal{S}^* \subset \mathcal{S}$ of consecutive disjoint replenishment cycles that covers our planning horizon at the minimum cost.

We shall now show that the optimal solution to this relaxation is given by the shortest path in a *state space graph* from a given initial node to a final node (boundary condition) where each arc represents a replenishment cycle cost. If N is the number of periods in the planning horizon of the original problem, we introduce $N+1$ nodes. Since we assume that an order is always placed at period 1, we take node 1, which represents the beginning of the planning horizon, as the initial node. Node $N+1$ represents the end of the planning horizon.

Definition. The cycle cost, $c(i,j)$, denotes the expected cost of the optimal policy for $T(i,j)$. It can be expressed as

$$c(i,j) = a + h(j-i+1)b(i,j) + h\sum_{t=i}^{j} (t-i)\tilde{d}_t. \tag{6}$$

The cycle cost is the sum of two components. A fixed ordering cost a, that is charged at the beginning of the cycle when an order is placed, and a variable holding cost h_t charged at the end of each time period within the replenishment cycle and proportional to the amount of stock held in inventory.

For each possible replenishment cycle $T(i,j-1)$ such that $i,j \in \{1, \ldots, N+1\}$ and $i<j$, we introduce an arc (i,j) with associated cost $c(i,j-1)$ (Fig. 1). Since we are dealing with a one-way temporal feasibility problem (Wagner and Whitin, 1958), when $i \ge j$, we introduce no arc. As shown in Tarim (1996), the cost of the shortest path from node 1 to node $N+1$ in the given graph is a valid lower bound for the original problem, as it is a solution of the relaxed problem. A shortest path can be efficiently found by applying Dijkstra's algorithm that runs in $O(n^2)$ time, where n is the number of nodes in the graph. Details on efficient implementations of Dijkstra's algorithm can be found in Sedgewick (1988).

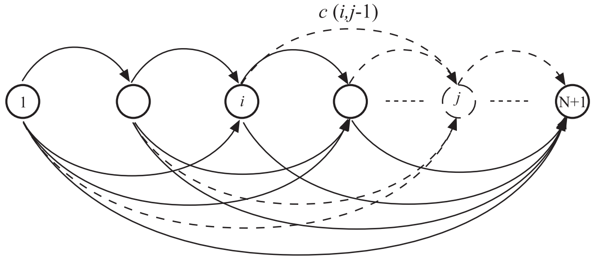


Fig. 1. Shortest path problem graph.

Please cite this article as: Rossi, R., et al., A state space augmentation algorithm for the replenishment cycle inventory policy. International Journal of Production Economics (2010), doi:10.1016/j.ijpe.2010.04.017

It is easy to map the optimal solution for the relaxed problem, that is the set of arcs participating to the shortest path, to an assignment for the original problem by noting that each arc (i,j) represents a replenishment cycle $T(i,j-1)$. The set of arcs in the optimal path, therefore, uniquely identifies a set of disjoint replenishment cycles, that is a replenishment plan. Furthermore for each period $t \in \{i,\dots,j-1\}$ in cycle $T(i,j-1)$ we already showed that all the expected closing-inventory-levels $\tilde{I}_t$, $t \in \{i,\dots,j-1\}$, are known. This produces a complete assignment for decision variables in our model. The feasibility of an assignment with respect to the original problem can be checked by verifying that it satisfies every relaxed constraint, that is no negative expected order quantity is scheduled.

5. A filtering procedure for the relaxed state space graph

In the previous section we presented a known relaxation for the deterministic equivalent formulation of the (R^n,S^n) policy. In this relaxation we solve a shortest path problem over a given graph in order to find a lower bound for the cost of the optimal solution for the original problem.

We now aim to reduce *a priori* as much as possible the number of arcs in the graph we defined in the previous section. To do so we exploit a reduction procedure based on an upper bound for replenishment cycle lengths that was originally presented by Tarim and Smith (2008).

Definition. Cycle opening inventory level, $R(i,j)$, denotes the minimum opening inventory level in period i to meet demand until period $j+1$ and $R(i,j)=b(i,j)+\sum_{t=i}^{j}\tilde{d}_t$.

```
input : a relaxed and filtered state space graph$RSG(S,T)$
output : an augmented state space graph $ASG(S',T')$
begin
    $i'=N+1$;
    $ASG(S',T') \leftarrow RSG(S,T)$;
    for each node $i=1,\dots,N$ in $S'$ do
        for each arc $(p,i)$ in $T'$ do
            let $b^*$ be the buffer stock associate to $(p,i)$
            for each arc $(i,j)$ in $T'$ do
                If $b^* > R(i,j-1)$ then
                    $i'=i'+1$;
                    create a new node $i'$ in $S'$;
                    introduce arc $(p,i')$ in $T'$ with associated buffer stock $b^*$;
                    remove arc $(p,i)$ from $T'$;
                    let $t>i$ be the minimum index for which
                    $b^* \le R(i,t-1) \le R(i,t) \le \cdots \le R(i,N)$;
                    introduce arc $(i',t)$ $T'$ with buffer stock $b(i,t-1)$;
                    for each arc $(i,k)$, $k=t+1,\dots,N+1$ in $T'$ do
                        introduce arc $(i',k)$ in $T'$ with associated buffer stock
                        $b(i,k-1)$;
                    let $t-1>i$ be the maximum index for which
                    $b^* > \cdots \ge R(i,t-2)$;
                    introduce arc $(i',t-1)$ in $T'$ with associated buffer stock
                    $b^*-\sum_{k=i}^{t-1}\tilde{d}_k$
end
```

Let us assume now that period i is a replenishment period. It is not generally possible, prior to obtaining the optimal solution to an instance of the problem, to determine the length of the optimum replenishment cycle for a particular replenishment period; however, an upper bound on the length can be determined using Proposition 1.

Proposition 1 (Tarim and Smith, 2008). *If* $\forall k \in \{i,\dots,j-1\}$, $(c(i,k)+c(k+1,j)>c(i,j)) \vee (b(i,k)>R(k+1,j))$ *and* $\exists k \in \{i,\dots,j\}$ $(c(i,k)+c(k+1,j+1) \le c(i,j+1)) \wedge (b(i,k) \le R(k+1,j+1))$ *then for period* i *the optimum length replenishment cycle is* $T(i,p)^*$ *where* $i \le p \le j$, *and* j *indicates an upper bound.*

Since we have an upper bound j for the length of an optimum replenishment cycle starting at period i, we can remove from our graph every arc (i,t), where $t>j+1$.

6. An augmenting procedure for the relaxed state space graph

Once the shortest path problem on the graph constructed as shown above is solved, we can easily verify if every relaxed constraint is satisfied by the solution found, that is, if no expected negative replenishment quantity is scheduled in the optimal replenishment plan. In this case, the solution found is feasible and optimal for the original problem. If, on the other hand, the solution is not feasible for the original model and it schedules expected negative replenishment quantities, we can augment the graph with additional nodes and arcs in such a way that the shortest path on the augmented graph is guaranteed to provide a feasible and optimal solution for the original problem. In what follows we shall show how to augment the graph and efficiently compute an optimal solution for the original problem.

Algorithm 1. Augmenting procedure

For convenience, instead of associating a cost $c(i,j-1)$ to each arc (i,j) in the graph, we will now associate the respective cycle buffer stock, $b(i,j-1)$, as defined above. From the definitions

Please cite this article as: Rossi, R., et al., A state space augmentation algorithm for the replenishment cycle inventory policy. International Journal of Production Economics (2010), doi:10.1016/j.ijpe.2010.04.017

given, it is easy to see that, once this expected buffer stock level is fixed, also the cost $c(i,j-1)$ is uniquely defined.

Let $RSG(S,T)$ be a relaxed state space graph built according to the discussion in Section 4 and filtered according to the discussion in Section 5. Let S denotes the set of nodes and T the set of arcs in the graph. The pseudo-code for the proposed augmenting procedure is presented in Algorithm 1. The procedure eventually generates an augmented state space graph $ASG(S',T')$, where S' is the set of nodes and T' is the set of arcs in the augmented graph.

Algorithm 1 initially creates a copy $ASG(S',T')$ of $RSG(S,T)$ (line 3). Then it considers each node in S' in order (line 4), starting from node 1 up to node N. Note that node $N+1$ has no outbound arcs, so we do not have to consider it. The process is repeated for each node i, therefore we will only describe the steps performed on a single node.

We consider every inbound arc at node i (line 5) and we operate in the following fashion. Given an inbound arc (p,i) with associated buffer stock b^* (line 6), for each outbound arc (i,j) in the graph (line 7) we check that $b^* \le R(i,j-1)$. If this condition is satisfied for every outbound arc, then we preserve the inbound arc (p,i) at node i with the associated buffer stock b^* (Fig. 2). Otherwise, if $b^* > R(i,j-1)$ (line 8), for a subsequent pair of replenishment cycles a negative order quantity is scheduled. In order to resolve this infeasibility we perform the following transformation (lines 10…18). We introduce a new node i' in the graph. We remove arc (p,i) and we introduce a new arc (p,i') with associated buffer stock b^* (Fig. 3). Then we connect this new node in the following way.

Let $t>i$ be the minimum index for which $b^* \le R(i,t-1) \le R(i,t) \le \cdots \le R(i,N)$. We introduce arc (i',t) with buffer stock $b(i,t-1)$. Then, for each arc $(i,t+1),\ldots,(i,N+1)$ *in the graph*, we also introduce $(i',t+1),\ldots,(i',N+1)$ with buffer stock, respectively, $b(i,t),\ldots,b(i,N)$. It should be noted that some of the arcs $(i,t+1),\ldots,(i,N+1)$ may have been removed by the filtering described in Section 5.

Let $t-1>i$ be the maximum index for which $b^* > \cdots \ge R(i,t-2)$. We introduce arc $(i',t-1)$ with buffer stock $b^* - \sum_{k=i}^{t-1} \tilde{d}_k$. Obviously arcs $(i',t-2),(i',t-3),\ldots$ are suboptimal and should not be introduced, since the inventory carried on from the previous replenishment cycle is enough to cover subsequent periods up to $t-1$.

Note that, when the process is iterated on subsequent nodes $i+1,\ldots,N$, the new inbound arcs that may have been introduced must also be considered among all the possible ones for a given node.

By starting from node 1 and by iterating this process for each node i, $1 \le i \le N$, we obtain an augmented graph. By construction the cost of the shortest path in this augmented graph is the optimal solution cost for our original problem since every possible negative order quantity scenario has been considered and replaced with the respective feasible possible courses of action. Nevertheless, as a consequence of the original filtering performed on the relaxed graph, the augmented graph will typically feature a very limited number of node and arcs. This will be shown in the following sections.

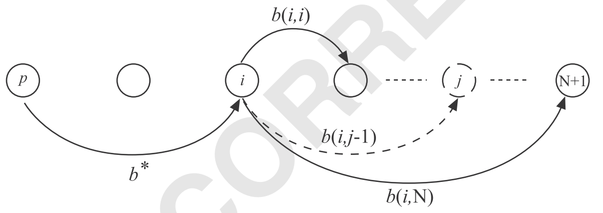


Fig. 2. Feasible node point.

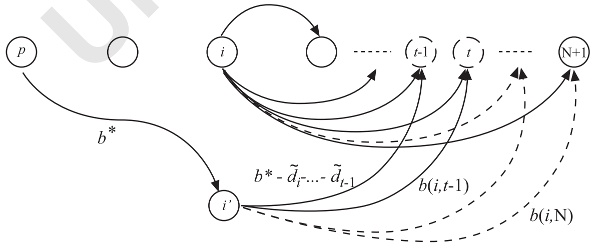


Fig. 3. Infeasible node point.

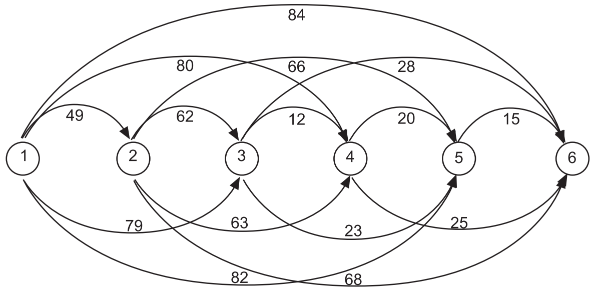


Fig. 4. Connection matrix with expected buffer stock levels.

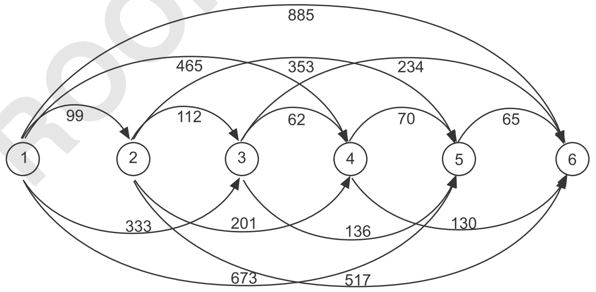


Fig. 5. Connection matrix with expected cycle costs.

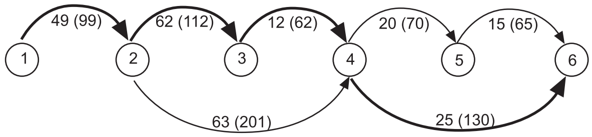


Fig. 6. Filtered connection matrix. Expected buffer stock levels and expected cycle costs (in parentheses) are shown for each arc. The shortest path is highlighted.

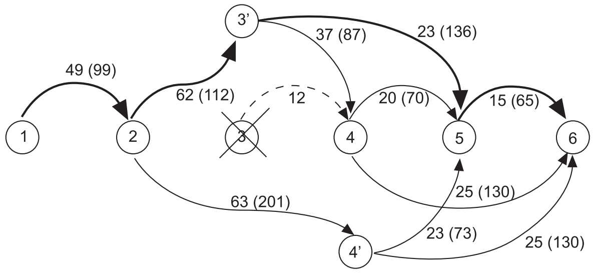


Fig. 7. Augmented connection matrix. Expected buffer stock levels and expected cycle costs (in parentheses) are shown for each arc. The shortest path is highlighted. Node 3 (and obviously arc (3,4)) has been removed from the network since the augmenting procedure removed all its inbound arcs.

Please cite this article as: Rossi, R., et al., A state space augmentation algorithm for the replenishment cycle inventory policy. International Journal of Production Economics (2010), doi:10.1016/j.ijpe.2010.04.017

Table 1
Test set P5.

a	N	$\alpha=0.95$				$\alpha=0.99$			
		CP		DP		CP		DP	
		Nod	Sec	Graph	Sec	Nod	Sec	Graph	Sec
		$\sigma_t/\tilde{d}_t=1/3$							
25	50	857	45	$\langle 64;76\rangle$	0.33	2474	170	$\langle 67;82\rangle$	0.30
	75	41 386	5400	$\langle 102;125\rangle$	0.38	180 000*	20 000*	$\langle 106;133\rangle$	0.37
50	50	441	17	$\langle 66;84\rangle$	0.34	1242	170	$\langle 69;88\rangle$	0.33
	75	23 805	2400	$\langle 104;133\rangle$	0.37	180 000*	20 000*	$\langle 108;139\rangle$	0.37
100	50			$\langle 51;87\rangle$	0.21	104	5	$\langle 73;109\rangle$	0.34
	75			$\langle 76;134\rangle$	0.24	329	30	$\langle 113;167\rangle$	0.21
200	50			$\langle 51;139\rangle$	0.23			$\langle 51;131\rangle$	0.24
	75			$\langle 76;212\rangle$	0.26			$\langle 76;200\rangle$	0.28
		$\sigma_t/\tilde{d}_t=1/6$							
25	50	22	1	$\langle 58;65\rangle$	0.34	325	17	$\langle 61;70\rangle$	0.33
	75	245	35	$\langle 90;103\rangle$	0.2	10 118	970	$\langle 98;116\rangle$	0.20
50	50			$\langle 51;69\rangle$	0.20	70	3	$\langle 63;80\rangle$	0.42
	75			$\langle 76;106\rangle$	0.14	155	14	$\langle 100;126\rangle$	0.37
100	50			$\langle 51;103\rangle$	0.13			$\langle 51;94\rangle$	0.12
	75			$\langle 76;161\rangle$	0.26			$\langle 76;145\rangle$	0.25
200	50			$\langle 51;158\rangle$	0.22			$\langle 51;184\rangle$	0.15
	75			$\langle 76;242\rangle$	0.27			$\langle 76;226\rangle$	0.26

A figure marked with * means that the instance could not be solved in the given limit of 20 000 s (5.55 h).

Before demonstrating our method on a simple numerical example, it is worth mentioning the following. Our model, for the sake of simplicity, assumes a zero unit purchase/production cost, also in line with the model in Tarim and Smith (2008). Nevertheless, the extension of our algorithm to the case of a non-zero unit production/purchasing cost is quite straightforward. In fact, as shown in Tarim and Kingsman (2004, p. 113), the total unit variable cost can be reduced to a function of the expected closing-inventory-level of the very last period N. Therefore, considering such an effect in our algorithm is easy, since it only requires us to modify, in the graph connection matrix, the costs that appear in the rightmost column, which represents every possible replenishment cycle that ends in period N.

7. An example

We shall consider here a simple example in detail, to show how in practice it is possible to apply the procedure described.

A single problem over a 5-period planning horizon is considered and the expected values for period demand are [100, 125, 25, 40, 30]. We assume an initial null inventory level and a normally distributed demand for every period with a coefficient of variation $\sigma_t/\tilde{d}_t=0.3$ for each $t\in\{1,\dots,N\}$, where σ_t denotes the standard deviation of the demand in period t. We consider an ordering cost value $a=50$ and a holding cost $h=1$ per unit per period. The non-stock-out probability in each period is set to $\alpha=0.95$.

Firstly we build the connection matrix for the relaxed problem as described in Section 4. In Fig. 4 we show the connection matrix with the respective expected buffer stock level $b(i,j-1)$ associated with each arc (i,j).[1] In Fig. 5 instead with each arc (i,j) we associate the respective expected cycle cost $c(i,j-1)$. It should be noted that the two representations are equivalent, since the expected cycle cost can be easily computed once the expected buffer stock level for a given cycle is fixed. In Fig. 6 the connection matrix is filtered according to the procedure presented in Section 5. Expected buffer stock levels and expected cycle costs (in parentheses) are indicated for each arc that has not been removed by the filtering. The shortest path in this reduced network has a cost of 403. The order periods and the order quantities are, respectively, [1, 2, 3, 4] and [149, 138, -25, 83]. This assignment is infeasible for the non-relaxed problem since the expected order quantity in period 3 is -25, therefore its cost is a lower bound for the optimal solution cost of our original problem. According to the procedure described in Section 6 we augment the filtered graph and we obtain the new graph in Fig. 7. The shortest path in this augmented network has a cost of 412 and represents the optimal solution cost of our original problem. The replenishment periods in this optimal solution can be obtained from the indexes of the nodes in the shortest path. The respective order quantities can also be easily obtained from the expected buffer stock levels associated with each arc in the shortest path. The order periods and the order quantities are therefore, respectively, [1, 2, 3, 5] and [149, 138, 26, 22].

[1] For clarity, in order to keep the graphical presentation as compact as possible, the expected buffer stock levels have been rounded to the nearest integer value.

Please cite this article as: Rossi, R., et al., A state space augmentation algorithm for the replenishment cycle inventory policy. International Journal of Production Economics (2010), doi:10.1016/j.ijpe.2010.04.017

8. Experimental results

We compared the results obtained with our approach with the results obtained with the state-of-the-art constraint programming (CP) approach in Tarim et al. (2009), based on the set of instances originally proposed in Berry (1972). All the experiments presented in this section were performed on an Intel(R) Centrino(TM) CPU 1.50 GHz with 500 Mb RAM. As in Tarim et al. (2009), the demand in each period is assumed to be normally distributed and we also assume that its coefficient of variation remains sufficiently low (i.e. less or equal to 1/3) to ensure that negative demand values can be ignored. We recall that in Tarim et al. (2009) period demands are generated from seasonal data with no trend: $\tilde{d}_t = 50[1+\sin(\pi t/6)]$. In addition to the "no trend" case (P1) three others are also considered:

(P2) positive trend case, $\tilde{d}_t = 50[1+\sin(\pi t/6)]+t$,
(P3) negative trend case, $\tilde{d}_t = 50[1+\sin(\pi t/6)]+(52-t)$,
(P4) life-cycle trend case, $\tilde{d}_t = 50[1+\sin(\pi t/6)]+\min(t,52-t)$.

Tests are performed using four different ordering cost values $a \in \{40,80,160,320\}$ and two different $\sigma_t/\tilde{d}_t \in \{1/3,1/6\}$. The planning horizon length takes even values in the range [24, 50] when the ordering cost is 40 or 80 and [14, 24] when the ordering cost is 160 or 320. The holding cost used in these tests is $h=1$ per unit per period. Tests consider two different service levels $\alpha = 0.95$ ($z_{\alpha=0.95} = 1.645$) and $\alpha = 0.99$ ($z_{\alpha=0.99} = 2.326$).

For almost all these instances our DP approach is either better—in terms of run time—than the CP approach or equivalent, with some exceptions for the smallest instances. When the number of periods considered in the planning horizon grows, our DP approach clearly scales better than the CP approach. The maximum improvement observed reaches a factor of 24. Nevertheless, for this set of instances the CP approach remains competitive and achieves reasonable run times of a few seconds also for the largest instances.

In what follows, we aim to highlight the limits of the CP approach and we want to show that our DP approach remains very effective even for those instances for which the CP approach performs poorly. In order to do so, we consider the following set of instances (test set P5). The expected period demands, $\tilde{d}_t$, are generated as uniformly distributed random numbers in [0, 100]. Empirically, in fact, we observed that generating random sequences of demands rather than seasonal patterns or trends makes the problem harder to solve. Again we consider four different ordering cost values $a \in \{25,50,100,200\}$ and two different $\sigma_t/\tilde{d}_t \in \{1/3,1/6\}$. The planning horizon length takes the values {50, 75}. The holding cost used in these tests is $h=1$ per unit per period. Again we consider two different service levels $\alpha = 0.95$ ($z_{\alpha=0.95} = 1.645$) and $\alpha = 0.99$ ($z_{\alpha=0.99} = 2.326$). Table 1 compares the CP and the DP approach for this new set of instances. In our test results, the heading "CP" refers to the state-of-the-art CP approach in Tarim et al. (2009), while "DP" refers to our novel DP approach. For the CP approach we report the number of nodes explored (Nod) and the run time in seconds (Sec); for our DP approach we report the size of the state space graph generated (Graph) and the run time in seconds (Sec). The size of the state space graph is described as a pair $\langle N;A\rangle$, where N is the number of nodes and A is the number of arcs. When a field is empty in the table, this means that the CP approach and the DP approach are equivalent, since for that particular instance the CP approach was able to prove optimality at the root node in polynomial time using the DP relaxation originally proposed in Tarim (1996).

Table 2
Test set P6.

a	N	$\sigma_t/\tilde{d}_t = 1/3$				$\sigma_t/\tilde{d}_t = 1/6$			
		$\alpha = 0.95$		$\alpha = 0.99$		$\alpha = 0.95$		$\alpha = 0.99$	
		Graph	Sec	Graph	Sec	Graph	Sec	Graph	Sec
2500	75	$\langle 102;125\rangle$	0.44	$\langle 106;133\rangle$	0.38	$\langle 90;103\rangle$	0.25	$\langle 98;116\rangle$	0.26
	110	$\langle 153;194\rangle$	0.36	$\langle 160;208\rangle$	0.36	$\langle 134;157\rangle$	0.31	$\langle 144;175\rangle$	0.28
	145	$\langle 197;246\rangle$	0.41	$\langle 208;266\rangle$	0.41	$\langle 174;202\rangle$	0.41	$\langle 185;222\rangle$	0.41
	180	$\langle 242;301\rangle$	0.48	$\langle 255;327\rangle$	0.49	$\langle 211;245\rangle$	0.46	$\langle 225;268\rangle$	0.47
	215	$\langle 284;350\rangle$	0.96	$\langle 297;376\rangle$	0.93	$\langle 248;285\rangle$	0.93	$\langle 263;310\rangle$	0.88
	250	$\langle 329;407\rangle$	0.79	$\langle 346;439\rangle$	0.79	$\langle 287;330\rangle$	0.75	$\langle 305;361\rangle$	0.77
5000	75	$\langle 104;133\rangle$	0.20	$\langle 108;139\rangle$	0.48	$\langle 76;107\rangle$	0.13	$\langle 100;126\rangle$	0.22
	110	$\langle 155;202\rangle$	0.57	$\langle 162;214\rangle$	0.29	$\langle 111;152\rangle$	0.18	$\langle 146;185\rangle$	0.28
	145	$\langle 199;255\rangle$	0.36	$\langle 210;272\rangle$	0.40	$\langle 146;198\rangle$	0.21	$\langle 187;234\rangle$	0.62
	180	$\langle 245;317\rangle$	0.46	$\langle 258;337\rangle$	0.55	$\langle 181;250\rangle$	0.30	$\langle 230;295\rangle$	0.49
	215	$\langle 287;366\rangle$	0.85	$\langle 300;386\rangle$	0.94	$\langle 216;296\rangle$	0.75	$\langle 268;338\rangle$	0.49
	250	$\langle 332;426\rangle$	0.76	$\langle 349;450\rangle$	0.74	$\langle 251;347\rangle$	0.61	$\langle 312;399\rangle$	0.95
10 000	75	$\langle 76;134\rangle$	0.13	$\langle 116;174\rangle$	0.34	$\langle 76;162\rangle$	0.15	$\langle 76;147\rangle$	0.04
	110	$\langle 170;270\rangle$	0.29	$\langle 171;256\rangle$	0.19	$\langle 111;230\rangle$	0.22	$\langle 111;211\rangle$	0.10
	145	$\langle 216;344\rangle$	0.62	$\langle 224;332\rangle$	0.35	$\langle 146;300\rangle$	0.26	$\langle 146;280\rangle$	0.19
	180	$\langle 271;439\rangle$	0.51	$\langle 279;422\rangle$	0.69	$\langle 181;377\rangle$	0.34	$\langle 181;354\rangle$	0.25
	215	$\langle 317;517\rangle$	0.85	$\langle 324;493\rangle$	0.61	$\langle 216;448\rangle$	0.70	$\langle 216;423\rangle$	0.58
	250	$\langle 365;593\rangle$	1.02	$\langle 375;569\rangle$	0.99	$\langle 251;512\rangle$	0.62	$\langle 251;485\rangle$	0.67
20 000	75	$\langle 76;212\rangle$	0.14	$\langle 76;201\rangle$	0.15	$\langle 76;242\rangle$	0.15	$\langle 76;228\rangle$	0.17
	110	$\langle 111;306\rangle$	0.21	$\langle 111;292\rangle$	0.13	$\langle 111;352\rangle$	0.17	$\langle 111;332\rangle$	0.16
	145	$\langle 146;408\rangle$	0.27	$\langle 146;388\rangle$	0.24	$\langle 146;460\rangle$	0.39	$\langle 146;437\rangle$	0.26
	180	$\langle 181;514\rangle$	0.35	$\langle 181;485\rangle$	0.49	$\langle 181;575\rangle$	0.38	$\langle 181;546\rangle$	0.54
	215	$\langle 216;617\rangle$	0.67	$\langle 216;585\rangle$	0.44	$\langle 216;685\rangle$	0.50	$\langle 216;652\rangle$	0.47
	250	$\langle 251;713\rangle$	0.68	$\langle 251;675\rangle$	0.60	$\langle 251;787\rangle$	0.80	$\langle 251;750\rangle$	0.79

Please cite this article as: Rossi, R., et al., A state space augmentation algorithm for the replenishment cycle inventory policy. International Journal of Production Economics (2010), doi:10.1016/j.ijpe.2010.04.017

It is immediately clear that for low a/h ratios (that is for the lowest ordering costs considered), the CP approach has to explore a large search space and requires a long time to prove optimality, while our DP approach still generates small state space graphs and achieves fast runtimes. As the ratio a/h increases, the CP approach performs better and, for some instance, it is equivalent to our DP approach.

In the last set of instances considered (test set P6) we aim to show that our approach is effective even when the planning horizon is significantly longer, and that the computation is not affected by the magnitude of the demands considered. The planning horizon length now ranges up to 250 periods, in order to show that our approach scales well in the number of periods. The expected period demands $\tilde{d}_t$ are generated as uniformly distributed random numbers in [0, 10 000], in order to show that large values for the expected demands do not affect the scalability of our approach. Once more, we consider four different ordering cost values $a \in \{2500, 5000, 10\,000, 20\,000\}$ and two different $\sigma_t/\tilde{d}_t \in \{1/3, 1/6\}$. The planning horizon length takes the following values {75, 110, 145, 180, 215, 250}. The holding cost used in these tests is $h=1$ per unit per period. Also in this case, we consider two different service levels $\alpha=0.95$ ($z_{\alpha=0.95}=1.645$) and $\alpha=0.99$ ($z_{\alpha=0.99}=2.326$). The computational results in Table 2 present that the graphs generated are still extremely compact and that the run times are mostly under one second even if a long planning horizon and large demands are considered.

9. Conclusions

We proposed a novel DP approach for computing (R^n,S^n) policy parameters. Our experimental results show that our approach, based on the described filtering algorithm for the state space graph and on the state space graph augmenting procedure, can solve instances over planning horizons comprising hundreds of periods. State space relaxation and state space augmentation are two known strategies in operations research, nevertheless, the idea of filtering a relaxed state space graph is, to the best of our knowledge, a novel contribution. As our computational experience shows, our DP reformulation performs significantly better than the original MIP approach proposed by Tarim and Kingsman and it also beats the state-of-the-art reformulations proposed by Tarim and Smith and Tarim et al. Furthermore our results are not affected by the magnitude of the demand considered in each period.

Acknowledgements

B. Hnich and A. Tarim are supported by the Scientific and Technological Research Council of Turkey under Grant no. SOBAG-108K027. A. Tarim is supported by Hacettepe University-BAB. We wish to thank the anonymous reviewers who significantly contributed to improve the quality of this manuscript.

References

Askin, R.G., 1981. A procedure for production lot sizing with probabilistic dynamic demand. AIIE Transactions 13 (2), 132–137.

Bellman, R.E., 1957. Dynamic Programming. Princeton University Press, Princeton, NJ.

Berry, W.L., 1972. Lot sizing procedures for requirements planning systems: a framework for analysis. Production and Inventory Management Journal 13 (2), 19–34.

Boland, N., Dethridge, J., Dumitrescu, I., 2006. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. Operations Research Letters 34 (1), 58–68.

Bookbinder, J.H., Tan, J.Y., 1988. Strategies for the probabilistic lot-sizing problem with service-level constraints. Management Science 34 (9), 1096–1108.

Christofides, N., Mingozzi, A., Toth, P., 1981. State space relaxation procedures for the computation of bounds to routing problems. Networks 11 (2), 145–164.

de Kok, A.G., 1991. Basics of inventory management: part 2. The (R,S)-model. Research Memorandum, FEW 521, Department of Economics, Tilburg University, Tilburg, The Netherlands.

de Kok, A.G., Inderfurth, K., 1997. Nervousness in inventory management: comparison of basic control rules. European Journal of Operational Research 103 (1), 55–82.

Dreyfus, S.B., Law, A.M., 1989. The Art and Theory of Dynamic Programming. Academic Press, New York.

Focacci, F., Milano, M., 2001. Connections and integrations of dynamic programming and constraint programming. In: Proceedings of the International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems CP-AI-OR 2001.

Heisig, G., 2002. Planning Stability in Material Requirements Planning Systems. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Mingozzi, A., Bianco, L., Ricciardelli, S., 1997. Dynamic programming strategies for travelling salesan problem with time windows and precedence constraints. Operations Research 45 (3), 365–377.

Pujawan, I.N., Silver, E.A., 2008. Augmenting the lot sizing order quantity when demand is probabilistic. European Journal of Operational Research 127 (3), 705–722.

Righini, G., Salani, M., 2008. New dynamic programming algorithms for the resource constrained elementary shortest path problem. Networks 51 (3), 155–170.

Rossi, R., Tarim, S.A., Hnich, B., Prestwich, S., 2008. A global chance-constraint for stochastic inventory systems under service level constraints. Constraints 13 (4), 490–517.

Sedgewick, R., 1988. Algorithms, second ed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Silver, E.A., 1978. Inventory control under a probabilistic time-varying demand pattern. AIIE Transactions 10 (4), 371–379.

Silver, E.A., Pyke, D.F., Peterson, R., 1998. Inventory Management and Production Planning and Scheduling. John-Wiley and Sons, New York.

Tang, C.S., 2006. Perpectives in supply chain risk management. International Journal of Production Economics 103 (2), 451–488.

Tarim, S.A., 1996. Dynamic lotsizing models for stochastic demand in single and multi-echelon inventory systems. Ph.D. Thesis, Lancaster University.

Tarim, S.A., Hnich, B., Rossi, R., Prestwich, S., 2009. Cost-based filtering techniques for stochastic inventory control under service level constraints. Constraints 14 (2), 137–176.

Tarim, S.A., Kingsman, B.G., 2004. The stochastic dynamic production/inventory lot-sizing problem with service-level constraints. International Journal of Production Economics 88 (1), 105–119.

Tarim, S.A., Manandhar, S., Walsh, T., 2006. Stochastic constraint programming: a scenario-based approach. Constraints 11 (1), 53–80.

Tarim, S.A., Smith, B., 2008. Constraint programming for computing non-stationary (*R*,*S*) inventory policies. European Journal of Operational Research 189 (3), 1004–1021.

Tempelmeier, H., 2007. On the stochastic uncapacitated dynamic single-item lotsizing problem with service level constraints. European Journal of Operational Research 181 (1), 184–194.

Wagner, H.M., Whitin, T.M., 1958. Dynamic version of the economic lot size model. Management Science 5 (3), 89–96.

Please cite this article as: Rossi, R., et al., A state space augmentation algorithm for the replenishment cycle inventory policy. International Journal of Production Economics (2010), doi:10.1016/j.ijpe.2010.04.017

An investigation of setup stability in non-stationary stochastic inventory systems

Onur A. Kilic[*]

Department of Operations, University of Groningen, The Netherlands

S. Armagan Tarim

Department of Management, Hacettepe University, Turkey

Abstract

In stochastic inventory planning systems unfolding uncertainties in demand lead to revision of earlier production/order plans. This results in different order release decisions in successive planning cycles, which in turn leads to an instability in inventory plan, or so-called system nervousness. In this paper, we provide the grounds for measuring system nervousness in non-stationary demand environments, and gauge the setup stability and cost performance of (R, S) and (s, S) inventory control policies. We conduct a numerical study using a test set including a variety of costs, demand patterns, and coefficients of demand variation. The results reveal that both the stability and the performance of inventory policies are affected by the demand pattern as well as the cost parameters. Furthermore, our analysis point out that (R, S) policy has the potential to replace the cost-optimal (s, S) policy, especially for systems characterized by a low degree of flexibility to setup changes.

Keywords: Inventory policies, Stability, Non-stationary demand

1 Introduction

In inventory planning systems, inventory plans experience changes in response to realized demand. In practice, when this is the case, the plan is regenerated for the rest of the planning horizon. This results in different order decisions causing planning instability or so-called *system nervousness* (Vollmann et al., 1988).

Inventory management involves application of various inventory control policies. These policies are extensively investigated in terms of their cost performance. However, in systems with a low degree of flexibility, the cost of implementing revisions in setup decisions may overcome the advantage of using the cost-efficient technique. In this context inventory control rules show different levels of instability. Thus system nervousness, as a performance criterion, can be of high importance in assessing inventory control rules. Omitting the planning instability can turn out to be a serious problem because it gives rise to a considerable amount of alteration efforts (Heisig, 2001).

The (s, S) control policy has been shown to be cost-optimal under very relaxed assumptions in both stationary and non-stationary cases (see Scarf, 1959; Iglehart, 1963). Heisig (1998, 2001), and de Kok and Inderfurth (1997) have questioned the performance of (s, S) policy with respect to the nervousness criterion

[*] Corresponding author: o.a.kilic@rug.nl

in the stationary case. Their research reveals the trade-off between cost effectiveness and nervousness and show that (s,S) policy exhibits the worst stability performance among a number of policies considered. Different strategies for dealing with the problem of nervousness are examined by Blackburn et al. (1986). They suggest an effective strategy based on freezing certain orders so they cannot be changed. In this regard, (R,S) policy, in which the timing of future orders are fixed, provides a means of dampening the setup instability. Silver et al. (1998) points out that (R,S) policy, which provides a rhythmic rather than a random replenishment pattern, is usually appealing from a practitioners point of view.

One major difficulty in the continuing development of inventory theory is to incorporate more realistic assumptions about demand into inventory models. In many stable environments it is an adequate approximation to treat period demands as identically distributed random variables. However, many times the demand pattern is heavily seasonal, or has a significant trend, especially in industrial settings with business cycles. On the other hand, as product life cycles get shorter, the randomness and unpredictability of demand processes become even greater. The essence of such situations can only be captured by means of finite horizon non-stationary inventory models.

Literature provides guidelines about the stability performance of inventory policies in stationary systems. However, those may not be directly generalized to non-stationary systems due to differences in their natures. In stationary systems policy parameters are also stationary, and therefore, the measure of stability of the whole system can be determined by means of observing any two consecutive planning cycles. However, in non-stationary systems, policy parameters are determined in connection with each and every period through the horizon, and consequently, stability is a function of the demand pattern. To the best of our knowledge, no work has been done on the measures of nervousness in non-stationary systems. In this paper, we aim to fill in this gap by investigating the system nervousness under non-stationary stochastic demand. Our contribution is two-fold. First we propose an exact method for measuring system nervousness in non-stationary demand environments. Secondly we gauge the setup stability of (R,S) and (s,S) type inventory control policies and demonstrate that (R,S) policy has the potential to replace the cost-optimal (s,S) policy, especially for systems characterized by a low degree of flexibility to setup changes.

The remainder of this paper is organized as follows. Section 2 investigates the related literature. Section 3 gives the notation and the definitions regarding the addressed inventory system. Section 4 provides the grounds for computing setup instability in non-stationary environments and proposes a method for the computation thereof. Section 5 gives the models and methods for computing (s,S) and (R,S) policies. Section 6 presents the computational experiments, tests the proposed inventory control rules with respect to setup instability, and clarifies and discusses our findings. Section 7 presents the conclusions and some likely extensions of the study.

2 Background

The most important issue in the investigation of the performance of inventory control rules with respect to stability is the definition of nervousness. Previous research specify two classes of general nervousness types: short/long term and setup/quantity oriented nervousness. The former involves rating order decision instability between the first two periods for the short term and between all consecutive periods for the long term. The latter distinguishes between considering adjustments on pure setup actions (cancellation of a planned order or placement an unplanned order) and adjustments on the setup quantities. It is noted by Inderfurth (1994) and Heisig (2001) that setup oriented system nervousness is considered as the most serious in practice. In this paper we address *long term, setup oriented* nervousness or so-called *setup instability*.

Generally it is hard to express instability in terms of cost. For this reason, rather than integrating nervousness into pure cost-based inventory models, like for instance in Kropp et al. (1983) and Kropp and Carlson (1984), we define stability as an independent attribute of an inventory control system and refer to the measures used in Jensen (1996) and Heisig (2001).

Early studies in nervousness involves a wide set of simulation studies, where the impact of different planning parameters on system nervousness are investigated in deterministic demand environments (see e.g. Blackburn et al., 1986, 1987; Sridharan et al., 1988; Minfie and Davis, 1990; Kadipasalioglu and Sridharan, 1997). A systematic development of nervousness measures in stochastic environments is given in Inderfurth (1994), de Kok and Inderfurth (1997), Heisig (1998) and Heisig (2001). Analytical results are presented in Inderfurth (1994) where the performance of (s, S) and (s, nQ) policies with respect to *short-term, setup-oriented* nervousness is analyzed. In de Kok and Inderfurth (1997), the *short-term, setup-oriented* as well as the *short-term, quantity-oriented* nervousness are examined for (s, S), (s, nQ) and (R, S) policies. *Long-term, setup-oriented* nervousness performance of (s, S) and (s, nQ) policies are analyzed in Heisig (1998) and Heisig (2001). Above mentioned studies define instability as the ratio of expected deviations over the maximum deviations that can take place in the worst case. These studies show that there exists a trade-off between cost effectiveness and stability performance, and therefore the (s, S) policy, which is optimal in terms of cost performance, exhibits the worst stability performance among all other policies considered.

Previous studies investigate nervousness in the rolling horizon framework. In this framework, although setups and associated quantities are computed over the entire planning horizon, only the first period decision is implemented, and then the schedule is rolled forward to the next period with new demand appended to the horizon. Within this approach, there are two sources of nervousness: demand uncertainty and rolling horizon planning (Kadipasalioglu and Sridharan, 1997). Demand uncertainty implies that actual demand may differ from the forecast, and therefore, leads to a revision of setups as necessary. Rolling horizon planning, however, may cause planned orders to change because of the new information obtained about future demands. In this study we adapt a re-planning approach rather than a rolling horizon framework. In a re-planning approach, new periods are not appended to the fixed length planning horizon and setup plans are generated only for the remaining periods. One positive side effect of this approach is that the inventory system is no longer exposed to the instability due to the rolling horizon planning. Hence it gives us the opportunity to investigate the sole effect of demand uncertainty on stability performance.

3 Notation and definitions

In this paper we address a multi-period stochastic inventory problem which is characterized by a finite horizon comprising N periods. The demand, d_t in period t is considered as a random variable with known probability density function, $g_t(d_t)$, and is assumed to occur instantaneously at the beginning of each period. The mean rate of demand may vary from period to period. Demands in different time periods are assumed independent. A fixed holding cost h is incurred on any unit carried in inventory over from one period to the next. Demands occurring when the system is out of stock are assumed to be backordered, and satisfied immediately the next replenishment order arrives. A fixed shortage cost p is incurred for each unit of demand backordered. A fixed setup cost K is incurred each time a replenishment order is placed, whatever the size of the order. For convenience, without loss of generality, the direct item cost is assumed to be zero, and the delivery lead-time is not incorporated. We assume that inventory plans are updated in response to the realized demands throughout the planning horizon. When this is the case, no-more-optimal inventory plans are replaced by optimal ones in successive *re-planning states* (see Definition 1).

Definition 1 (Re-planning State)**.** *A re-panning state is denoted by a pair $\xi = (t, k), \xi \in \Xi$, for any given period t and inventory level k.*

Definition 2 (Re-planning Period)**.** *For a re-planing state ξ, the re-planning period $\Theta(\xi) \in \{t+1, \ldots, N | \xi = (t, k)\}$ denotes the period the next replenishment is due.*

Definition 3 (Re-planning Transition Probability)**.** *Transition probabilities for successive re-planning states ξ_1 and ξ_2 are represented by a transition matrix P with entries $p(\xi_1, \xi_2) = \Pr(X_{n+1} = \xi_2 | X_n = \xi_1)$.*

Definition 4 (State Probability)**.** *State probability $\pi(\xi)$ of state ξ is the probability of visiting state ξ starting from a given initial state.*

Definition 5 (Opening Inventory Level)**.** *For any given state $\xi = (t, k)$, the opening inventory level $I(\xi)$ equals $k + Q$ where Q is the order quantity.*

The proposed inventory system can be expressed as a stochastic process defined over the state space Ξ with transition probabilities P. One of the aims of this paper is to propose a measure of system nervousness and develop a methodology for the computation thereof for any given inventory policy. Such a measure and a methodology can be characterized over the aforementioned stochastic process. This is done in the following section.

4 Setup instability

In the inventory setting described in Section 3 the measure of *long term, setup oriented* system nervousness is given in Definition 6.

Definition 6 (Setup instability)**.** *For any given inventory system the setup instability measure is the expected number of setup changes (a new setup is scheduled or a formerly planned setup is canceled) throughout the planning horizon.*

An expression of the setup instability between two states $\xi_1 = (t_1, k_1)$ and $\xi_2 = (t_2, k_2)$ of consecutive re-planning periods (i.e. $t_2 = \Theta(\xi_1)$) is given in (1).

$$\sum_{t=t_2}^{N} |T(\xi_1, t) - T(\xi_2, t)| \tag{1}$$

where $T(\xi, h)$ is a binary variable equals to 1 if the setup plan at state $\xi = (t, k)$ calls for a replenishment for period $h, h > t$ and 0 otherwise. In (2) we use (1) to express the total instability throughout the planning horizon, $\mathcal{N}$.

$$\mathcal{N} = \sum_{\xi_1} \sum_{\xi_2} \pi(\xi_1) p(\xi_1, \xi_2) \sum_{t=t_2}^{N} |T(\xi_1, t) - T(\xi_2, t)| \tag{2}$$

In order to compute the nervousness we need to determine the state probabilities $\pi(\xi)$, and the transition probabilities $p(\xi_1, \xi_2)$. It is clear that the recursive formulation (3) gives the state probabilities.

$$\pi(\xi_2) = \sum_{\xi_1} \pi(\xi_1)p(\xi_1, \xi_2) \tag{3}$$

Transition probabilities can be written as follows:

$$p(\xi_1, \xi_2) = \Pr(I(\xi_1) - D(t_1, t_2) = k_2), \qquad \xi_j = (t_j, k_j) \text{ and } t_2 = \Theta(\xi_1) \tag{4}$$

where $D(t_1, t_2) = \sum_{t=t_1}^{t_2-1} d_t$.

5 Non-stationary (s, S) and (R, S) policies

The optimal values of decision variables, $\Theta(\xi)$, $I(\xi)$, and $T(\xi, h)$, given in the previous section, can only be determined in connection with an inventory control rule. This section provides the models for computing these decision variables under (s, S) and (R, S) type inventory policies.

5.1 (s, S) policy

In this section we present a dynamic programming approach following Bollapragada and Morton (1997) to compute optimal policy parameters for the non-stationary (s, S) problem. This approach is based on the dynamic programming formulation given below,

$$J_{t,N} = \min\{L_t(x) + \mathrm{E}(J_{t+1,N}(x - d_t)), K + L_t(S_t) + \mathrm{E}(J_{t+1,N}(S_t - d_t))\}, \quad t = 1, \ldots, N \tag{5a}$$

$$J_{N+1,N}(i) = 0. \tag{5b}$$

The state variable is the inventory on hand at the beginning of the time period, x. $J_{t,N}(i)$ denotes the expected cost of following the optimal policy from period t onwards, $L_t(i)$ represents the expected period cost function if the opening inventory level is i, and K is the fixed setup cost. The K-convexity property (Scarf, 1959) is used in obtaining a solution in each stage of the dynamic program.

Optimal parameters of (s, S) policy are independent of the current state of the system, every period is a re-planning period, and optimal policy parameters do not propose a clear setup plan. The lack of a clear setup plan demands the use of an *expected setup plan* based on the expected value problem (i.e. expected demand replaces random demand) as suggested in Heisig (1998) and Heisig (2001). In this study we adopt their approach.

5.2 (R, S) policy

In this paper we employ the non-stationary (R, S) model presented in Tarim and Kingsman (2006). They propose a certainty equivalent mixed integer programming (MIP) model to compute the optimal policy parameters. Since (R, S) is a periodic review policy in contrast to (s, S), it can be used both with and without re-planning. In the rest of this paper $(R, S)'$ will donate (R, S) policy with re-planning. (R, S) policy results in zero setup instability since setup periods are fixed in the beginning of the planning horizon. On the other hand in $(R, S)'$ policy only the foremost setup is certain and the rest of the setup plan is exposed to revisions in the following re-planning periods. It should also be noted that policy parameters of (R, S) and $(R, S)'$ policies depend on the current state of the system.

As one might expect, using $(R,S)^{'}$ yields lower expected costs compared to (R,S) in the expense of increasing instability. That is, $\mathrm{E}\{C_{(R,S)}\} \geq \mathrm{E}\{C_{(R,S)'}\}$ and $\mathcal{N}^{'}_{(R,S)} \geq \mathcal{N}_{(R,S)} = 0$, where C_Ω and $\mathcal{N}_\Omega$ are the realized cost and the setup instability under policy Ω respectively. $\mathrm{E}\{C_{(R,S)}\}$ can be obtained by solving the MIP model. However, in computing $\mathrm{E}\{C_{(R,S)'}\}$ one should consider the possible states and corresponding state probabilities as well as the expected cost of the immediate replenishment cycle, $\mathrm{E}\{C^{\mathrm{cyc}}_{(R,S)'}(\xi)\}$. The exact expected cost calculations can be carried out by means of the following expression.

$$\mathrm{E}\{C_{(R,S)'}\} = \sum_{\xi} \pi(\xi)\mathrm{E}\{C^{\mathrm{cyc}}_{(R,S)'}(\xi)\} \tag{6}$$

6 Numerical Experiments

In this section we evaluate (s,S), (R,S) and $(R,S)^{'}$ policies with respect to *long-term, setup oriented* system nervousness and expected cost, and investigate the effects of (i) setup frequency, (ii) uncertainty, and (iii) non-stationarity on those measures. To serve this purpose, we vary setup cost, coefficient of variation, and demand pattern to build a test set of 64 instances. We use 4 different setup costs $K = \{10, 250, 500, 1000\}$, 4 different coefficient of variations $\sigma/\mu = \{0.10, 0.15, 0.20, 0.25\}$, and 4 different demand patterns (static, sinusoidal, life-cycle, and erratic) adopted from Berry (1972) (see Figure 1). It is assumed that demand is normally distributed. The planning horizon is set to 20 periods with no initial inventory. Without loss of generality, holding and penalty costs are set at 1 and 10 per unit per period respectively, and unit cost is ignored. Expected costs and setup instabilities of test instances are computed under (s,S), (R,S) and $(R,S)^{'}$ policies using the models and methods described in Sections 4 and 5.

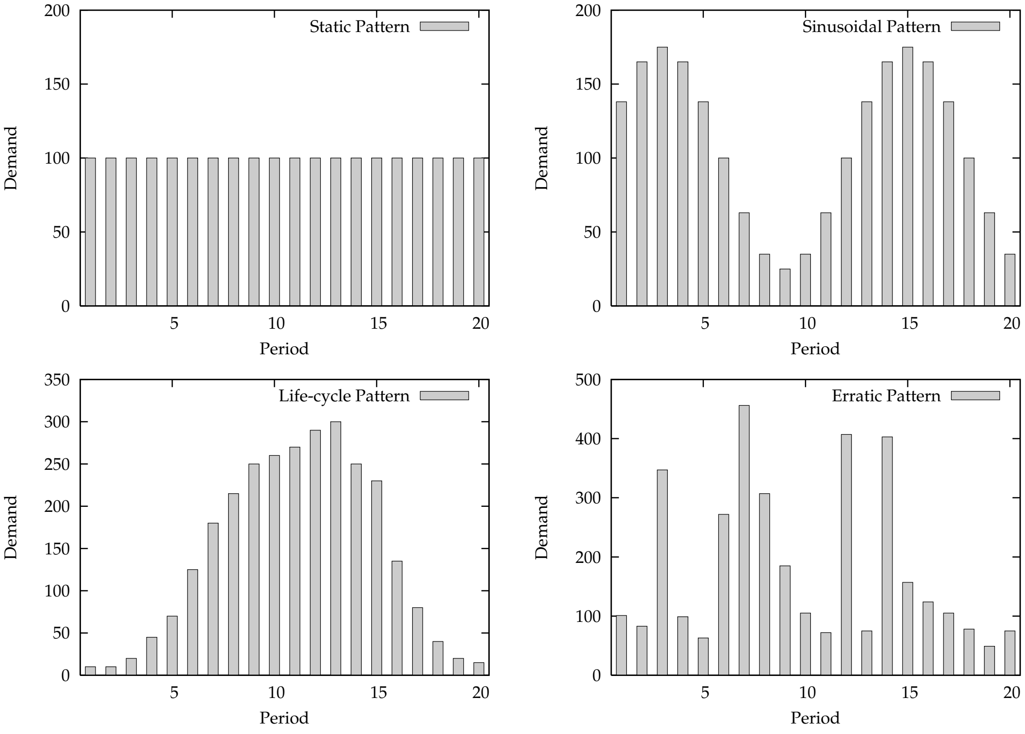


Figure 1: Demand Patterns

Since (s, S) policy is optimal in terms of cost performance, we present the cost performances of (R, S) and $(R, S)^{'}$ policies, $\Delta_{(R,S)}$ and $\Delta_{(R,S)'}$ respectively, on % differences from the optimal cost. Note that, (R, S) has zero instability by definition, and therefore, not included in the table. The results are listed in Table 1. We investigate the behavior of the cost performance of (R, S) and $(R, S)^{'}$ and stability performance of (s, S) and $(R, S)^{'}$ policies by elaborating upon the standalone effects of setup costs, coefficient of variation and demand patterns. These are summarized in Figures 1 to 3. In each figure, we plot (i) the average cost increment due to the use of non-optimal policy for (R, S) and $(R, S)^{'}$ policies, and (ii) the setup instability of (s, S) and $(R, S)^{'}$ policies against the values of one of the three parameters. The values on the y-axis are the averages of all instances having the same specified parameters. We summarize our results as follows.

Effects of setup cost: Consider the extreme cases of setup costs such that $K = 0$ and $K = \infty$. The former leads to a setup in all periods, and the latter leads to a single setup only in the first period. Both cases result in zero setup instability since regenerated plans are the same as the original one. Our observations on test results verify these comments. The cost and stability performances with respect to setup cost are given in Figure 2. It is clear that low setup costs encourages frequent setups whereas large setup costs reduce setup frequency. For extreme values of setup cost such as 10 and 1000 all policies have lower instabilities than for setup costs 250 and 500. In a similar fashion, cost penalty of using (R, S) and $(R, S)^{'}$ policies rather than (s, S) is smaller for the extreme values of setup cost.

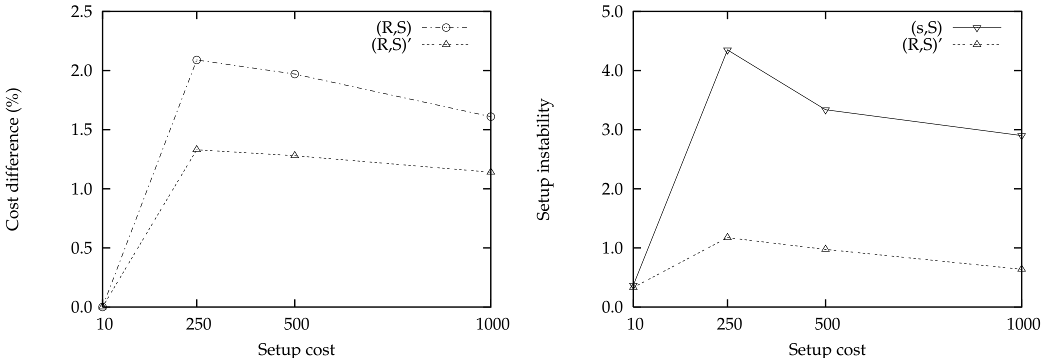


Figure 2: Effects of setup cost

Effects of demand variability: Figure 3 illustrates the average cost and instability performances with respect to demand uncertainty. We can observe that the coefficient of variation has adverse effects on both instability and expected cost of all inventory policies as one might expect. It should be noted that the effect of uncertainty is rather weak for small values of coefficient of variation and grows exponentially as coefficient of variation increases.

Effects of demand demand patterns: Figure 4 illustrates the effects of demand patterns on the cost and stability performances of inventory polices. The experiments indicate that, as the erraticity of demand increases (i.e. from static to erratic), the expected costs increase, and the instability decreases for all inventory control policies considered. Although this observation is valuable on its own sake, it is more important to investigate the effects of the demand pattern on the cost-instability trade-off. As it can be observed from Figure 4, for static, sinusoidal and life-cycle patterns, the cost difference from the optimal policy is rather low for both (R, S) and $(R, S)^{'}$. On the other hand, especially for static and sinusoidal patterns, the difference in the instability is rather large. Therefore, we can conclude that the trade-off between the cost and the stability performance of inventory policies is a function of the demand pattern.

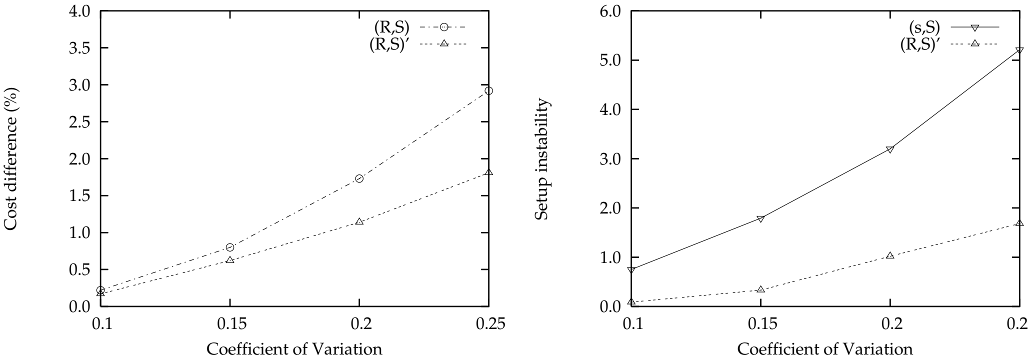


Figure 3: Effects of coefficient of variation

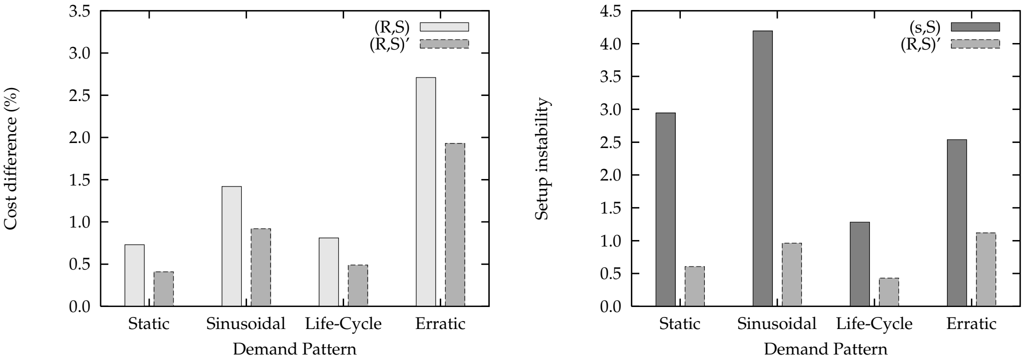


Figure 4: Effects of demand pattern

To summarize, we observe that the extent of cost and instability performances of inventory policies depend on the system parameters in our numerical study. The cost differences of (R,S) and $(R,S)'$ policies from the optimal cost of (s,S) policy are lower than 0.01% for 46 and 0.05% for 78 of 80 test instances respectively. For all 80 instances $(R,S)'$ is superior to (s,S) policy in terms of instability performance.

To provide more insight it might be useful to examine an instance in detail. Consider the instance with static demand pattern, setup cost 250 and coefficient of variation 0.25. When (s,S) policy is employed, the expected number of setup changes is 11.3. If $(R,S)'$ policy is used the expected number of setup changes decreases to 1.5, but the expected cost increases by 0.88%. On the other hand, if (R,S) policy is employed, there will be no instability and expected cost is increases by 1.52%.

7 Conclusions and extensions

In this study, we extended first the system nervousness definitions in the literature to cover non-stationary stochastic demand, and we investigated the cost and stability performances of (R,S) and cost-optimal (s,S) inventory policies in terms of system nervousness. In contrast to previous studies using a rolling horizon framework which itself is a source of nervousness, we employed a re-planning approach and we analyzed the nervousness resulting from pure demand uncertainty. With the strategy we proposed we obtain the expected cost and instability values of a given inventory problem simultaneously. We characterized the effects of cost parameters, demand variability and non-stationary demand patterns on the cost and instability performances of inventory policies. Our findings can be used to assess inventory control policies under different system settings and can be of help to choose the most efficient inventory policy depending on the flexibility of the system to setup changes and importance of cost.

We showed that (R, S) policy is clearly dominant to (s, S) policy in terms of stability performance even when the inventory plan is regenerated through the planning horizon, and the cost penalty of using (R, S) policy, rather than cost-optimal (s, S) policy, is fairly small under general settings. In addition to that, we clarified that the cost performance of (R, S) policy can be improved to some extent by employing a re-planning approach in expense of some degree of instability. These all together shows that (R, S) can be a strong alternative to (s, S) with its superior stability performance, especially for systems characterized by a low degree of flexibility to setup changes.

In this study we assumed that the setup changes in the imminent and the following periods effect the instability value identically. Therefore, our results do not account for the cases where the importance of the instabilities diminish in time. Investigating such a case, possibly by using a discount factor, is one of the interesting directions for further research. Another potential extension is to consider different inventory policies.

		Static Pattern					Sinusoidal Pattern				
K	σ/μ	$\mathrm{E}\{C_{(s,S)}\}$	$\Delta_{(R,S)}$	$\Delta_{(R,S)'}$	$\mathcal{N}_{(s,S)}$	$\mathcal{N}_{(R,S)'}$	$\mathrm{E}\{C_{(s,S)}\}$	$\Delta_{(R,S)}$	$\Delta_{(R,S)'}$	$\mathcal{N}_{(s,S)}$	$\mathcal{N}_{(R,S)'}$
10	0.10	559.90	0.00%	0.00%	0.0	0.0	592.40	0.00%	0.00%	0.0	0.0
	0.15	739.76	0.00%	0.00%	0.0	0.0	788.46	0.00%	0.00%	0.0	0.0
	0.20	919.86	0.00%	0.00%	0.0	0.0	983.32	0.00%	0.00%	0.2	0.1
	0.25	1098.77	0.00%	0.00%	0.0	0.0	1177.64	0.00%	0.00%	0.5	0.3
250	0.10	3910.71	0.00%	0.00%	0.0	0.0	3829.29	0.26%	0.25%	3.8	0.0
	0.15	4114.62	0.00%	0.00%	0.3	0.0	4021.89	1.27%	1.00%	7.5	0.6
	0.20	4304.22	0.28%	0.13%	2.4	0.5	4199.00	2.59%	1.62%	11.4	2.9
	0.25	4452.74	1.52%	0.88%	11.3	1.5	4357.01	4.13%	2.27%	13.0	4.0
500	0.10	5815.55	0.03%	0.03%	0.3	0.0	5596.82	0.33%	0.29%	2.6	0.1
	0.15	5999.73	0.44%	0.35%	1.6	0.2	5805.48	1.05%	0.77%	3.9	0.5
	0.20	6138.86	1.56%	1.06%	4.3	1.0	5993.94	2.16%	1.32%	5.0	1.9
	0.25	6233.83	3.36%	2.19%	12.4	2.0	6152.26	3.68%	2.03%	6.2	2.9
1000	0.10	8387.88	0.04%	0.03%	0.2	0.1	7993.25	0.47%	0.12%	1.8	0.4
	0.15	8549.52	0.43%	0.23%	1.9	0.5	8200.83	1.13%	0.98%	2.7	0.3
	0.20	8666.46	1.33%	0.72%	4.8	1.2	8400.25	2.17%	1.49%	3.6	0.5
	0.25	8757.73	2.67%	0.95%	7.6	2.7	8571.73	3.57%	2.54%	4.9	0.9

		Life-Cycle Pattern					Erratic Pattern				
K	σ/μ	$\mathrm{E}\{C_{(s,S)}\}$	$\Delta_{(R,S)}$	$\Delta_{(R,S)'}$	$\mathcal{N}_{(s,S)}$	$\mathcal{N}_{(R,S)'}$	$\mathrm{E}\{C_{(s,S)}\}$	$\Delta_{(R,S)}$	$\Delta_{(R,S)'}$	$\mathcal{N}_{(s,S)}$	$\mathcal{N}_{(R,S)'}$
10	0.10	706.17	0.00%	0.00%	0.0	0.0	842.74	0.00%	0.00%	0.3	0.3
	0.15	959.13	0.00%	0.00%	0.0	0.0	1175.16	0.00%	0.00%	0.7	0.7
	0.20	1211.27	0.00%	0.00%	0.3	0.2	1518.89	0.00%	0.00%	1.3	1.3
	0.25	1462.26	0.00%	0.00%	0.8	0.6	1872.41	0.00%	0.00%	1.8	1.8
250	0.10	4082.16	0.07%	0.02%	0.2	0.1	4317.02	1.16%	1.03%	0.9	0.2
	0.15	4341.71	0.46%	0.22%	0.8	0.4	4607.10	3.47%	2.95%	3.0	1.0
	0.20	4586.44	1.20%	0.67%	1.5	0.8	4859.34	6.33%	4.00%	5.1	2.5
	0.25	4822.74	2.08%	1.12%	1.7	1.0	5107.56	8.62%	5.13%	6.7	3.3
500	0.10	6026.57	0.10%	0.09%	0.1	0.0	6505.49	0.76%	0.53%	1.0	0.2
	0.15	6325.25	0.33%	0.16%	0.6	0.2	6809.34	2.41%	1.57%	2.2	0.8
	0.20	6579.55	1.27%	0.69%	2.0	0.7	7089.95	4.49%	3.08%	3.4	1.4
	0.25	6791.46	2.84%	1.50%	3.5	1.9	7357.34	6.71%	4.82%	4.3	1.8
1000	0.10	8858.61	0.06%	0.06%	0.4	0.0	9977.06	0.25%	0.25%	0.4	0.0
	0.15	9182.33	0.44%	0.39%	1.3	0.1	10349.20	1.36%	1.34%	2.2	0.0
	0.20	9465.45	1.30%	0.99%	2.3	0.3	10666.60	3.02%	2.51%	3.6	1.0
	0.25	9690.07	2.78%	1.95%	5.0	0.6	10943.00	4.71%	3.63%	3.7	1.6

Table 1: Results of 64 test instances

References

W. L. Berry. Lot sizing procedures for requirement planning systems: Framework for analysis. *Production and Inventory Management*, 13:19–34, 1972.

J. Blackburn, D. Kropp, and R. Millen. Comparison of strategies to dampen nervousness in MRP systems. *Management Science*, 32: 413–429, 1986.

J. D. Blackburn, D. H. Kropp, and R. A. Millen. Alternative approaches to schedule instability: A comparative analysis. *International Journal of Production Economics*, 25:1739–1749, 1987.

S. Bollapragada and T. Morton. A simple heuristic for computing non-stationary (s, S) policies. *Operations Research*, 47:576–584, 1997.

T. de Kok and K. Inderfurth. Nervousness in inventory management: Comparison of basic control rules. *European Journal of Operations Research*, 103:55–82, 1997.

G. Heisig. Planning stability under (s, S) inventory control rules. *OR Spektrum*, 20:215–228, 1998.

G. Heisig. Comparison of (s, S) and (s, nQ) inventory control rules with respect to planning stability. *International Journal of Production Economics*, 73:59–82, 2001.

D. L. Iglehart. Dynamic programming and stationary analysis of inventory problems. In H. E. Scarf, D. Gilford, and M. Shelley, editors, *Multistage Inventory Models and Techniques*, pages 1–31. Stanford University Press, Stanford, CA, 1963.

K. Inderfurth. Nervousness in inventory control: Analytical results. *OR Spektrum*, 16:113–123, 1994.

T. Jensen. *Planungsstabilität in der Material-Logistik*. Phd thesis, Physica-Verlag, Berlin, Heidelberg, 1996.

S. N. Kadipasalioglu and S. V. Sridharan. Measurement of instability in multi-level MRP systems. *International Journal of Production Research*, 35:713–737, 1997.

D. H. Kropp and R. C. Carlson. A lot-sizing algorithm for reducing nervousness in MRP systems. *Management Science*, 30:240–244, 1984.

D. H. Kropp, R. C. Carlson, and J. V. Jucker. Heuristic lot-sizing approaches for dealing with MRP system nervousness. *Decision Sciences*, 14(2):156–170, 1983.

J. R. Minfie and R. A. Davis. Interaction effects on MRP nervousness. *International Journal of Production Research*, 28:173–183, 1990.

H. E. Scarf. Optimality of (s, S) policies in the dynamic inventory problem. In K. Arrow, S. Karlin, and P. Suppes, editors, *Mathematical Methods in the Social Sciences*, pages 196–202. Stanford University Press, Stanford, CA, 1959.

E. A. Silver, D. A. Pyke, and R. Peterson. *Inventory Management and Production Planning and Scheduling*. John Wiley and Sons, New York, 3rd edition, 1998. ISBN 0471119474.

S. V. Sridharan, W. L. Berry, and V. Udayabhanu. Measuring master production schedule stability under rolling planning horizons. *Decision Sciences*, 19:147–166, 1988.

S. A. Tarim and B. G. Kingsman. Modelling and computing (R^n, S^n) policies for inventory systems with non-stationary stochastic demand. *European Journal of Operations Research*, 174:581–599, 2006.

T. E. Vollmann, W. L. Berry, and D. C. Whybark. *Manufacturing Planning and Control Systems*. Dow Jones-Irwin, Homewood, IL, 2nd edition, 1988.

A Survey on CP-AI-OR Hybrids for Decision Making under Uncertainty*

Brahim Hnich[1], Roberto Rossi[2], S. Armagan Tarim[3] and Steven Prestwich[4]

Faculty of Computer Science, Izmir University of Economics, Turkey[1]
brahim.hnich@ieu.edu.tr
Logistics, Decision and Information Sciences, Wageningen UR, the Netherlands[2]
roberto.rossi@wur.nl
Department of Management, Hacettepe University, Ankara, Turkey[3]
armtar@yahoo.com
Cork Constraint Computation Centre, University College Cork, Ireland[4]
s.prestwich@4c.ucc.ie

Abstract. In this survey we focus on problems of decision making under uncertainty. Firstly, we clarify the meaning of the word "uncertainty" and we describe the general structure of problems that fall into this class. Secondly, we provide a list of problems from the Constraint Programming, Artificial Intelligence and Operations Research literatures in which uncertainty plays a role. Thirdly, we survey existing modeling frameworks that provide facilities for handling uncertainty. A number of general purpose and specialized hybrid solution methods are surveyed, which deal with the problems in the list provided. These approaches are categorized into three main classes: stochastic reasoning-based, reformulation-based and sample-based. Finally, we provide a classification for other related approaches and frameworks in the literature.

1 Introduction

In this work we survey problems in which we are required to make decisions under uncertainty, and we categorize existing hybrid techniques in Constraint Programming (CP), Artificial Intelligence (AI) and Operations Research (OR) for dealing with them. The word *uncertainty* is used to characterize the existence, in these problems, of uncontrollable or "random" variables[1], which cannot be influenced by the decision maker. In addition to these random variables, problems also comprise controllable or "decision" variables, to which a value from given domains has to be assigned. More specifically, a problem classified as *deterministic* with respect to the degree of uncertainty does not include random variables, while a *stochastic* problem does.

Random variables are typically employed to model factors such as the customer demand for a certain product, the crop yield of a given piece of land during

* B. Hnich and A. Tarim are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027.

[1] Alternatively, in the literature, these variables are also denoted as "stochastic".

a year, the arrival rate of orders at a reservation center and so forth. A continuous or discrete domain of possible values that can be observed is associated with each random variable. A probabilistic measure — typically a probability distribution — over such a domain is assumed to be available in order to fully quantify the likelihood of each value (respectively, range of values in the continuous case) that appears in the domain.

The decision making process comprises one or more subsequent *decision stages*. In a decision stage, a decision is taken by the decision maker who assigns a value to each controllable variable related to this decision stage of the problem and, subsequently, the uncontrollable variables related to this stage are observed and their realized values become known to the decision maker.

It should be noted that, in this work, we do not consider situations in which the decision maker has the power to modify the probability distribution of a given random variable by using his decisions. Random variables are therefore fully uncontrollable. To clarify, this means that a situation in which the decision maker has the option of launching a marketing campaign to affect the distribution of customer demands will not be considered.

This work is structured as follows: in Section 2 we employ a motivating example and a well established OR modeling framework — Stochastic Programming — in order to illustrate key aspects associated with the process of modeling problems of decision making under uncertainty; in Section 3 we provide a list of relevant problems from the literature on hybrid approaches for decision making under uncertainty and, for each problem, we also provide a short description and a reference to the work in which such a problem has been proposed and tackled; in Section 4 we introduce frameworks, respectively from AI and from CP, that aim to model problems of decision making under uncertainty; in Section 5, we classify existing hybrid approaches for tackling problems of decision making under uncertainty into three classes: in the first class (Section 6) we identify general and special purpose approaches that perform "stochastic reasoning", in the second class (Section 7) we list approaches, general and special purpose, that use reformulation, and in the third class (Section 8) we categorize approximate techniques based on a variety of strategies employing sampling; finally, in Section 9 we point out connections with other related works, and in Section 10 we draw conclusions.

2 Decision making under uncertainty

Several interesting real world problems can be classified as "stochastic". In this section we use a variant of the Stochastic Knapsack Problem (SKP) discussed in [34] as a running example to demonstrate ideas and concepts related to stochastic problems.

Single-stage Stochastic Knapsack. A subset of k items must be chosen, given a knapsack of size c into which to fit the items. Each item i, if included in the knapsack, brings a deterministic profit r_i. The size ω_i of each item is stochastic and it is not known at the time the decision has to be made. Nevertheless, we assume that the decision maker knows the probability mass function PMF(ω_i) [31], for each $i = 1, \ldots, k$. A per unit penalty cost p has to be paid for exceeding the capacity of the knapsack. Furthermore, the probability of the plan not exceeding the capacity of the knapsack should be greater than or equal to a given threshold θ. The objective is to find the knapsack that maximizes the expected profit.

We now discuss Stochastic Programming, which is one of the most well known modeling approaches in OR for problems of decision making under uncertainty, such as the SKP. We arbitrarily chose to employ such a framework to introduce the key concepts of decision making under uncertainty. In the next sections, the following frameworks will be also introduced: Stochastic Boolean Satisfiability, Probabilistic Constraint Satisfaction Problems, Event-Driven Probabilistic Constraint Programming and Stochastic Constraint Programming.

Stochastic Programming (SP) [11, 32] is a well established technique often used for modeling problems of decision making under uncertainty. A *Stochastic Program* typically comprises a set of decision variables defined over continuous or discrete domains, a set of random variables also defined over continuous or discrete domains and, for each random variable, the respective probability density function (PDF) if continuous or probability mass function (PMF) if discrete. Decision and random variables are partitioned into decision stages. Within a decision stage, firstly, all the associated decision variables are assigned values; and secondly, all the associated random variables are observed. A set of constraints is usually enforced over decision and random variables in the model. These constraints may be *hard*, that is they should always be met regardless of the values that are observed for the random variables, or they may be *chance-constraints* [15]. Chance-constraints are constraints that should be satisfied with a probability exceeding a given threshold. If the problem is an optimization one, it may minimize/maximize an objective function defined over some expressions on possible realisations (for example, maximize the worst case performance of the stochastic system under control, or minimize the difference between the maximum and minimum values a performance measure may take to increase the robustness of a system) or some probabilistic measure — such as expectation or variance — of decision and random variables in the model.

To clarify these concepts we now introduce a Stochastic Programming model for the single-stage SKP (Fig. 1). The objective function maximizes the trade-off between the reward brought by the objects selected in the knapsack (those for which the binary decision variable X_i is set to 1) and the expected penalty paid for buying additional capacity units in those scenarios in which the available capacity c is not sufficient. Control actions that are performed after the uncertainty is resolved — such as buying additional capacity at a high cost — are called, in SP, “recourse actions”. The only chance-constraint in the model ensures that the capacity c is not exceeded with a probability of at least θ. There is only a single decision stage in the model. Decision stages define how uncertainty unfolds

Objective:

$$\max \left\{ \sum_{i=1}^{k} r_i X_i - p\mathbb{E}\left[\sum_{i=1}^{k} \omega_i X_i - c \right]^+ \right\}$$

Subject to:

$$\Pr\left\{ \sum_{i=1}^{k} \omega_i X_i \leq c \right\} \geq \theta$$

Decision variables:

$X_i \in \{0, 1\}$ $\quad \forall i \in 1, \ldots, k$

Random variables:

$\omega_i \rightarrow$ item i weight $\quad \forall i \in 1, \ldots, k$

Stage structure:

$V_1 = \{X_1, \ldots, X_k\}$

$S_1 = \{\omega_1, \ldots, \omega_k\}$

$L = [\langle V_1, S_1 \rangle]$

Fig. 1. A Stochastic Programming formulation for the single-stage SKP. Note that $[y]^+ = \max\{y, 0\}$ and $\mathbb{E}$ denotes the expected value operator.

in the decision making process. In other words, what the alternation should be between decisions and random variable observations. In a decision stage $\langle V_i, S_i \rangle$, first we assign values to all the decision variables in the set V_i, then we observe the realized values for all the random variables in the set S_i. More specifically, in the single decision stage $\langle V_1, S_1 \rangle$ of the SKP, first we select all the objects that should be inserted into the knapsack, that is we assign a value to every decision variable $X_i \in V_1$, $\forall i \in 1, \ldots, k$; second, we observe the realized weight $\omega_i \in S_1$ for every object $i \in 1, \ldots, k$.

We now introduce a numerical example for the single-stage SKP.

Example 1. Consider $k = 5$ items whose item rewards r_i are $\{16, 16, 16, 5, 25\}$. The discrete probability mass functions for the weight ω_i of item $i = 1, \ldots, 5$ are respectively:
PMF(ω_1) $= \{10(0.5), 8(0.5)\}$, PMF(ω_2) $= \{9(0.5), 12(0.5)\}$, PMF(ω_3) $= \{8(0.5),$ $13(0.5)\}$, PMF(ω_4) $= \{4(0.5), 6(0.5)\}$, PMF(ω_5) $= \{12(0.5), 15(0.5)\}$.
The figures in parenthesis represent the probability that an item takes a certain weight. The other problem parameters are $c = 30$, $p = 2$ and $\theta = 0.6$.

As discussed, the problem has a single decision stage. This means that every decision has to be taken in a proactive way, before any of the random variables is observed. Therefore the optimal solution can be expressed as a simple assignment for the decision variables X_i, $\forall i \in 1, \ldots, k$. More specifically, the optimal solution for Example 1 proactively selects items $\{1, 4, 5\}$ and achieves an expected profit of 45.75. Such a solution can be validated using a scenario tree, as shown in Fig. 2. This tree considers every possible future realisation for the random variables ω_i, $\forall i \in 1, \ldots, k$. Since every random variable in the problem takes each of the possible values in its domain with uniform probability, all the paths in the scenario tree are equally likely. Therefore, it is easy to compute the expected profit of such an assignment and the expected additional capacity required. By

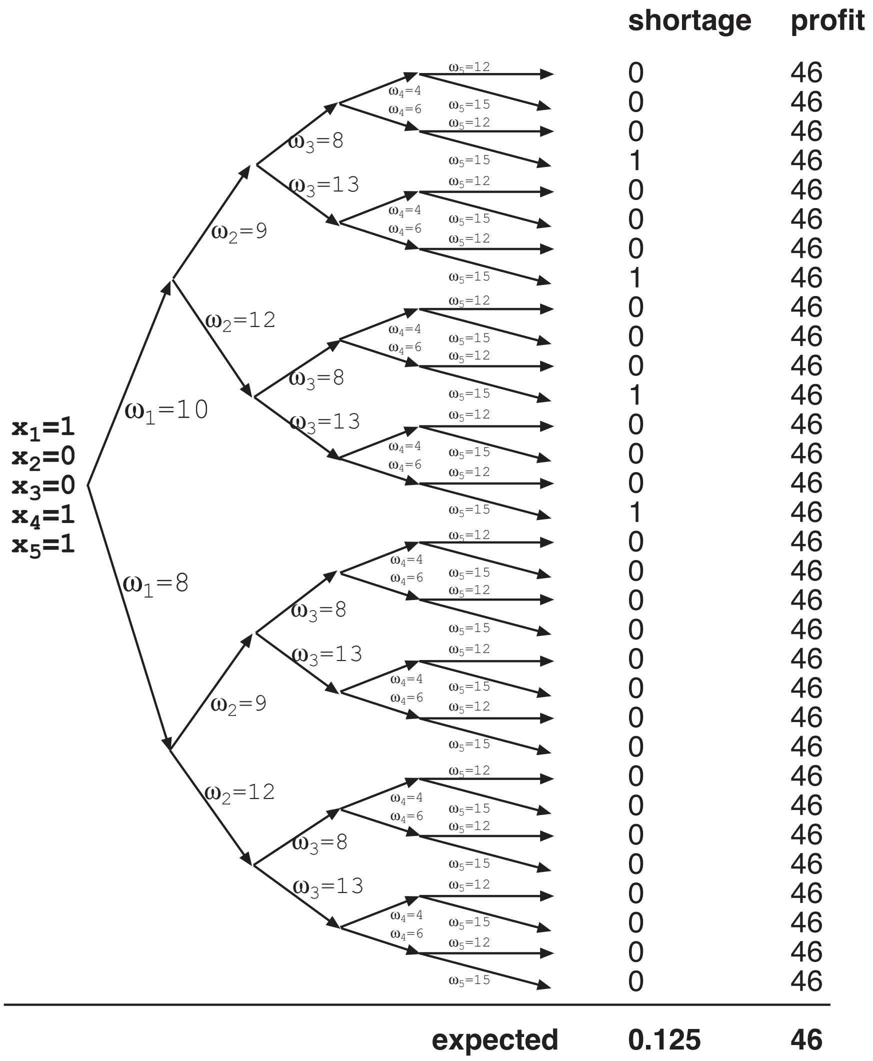


Fig. 2. Scenario tree representing the solution of the single-stage SKP in Example 1.

plugging these values into the objective function, the profit associated with this solution can be easily obtained (i.e. $46-2\cdot 0.125 = 45.75$). Finally, it can be easily verified that the chance-constraint in the model is also satisfied by this solution. In fact, a shortage is observed only in 4 out of 32 scenarios, therefore the chance constraint is satisfied, in this solution, with probability $0.875 \geq \theta = 0.6$.

The problem discussed in the former paragraphs only comprises a single decision stage. However, in general, stochastic programs may comprise multiple decision stages, that is a sequence of decisions and observations. In order to

Objective:

$$\max_{X_1}\{r_1X_1+\mathbb{E}_{\omega_1}\{\max_{X_2} r_2X_2+\mathbb{E}_{\omega_2}\{\ldots\{\max_{X_{k-1}} r_{k-1}X_{k-1}+$$
$$\mathbb{E}_{\omega_k}\{\max_{X_k} r_kX_k+p[\textstyle\sum_{i=1}^{k}\omega_iX_i-c]^+\}\}\ldots\}\}\}$$

Subject to:

$$\Pr\left\{\textstyle\sum_{i=1}^{k}\omega_iX_i\leq c\right\}\geq\theta$$

Decision variables:

$X_i\in\{0,1\}$ $\quad\forall i\in 1,\ldots,k$

Random variables:

$\omega_i\rightarrow$ item i weight $\forall i\in 1,\ldots,k$

Stage structure:

$V_i=\{X_i\}\ \forall i\in 1,\ldots,k$

$S_i=\{\omega_i\}\ \forall i\in 1,\ldots,k$

$L=[\langle V_1,S_1\rangle,\langle V_2,S_2\rangle,\ldots,\langle V_k,S_k\rangle]$

Fig. 3. Stochastic programming formulation for the multi-stage SKP.

clarify this, we slightly modify the SKP presented above in such a way as to allow for multiple decision stages. Therefore, we introduce the multi-stage SKP.

Multi-stage Stochastic Knapsack. The single-stage problem description and assumptions are valid here with the exception that the items are considered sequentially, starting from item 1 up to item k. In other words, first we take the decision of inserting or not a given object into the knapsack, then we immediately observe its weight, which is a random variable, before any further item is taken into account.

A stochastic programming model for the multi-stage SKP is shown in Fig. 3. The model is similar to the one presented in Fig. 1, but the structure of the objective function is different. In this new model, expectation ($\mathbb{E}_{\omega_i}$) and $\max_{X_i}$ operators are nested and parameterized each by, respectively, the random variable ω_i over which the expectation is computed and the decision variable X_i that should be assigned in order to maximize the objective function value. This means, in practice, that an object may be selected or not, depending on the realized weights for previous objects. The stage structure is also different, because now the problem comprises multiple decision stages that alternate decisions and observations according to the arrival sequence of the objects.

We refer, once more, to the Example 1 presented above. The numerical data introduced there can be used to obtain an instance of the multi-stage SKP. As discussed, the problem now has multiple decision stages. This means that decisions are taken in a dynamic way, and they are alternated with observations for random variables. Therefore, the optimal solution is now expressed by using a *solution tree.* A solution tree encodes full information on how to act at a certain decision stage, when some random variables have been already observed. More specifically, the optimal solution tree for the instance of the multi-stage SKP

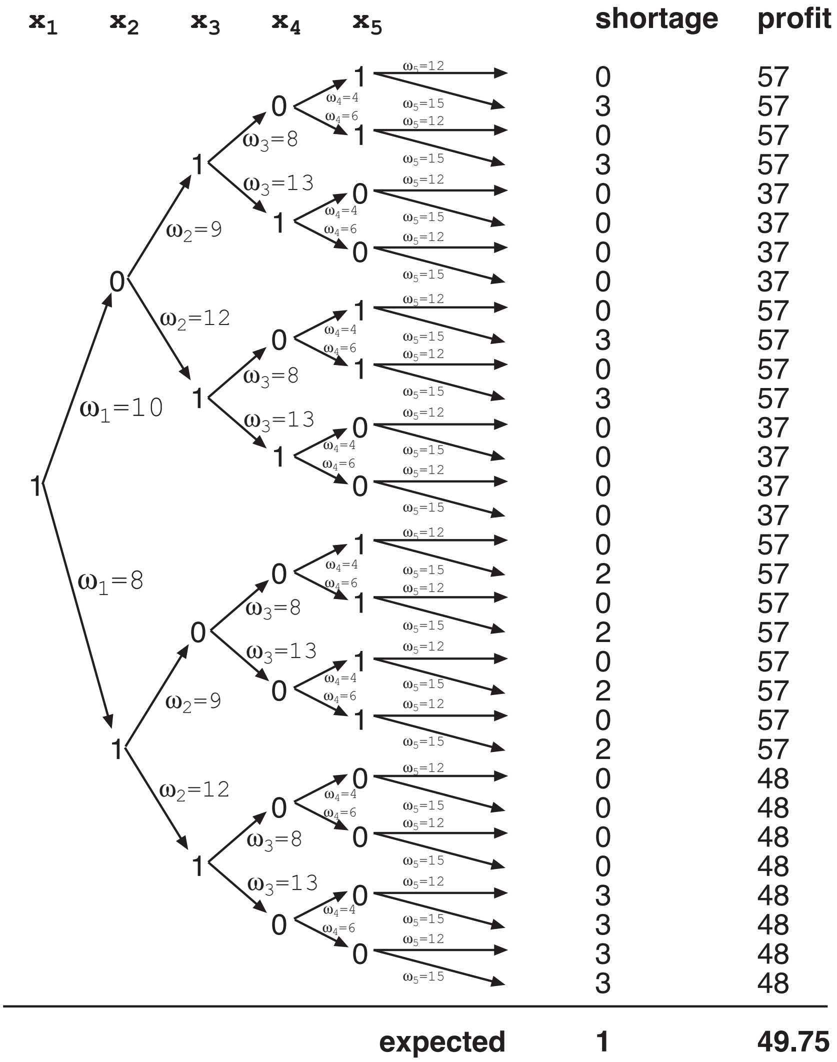


Fig. 4. Solution tree for the multi-stage SKP in Example 1.

defined by the data in Example 1 achieves an expected profit of 47.75 and it is shown in Fig. 4. To clarify: at the root node no uncertainty has been unfolded. The optimal solution tree in Fig. 4 shows that it is always optimal to take item 1 in the knapsack. Nevertheless, depending on the observed value for the weight of item 1, two alternative decisions may be optimal: not taking item 2 if the observed weight for item 1 is 10; or taking item 2 if the observed weight for item 1 is 8. To reiterate, since every random variable in the problem takes each of the possible values in its domain with uniform probability, all the paths in the solution tree are equally likely. Therefore, it is easy to compute the expected profit of such an assignment and the expected additional capacity required. By plugging these values into the objective function, the profit associated with this solution can be easily obtained. Finally, it can be also easily verified that the chance-constraint in the model is also satisfied by this solution. In fact, a shortage is observed only in 12 out of 32 scenarios, therefore the chance constraint is satisfied, in this solution, with probability $0.625 \geq \theta = 0.6$.

In this section we discussed the SKP; in Section 3 we provide a further list of problems from the literature discussing hybrid approaches to decision making under uncertainty.

3 A Collection of Stochastic Problems

In this section we provide a list of 9 other problems of decision making under uncertainty for which hybrid approaches have been proposed in the literature. This list is comprehensive, in the sense that it contains representative problems for each hybrid CP-AI-OR approach for decision making under uncertainty surveyed in this work.

The problems are:

- Stochastic queueing control problem [8, 75, 74]
- Scheduling Conditional Task Graphs [40]
- Stochastic reservation [5]
- Job shop scheduling with probabilistic durations [3]
- Two-stage stochastic matching problem [33]
- Production/inventory management [78]
- Stochastic template design [52, 71]
- Scheduling internal audit activities [60]
- Stochastic sequencing with release times and deadlines [57].

For each of these problems we provide a textual description; the reader may refer to the respective works where these problems were first introduced to obtain a more detailed description. In Section 5 we discuss and classify the hybrid solution methods proposed for modeling and solving these problems.

3.1 Stochastic queueing control problem

In a facility with front room and back room operations, the aim is to switch workers between the rooms in order to cope with changing customer demand. Customer arrival and service time are stochastic and the decision maker seeks a policy for switching workers such that the expected customer waiting time is minimized, while the staff in the back room remains sufficient to perform all work. The problem was originally proposed and analyzed in [8]. Terekhov and Beck investigated it in [75, 74].

3.2 Scheduling conditional task graphs

This is the problem, discussed in [40], of scheduling conditional task graphs in presence of unary and cumulative resources, minimizing the expected makespan. Conditional task graphs are directed acyclic graphs containing activities linked by precedence relations. Some of the activities represent branches. At run time only one of the successors of a branch is chosen for execution, depending on the occurrence of a condition labeling the corresponding arc. Since the truth or the falsity of those conditions is not known a priori, the problem is stochastic. Therefore all the possible future scenarios must be taken into account while constructing the schedule.

3.3 Stochastic reservation

This problem, introduced in [5], is a particular application of the stochastic multi-knapsack problem. A travel agency may aim at optimizing the reservation of holiday centers during a specific week with various groups in the presence of stochastic demands and cancellations. The requests are coming according a given probability distribution and they are characterized by the size of the group and the price the group is willing to pay. The requests cannot specify the holiday center. However, the travel agency, if it accepts a request, must inform the group of its destination and must commit to it. Groups can also cancel the requests at no cost. Finally, the agency may overbook the centers, in which case the additional load is accommodated in hotels at a fixed cost.

3.4 Job shop scheduling with probabilistic duration

This problem was originally proposed in [3]. The problem is a classic Job Shop Scheduling (JSP) (see [23], p. 242) in which the objective is to find the minimum makespan. In contrast to the classic formulation for the JSP presented in [23] the authors assume, in this case, that the job durations are probabilistic. The objective is therefore accordingly modified to account for uncertainty: the authors search for a proactive plan, consisting of a partial order among activities and of resource-activity allocations, which attains the lowest possible makespan with probability greater or equal to a given threshold.

3.5 Two-stage stochastic matching problem

We consider the minimum cost maximum bipartite matching problem discussed in [33]. The task is to buy edges of a bipartite graph which together contain a maximum-cardinality matching in the graph. The problem is formulated as a two-stage stochastic program with recourse, therefore edges can be bought either during the first stage, or with a recourse action after uncertainty has been resolved. There are two possible variants of this problem. In the first, the uncertainty is in the second stage edge-costs, that is the cost of an edge can either increase or decrease in the second stage. In the second variant all edges become more expensive in the second stage, but the set of nodes that must be matched is unknown. This problem can model real-life stochastic integral planning problems such as commodity trading, reservation systems and scheduling under uncertainty.

3.6 Production/inventory management

Uncertainty plays a major role in production and inventory management. In this simplified production/inventory planning example there are a single product, a single stocking point, production capacity constraints, service level constraints and a stochastic demand. The objective is to find a replenishment plan associated with the minimum expected total cost. The cost components taken into account are inventory holding costs and fixed replenishment (or setup) costs. The optimal plan gives the timing of the replenishments as well as the order quantities, which depend upon the previously realized demand. This production/inventory management problem has been investigated in [71, 78]. In [69] the authors investigate the same problems under the assumption that the production capacity constraints are relaxed.

3.7 Stochastic template design

The deterministic template design problem (prob002 in CSPLib[2]) is described as follows. We are given a set of variations of a design, with a common shape and size and such that the number of required pressings of each variation is known. The problem is to design a set of templates, with a common capacity to which each must be filled, by assigning one or more instances of a variation to each template. A design should be chosen that minimises the total number of runs of the templates required to satisfy the number of pressings required for each variation. As an example, the variations might be for cartons for different flavours of cat food, such as fish or chicken, where ten thousand fish cartons and twenty thousand chicken cartons must be printed. The problem would then be to design a set of templates by assigning a number of fish and/or chicken designs to each template such that a minimal number of runs of the templates is required to print all thirty thousand cartons. Proll and Smith [55] address this problem by fixing the number of templates and minimising the total number of pressings. In the stochastic version of the problem [52] the demand for each variation is uncertain. In compliance with production/inventory theory, the authors incorporate two conventional cost components: scrap cost, incurred for each template that is produced in excess of the realized demand, and shortage cost, incurred for each unit of demand not fulfilled. The objective is then to minimize the expected total cost.

[2] http://www.csplib.org

3.8 Scheduling internal audit activities

Based on costs and benefits that change over time, the focus of the internal audit scheduling problem is how often to conduct an internal audit on an auditable unit. Auditable units are the units upon which internal control procedures are applied, in order to safeguard assets and assure the reliability of information flows. The problem, originally introduced in [60], can be stated as follows. We consider a planning horizon comprising of N time periods. We are given a set of M audit units over which random losses may accrue over time. Losses in each period are assumed to have a known probability mass function that could easily be estimated from available historical data. The distribution of losses may vary from period to period, i.e., it is non-stationary. Losses at different periods are assumed to be independent. Auditing is a time-consuming task, and the auditing team is given a strict deadline for performing an audit. Specifically, an audit must be completed in T time periods. Therefore after T periods the accrued losses drop to zero. If a team has already started auditing a unit at a given time period, then no other audit can be initiated during this period for the given audit team. The timing of audits are fixed once and for all at the beginning of the planning horizon and cannot be changed thereafter, even if it is suspected that certain auditable units have accrued unexpected losses. The objective is to find the optimal audit schedule while respecting the maximum loss criteria. That is, the invariant audit cost (i.e., fixed audit costs incurred each time an audit is conducted) and expected total discounted audit losses (i.e., cumulative losses accrued at the end of each period) are minimized by satisfying a minimum probability α that the losses will not exceed a predetermined level (allowed maximum loss) in any given audit period for any auditable unit.

3.9 Stochastic sequencing with release times and deadlines

The problem, introduced in [57], consists in finding an optimal schedule to process a set of orders using a set of parallel machines. The objective is to minimize the expected total tardiness of the plan. Processing an order can only begin after its release date and should be completed at the latest by a given due date for such an order. An order can be processed on any of the machines. The processing time of a given order, when processed on a certain machine, is a random variable. A solution for this problem consists in an assignment for the jobs on the machines and in a total order between jobs on the same machine. A job will be processed on its release date if no other previous job is still processing, or as soon as the previous job terminates.

4 Frameworks for decision making under uncertainty in CP and AI

In Section 2 we introduced SP, a well established OR framework for decision making under uncertainty. In this section, we introduce other existing frameworks for decision making under uncertainty from, respectively, AI and CP.

Stochastic Boolean Satisfiability extends a well established AI modeling framework, Propositional Satisfiability, by considering uncertainty. Probabilistic CSP, Event-Driven Probabilistic Constraint Programming and Stochastic Constraint Programming set the scene for dealing with uncertainty in CP. Where appropriate, we describe connections and similarities among these different frameworks.

4.1 Stochastic Boolean Satisfiability

The Boolean Satisfiability (SAT) community have investigated problems involving uncertainty, with the *Stochastic Satisfiability* (SSAT) framework. SSAT aims to combine features of logic and probability theory, and has been applied to probabilistic planning, belief networks and trust management. We base our discussion on a recent survey [43].

Definitions The SAT problem is to determine whether a Boolean expression has a satisfying labelling (set of truth assignments). The problems are usually expressed in conjunctive normal form: a conjunction of clauses $c_1 \wedge \ldots \wedge c_m$ where each clause c is a disjunction of literals $l_1 \vee \ldots \vee l_n$ and each literal l is either a Boolean variable v or its negation $\bar{v}$. A Boolean variable can be labelled true (T) or false (F). Many constraint problems can be SAT-encoded (modelled as a SAT problem) and vice-versa. In fact any SAT problem can be viewed as a Constraint Satisfaction Problem (CSP) with binary domains and non-binary constraints via the *non-binary encoding* [77]: for example a clause $a \vee b \vee \bar{c}$ corresponds to the constraint (or conflict) preventing the assignments $\{a \leftarrow F,\ b \leftarrow F,\ c \leftarrow T\}$. The SSAT terminology is somewhat different than that of SP but there are many correspondences.

An SSAT problem $\Phi = Q_1 v_1 \ldots Q_n v_n \phi$ is specified by:

- a *prefix* $\Phi = Q_1 v_1 \ldots Q_n v_n$ that orders the Boolean variables $v_1 \ldots v_n$ of the problem and *quantifies* them. Each variable v_i is quantified by its quantifier Q_i either as *existential* ($\exists$) or *randomised* (ᴚ);
- a *matrix* ϕ: a Boolean formula containing the variables, usually in conjunctive normal form (CNF).

An existential variable is a standard SAT variable (corresponding to a decision variable in SP), while a randomised variable v_i is a Boolean variable that is true with associated probability π_i (corresponding to a random variable in SP). Sequences of similarly quantified variables may be grouped together into (existential or randomised) *blocks*, and an SP stage corresponds to an existential block followed by a randomised block. The values of existential variables may be contingent on the values of (existential or randomised) variables earlier in the prefix, so an SSAT solution takes the form of an *assignment tree* (corresponding to the solution tree in SP) specifying an assignment to each existential variable for each possible instantiation of the randomised variables preceding it in the prefix. An *optimal assignment tree* is one that yields the maximum probability of satisfaction; alternatively, the decision version of SSAT asks whether the probability of satisfaction exceeds a threshold θ.

SSAT is simpler than a Stochastic Program in three ways: the variable domains are Boolean only (as in SAT), the constraints (clauses) are of a fixed type (as in SAT), and no distinction is made between scenarios in which different clauses are violated. The latter means that SSAT is akin to a stochastic program with a single chance-constraint.

Restrictions and generalizations Some special cases have been identified in the literature: if all variables are randomised then we have a MAJSAT problem; if the prefix has only an existential block followed by a randomised block then we have an E-MAJSAT problem; and if each block contains a single variable then we have an Alternating SSAT (ASSAT) problem. SSAT has also been extended by the addition of *universal* quantifiers ($\forall$) to give Extended SSAT (XSSAT). A formula $\forall v \phi$ must be true for both $v = T$ and $v = F$. XSSAT subsumes Quantified Boolean Formulae (QBF), which is the archetypal PSPACE-complete problem: QBF is XSSAT without randomised quantifiers.

4.2 Probabilistic Constraint Satisfaction Problems

The Probabilistic CSP framework, proposed in [19], is an extension of the CSP framework [1] that deals with some decisions problems under uncertainty. This extension relies on a differentiation between the agent-controllable decision variables and the uncontrollable parameters whose values depend on the occurrence of uncertain events. The uncertainty on the values of the parameters is assumed to be given under the form of a probability distribution.

Definitions A probabilistic CSP is a CSP equipped with a partition between (controllable) decision variables and (uncontrollable) parameters, and a probability distribution over the possible values of the parameters. More specifically, the authors define a Probabilistic CSP as a 6-tuple $\mathcal{P} = \langle \varLambda, W, X, D, \mathcal{C}, pr \rangle$, where $\varLambda = \{\lambda_1, \ldots, \lambda_p\}$ is a set of parameters; $W = W_1 \times \cdots \times W_p$, where W_i is the domain of λ_i; $X = \{x_1, \ldots, x_n\}$ is a set of decision variables; $D = D_1 \times \ldots \times D_n$, where D_i is the domain of x_i; $\mathcal{C}$ is a set of constraints, each of them involving at least one decision variable; and $pr : W \to [0, 1]$ is a probability distribution over the parameter assignments. Constraints are defined as in classical CSP. A complete assignment of the parameters (resp. of the decision variables) is called a "world" (resp. a "decision").

The authors consider successively two assumptions concerning the agents awareness of the parameter values at the time the decision must imperatively be made.

- "No more knowledge": the agent will never learn anything new before the deadline for making a decision; all it will ever know is already encoded by the probability distribution.
- "Complete knowledge": the actual parameters will be completely revealed before the deadline is reached (possibly, just before), so that it it useful to

the agent to compute off-line a ready-to-use conditional decision, that the agent will be able to instantiate on-line, as soon as it knows what the actual parameters are.

For the first case, a solution is an unconditional decision that is most likely to be feasible according to world probabilities. For the second case, a solution provides a set of decisions with their conditions of applicability — i.e. under which world(s) a given decision should be used — together with the likelihood of occurrence of these conditions, which also follows from world probabilities.

4.3 Event-driven Probabilistic Constraint Programming

In Event-driven Probabilistic Constraint Programming (EDP-CP), which is an extension of the Probabilistic CSP framework, some of the constraints can be designated by the user as *event constraints*. The user's objective is to maximize his/her chances of realizing these "events". In each world — as defined in the Probabilistic CSP framework — events are subject to certain pre-requisite constraints and to certain conditions. If a pre-requisite is unsatisfied in a given world then the event is also classed as unsatisfied in that world; and if a condition is unsatisfied in a world then the event is classed as satisfied in that scenario. Intuitively, this means that in EDP-CP it is possible to express the fact that the feasibility of certain event constraints may depend on the satisfaction of other constraints (denoted as "pre-requisite constraints") under certain "conditions". In order to model such situations, a new meta-constraint — the *dependency meta-constraint* — is introduced.

Definitions An EDP-CP is a 9-tuple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \Lambda, \mathcal{W}, \mathcal{E}, \mathcal{C}, \mathcal{H}, \Psi, \Pr \rangle$ where:

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of decision variables;
- $\mathcal{D} = D_1 \times \ldots \times D_n$, where D_i is the domain of X_i;
- $\Lambda = \{\lambda_1, \ldots, \lambda_l\}$ is a set of uncertain parameters;
- $\mathcal{W}= W_1 \times \ldots \times W_l$, where W_i the domain of λ_i;
- $\mathcal{E} = \{e_1, \ldots, e_m\}$ is a set of event constraints. Each e_i may either be probabilistic (involving a subset of $\mathcal{X}$ and a subset of Λ) or deterministic (involving only a subset of $\mathcal{X}$);
- $\mathcal{C} = \{c_1, \ldots, c_o\}$ is a set of dependency meta-constraints. For each dependency meta-constraint $c_i : \textsc{Dependency}(e, p, f)$ we have $e \in\mathcal{E}$, where p may be either a probabilistic or a deterministic pre-requisite constraint, and f is a deterministic condition constraint;
- $\mathcal{H}= \{h_1, \ldots, h_p\}$ is a set of hard constraints. Each h_i may either be probabilistic (involving a subset of $\mathcal{X}$ and a subset of Λ) or deterministic (involving only a subset of $\mathcal{X}$);
- Ψ is any expression involving the event realization measures on the event constraints in $\mathcal{E}$;
- $\Pr : \mathcal{W} \rightarrow [0, 1]$ is a probability distribution over uncertain parameters.

An optimal solution to an EDP-CP $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \Lambda, \mathcal{W}, \mathcal{E}, \mathcal{C}, \mathcal{H}, \Psi, Pr \rangle$ is any assignment S to the decision variables such that:

1. the hard constraints are satisfied in each possible world; and
2. there exists no other assignment satisfying all the hard constraints with a strictly better value for Ψ, according to the DEPENDENCY constraints introduced in the model.

Relations to other Frameworks The Event-driven Probabilistic Constraint Programming (EDP-CP) framework, proposed in [67], extends both the Probabilistic CSP framework [19] and the Dependent-chance Programming framework [38]. In contrast to probabilistic CSP, which treats all probabilistic constraints uniformly, EDP-CP distinguishes between event, pre-requisite, condition, and hard constraints. Furthermore, in Dependent-chance Programming a feasible solution satisfies all event constraints, whilst in EDP-CP such a requirement is relaxed. This gives the decision-maker more flexibility in modeling. Finally, the notion of constraint dependency introduced in [67] comprises condition constraints, in addition to the event and pre-requisite constraints. As the authors remark, constraint dependency without condition constraints does not guarantee optimal plans since in certain instances common variables may take values which break the link between two dependent constraints.

4.4 Stochastic Constraint Programming

Stochastic Constraint Programming (SCP) was first introduced in [78] in order to model combinatorial decision problems involving uncertainty and probability. According to Walsh, SCP combines together the best features of CP (i.e. global constraints, search heuristics, filtering strategies, etc.), of SP (expressiveness in representing problems involving random variables), and of Stochastic Satisfiability.

Definitions An m-stage Stochastic Constraint Satisfaction Problem (SCSP) is defined, according to [78], as a 7-tuple $\langle V, S, D, P, C, \theta, L \rangle$[3], where V is a set of decision variables and S is a set of random variables, D is a function mapping each element of V and each element of S to a domain of potential values. In what follows we assume that both decision and random variable domains are finite. P is a function mapping each element of S to a probability distribution for its associated domain. C is a set of chance-constraints over a non-empty subset of decision variables and a subset of random variables. θ is a function mapping each

[3] The original formulation, proposed in [78], does not directly encode the stage structure in the tuple and actually defines a SCSP as a 6-tuple; consequently the stage structure is given separately. We believe that a more adequate formulation is the one proposed in [30], that explicitly encodes the stage structure as a part of the tuple, giving a 7-tuple.

Objective:

$$\max\left\{\sum_{i=1}^{k} r_i X_i - p\mathbb{E}\left[\sum_{i=1}^{k} \omega_i X_i - c\right]^{+}\right\}$$

$\langle V, S, D, P, C, \theta, L\rangle$**:**

$$V = \{X_1, \ldots, X_k\}$$
$$S = \{\omega_1, \ldots, \omega_k\}$$
$$D = \{X_1, \ldots, X_k \in \{0,1\}, D(\omega_1), \ldots, D(\omega_k)\}$$
$$P = \{PDF(\omega_1), \ldots, PDF(\omega_k)\}$$
$$C = \left\{\Pr\left\{\sum_{i=1}^{k} \omega_i X_i \le c\right\} \ge \theta\right\}$$
$$L = [\langle\{X_1, \ldots, X_k\}, \{\omega_1, \ldots, \omega_k\}\rangle]$$

Fig. 5. Stochastic Constraint Programming formulation for the single-stage SKP.

chance-constraint $h \in C$ to θ_h which is a threshold value in the interval $(0, 1]$, indicating the minimum satisfaction probability for chance-constraint h. Note that a chance-constraint with a threshold of 1 (or without any explicit threshold specified) is equivalent to a hard constraint. $L = [\langle V_1, S_1\rangle, \ldots, \langle V_i, S_i\rangle, \ldots, \langle V_m, S_m\rangle]$ is a list of *decision stages* such that each $V_i \subseteq V$, each $S_i \subseteq S$, the V_i form a partition of V, and the S_i form a partition of S.

To solve an m-stage SCSP an assignment to the variables in V_1 must be found such that, given random values for S_1, assignments can be found for V_2 such that, given random values for S_2, ..., assignments can be found for V_m so that, given random values for S_m, the hard constraints are satisfied and the chance constraints are satisfied in the specified fraction of all possible scenarios. The solution of an m-stage SCSP is represented by means of a *policy tree*. A policy tree is a set of decisions where each path represents a different possible scenario and the values assigned to decision variables in this scenario. The policy tree, in fact, corresponds to the solution tree adopted in SP.

Let $\mathcal{S}$ denote the space of policy trees representing all the solutions of a SCSP. We may be interested in finding a feasible solution, i.e. a policy tree $s \in \mathcal{S}$, that maximizes the value of a given objective function $f(\cdot)$ over a set $\widehat{S} \subseteq S$ of random variables (edges of the policy tree) and over a set $\widehat{V} \subseteq V$ of the decision variables (nodes in the policy tree). A *stochastic constraint optimization problem* (SCOP) is then defined in general as $\max_{s \in \mathcal{S}} f(s)$.

Unlike SP, SCP offers a richer modeling language which supports chance-constraints over global, nonlinear, and logical constraints in addition to linear ones.

It is easy to reformulate the running example discussed in Section 2 (SKP) as a single-stage SCOP, the respective model is given in Fig. 5. As in the SP model, in the SCP model we have sets of decision and random variables with their respective domains. For the random variables the respective probability mass function is specified. There is a chance-constraint with an associated threshold θ. In fact, the SCOP in Fig. 5 fully captures the structure of the stochastic program in Fig. 1.

5 A Classification of Existing Approaches

In previous sections we stressed the fact that this survey is centered on "uncertainty", and we also clarified the precise meaning we associate with the term uncertainty. Other literature surveys tend to merge uncertainty with other concepts; in Section 9 we will briefly discuss related works in these different areas, and the reader may refer to these surveys for more details. Furthermore, there exist surveys that are more explicitly focused on pure AI [10] or OR [62] techniques, but little attention has been dedicated so far to hybrid techniques.

In this section, we propose a classification for existing hybrid approaches and frameworks that blend CP, AI and OR for decision making under uncertainty. The integration of CP, AI and OR techniques for decision making under uncertainty is a relatively young research area. We propose to classify existing approaches in the literature within three main classes (Fig. 10).

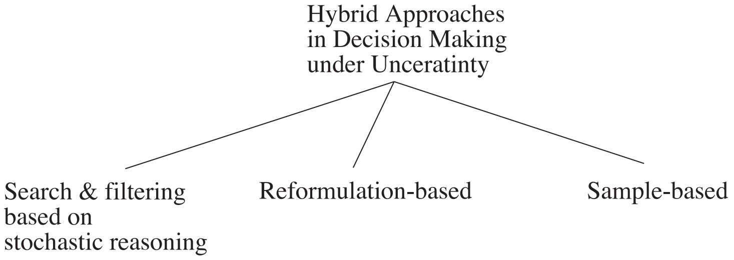

Hybrid Approaches in Decision Making under Unceratinty
Search & filtering based on stochastic reasoning
Reformulation-based
Sample-based

Fig. 6. A classification of hybrid approaches in CP-AI-OR for decision making under uncertainty.

- The first class comprises those approaches that perform some form of "stochastic reasoning" by using dedicated — general or special purpose — search procedures, filtering algorithms, neural networks, genetic algorithms etc.
- The second class, in contrast, includes approaches that exploit reformulation — once again employing either a specialized analytical derivation for a given problem, or general purpose techniques — in order to produce a deterministic model that can be solved using existing solvers.
- Finally, the third class comprises incomplete approaches that exploit sampling in order to attain a near-optimal solution for problems of optimization under uncertainty. We believe that approaches based on sampling are particularly attractive and deserve a dedicated class. In fact, a high level of complexity is a typical trait of decision problems involving uncertainty, therefore it seems that the only feasible way of tackling many of these problems consists in developing effective approximation strategies.

Before discussing further this classification, it is worth mentioning that we believe it would be impractical to list all existing applications of hybrid methods from CP, AI, and OR in decision making under uncertainty. For this reason we aim rather to classify the different strategies — and not the specific applications — adopted in the literature for solving this class of problems using hybrid approaches. Nevertheless, for each strategy mentioned in this section, we will report some of the respective applications.

In Section 6, we will discuss approaches performing "stochastic reasoning"; in Section 7 we will discuss approaches that exploit reformulation; and finally in Section 8 we will discuss incomplete approaches that exploit sampling.

6 Approaches based on stochastic reasoning

In this section we will analyze existing approaches that perform some sort of "stochastic reasoning" by using dedicated — general or special purpose — techniques. These techniques take several different forms: search procedures, filtering algorithms, neural networks, genetic algorithms etc.

Firstly, we shall distinguish between *general purpose* and *problem specific* strategies (Fig. 7).

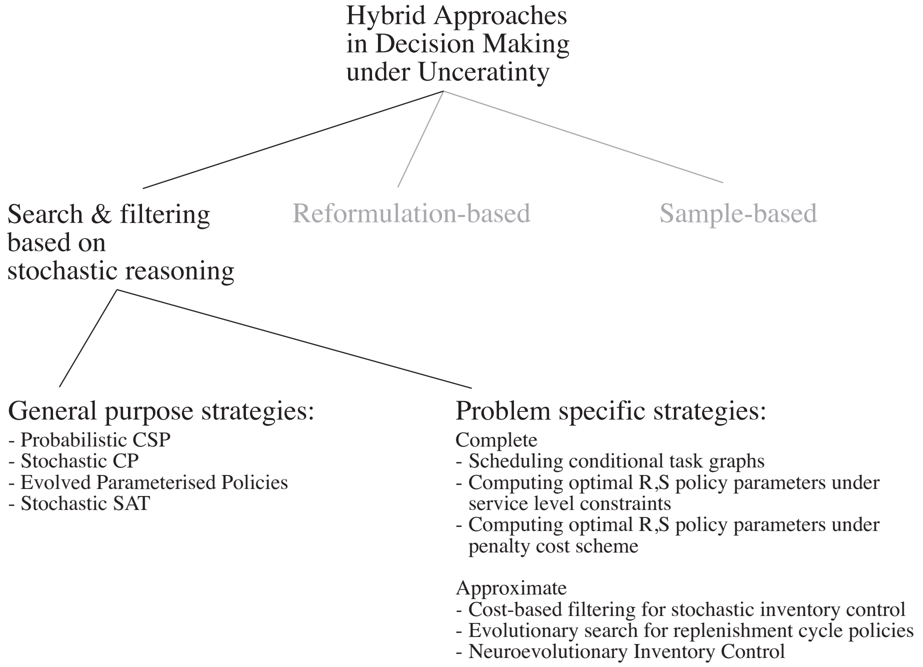

Hybrid Approaches
in Decision Making
under Unceratinty
Search & filtering
based on
stochastic reasoning
Reformulation-based
Sample-based
General purpose strategies:
- Probabilistic CSP
- Stochastic CP
- Evolved Parameterised Policies
- Stochastic SAT
Problem specific strategies:
Complete
- Scheduling conditional task graphs
- Computing optimal R,S policy parameters under service level constraints
- Computing optimal R,S policy parameters under penalty cost scheme
Approximate
- Cost-based filtering for stochastic inventory control
- Evolutionary search for replenishment cycle policies
- Neuroevolutionary Inventory Control

Fig. 7. A classification of hybrid approaches in CP-AI-OR for decision making under uncertainty: approaches based on stochastic reasoning.

General purpose strategies aim to develop frameworks that provide modeling and solving facilities to handle generic problems of decision making under uncertainty. The modeling frameworks proposed in the literature typically aggregate concepts from different domains, for instance *global constraints* from CP, *chance-constraints* and *random variables* from SP (OR). These frameworks exploit well established AI strategies, such as forward checking procedures and genetic algorithms, in the solution process.

Problem specific strategies typically develop specialized reasoning algorithms that, during the search, are able to perform inference by exploiting the specific structure of the problem. For instance a typical approach is to encapsulate the reasoning within a dedicated global constraint that prunes decision variable domains according to the underlying stochastic reasoning.

In addition, both general purpose and problem specific strategies may be complete or heuristic. We shall now discuss in more detail these two different classes of approaches based on stochastic reasoning, by providing pointers to works in the literature.

6.1 General purpose strategies

We survey four different general purpose strategies for modeling and solving different classes of problems of decision making under uncertainty. These are: Probabilistic CSP, Stochastic CP, Evolving Parameterised Policies, and Stochastic SAT.

Probabilistic CSP. One of the first general purpose frameworks for modeling uncertainty in CP is the Probabilistic CSP [19]. In the Probabilistic CSP a distinction is made between *controllable* and *uncontrollable* variables which correspond, respectively, to decision and random variables in SP. As in SP, a probability density function is associated with each uncontrollable variable. The authors discuss two different settings. Under the first of these settings, for each of the possible realizations that may be observed for the uncontrollable variables, the best decision is determined. This strategy corresponds to the wait-and-see policy in SP ([32], pp. 8) and it presents a posterior analysis. The second setting simply corresponds to a conventional single stage stochastic program where an optimal decision has to be taken before observing the realized values for the uncontrollable variables. The optimal decision, in this second case, is the one that guarantees the maximum likelihood to result feasible with respect to the given probability density functions for the uncontrollable variables.

The authors propose two algorithms for solving Probabilistic CSPs. The first algorithm, used for solving problems formulated under the first setting discussed, borrows ideas from solution methods developed in for solving Dynamic CSPs [18] and, in particular, reuses a procedure proposed in [21]. The second proposed algorithm consists of a depth first branch and bound algorithm and of a forward checking procedure. These are employed to solve problems formulated under the second setting discussed.

Stochastic Constraint Programming. The Probabilistic CSP represents the first attempt to include random variables, and thus uncertainty, within the CP framework. Nevertheless, only in [78] is a clear link established between CP and SP with the introduction of SCP. We have already discussed in detail SCP as a modeling framework in Section 4.4. In [78] Walsh discusses the complexity of Stochastic CSPs, and proposes a number of complete algorithms and of approximation procedures for solving them. Namely, a backtracking algorithm and a forward checking procedure are proposed, which resemble those proposed in [19] for Probabilistic CSPs. Nevertheless, we want to underscore the fact that the key difference between a Probabilistic CSP and a Stochastic CSP is the fact that the former does not handle multiple decision stages.

In [2] Balafoutis et al. build on the SCP framework introduced in [78], they correct a flaw in the original forward checking procedure for Stochastic CSPs and they also extend this procedure in order to better take advantage of probabilities and thus to achieve stronger pruning. In addition, *arc-consistency* is defined for Stochastic CSPs and an arc-consistency algorithm able to handle constraint of any arity is introduced. Tests are carried on random binary Stochastic CSPs formulated as single and multi-stage problems.

In [13] Bordeaux and Samulowitz investigate two extensions to the original SCP framework. Firstly, they investigate situations in which variables are not ordered sequentially, corresponding to situations in which the future can follow different branches; they show that minor modifications allow the framework to deal with non-sequential forms. Secondly, they investigate how to extend the framework in such a way as to incorporate multi-objective decision making. An algorithm is proposed, which solves multi-objective stochastic constraint programs in polynomial space.

Global chance-constraints — which we discussed in Section 4.4 — were introduced first in [58], and they bring together the reasoning power of global constraints from CP and the expressive power of chance-constraints from SP. A general purpose approach for filtering global chance-constraints is proposed in [30]. This approach is able to reuse existing propagators available for the respective deterministic global constraint which corresponds to a given global chance-constraint when all the random variables are replaced by constant parameters. In addition, in [57] Rossi et al. discuss some possible strategies to perform cost-based filtering for certain classes of Stochastic COPs. These strategies exploit well-known inequalities borrowed from SP and used to compute valid bounds for any given Stochastic COP that respects some mild assumptions. Examples are given for a simplified version of the stochastic knapsack problem previously discussed and for the stochastic sequencing problem discussed in Section 3.9.

Evolved Parameterised Policies. Inspired by the success of machine learning methods for stochastic and adversarial problems, a recent approach to Stochastic CSPs/COPs called *Evolved Parameterised Policies* (EPP) is described in [53]. Instead of representing a policy explicitly in a Stochastic Constraint Program, an attempt is made to find a rule that decides, at each decision stage, which domain value to assign to the decision variable(s) at that stage. The quality of a rule can be determined by constructing the corresponding policy tree and observing the satisfaction probability of each chance constraint (and the value of the objective function if there is one). Evolutionary or other non-systematic search algorithms can be used to explore the space of rules.

EPP treats a Stochastic CSP/COP problem as an unconstrained noisy optimisation problem with at worst the same number of (real-valued) variables. This allows a drastic compression of the policy tree into a small set of numbers, and this compression together with the use of evolutionary search makes EPP scalable to large multi-stage Stochastic CSPs/COPs. It has the drawback that only policies of a relatively simple form can be discovered, but it results much more robust than a scenario-based approach on a set of random multi-stage problems [53]. Moreover, arbitrarily complex rules could be discovered by using artificial neural networks instead of these simple functions, a *neuroevolutionary* approach that has been successfully applied to many problems in control [24, 29, 64].

Stochastic SAT. Another general purpose framework for modeling and solving a well established class of problems under uncertainty in AI — and especially in planning under uncertainty — is Stochastic SAT. We introduced the modeling framework in Section 4.1. Current SSAT algorithms fall into three classes: *systematic*, *approximation*, and *non-systematic*.

The systematic algorithms are based on the standard SAT backtracking algorithm — the Davis-Putnam-Logemann-Loveland (DPLL) algorithm [17, 16] — and correspond roughly to some current SCSP algorithms. The first such algorithms were described in [36], in particular the `evalssat` algorithm for XSSAT which formed the basis for future systematic SSAT algorithms. `evalssat` did not use branching heuristics as in current SAT and CSP solvers, though [36] also used some restricted branching heuristics, but assigned variables in the order specified by the prefix. However, it did use SAT-based techniques (unit propagation and pure variable elimination) and reasoning on the probability threshold θ to prune the search tree. The policy-based SCSP algorithm of [78] is essentially `evalssat` with forward checking. Systematic algorithms have also been devised for special cases of XSSAT. MAXPLAN [45], ZANDER [46] and DC-SSAT [44] all use special techniques for planning problems modelled as XSSAT problems.

The `sampleevalssat` approximation algorithm uses random sampling to select paths, then uses SAT techniques to search the restricted tree to maximise θ. The `APPSSAT` algorithm [42] considers scenarios in decreasing order of probability to construct a partial tree for the special case of planning problems modelled as SSAT problems.

The `randevalssat` algorithm [36] is based on the `sampleevalssat` algorithm mentioned above, but applies stochastic local search to the existential variables in a random set of scenarios, thus it is non-systematic. Other ways of applying local search were described in [41], including periodically restarting `randevalssat` with different sampled scenarios, an approach used by the WALKSSAT algorithm [79].

6.2 Problem specific strategies

In the previous section we discussed general purpose solution methods that bring together CP, AI and OR techniques for decision making under uncertainty. We will now discuss some special purpose approaches proposed in the literature that perform stochastic reasoning on specific problems.

Scheduling conditional task graphs. The work of [40] describes a complete, special purpose approach that concerns the problem — discussed in Section 3.2 — of scheduling conditional task graphs. Similarly to the approach in [56], the authors propose an analytical formulation of the stochastic objective function, in this case based on the task graph analysis, and a conditional constraint able to handle such a formulation efficiently. The authors show the benefit of such an approach by comparing the results with a deterministic model, which disregards uncertainty, and with a scenario-based formulation [71] that requires an exponential number of scenarios to fully represent the stochastic objective function.

Computing optimal R,S policy parameters under service level constraints. Another special purpose strategy is presented in [58], and proposes a dedicated global chance-constraint for computing replenishment cycle inventory policy parameters under service level constraints. More specifically, the problem considered in this work is the production/inventory problem described in Section 3.6. Computing optimal replenishment cycle policy parameters for such a problem is a complex task [69]. By using a dedicated global chance-constraint the authors were able to perform the complex stochastic reasoning required to compute optimal replenishment cycle policy parameters. Such a complete algorithm performs a numerical integration step in order to compute the real service level provided in each period by a given set of policy parameters and the associated expected total cost.

Computing optimal R,S policy parameters under penalty cost scheme. Similarly, a dedicated global constraint has been proposed in [56] in order to solve to optimality the problem of computing optimal replenishment cycle policy parameters under a penalty cost scheme. Such a problem has been investigated in [70], but in this work the authors could only solve the problem in a heuristic way, by employing a piecewise linear approximation of the convex cost function in the problem in order to build up a deterministic equivalent MIP model. In [56] the authors were able to embed a closed-form non-linear analytical expression for such a convex cost function within a global constraint, thus obtaining a complete model able to compute optimal replenishment cycle policy parameters.

Cost-based filtering for stochastic inventory control. The work in [68] has a different flavor. In this case, the underling model is the deterministic equivalent CP formulation proposed in [73] for computing near-optimal replenishment cycle policy parameters under service level constraints. The CP formulation was originally proposed as a reformulation of the MIP model in [69]. Such a reformulation showed significant benefits in terms of efficiency. The authors, in [68], propose three independent cost-based filtering strategies that perform stochastic reasoning and that are able to significantly speed up the search when applied to the original CP model in [73].

Evolutionary search for replenishment cycle policies. A recent application of a genetic algorithm to a multi-stage optimisation problem in inventory control is described in [51]. Each chromosome represents a replenishment cycle policy plan as a list of order-up-to levels, with a level of 0 representing no order, and the fitness of a chromosome is averaged over a large number of scenarios. This approach is enhanced in [50] by hybridising the genetic algorithm with the SARSA temporal difference learning algorithm [61]. This is shown to greatly improve the performance of genetic search for replenishment cycle policies, both with and without order capacity constraints.

Neuroevolutionary Inventory Control. One may evolve an artificial neural network to optimally control an agent in an uncertain environment. The network inputs represent the environment and its outputs the actions to be taken. This combination of evolutionary search and neural networks is called *neuroevolution*. A recent paper [54] applies neuroevolution to find optimal or near-optimal plans in inventory control, following no special policy. The problems are multi-stage and involve multi-echelon systems (they have more than one stocking point). Such problems have no known optimal policy and rapidly become too large for exact solution. The inputs to the network are the current stock levels and the outputs are the order quantities.

7 Reformulation-based approaches

In this section, we will analyze existing approaches that are based on a reformulation that produces a deterministic model, which can be solved using an existing solver.

Once more, we shall distinguish between *general purpose* and *problem specific* strategies (Fig. 8).

Hybrid general purpose reformulation strategies have recently appeared especially at the borderline between CP and OR. These typically take the form of a high level language — such as Stochastic OPL — used to formulate the problem under uncertainty, and of a general purpose compiler that can handle the high level stochastic model and produce a compiled deterministic equivalent one. Often, the compilation relies on a well known technique in SP: *scenario-based modeling*. In addition, due to the complexity of stochastic programs in general, approximation strategies are often proposed in concert with these general purpose frameworks in order to make the size of the compiled model manageable.

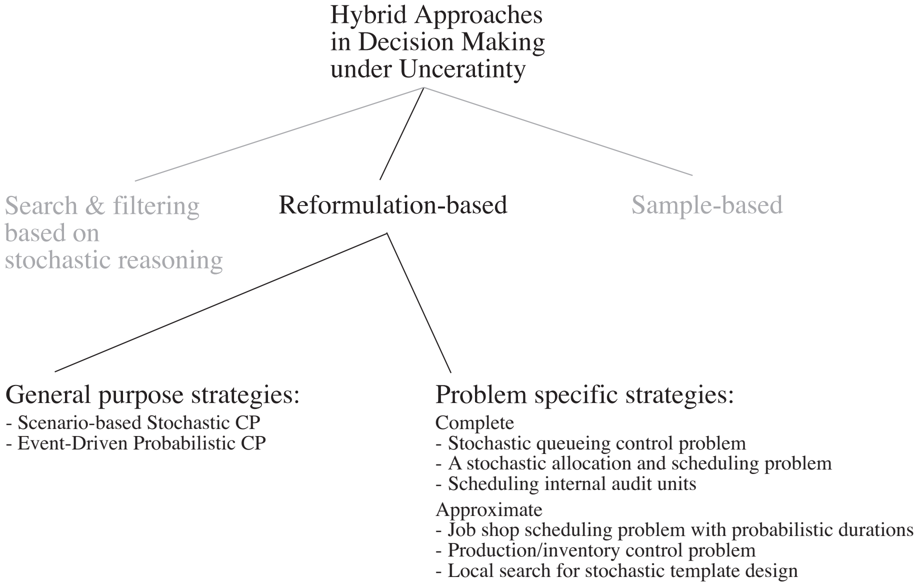

Hybrid Approaches
in Decision Making
under Unceratinty
Search & filtering
based on
stochastic reasoning
Reformulation-based
Sample-based
General purpose strategies:
- Scenario-based Stochastic CP
- Event-Driven Probabilistic CP
Problem specific strategies:
Complete
- Stochastic queueing control problem
- A stochastic allocation and scheduling problem
- Scheduling internal audit units
Approximate
- Job shop scheduling problem with probabilistic durations
- Production/inventory control problem
- Local search for stochastic template design

Fig. 8. A classification of hybrid approaches in CP-AI-OR for decision making under uncertainty: approaches based on a deterministic reformulation.

In contrast, problem specific strategies aim to fully exploit the structure of the problem in order to produce a deterministic — and possibly equivalent — model that can be handled efficiently by existing solver. In many cases, in order to obtain a model that is manageable by existing solvers, it is necessary to introduce some assumptions that affect the completeness and, thus, the quality of the solution found the the deterministic model. We will provide examples of applications in which a special purpose deterministic equivalent model is built, which is equivalent to the original model and also examples in which the deterministic model can only approximate the original stochastic model.

7.1 General purpose strategies

We survey two different general purpose strategies based on reformulation for modeling and solving classes of problems of decision making under uncertainty. These are Scenario-based Stochastic CP and Event-Driven Probabilistic Constraint Programming.

Scenario-based Stochastic Constraint Programming. The first general purpose framework based on reformulation that we present is Scenario-based Stochastic Constraint Programming, which was proposed by Tarim et al. in [71]. The novelty in this work is the fact that the authors adopt a semantics for stochastic constraint programs based on scenario trees. By using this semantics the authors can compile stochastic constraint programs into conventional (non-stochastic) constraint programs and they can therefore use existing constraint solvers to effectively solve this class of problems.

In a scenario based approach — frequently used in SP [11] — a scenario tree is generated which incorporates all possible realizations of discrete random variables into the model explicitly. A path from the root to an extremity of the event tree represents a scenario. With each scenario a given probability is associated. Within each scenario we have a conventional (non-stochastic) constraint program to solve. All we need to do is replace the random variables by the values taken in the scenario, and ensure that the values found for the decision variables are consistent across scenarios, as certain decision variables are shared across scenarios. Constraints are defined (as in traditional constraint satisfaction) by relations of allowed tuples of values, and can be implemented with specialized and efficient algorithms for consistency checking. Furthermore, the scenario-based view of stochastic constraint programs also allows later-stage random variables to take values which are conditioned by the earlier-stage random variables. This is a direct consequence of employing the scenario representation, in which random variables are replaced with their scenario dependent values.

Scenario-based SCP has been outlined in Section 4.4. Tarim et al. [71] not only defined a general way to compile stochastic constraint programs into conventional constraint programs, but they also proposed a language, Stochastic OPL, which is based on the OPL constraint modeling language [28]. Using this language the authors modeled optimization problems under uncertainty from a variety of fields, such as portfolio selection, agricultural planning, and production/inventory management (Section 3.6). We will not discuss the language in detail, but in the Appendix we show how to model the single and multi-stage SKP problems of Section 2 by using the Stochastic OPL.

Among the benefits of the scenario based approach in [71] is the fact that it allows multiple chance-constraints and a range of different objectives to be modeled. The authors point out that each of these changes would require substantial modifications in the backtracking and forward checking algorithms proposed in [78]. The scenario based view allows each of these extensions to be modeled easily using stochastic OPL, compiled down into standard OPL, and solved by means of existing solvers. It should be noted that the approach is general and the compilation need not necessarily be performed using OPL, but it can be implemented using any available CP language and/or software package. The main drawback of this approach is the fact that the scenario tree required to model a given problem grows exponentially in size when random variable domains are large, thus leading to large models that are difficult to solve.

In addition to this general purpose modeling/solving framework the authors also proposed some techniques to improve the efficiency of the solution process. In order to do so, they proposed scenario reduction techniques, such as Monte Carlo Sampling or Latin Hypercube Sampling [65], to reduce the number of scenarios considered in the model. Their experimental results show the effectiveness of this approach, which in practice is able to find high quality solutions using a small number of scenarios. Finally, inspired by robust optimization techniques used in OR [35], the authors also proposed some techniques to generate robust solutions, that is solutions that adopt similar (or the same) decisions under different scenarios.

Event-Driven Probabilistic Constraint Programming. We now briefly discuss a second general purpose framework based on reformulation: Event-Driven Probabilistic Constraint Programming [67]. This framework was introduced to address different problems than those for which SCP is a suitable modeling tool. Event-Driven Probabilistic Constraint Programming, as the name suggest, is connected to Probabilistic CSPs and, mainly, to Dependent-chance Programming [37, 38].

Sometimes a complex probabilistic decision system undertakes multiple tasks, called *events* here, and the decision-maker wishes to maximize chance functions which are defined as the probabilities of satisfying these events. This is especially useful in situations where a particular measure of the "reliability" or "robustness" of a given plan has to be maximized. The Event-Driven Probabilistic Constraint Programming modeling framework allows users to designate certain probabilistic constraints, involving both decision and random variables, as *events* whose chance of satisfaction must be maximized, subject to hard constraints which should be always satisfied, and also logical dependencies among constraints. Event-Driven Probabilistic Constraint Programming builds on Dependent-chance Programming and provides more expressiveness to the user, in order to capture a more realistic and accurate measure of plan reliability [59]. It also provides an exact solution method, employing scenario-based reformulation, in contrast to the approximate genetic algorithm in [38].

7.2 Problem specific strategies

We now discuss some problem specific strategies based on deterministic equivalent reformulations.

Stochastic queueing control problem. In [75, 74] the authors propose a set of deterministic equivalent CP models for solving the stochastic queueing control problem discussed in Section 3.1. [75] not only provides the first application of CP to solving a stochastic queueing control problem, but it also provides a complete approach for a problem for which only a heuristic algorithm [8] existed. Three deterministic equivalent constraint programming models and a shaving procedure are proposed. The complete models provide satisfactory performances when compared with the heuristic procedure, which nevertheless remains superior in terms of solution quality over time. A hybrid method is therefore proposed, which combines the heuristic in [8] with the best constraint programming method. Such a hybrid approach performs better than either of these approaches separately.

The interesting aspect of this work is that, as in [60], all the stochastic information is encoded as constraints and expected values, and there is no need of random variables or scenarios. The three models proposed explore different sets of variables and different configurations for the constraint set, for instance using duality. Nevertheless, all the three models use predefined constraints available in standard CP solvers.

A stochastic allocation and scheduling problem. The problem, discussed in [39], is the scheduling problem described in Section 3.2 applied to multiprocessor systems on chip: given a conditional task graph characterizing a target application and a target architecture, with alternative memory and computation resources, the authors compute an allocation and schedule that minimize the expected value of communication costs, since — as they point out — communication resources are one of the major bottlenecks in modern multiprocessor systems on chips. The approach they propose is complete and efficient. As in the previous cases, it is based on a deterministic equivalent reformulation of the original stochastic integer linear programming model. More specifically, the authors employ logic based Benders decomposition. The stochastic allocation problem is solved through an Integer Programming solver, while the scheduling problem with conditional activities is handled with CP. The two solvers interact through no-goods. Once more, one of the main contributions is the derivation of an analytical deterministic expression employed in order to compute the expected value of communication costs in the objective function. This expression makes it possible for the authors to transform the original stochastic allocation problem into a deterministic equivalent one that can be solved using any available Integer Programming solver.

Scheduling internal audit units. In [60] the authors analyze the problem of scheduling internal audit units discussed in Section 3.8. A stochastic programming formulation is proposed with Mixed Integer Linear Programming and CP certainty-equivalent models. Both the models transform analytically the chance-constraints in the model into deterministic equivalent ones. In experiments neither approach dominates the other. However, the CP approach is orders of magnitude faster for large audit times, and almost as fast as the MILP approach for small audit times.

Finally, we discuss works in which the deterministic model obtained through reformulation for a given stochastic program is not “equivalent”; rather, it is based on some simplifying assumption that makes it possible to obtain a compact deterministic formulation able to provide a near-optimal solution and an approximate value for the cost of such a solution, or a bound for such a cost.

Job shop scheduling with probabilistic durations. In [3] an approximate deterministic reformulation is employed to compute valid bounds to perform cost-based filtering. In this work the authors analyze the Job Shop Scheduling problem discussed in Section 3.4, in which the objective is to find the minimum makespan. In contrast to the classic formulation presented in [23], in [3] the authors assume that the job durations are probabilistic. The objective is therefore accordingly modified to account for uncertainty. More specifically, the authors search for a proactive plan, consisting of a partial order among activities and of resource-activity allocations, which attains the lowest possible makespan with probability greater or equal to a given threshold. For this problem the authors propose a deterministic formulation, which depends on a given non-negative parameter q. A correct choice of such a parameter guarantees that the minimum makespan for the deterministic model is a lower bound for the minimum makespan that can be attained with a certain threshold probability in the original model. This deterministic model can be efficiently solved with classic constraint programming techniques and can provide tight bounds at each node of the search tree that are employed to perform cost-based filtering. A number of heuristic techniques are proposed for correctly choosing a “good” value for the parameter q.

Production/inventory control problem. Consider the production/inventory problem discussed in Section 3.6. The deterministic reformulation proposed in Tarim et al. [73] relies on some mild assumptions — discussed in [69] — concerning order-quantities. Under these assumptions, it was possible for the authors to obtain analytical deterministic expressions for enforcing the required service level in each period of the planning horizon, and to compute the expected total cost associated with a given set of policy parameters. By using these expressions, it was possible for the authors to formulate a deterministic model by employing standard constraints available in any CP solver. In [58], the authors compare the solutions obtained through a complete formulation with those obtained with the model in [73]. This comparison shows that the assumptions do not significantly compromise optimality, whereas they allow the construction of a model that can significantly outperform the complete one, and solve real-world instances comprising long planning horizons and high demand values.

Local search for stochastic template design. In [52] the stochastic template design problem discussed in Section 3.7 is reformulated as a deterministic equivalent constrained optimisation problem, using all possible scenarios and a novel modeling technique to eliminate non-linear constraints. The result is a standard integer linear program that proved to be hard to solve by branch-and-bound. However, a local search algorithm design for linear integer programs performed very well, and was more scalable than the Bender’s decomposition algorithm in [72].

8 Approaches based on sampling

In this section we will discuss sample-based approximation strategies for solving problems of decision making under uncertainty. Due to the complexity of these problems in general, several works in the literature have been devoted to analyzing the effectiveness of heuristic approaches based on sampling. In Fig. 9 it is possible to observe how three main trends have been identified in the CP and AI literature, which apply sampling in a hybrid setting for solving problems of decision making under uncertainty: the Sample Average Approximation approach (SAA), Forward Sampling and Sample Aggregation.

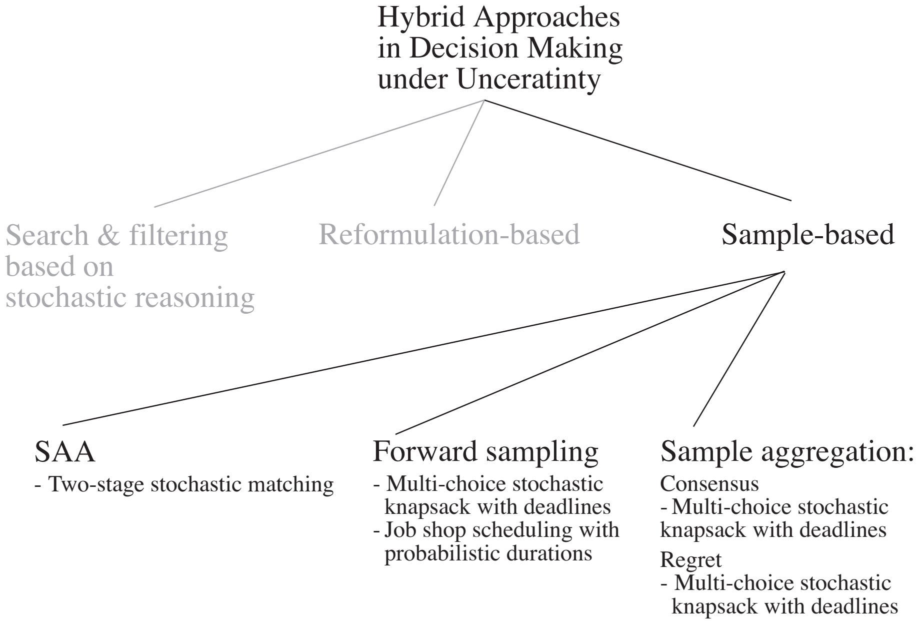

Hybrid Approaches in Decision Making under Unceratinty
Search & filtering based on stochastic reasoning
Reformulation-based
Sample-based
SAA
- Two-stage stochastic matching
Forward sampling
- Multi-choice stochastic knapsack with deadlines
- Job shop scheduling with probabilistic durations
Sample aggregation:
Consensus
- Multi-choice stochastic knapsack with deadlines
Regret
- Multi-choice stochastic knapsack with deadlines

Fig. 9. A classification of hybrid approaches in CP-AI-OR for decision making under uncertainty: approaches based on sampling.
text under figure

- In OR, and particularly in SP, the state-of-the-art technique that applies sampling in combinatorial optimization is the Sample Average Approximation approach [34]. In this approach a given number of samples is drawn from the random variable distributions, and the combinatorial problem of interest is repeatedly solved by considering different samples as input in each run. The real expected cost/profit of a solution produced for a given sample is then computed by simulating a sufficient number of samples. Among all the solutions computed, the one that provides the minimum expected cost (or the maximum expected profit) is retained. Two criteria are given by the authors: one for deciding when a given sample size is no more likely to produce better solutions, and one to decide if it increasing the sample size may lead to better solutions.

- Forward sampling, as the name suggests, is a sort of forward checking that employs samples in order to make inference about which values are not consistent in decision variable domains or about the expected cost/profit of associated with a given (partial) assignment for decision variables, which is assessed against the generated samples by computing, for instance, the expected profit/cost of such an assignment with respect to these samples.
- Sample aggregation is a strategy in which a number of samples is generated, for each of these samples a deterministic problem is solved, then the results obtained for all these samples are aggregated and analyzed according to some rule. The "best" among these decisions is implemented in practice. For instance, a possible rule may always choose the decision that is optimal for the highest number of samples.

In the CP and AI literature, sampling is often applied in concert with a so called "online" optimization strategy. Online refers to the fact that decisions and observations are interleaved in the problem, and each time an observation occurs an optimization step takes place to compute the next decision, by taking into account the probability density function of future random variables and the observed values for the past ones. It is easy to notice that a multi-stage stochastic program subsumes an online strategy if the decision maker has a complete knowledge of the probability density function of the random variables in the problem. In this case we may compute the entire solution tree at the beginning, and use it in order to find the best following decision each time a random variable is observed. Nevertheless, several reasons justify the use of an online strategy (also called a "rolling horizon" approach in the OR literature and especially in Inventory Control). The most compelling reason for using an online approach is that it does not require the decision maker to have a complete knowledge of the probability density functions of the random variables. Consider, for instance, the Stochastic Knapsack Problem introduced in the previous sections. If the problem is formulated as a multi-stage stochastic program and we have a full knowledge about the possible weights that can be observed for all the objects, the policy tree will prescribe exactly what to do in each possible future course of action. Nevertheless, if at some stage one of the objects takes a weight that is not part of the probability density function we considered for such an object, the policy tree will not be able to prescribe an appropriate action. In contrast, an online approach would simply take into consideration this weight in the following optimization step and it would however provide a valid decision to be implemented next.

Stochastic problems solved using online strategies, and to which either forward sampling or sample aggregation strategies are applied, appear in a number of works within the CP and AI literatures. In what follow we shall classify some of these works on the basis of which sampling technique is applied.

8.1 Sample Average Approximation

In this section we provide a pointer to a work that proposes to apply SAA to a modified version of a classic matching problem: the two-stage stochastic matching problem.

Two-stage stochastic matching problem. In [33] Katriel et al. consider the two-stage stochastic matching problem discussed in Section 3.5. The authors prove lower bounds and analyze efficient strategies. We do not provide here a general survey for this work, as the reader may refer to the cited article for more details. Instead, we focus on one of the authors' contributions in which they firstly observe that, in this problem, with independently activated vertices the number of scenarios is extremely large. However, in such a situation there is often a black box sampling procedure that provides, in polynomial time, an unbiased sample of scenarios; then they observe that one can use the SAA method to simulate the explicit scenarios case and, under some mild assumptions, obtain a tight approximation guarantee. The main observation is that the value of the solution defined by taking a polynomial number of samples of scenarios tightly approximates the value of the solution defined by taking all possible scenarios.

8.2 Forward Sampling

In this section we survey two relevant works in which forward sampling is applied: the multi-choice stochastic knapsack with deadlines and the job shop scheduling with probabilistic durations.

Multi-Choice Stochastic Knapsack with Deadlines. In [5] the authors analyze different techniques for performing online stochastic optimization. A benchmark problem is proposed in order to assess all these different techniques. The benchmark stemmed from the authors' industrial experience and it consist of a Multi-Choice Stochastic Knapsack with Deadlines. This problem corresponds, in practice, to the stochastic reservation problem discussed in Section 3.3 and it is used to test four different online strategies exploiting combinations of the stochastic and combinatorial aspects of the problem. These strategies are, respectively, forward sampling, average values, most likely scenario analysis and yield management techniques.

Initially, the authors propose two naive order handling policies: a first-come/first-serve policy and a best-fit policy. Furthermore, in order to assess the quality of a given policy, the authors also discuss "far seeing" strategies, which assume advanced knowledge of the realized demand and can therefore solve the associated deterministic multi-choice knapsack problem.[4]

One of the strategies used in this work to estimate the quality of a given policy — for instance first-come/first-serve or best-fit — employs forward sampling in

[4] We recall that in SP this corresponds to using a wait-and-see policy and performing a posterior analysis.

order to generate samples from the current date to the end of the planning horizon. The evaluation of a sample can be done, for instance, by simulating the behavior of a best-fit strategy for the specific sample. The policy evaluation then will be a measure (for instance the average) over the evaluations of many generated samples.

Job Shop Scheduling with probabilistic durations. Forward sampling is also employed in [3]. We recall that in this work the authors analyze the Job Shop Scheduling problem discussed in Section 3.4, in which the authors assume that the job durations are probabilistic. A number of algorithms are proposed for solving this problem through sampling. Firstly, a branch-and-bound procedure is introduced, which exploits at each node of the search tree a Monte Carlo simulation approach to compute — with respect to the partial assignment associated with such a node — a valid lower bound for the minimum possible makespan that may be attained with a probability greater or equal to the given threshold. Since sampling is employed for computing the bound, confidence interval analysis is employed to estimate if the attainment probability associated with the given makespan is a sufficiently reliable estimate. Secondly, the authors propose a number of heuristic techniques that aim to limit the amount of time spent on Monte Carlo simulation during the search, by using the deterministic makespan as an oracle for selecting and simulating only the most promising plans in order to save CPU time and to dedicate more time to the exploration of the search space rather than on simulating non-promising plans. Finally, dedicated tabu search strategies are proposed in order to propose a valid alternative to the constructive search techniques above, which are mainly based on tree-search.

8.3 Sample Aggregation

In this section, we discuss works in which two alternative sample aggregation strategies are employed: the “Consensus” strategy and the “Regret” strategy. The problem to which these strategies are applied is, once more, the multi-choice stochastic knapsack with deadlines.

Multi-Choice Stochastic Knapsack with Deadlines. In [27] the authors consider the same Online Multi-Choice Knapsack with Deadlines problem considered in [5]. In order to solve this problem the authors employ the following online algorithm. The algorithm receives a sequence of online requests and starts with an empty allocation. At each decision point the algorithm considers the current allocation and the current request, and chooses a bin in which to allocate the request, which is then included in the current assignment. Eventually, the algorithm returns the final allocation and the respective value. In order to decide in which bin to allocate a given request, the algorithm employs a function “chooseAllocation” which is based on two black boxes: a function “getSample” that returns a sample of the arrival distribution; and a function “optSol” that, given the current assignment and a request, returns an optimal allocation of the request by taking into account the past decisions. The authors then consider four possible options for implementing “chooseAllocation”:

– The best-fit strategy discussed in [5].
– A strategy called “Expectation” — in practice performing a forward sampling —- that generates future requests by sampling and that evaluates each possible allocation for a given request (i.e. in which bin to fit such a request) against the samples.
– A strategy called “Consensus”, which was introduced in [47], and whose key idea is to solve each sample only once. More specifically, instead of evaluating each possible bin at a given time point with respect to each sample, “consensus” executes the optimization algorithm only once per sample. The bin to which the request is eventually allocated by this optimization step is then credited with the respective profit, while the other bins receive no credit. The algorithm eventually returns the bin with which the highest profit is associated.
– A strategy called “Regret” [6, 7] based on a sub-optimality approximation, which is a fast estimation of the loss caused by sub-optimal allocations. The key steps in the process of choosing a bin resemble the “consensus” algorithm. But in “regret”, instead of assigning some credit only to the bin selected by the optimal solution, the sub-optimality approximation is used to compute, for each possible request allocation, an approximation of the best solution that makes such a choice. Therefore every available bin is given an evaluation for every sample at a given time, at the cost of a single optimization.

Consensus and regret are two examples of what we previously defined as “sample aggregation” strategies.

9 Related works

In this section we will first briefly discuss Stochastic Dynamic Programming, a related and well established technique in OR that deals with decision making under uncertainty. We will also clarify why this technique has not been covered in the former sections. Secondly, we will cast our work within a broader picture, and contrast our survey with existing similar works that address the topics of uncertainty and change.

9.1 Stochastic Dynamic Programming

An alternative and effective technique for modeling problems of decision making under uncertainty is Dynamic Programming. In [4] Bellman explicitly states that Dynamic Programming was initially conceived for modeling multi-stage decision processes. He also argues that these processes arise in practice in a multitude of diverse field and in many real life problems, for instance in stock control, scheduling of patients through a medical clinic, servicing of aircraft at an airfield, etc. Dynamic Programming has been applied to a multitude of deterministic multi-stage decision problems, but in [4] Bellman also discussed its application to stochastic multi-stage decision processes. As in the deterministic case, in the stochastic case the modeling also relies mainly on the development of adequate functional equations capturing the dynamics of the system, and the expected cost (or profit) function associated with the possible decisions and affected by the random variables in the problem. The multi-stage decision process, in Dynamic Programming, is typically defined recursively, starting from a bounding condition that describes a degenerate state of the system that can be easily characterized. Depending on the specific nature of the process being analyzed (Markovian, Semi-Markovian, etc. — see [25], Chapter 8) it is possible to exploit its structure to devise efficient solution methods or closed form solutions for the optimal control policy, which corresponds to the policy tree that constitutes a solution of a given Stochastic Program.

In this work we mainly focused on the connections between and integration of SP, CP and AI. So far Dynamic Programming has not played a role as significant as SP in the development of hybrids approaches for decision making under uncertainty. For this reason, Stochastic Dynamic Programming and its extension to infinite horizon case Markov Decision Processes are not thoroughly covered here. For more details on Stochastic Dynamic Programming the reader may refer to the seminal work of Bellman [4], and to the works of Bertsekas [9], Warren [49], Sutton and Barto [66] and Gosavi [25].

9.2 Related Modeling Frameworks

Recently, the topic of decision making under *uncertain* and *dynamic* environment has been discussed in two literature surveys [76, 14]. Nevertheless, these two works discuss a variety of different problems that can hardly be classified within a unique group. For instance, consider a problem whose structure changes dynamically over time. As an example we may refer to the Dynamic Constraint Network discussed in [18], in which, from time to time, new facts that become known about the model induce a change in the constraint network. We find that such a problem has almost nothing in common with a problem where some parameters are random — thus may assume a certain value with a given probability — and a decision has to be taken proactively, before the realized values for these parameters are known. As an example for this second class, we may consider the proactive Job Shop Scheduling problem discussed in [3], in which an optimal plan — that achieves a minimum makespan with a certain probability — has to be determined before the actual job durations are known. Also consider, as in [22], a constraint satisfaction problem in which we allow some of the constraints to be violated by a solution, and in which we search for a solution that tries to satisfy the original constraint problem as much as possible; or, alternatively, consider a constraint satisfaction problem, as in [26], in which some of the values in the decision variable domains may suddenly become unavailable after a solution has been computed and for which we are looking for robust solutions that can be "repaired" with little effort. These two latter examples, again, significantly differ from the previous ones and among each others.

A clear and comprehensive classification of all these different problems and frameworks is still missing. For this reason, in this section we propose a classification in three distinct classes and we try to position in each of these classes some of the frameworks proposed in the literature.

In our classification (Fig. 10) there are three criteria based on which a particular framework is classified: Degree of Change, Degree of Satisfiability and Degree of Uncertainty.

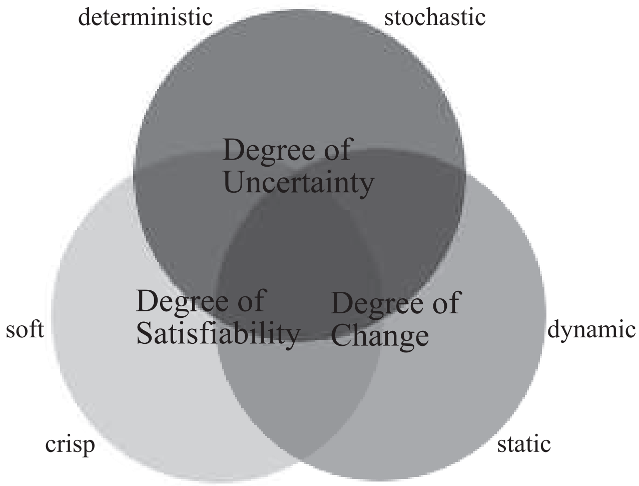


Fig. 10. A classification for existing frameworks based on problem structure.

- With respect to the **Degree of Change**, "static" refers to a classic, static CSP, while "dynamic" refers to the fact that the model is assumed to change dynamically, since constraints are added/removed. The solution has to be flexible enough to be adapted to these changes without too many modifications and with limited computational effort. Existing frameworks that, with respect to the Degree of Change, are classified as "dynamic" are: Dynamic Constraint Satisfaction (Dechter [18]); Conditional CSP (Minton et al. [48]); and Super-solutions in CP (Hebrard et al. [26]).
- With respect to the **Degree of Satisfiability** "crisp" refers to a classic CSP in which all the constraints have to be satisfied by a given solution, while "soft" refers to the fact that some of the constraints in the model may be violated by a solution. The aim is to find a solution that typically violates the minimum number of constraints or that, in general, minimizes some violation measure. Existing frameworks that, with respect to the Degree of Satisfiability, are classified as "soft" are: Partial Constraint Satisfaction (Freuder [20]); Constraint solving over semi-rings (Bistarelli et al. [12]); and Valued Constraint Satisfaction (Schiex et al. [63]).

- With respect to the **Degree of Uncertainty**, “deterministic” refers to classic CSPs, while “stochastic” refers to the existence of uncontrollable (random) variables in the model for which a probability distribution is given. Stochastic problems present an alternation of decisions and observations. Constraints are assigned a satisfaction threshold that must be met by any given solution.

Some of the frameworks presented in the literature do, in fact, cover more than one of the classes presented, and for this reason the circles are intersecting each others. Clearly, this classification does not cover several other frameworks that in the years have been proposed to deal with other problem classes.

We have introduced pointers to relevant frameworks that can be either classified under Degree of Change (“dynamic”) or Degree of Satisfiability (“soft”). Problems that are classified as “stochastic” with respect to their Degree of Uncertainty have been widely surveyed in the former part of this work. We argue that such a classification better positions existing works with respect to aspects that are, in fact, orthogonal among each others.

10 Conclusions

In this survey we focused on hybrid CP-AI-OR methods for decision making under uncertainty. Firstly, we explicitly defined what “uncertainty” is and how it is possible to model it by using SP, a well established existing modeling framework in OR. We surveyed additional existing frameworks — one from AI and one from CP — for modeling problems of decision making under uncertainty and we also identified the relevant connections among these frameworks. Secondly, we introduced a list of problems from the literature in which uncertainty plays a role and we categorized existing hybrid techniques that have been proposed for tackling these problems into three classes. In the first class we identified general and special purpose approaches that perform “stochastic reasoning”. In the second class we listed approaches, once more general and special purpose, that use reformulation. In the third class we categorized approximate techniques based on a variety of strategies employing sampling. Finally, we pointed out connections with other related works.

11 Appendix

In [71] Stochastic OPL, a language for modeling stochastic constraint programs, is proposed. We will now show how the single and multi-stage SKP problems introduced in Section 2 can be easily modeled using such a language.

In Fig. 11 the Stochastic OPL model for the single stage SKP is presented. As in the model presented in Fig. 1, the objective function maximizes the revenue brought by the objects in the knapsack minus the expected penalty for exceeding capacity. Chance-constraint `prob(sum(i in Items) W[i]*x[i] <= c) >=` θ ensures that the capacity is not exceeded with a probability higher than θ.

```
int k = ...;
int p = ...;
int c = ...;
float θ = ...;
range Items 1..k;
range onestage 1..1;
stoch myrand[onestage]=...;
float W[Items,onestage]^myrand = ...;
float r[Items] = ...;
dvar float+ z;
dvar int x[Items] in 0..1;

maximize sum(i in Items) x[i]*r[i] - expected(p*z)
subject to{
   z >= sum(i in Items) W[i]*x[i] - c;
   prob(sum(i in Items) W[i]*x[i] <= c) >= θ;
};
```

Fig. 11. Stochastic OPL formulation for the single-stage SKP.

We now refer to the numerical Example 1 for SKP. In Fig. 12 the Stochastic OPL data file corresponding the numerical instance in Example 1 is presented. We recall that the strategy proposed in [71] employs a scenario-based formulation. In fact it is easy to see that, given the random variables in the example and the values in their domains, there are a total of 32 scenarios that should be considered. Each row for variable `W` in Fig. 12 has, in fact, 32 entries (i.e. `[<10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10, 8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8>]`). There are in total 5 rows, each having 32 entries, so a column — containing all the entries at the same position in each row — therefore fully encodes one of the possible 32 scenarios. The probability of each of the 32 scenarios is provided using the array `myrand`. By using the compilation strategy proposed in [71], any model and data file written using Stochastic OPL can be easily compiled into a classic (deterministic) constraint program and solved by using classic solvers. The optimal solution for Example 1 — computed using the compiled OPL code obtained from the Stochastic OPL model and data file presented — selects items $\{1, 4, 5\}$ and achieves an expected profit of 45.75, as shown in Fig 2.

The SKP can be also formulated as a multi-stage stochastic constraint program as shown in Fig 3. In Fig. 13 the Stochastic OPL model for the multi-stage SKP is presented. The model is similar to the one presented in Fig. 11. Nevertheless, now the weight of each object is observed at a different decision stage. Therefore we have an array of k random variables (`stoch W[Items]`) in contrast to the previous model that only had one random variable (`myrand`) to model the probability distribution of the possible scenarios. In Fig. 14 the data file corresponding to the numerical instance in Example 1 is presented. The opti-

```
k = 5;
p = 2;
c = 30;
θ = 0.6;
W = [
          [<10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,8,8,8,
            8,8,8,8,8,8,8,8,8,8,8,8,8>],
          [<9,9,9,9,9,9,9,9,12,12,12,12,12,12,12,12,9,9,9,9,9,9,
            9,9,12,12,12,12,12,12,12,12>],
          [<8,8,8,8,13,13,13,13,8,8,8,8,13,13,
            13,13,8,8,8,8,13,13,13,13,8,8,8,8,13,13,13,13>],
          [<4,4,6,6,4,4,6,6,4,4,6,6,4,4,6,6,4,4,
            6,6,4,4,6,6,4,4,6,6,4,4,6,6>],
          [<12,15,12,15,12,15,12,15,12,15,12,15,12,15,12,15,
            12,15,12,15,12,15,12,15,12,15,12,15,12,15,12,15>]
];
myrand = [
          <0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),
          0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),
          0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),
          0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),
          0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),
          0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),
          0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),
          0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125)>
];
r = [16,16,16,5,25];
```

Fig. 12. Stochastic OPL Data File for the single-stage SKP.

mal solution for Example 1, when the problem is formulated as a multi-stage Stochastic COP, can be computed using the compiled OPL code obtained from the Stochastic OPL model in Fig. 13 and from the data file presented in Fig. 14. This solution takes the form of a policy tree — graphically rendered in Fig. 4 — and achieves an expected profit of 47.75.

References

1. K. Apt. *Principles of Constraint Programming.* Cambridge University Press, Cambridge, UK, 2003.
2. T. Balafoutis and K. Stergiou. Algorithms for stochastic csps. In Frédéric Benhamou, editor, *Principles and Practice of Constraint Programming, CP 2006, Proceedings*, volume 4204 of *Lecture Notes in Computer Science*, pages 44–58. Springer, 2006.
3. J. C. Beck and N. Wilson. Proactive algorithms for job shop scheduling with probabilistic durations. *J. Artif. Intell. Res. (JAIR)*, 28:183–232, 2007.

```
int k = ...;
int p = ...;
int c = ...;
float θ = ...;
range Items 1..k;
stoch W[Items]=...;
float r[Items] = ...;
dvar float+ z;
dvar int x[Items] in 0..1;

maximize expected(sum(i in Items) x[i]*r[i]) - p*expected(z)
subject to{
   z >= sum(i in Items) W[i]*x[i] - c;
   prob(sum(i in Items) W[i]*x[i] <= c) >= θ;
};
```

Fig. 13. Stochastic OPL formulation for the multi-stage SKP.

```
k = 5;
p = 2;
c = 30;
θ = 0.6;
W = [
          <10(0.5),8(0.5)>,
          <9(0.5),12(0.5)>,
          <8(0.5),13(0.5)>,
          <4(0.5),6(0.5)>,
          <12(0.5),15(0.5)>
];
r = [16,16,16,5,25];
```

Fig. 14. Stochastic OPL Data File for the multi-stage SKP.

4. R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.
5. T. Benoist, E. Bourreau, Y. Caseau, and B. Rottembourg. Towards stochastic constraint programming: A study of online multi-choice knapsack with deadlines. In Toby Walsh, editor, *Principles and Practice of Constraint Programming, CP 2001, Proceedings*, volume 2239 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2001.
6. R. Bent and P. Van Hentenryck. Regrets only! online stochastic optimization under time constraints. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pages 501–506, 2004.
7. R. Bent, I. Katriel, and P. Van Hentenryck. Sub-optimality approximations. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP*

2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings, volume 3709 of *Lecture Notes in Computer Science*, pages 122–136. Springer, 2005.

8. O. Berman, J. Wang, and K. P. Sapna. Optimal management of cross-trained workers in services with negligible switching costs. *European Journal of Operational Research*, 167(2):349–369, 2005.
9. D. P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 1995.
10. L. Bianchi, M .Dorigo, L. Gambardella, and W. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2):239–287, 2009.
11. J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming.* Springer Verlag, New York, 1997.
12. S. Bistarelli, U. Montanari, and F. Rossi. Constraint solving over semirings. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI '95*, pages 624–630, 1995.
13. L. Bordeaux and H. Samulowitz. On the stochastic constraint satisfaction framework. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 316–320, New York, NY, USA, 2007. ACM.
14. K. N. Brown and I. Miguel. Uncertainty and change. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 21. Elsevier, 2006.
15. A. Charnes and W. W. Cooper. Deterministic equivalents for optimizing and satisficing under chance constraints. *Operations Research*, 11(1):18–39, 1963.
16. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.
17. M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.
18. R. Dechter and A. Dechter. Belief maintenance in dynamic constraint networks. In *Proceedings of the 7th National Conference on Artificial Intelligence, AAAI '88*, pages 37–42, 1988.
19. H. Fargier, J. Lang, R. Martin-Clouaire, and T. Schiex. A constraint satisfaction framework for decision under uncertainty. In *UAI '95: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, August 18-20, 1995, Montreal, Quebec, Canada*, pages 167–174, 1995.
20. E. C. Freuder. Partial constraint satisfaction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, IJCAI '89*, pages 278–283. Morgan Kaufmann, 1989.
21. E. C. Freuder and P. D. Hubbe. Extracting constraint satisfaction subproblems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI '95, Montral, Qubec, Canada, August 20-25*, pages 548–557. Morgan Kaufmann, 1995.
22. E. C. Freuder and R. J. Wallace. Partial constraint satisfaction. *Artif. Intell.*, 58(1-3):21–70, 1992.
23. M. R. Garey and D. S. Johnson. *Computer and Intractability. A guide to the theory of NP-Completeness.* Bell Laboratories, Murray Hill, New Jersey, 1979.
24. F. J. Gomez, J. Schmidhuber, and R. Miikkulainen. Efficient non-linear control through neuroevolution. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*, volume 4212 of *Lecture Notes in Computer Science*, pages 654–662. Springer, 2006.

25. A. Gosavi. *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning.* Kluwer Academic Publishers, Norwell, MA, USA, 2003.
26. E. Hebrard, B. Hnich, and T. Walsh. Super solutions in constraint programming. In Jean-Charles Régin and Michel Rueher, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, First International Conference, CPAIOR 2004, Nice, France, April 20-22, 2004, Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 157–172. Springer, 2004.
27. P. Van Hentenryck, R. Bent, and Y. Vergados. Online stochastic reservation systems. In J. Christopher Beck and Barbara M. Smith, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Third International Conference, CPAIOR 2006, Cork, Ireland, May 31 - June 2, 2006, Proceedings*, volume 3990 of *Lecture Notes in Computer Science*, pages 212–227. Springer, 2006.
28. P. Van Hentenryck, L. Michel, L. Perron, and J.-C. Régin. Constraint programming in opl. In Gopalan Nadathur, editor, *Proceedings of the International Conference on Principles and Practice of Declarative Programming (PPDP'99)*, volume 1702 of *Lecture Notes in Computer Science*, pages 98–116, September 29 - October 1 1999.
29. N. M. Hewahi. Engineering industry controllers using neuroevolution. *AI EDAM*, 19(1):49–57, 2005.
30. B. Hnich, R. Rossi, S. A. Tarim, and S. D. Prestwich. Synthesizing filtering algorithms for global chance-constraints. In *Principles and Practice of Constraint Programming, CP 2009, Proceedings*, volume 5732 of *Lecture Notes in Computer Science*, pages 439–453. Springer, 2009.
31. H. Jeffreys. *Theory of Probability.* Clarendon Press, Oxford, UK, 1961.
32. P. Kall and S. W. Wallace. *Stochastic Programming.* John Wiley & Sons, 1994.
33. I. Katriel, C. Kenyon-Mathieu, and E. Upfal. Commitment under uncertainty: Two-stage stochastic matching problems. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 171–182. Springer, 2007.
34. A. J. Kleywegt, A. Shapiro, and T. Homem-De-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal of Optimization*, 12(2):479–502, 2001.
35. M. L. Littman, J. Goldsmith, and M. Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1–36, 1998.
36. M. L. Littman, S. M. Majercik, and T. Pitassi. Stochastic boolean satisfiability. *J. Autom. Reasoning*, 27(3):251–296, 2001.
37. B. Liu. Dependent-chance programming: A class of stochastic optimization. *Computers & Mathematics with Applications*, 34:89–104, 1997.
38. B. Liu and K. Iwamura. Modelling stochastic decision systems using dependent-chance programming. *European Journal of Operational Research*, 101:193–203, 1997.
39. M. Lombardi and M. Milano. Stochastic allocation and scheduling for conditional task graphs in mpsocs. In Frédéric Benhamou, editor, *Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings*, volume 4204 of *Lecture Notes in Computer Science*, pages 299–313. Springer, 2006.

40. M. Lombardi and M. Milano. Scheduling conditional task graphs. In Christian Bessiere, editor, *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, volume 4741 of *Lecture Notes in Computer Science*, pages 468–482. Springer, 2007.

41. S. M. Majercik. *Planning under uncertainty via stochastic satisfiability.* PhD thesis, Durham, NC, USA, 2000. Supervisor-Littman, Michael L.

42. S. M. Majercik. Appssat: Approximate probabilistic planning using stochastic satisfiability. *Int. J. Approx. Reasoning*, 45(2):402–419, 2007.

43. S. M. Majercik. *Stochastic Boolean Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 27, pages 887–925. IOS Press, February 2009.

44. S. M. Majercik and B. Boots. Dc-ssat: A divide-and-conquer approach to solving stochastic satisfiability problems efficiently. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 416–422. AAAI Press / The MIT Press, 2005.

45. S. M. Majercik and M. L. Littman. Maxplan: A new approach to probabilistic planning. In *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems, Pittsburgh Pennsylvania, USA*, pages 86–93. AAAI Press, 1998.

46. S. M. Majercik and M. L. Littman. Contingent planning under uncertainty via stochastic satisfiability. *Artif. Intell.*, 147(1-2):119–162, 2003.

47. L. Michel and P. Van Hentenryck. Iterative relaxations for iterative flattening in cumulative scheduling. In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004), June 3-7 2004, Whistler, British Columbia, Canada*, pages 200–208. AAAI, 2004.

48. S. Minton, M. D. Johnston, A. B. Philips, and P. Laird. Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artif. Intell.*, 58(1-3):161–205, 1992.

49. W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2007.

50. S. D. Prestwich, A. Tarim, R. Rossi, and B. Hnich. A cultural algorithm for pomdps from stochastic inventory control. In Maria J. Blesa, Christian Blum, Carlos Cotta, Antonio J. Fernández, José E. Gallardo, Andrea Roli, and Michael Sampels, editors, *Hybrid Metaheuristics, 5th International Workshop, HM 2008, Málaga, Spain, October 8-9, 2008. Proceedings*, volume 5296 of *Lecture Notes in Computer Science*, pages 16–28. Springer, 2008.

51. S. D. Prestwich, A. Tarim, R. Rossi, and B. Hnich. A steady-state genetic algorithm with resampling for noisy inventory control. In Günter Rudolph, Thomas Jansen, Simon M. Lucas, Carlo Poloni, and Nicola Beume, editors, *Parallel Problem Solving from Nature - PPSN X, 10th International Conference Dortmund, Germany, September 13-17, 2008, Proceedings*, volume 5199 of *Lecture Notes in Computer Science*, pages 559–568. Springer, 2008.

52. S. D. Prestwich, S. A. Tarim, and B. Hnich. Template design under demand uncertainty by integer linear local search. *International Journal of Production Research*, 44(22):4915–4928, 2006.

53. S. D. Prestwich, S. A. Tarim, R. Rossi, and B. Hnich. Evolving parameterised policies for stochastic constraint programming. In *Principles and Practice of Constraint Programming, CP 2009, Proceedings*, volume 5732 of *Lecture Notes in Computer Science*, pages 684–691. Springer, 2009.

54. S. D. Prestwich, S. A. Tarim, R. Rossi, and B. Hnich. Neuroevolutionary inventory control in multi-echelon systems. In *1st International Conference on Algorithmic Decision Theory*, volume 5783 of *Lecture Notes in Computer Science*, pages 402–413. Springer, 2009.

55. L. Proll and B. Smith. Integer linear programming and constraint programming approaches to a template design problem. *INFORMS J. on Computing*, 10(3):265–275, 1998.

56. R. Rossi, S. A. Tarim, B. Hnich, and S. D. Prestwich. Replenishment planning for stochastic inventory systems with shortage cost. In Pascal Van Hentenryck and Laurence A. Wolsey, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 4th International Conference, CPAIOR 2007, Brussels, Belgium, May 23-26, 2007, Proceedings*, volume 4510 of *Lecture Notes in Computer Science*, pages 229–243. Springer Verlag, 2007.

57. R. Rossi, S. A. Tarim, B. Hnich, and S. D. Prestwich. Cost-based domain filtering for stochastic constraint programming. In P. J. Stuckey, editor, *Principles and Practice of Constraint Programming, 14th International Conference, CP 2008, Sydney, Australia, September 14-18, 2008. Proceedings*, volume 5202 of *Lecture Notes in Computer Science*, pages 235–250. Springer, 2008.

58. R. Rossi, S. A. Tarim, B. Hnich, and S. D. Prestwich. A global chance-constraint for stochastic inventory systems under service level constraints. *Constraints*, 13(4):490–517, 2008.

59. R. Rossi, S. A. Tarim, B. Hnich, S. D. Prestwich, and Cahit Guran. A note on liu-iwamura's dependent-chance programming. *European Journal of Operational Research*, 198(3):983–986, 2009.

60. R. Rossi, S. A. Tarim, B. Hnich, S. D. Prestwich, and S. Karacaer. Scheduling internal audit activities: a stochastic combinatorial optimization problem. *Journal of Combinatorial Optimization*, 2008.

61. G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. Technical report, CUED/F-INFENG/TR 166, Cambridge University, 1994.

62. N. V. Sahinidis. Optimization under uncertainty: State-of-the-art and opportunities. *Computers and Chemical Engineering*, 28:971–983, 2004.

63. T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI '95*, pages 631–639. Morgan Kaufmann, 1995.

64. K. O. Stanley and R. Miikkulainen. Evolving neural network through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

65. M. L. Stein. Large sample properties of simulation using latin hypercube sampling. *Technometrics*, 29:143–151, 1987.

66. R. S. Sutton and A/ G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, March 1998.

67. S. A. Tarim, B. Hnich, S. D. Prestwich, and R. Rossi. Finding reliable solution: Event-driven probabilistic constraint programming. *Annals of Operations Research*, 171(1):77–99, 2008.

68. S. A. Tarim, B. Hnich, R. Rossi, and S. D. Prestwich. Cost-based filtering techniques for stochastic inventory control under service level constraints. *Constraints*, 14(2):137–176, 2009.

69. S. A. Tarim and B. G. Kingsman. The stochastic dynamic production/inventory lot-sizing problem with service-level constraints. *International Journal of Production Economics*, 88:105–119, 2004.

70. S. A. Tarim and B. G. Kingsman. Modelling and Computing (R^n,S^n) Policies for Inventory Systems with Non-Stationary Stochastic Demand. *European Journal of Operational Research*, 174:581–599, 2006.

71. S. A. Tarim, S. Manandhar, and T. Walsh. Stochastic constraint programming: A scenario-based approach. *Constraints*, 11(1):53–80, 2006.

72. S. A. Tarim and I. Miguel. A hybrid benders' decomposition method for solving stochastic constraint programs with linear recourse. In Brahim Hnich, Mats Carlsson, François Fages, and Francesca Rossi, editors, *Recent Advances in Constraints, Joint ERCIM/CoLogNET International Workshop on Constraint Solving and Constraint Logic Programming, CSCLP 2005, Uppsala, Sweden, June 20-22, 2005, Revised Selected and Invited Papers*, volume 3978 of *Lecture Notes in Computer Science*, pages 133–148. Springer, 2005.

73. S. A. Tarim and B. Smith. Constraint Programming for Computing Non-Stationary (R,S) Inventory Policies. *European Journal of Operational Research*, 189:1004–1021, 2008.

74. D. Terekhov and J. C. Beck. A constraint programming approach for solving a queueing control problem. *J. Artif. Intell. Res. (JAIR)*, 32:123–167, 2008.

75. D. Terekhov and J. Christopher Beck. Solving a stochastic queueing control problem with constraint programming. In Pascal Van Hentenryck and Laurence A. Wolsey, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 4th International Conference, CPAIOR 2007, Brussels, Belgium, May 23-26, 2007, Proceedings*, volume 4510 of *Lecture Notes in Computer Science*, pages 303–317. Springer, 2007.

76. G. Verfaillie and N. Jussien. Constraint solving in uncertain and dynamic environments: A survey. *Constraints*, 10(3):253–281, 2005.

77. T. Walsh. Sat v csp. In Rina Dechter, editor, *Principles and Practice of Constraint Programming, CP 2000, Proceedings*, volume 1894 of *Lecture Notes in Computer Science*, pages 441–456. Springer, 2000.

78. T. Walsh. Stochastic constraint programming. In Frank van Harmelen, editor, *European Conference on Artificial Intelligence, ECAI'2002, Proceedings*, pages 111–115. IOS Press, 2002.

79. Y. Zhuang and S. M. Majercik. Walkssat: An approach to solving large stochastic satisfiability problems with limited time. Technical report.

Constraint-based Local Search for Inventory Control under Stochastic Demand and Lead time

Roberto Rossi
Logistics, Decision and Information Sciences, Wageningen UR, the Netherlands, roberto.rossi@wur.nl

S. Armagan Tarim
Department of Management, Hacettepe University, Turkey, armtar@yahoo.com

Ramesh Bollapragada
Decision Sciences Department, College of Business, San Francisco State University, 1600 Holloway Avenue, San Francisco, California, USA, rameshb@sfsu.edu

In this paper, we address the general multi-period production/inventory problem with non-stationary stochastic demand and supplier lead time under service-level constraints. A replenishment cycle policy is modeled. We propose two hybrid algorithms that blend Constraint Programming and Local Search for computing near-optimal policy parameters. Both the algorithms rely on a coordinate descent Local Search strategy, what differs is the way this strategy interacts with the Constraint Programming solver. These two heuristics are, firstly, compared for small instances against an existing optimal solution method. Secondly, they are tested and compared with each other in terms of solution quality and run time on a set of larger instances that are intractable for the exact approach. Our numerical experiments show the effectiveness of our methods.

Key words: inventory control; demand uncertainty; supplier lead-time uncertainty; constraint-based local search; heuristic
History: submitted February 2010.

1. Introduction

Inventory theory provides methods for managing and controlling inventories under different constraints and environments. An interesting class of production/inventory control problems is the one that considers the single-location, single-product case under non-stationary stochastic demand and service level constraints. Such a problem has been widely studied because of its key role in practice.

Different inventory control policies can be adopted for the above mentioned problem. For a discussion of inventory control policies see [33]. One of the possible policies that can be adopted is the replenishment cycle policy, (R, S). A detailed discussion on the characteristics of (R,S) can be found in [12]. In this policy an order is placed every R periods to raise the inventory level to the order-up-to-level S. This provides an effective means of dampening planning instability (deviations in planned orders, also known as *nervousness* [13, 19]) and coping with demand uncertainty. As pointed out by Silver et al. ([33], pp. 236–237), (R,S) is particularly appealing when items are ordered from the same supplier or require resource sharing. In these cases all items in a coordinated group can be given the same replenishment period. Periodic review also allows a reasonable prediction of the level of the workload on the staff involved, and is particularly suitable for advanced planning environments and risk management [35]. For these reasons, as stated by Silver et al. [33], (R, S) is a popular inventory policy.

Under the non-stationary demand assumption this policy takes the form (R^n,S^n), where R^n denotes the length of the nth replenishment cycle, and S^n the order-up-to-level value for the nth replenishment. It should be noted that this inventory control policy yields at most $2N$ policy parameters fixed at the beginning of an N-period planning horizon, therefore it is particularly easy to be implemented.

Due to its combinatorial nature, the computation of optimal (R^n,S^n) policy parameters, even in the absence of stochastic lead time, presents a difficult problem to solve to optimality. An early work in this area, by Bookbinder and Tan [9], proposes a two-step heuristic method. Tarim and Kingsman [38, 39] and Tempelmeier [42] propose a mathematical programming approach to compute policy parameters. Tarim and Smith [41] give a computationally efficient Constraint Programming formulation. An exact formulation of the policy and a solution method are presented in Rossi et al. [30].

All the above mentioned research assumes either zero or a fixed (deterministic) supplier lead time (i.e., replenishment lead time). However, the lead time uncertainty, in various industries an inherent part of the business environment, is having a detrimental effect on inventory systems. For this reason, there is a vast inventory control literature analyzing the impact of supplier lead time uncertainty on the ordering policy (Whybark and Williams [43], Speh and Wagenheim [34], Nevison and Burstein [23]). A comprehensive work on stochastic supplier lead time in continuous-time inventory systems is presented in Zipkin [44]. Kaplan [21] characterizes the optimal policy for a dynamic inventory problem, where the lead time is a discrete random variable with known distribution and the demands in successive periods are assumed to form a stationary stochastic process. Since tracking all the outstanding orders through the use of Dynamic Programming requires a large multi-dimensional state vector, Kaplan assumes that orders do not cross in time and supplier lead time probabilities are independent of the size/number of outstanding orders (for details on order-crossover, see Hayya et al. [18]). The assumption that orders do not cross in time is valid for systems where supplier's production system has a single-server queue structure operating under a FIFO policy. Nevertheless, there are settings in which this assumption is not valid and orders do cross in time. This has been recently investigated in Hayya et al. [17], Riezebos [28], Bashyam and Fu [5]. In a recent work, Babaï et al. [3] analyze a dynamic re-order point control policy for a single-stage, single-item inventory system with non-stationary demand and lead time uncertainty. We argue that incorporating both a non-stationary stochastic demand and a stochastic supplier lead time in an optimization model that computes (R^n,S^n) policy parameters — without assuming that orders do not cross in time — is a relevant and novel contribution. To the best of our knowledge, the only existing work that addresses the computation of optimal (R^n,S^n) policy parameters under these assumptions is the one proposed in Rossi et al. [31]. Nevertheless, the approach proposed in [31] is only able to solve, in reasonable time, instances comprising a limited number of periods and a stochastic lead time that ranges over a small finite support.

In order to address this efficiency issue, in this paper we propose two heuristic techniques for computing (R^n,S^n) policy parameters under stochastic supplier lead time. We build on the work of Eppen and Martin [14], and by following a similar *scenario-based* approach (see also Birge and Louveaux [7]), we develop two *constraint-based local search* methods, based on a *coordinate descent* strategy, for finding near-optimal (R^n,S^n) policy parameters under non-stationary stochastic demand and supplier lead time (for a complete discussion on local search strategies in the literature refer to Focacci et al. [15], Nocedal and Wright [24]).

In the first part of this paper, we develop a technique that is analogous to a classical strategy of blending constraints with local search procedures (Backer et al. [4], Pesant and Gendreau [25]). In this approach, the local search engine is used to "guide" the search, while Constraint Programming is used to explore promising neighborhoods. In order to implement this strategy, we exploited Tarim and Smith's model [41] within a *coordinate descent* local search approach. In the second part of this work, we adopt an alternative strategy for integrating Constraint Programming and Local Search. In this strategy, Local Search techniques are introduced within a constructive global search algorithm (Cesta et al. [10]). In order to do so, we realized a deterministic equivalent modeling of the chance-constraints (Charnes and Cooper [11]) enforcing the required service level, by employing a scenario based approach, and once more a *coordinate descent* heuristic for propagating these constraints. In this second strategy, *cost-based filtering* (Focacci et al. [16]) is employed to speed up the search. The results obtained with these two heuristic approaches are compared, for small instances, with the optimal solution produced by the approach presented in Rossi et al. [31].

Experimental results show that the approach proposed in Section 6 typically performs better the one discussed in Section 5 in terms of solution quality. Nevertheless, the approach in Section 5 scales better and is faster than the approach in Section 6 in solving larger instances. Both the approaches run faster than the complete approach presented in Rossi et al. [31], which is able to solve only very small instances.

The paper is organized as follows. In Section 2, we introduce the problem and the assumptions adopted throughout the paper. In Section 3, we provide a Stochastic Programming formulation of the problem. In Section 4, we discuss a deterministic equivalent non-linear formulation of the Stochastic Programming model. In Section 5, we introduce a first heuristic solution method for the non-linear model discussed in Section 4. A second heuristic strategy is discussed in Section 6. Computational results are presented in Section 7. Summary and Conclusions are presented in Section 8.

2. Problem Definition

We discuss the general multi-period production/inventory control problem with non-stationary stochastic demand and lead time.

We consider a finite planning horizon of N periods and a demand d_t for each period $t \in \{1, \ldots, N\}$, which is a random variable with probability density function $g_t(d_t)$. The demand we consider is non-stationary, that is it can vary from period to period, and we also assume that demands in different periods are independent.

As in Eppen and Martin [14], an order placed in period $t \in \{1, ..., N\}$ is subject to a stochastic lead time l_t with probability mass function $f_t(\cdot)$. Note that $\{l_t\}$ are mutually independent and they are also independent of the respective order quantity. Since we consider a discrete stochastic lead time with probability mass function $f_t(\cdot)$ in each period $t = 1, \ldots, N$, this implies that an order placed in period t will be received exactly after k periods with probability $f_t(k)$. Since $f_t(k)$ is discrete, we assume that there is a maximum lead time L for which $\sum_{k=0}^{L} f_t(k) = 1$, $i = 1, \ldots, N$. The probability of observing any lead time length $p > L$ will always be 0. Therefore, the possible lead time lengths are limited to $\Lambda = \{0, \ldots, L\}$, and the probability mass function is defined on the finite set Λ.

A fixed delivery cost a is incurred for each order at the time such an order is placed. A linear holding cost h is incurred for each unit of product carried in inventory from one period to the next, as well as those that are part of an outstanding order. This reflects the fact that we charge interests not only on the actual amount of items we have in stock, but also on outstanding orders. Doing so often makes sense, since companies may assess holding cost on their total invested capital and not simply on items in stock — this cost accounting strategy has been observed during our collaboration with Alcatel-Lucent manufacturing divisions. A further detailed justification for this cost accounting strategy can be found in Hunt [20].

Demands not met are assumed to be back-ordered, and satisfied as soon as the next replenishment order arrives. We assume that it is not possible to sell back excess items to the vendor at the end of a period and that negative orders are not allowed. If the actual inventory exceeds the order-up-to-level for that review, this excess stock is carried forward and not returned to the supply source. However, such occurrences are regarded as rare events and accordingly the cost of carrying this excess stock and its effect on the service levels of subsequent periods are ignored. This assumption is consistent with previous works in the literature (Bookbinder and Tan [9], Tarim and Kingsman [38], Tarim and Smith [41], Tempelmeier [42] and Tarim et al. [37]). Furthermore, a computational study in Rossi et al. [30] showed that such an assumption does not significantly impact the quality of the optimal solution obtained.

As a service level constraint, we require the probability that at the end of every period the net inventory will be non-negative to be at least a given value α. This value is assumed to be set by the management to a reasonably high threshold, therefore we will not consider values of α that are less than 0.5 — in real applications, α takes greater values, i.e. 0.95. Our aim is to minimize the expected total cost, which is composed of ordering costs and inventory holding costs, over the N-period planning horizon, satisfying the service level constraints.

The actual sequence of ordering and delivery is similar to the one described in Kaplan [21]. We adopt the same sequence of actions described in his paper, since it handles all the deliveries symmetrically, and allows for some delay in the arrival deliveries at the beginning of a period. The sequence is therefore as follows. At the beginning of a period, the inventory on-hand after the realization of demands from the previous periods is known. Since we are assuming complete backlogging, this quantity may be negative. We also know the orders placed in previous periods, that have not been delivered yet. On the basis of this information, an ordering decision is made for the current period. All deliveries made during a period are assumed to arrive immediately after this ordering decision, and hence are on hand at the beginning of the period. A further discussion that states the convenience of this sequence of events can be found in Kaplan [21]. To summarize there are three successive events at the beginning of each period. First, the inventory on-hand and outstanding orders are determined. Second, an ordering decision is made on the basis of this information. Third, all supplier deliveries for the current period, including the most recent orders, are received.

3. Stochastic Programming Formulation

A stochastic programming formulation for the problem discussed in the previous section is given below,

$$\begin{aligned}
&\min \;\; E\{TC\} = \\
&\int_{d_1} \cdots \int_{d_N} \sum_{t=1}^{N} \left[a \cdot \delta_t + h \cdot \max\left(\sum_{k=1}^{t} (X_k - d_k), 0 \right) \right] \qquad (1)\\
&g_1(d_1) g_2(d_2) \ldots g_N(d_N) \mathrm{d}(d_1) \mathrm{d}(d_2) \ldots \mathrm{d}(d_N)
\end{aligned}$$

subject to,

$$I_t = I_0 + \sum_{\{k | k \geq 1, l_k \leq t-k\}} X_k - \sum_{k=1}^{t} d_k \qquad t = 1, \ldots, N \qquad (2)$$

$$\delta_t = \begin{cases} 1, & \text{if } \; X_t > 0 \\ 0, & \text{otherwise} \end{cases} \qquad t = 1, \ldots, N \qquad (3)$$

$$\Pr\{I_t \geq 0\} \geq \alpha \qquad t = L+1, \ldots, N \qquad (4)$$

$$I_t \in \mathbb{R}, \quad X_t \geq 0, \qquad t = 1, \ldots, N. \qquad (5)$$

where

$E\{.\}$: the expectation operator,

TC : total cost,

d_t : the demand in period t, a random variable with probability density function, $g_t(d_t)$,

a : the fixed ordering cost (incurred when an order is placed),

h : the proportional inventory holding cost,

l_t : the lead time length of the order placed in period t, a discrete random variable with probability mass function $f_t(\cdot)$.

δ_t : a {0,1} variable that takes the value of 1 if a replenishment occurs in period t and 0 otherwise,

I_t : the inventory level (stock on hand minus back-orders) at the end of period t,

I_0 : the initial inventory,

X_t : the size of the replenishment order placed in period t, $X_t \geq 0$, (received in period $t + L$).

In this model, the objective function (Eq. 1) minimizes the expected total cost, which is comprised of ordering costs and inventory holding costs. As discussed earlier, the latter are charged in each period on delivered and outstanding orders, for this reason the stochastic lead time does not play a direct role in the objective function. Eq. 2 represents the inventory balance constraint, which states that the inventory level at period t, I_t, is the sum of the initial inventory, I_0, and of all the subsequent order quantities that are delivered by period t, $\sum_{\{k|k\geq 1, l_k \leq t-k\}} X_k$, minus the cumulative demand up to period t, $\sum_{k=1}^{t} d_k$. Eq. 3 states that if a replenishment occurs in period t — i.e. the order quantity X_t is greater than 0 — then the corresponding indicator variable δ_t must take a value of 1. Eq. 4 enforces the required service level constraint in each period. That is, the probability inventory level at the end of each period is positive, must be greater or equal to the threshold α. Finally, the inventory levels, I_t, are real valued decision variables and the order quantities, X_t, must be positive. Note that, depending on the lead time probability mass functions, it may not be possible to provide the required service level for some initial periods. In general, reasoning in a worst case scenario, it is always possible to provide the required service level α starting from period $L+1$. For this reason the service level constraints are only enforced over periods $L+1, \ldots, N$.

4. Deterministic Equivalent Modeling

In order to solve the above model, it is necessary to reformulate the service level constraints in Eq. 4, in terms of deterministic equivalent expressions. To do so, we blend a scenario based approach — since the lead time probability distribution is assumed to be discrete — with a strategy similar to Bookbinder and Tan's "static-dynamic" uncertainty [9].

To begin, we discuss how to obtain a deterministic equivalent formulation for the chance-constraints that enforce the required service level when the lead time in each period varies and assumes a given deterministic value. Subsequently, we will generalize the same reasoning to the case in which the lead time is stochastic and assumes a different distribution from period to period.

When a dynamic deterministic lead time $L_t \geq 0$ is considered in each period $t = 1, \ldots, N$, an order placed in period t will be received only at period $t + L_t$. Eq. 2 therefore becomes,

$$I_t = I_0 + \sum_{\{k|k\geq 1, L_k+k\leq t\}} X_k - \sum_{k=1}^{t} d_k \qquad t = 1, ..., N. \tag{6}$$

Let us denote the inventory position (the total amount of inventory on-hand plus outstanding orders minus backorders) at the end of period t as P_t. It directly follows that

$$P_t = I_t + \sum_{\{k|1\leq k\leq t, L_k+k>t\}} X_k, \tag{7}$$

where we assume $P_0 = I_0$. It is easy, then, to reformulate the model using the inventory position. Furthermore, consider the expectation operator $E\{\cdot\}$, and since the demands $\{d_t\}$ are assumed to be mutually independent, we may rewrite the objective function as

$$\min \quad E\{TC\} = \sum_{t=1}^{N} \left(h \cdot E\left\{\max(P_t, 0)\right\} + a \cdot \delta_t\right). \tag{8}$$

When a stock-out occurs, all demand is back-ordered and fulfilled as soon as an adequate supply arrives. Following Bookbinder and Tan [9], since we have assumed that the management will set the non-stockout probability to a reasonably high level — certainly greater than 0.5 — we can safely replace the term $E\{\max(P_t, 0)\}$ with the term $E\{P_t\}$.

The general stochastic programming formulation can then be modified to incorporate the "replenishment cycle policy". Consider a review schedule, which has m reviews over the N period planning horizon with orders placed at $T_1, T_2, \ldots, T_m$, where $T_i > T_{i-1}$, $T_m \leq N - L_{T_m}$. For convenience, T_1 is defined as the start of the planning horizon and $T_{m+1} = N + 1$ as the period immediately after the end of the planning horizon.[1] The associated inventory reviews

[1] The review schedule may be generalized to consider the case where $T_1 > 1$, if the opening inventory I_0 is sufficient to cover the immediate needs at the start of the planning horizon.

will take place at the beginning of periods T_i, $i = 1, \ldots, m$. In the replenishment cycle policy considered here, clearly the orders X_i are all equal to zero except at replenishment periods $T_1, T_2, \ldots, T_m$. The inventory level I_t carried from period t to period $t+1$ is the opening inventory plus any orders that have arrived up to and including period t less the total demand to date. Hence, the inventory balance equation becomes,

$$I_t = I_0 + \sum_{\{i|L_{T_i}+T_i \leq t\}} X_{T_i} - \sum_{k=1}^{t} d_k, \quad t = 1, \ldots, N. \tag{9}$$

Define $T_{p(t)}$ as the latest review before period t in the planning horizon, for which all the former orders, including the one placed in $T_{p(t)}$, are delivered within period t. Therefore,

$$p(t) = \max\left\{i|\forall j \in \{1, \ldots, i\}, T_j + L_{T_j} \leq t, \quad i = 1, \ldots, m\right\}. \tag{10}$$

The inventory level I_t at the end of period t (Eq. 9) can be expressed as

$$I_t = I_0 + \sum_{i=1}^{p(t)} X_{T_i} + \sum_{\{i|i>p(t), L_{T_i}+T_i \leq t\}} X_{T_i} - \sum_{k=1}^{t} d_k, \quad t = 1, \ldots, N. \tag{11}$$

We now want to reformulate the constraints of the chance-constrained model in terms of a new set of decision variables R_{T_i}, $i = 1, \ldots, m$.

Define,

$$P_t = R_{T_i} - \sum_{k=T_i}^{t} d_k, \quad T_i \leq t < T_{i+1}, \quad i = 1, \ldots, m \tag{12}$$

where R_{T_i} can be interpreted as the inventory position up to which inventory should be raised after placing an order at the ith review period T_i. We now express Eq. 11 using R_{T_i} as decision variables

$$I_t = R_{T_{p(t)}} + \sum_{\{i|i>p(t), L_{T_i}+T_i \leq t\}} \left(R_{T_i} - R_{T_{i-1}} + d_{T_{i-1}} + \ldots + d_{T_i - 1}\right) - \sum_{k=T_{p(t)}}^{t} d_k, \quad t = 1, \ldots, N. \tag{13}$$

As mentioned earlier, α is the desired minimum probability that the net inventory level in any time period is non-negative. Depending on the values assigned to L_t, it may not be possible to provide the required service level for some initial periods. In general, we provide the required service level α starting from the period t, for which the value $t + L_t$ is minimum. Let M be this period. Note that, it will never be optimal to place any order in a period t

such that $t + L_t > N$, since such an order will not be received within the given planning horizon.

By substituting I_t with the right hand term in Eq. 13 we obtain

$$G_S\left(R_{T_{p(t)}} + \sum_{\{i|i>p(t),L_{T_i}+T_i\leq t\}} (R_{T_i} - R_{T_{i-1}})\right) \geq \alpha, \qquad t = M, \ldots, N. \tag{14}$$

where $S = \sum_{k=T_{p(t)}}^{t} d_k - \sum_{\{i|i>p(t),L_{T_i}+T_i\leq t\}}(d_{T_{i-1}} + \ldots + d_{T_i-1})$, and $G_S(.)$ is the cumulative distribution function of S. The service level constraints are now deterministic and are expressed only in terms of the order-up-to-positions.

It is now relatively easy to obtain a deterministic equivalent model in which lead times are stochastic, under the original assumption that the lead time in each period is a discrete random variable l_t.

We first reformulate the chance-constrained model under stochastic lead time using the inventory position,

$$\min \;\; E\{TC\} = \int_{d_1} \ldots \int_{d_N} \sum_{t=1}^{N} (a\delta_t + h \cdot P_t) \tag{15}$$

$$g_1(d_1)g_2(d_2)\ldots g_N(d_N)\mathrm{d}(d_1)\mathrm{d}(d_2)\ldots\mathrm{d}(d_N)$$

subject to,

$$\delta_t = \begin{cases} 1, & \text{if} \quad X_t > 0 \\ 0, & \text{otherwise} \end{cases} \qquad t = 1, ..., N \tag{16}$$

$$P_t = I_0 + \sum_{k=1}^{t}(X_k - d_k) \qquad t = 1, ..., N \tag{17}$$

$$\Pr\{P_t \geq \sum_{\{k|1\leq k\leq t, l_k>t-k\}} X_k\} \geq \alpha \qquad t = L+1, ..., N \tag{18}$$

$$P_t \geq 0, \quad X_t \geq 0, \qquad t = 1, ..., N. \tag{19}$$

Also in this case, we want to adopt a replenishment cycle policy and express the whole model in terms of the new set of variables R_{T_i}, so that order quantities are decided only after the demand in the former periods is realized.

Similar to the dynamic deterministic lead time case, we now express the service level constraint as a relation between the opening-inventory-positions, such that the overall service level provided at the end of each period is at least α. In order to express this service level constraint, we propose a scenario based approach over the discrete random variables $\{l_t\}$. In a scenario based approach (see also Birge and Louveaux [7] and Tarim et al. [40]), a scenario tree is generated which incorporates all possible realizations of discrete random variables into the model explicitly, yielding a fully deterministic model under the non-anticipativity constraints.

In our problem, we can divide random variables into two sets: the random variables $\{l_t\}$ which represent lead times and the random variables $\{d_t\}$ which represent demands. We deal with each set in a separate fashion, by employing a scenario based approach for the $\{l_t\}$ and a deterministic equivalent modeling approach for the $\{d_t\}$. This is possible since under a given scenario discrete random variables are treated as constants. The problem is then reduced to the general multi-period production/inventory problem with dynamic deterministic lead time and stochastic demands, which has been previously analyzed.

Consider a review schedule, which has m reviews over the N period planning horizon with orders placed at $T_1, T_2, \ldots, T_m$. A scenario ω_t is a possible lead time realization for all the orders placed up to period t in a given review schedule. Let $(l_{T_i}|\omega_t)$ be the realized lead time in scenario ω_t for the order placed in period T_i. Finally, let Ω_t be the set of all the possible scenarios ω_t.

Under a given scenario ω_t, the service level constraint for a period t can be easily expressed using Eq. 14. It follows that the service level constraint is always a relation between at most $L+1$ decision variables R_{T_i} that represent the order-up-to-positions of the replenishment cycles covering the span $t-L, \ldots, t$. Let $p_\omega(t)$ be the value of $p(t)$ under a given scenario ω_t, when a review schedule Z is considered. In order to satisfy the service level constraints in our original model, we require that the overall service level under all possible scenarios, for each set of at most $L+1$ decision variables, is at least α. Equivalently, by using Eq. 14,

$$\sum_{\omega_t \in \Omega_t} \Pr\{\omega_t\} \cdot G_S \left(R_{T_{p_\omega(t)}} + \sum_{\{i|i>p_\omega(t),(l_{T_i}|\omega_t) \leq t-T_i\}} (R_{T_i} - R_{T_{i-1}}) \right) \geq \alpha, \qquad t = L+1, \ldots, N, \tag{20}$$

where $S = \sum_{k=T_{p_\omega(t)}}^{t} d_k - \sum_{\{i|i>p_\omega(t),(l_{T_i}|\omega_t)\leq t-T_i\}} (d_{T_{i-1}} + \ldots + d_{T_i-1})$. It should be noted that this equation is non-linear. In the remainder of the paper, we refer to Eq. 20 as “service level constraints” or “SL Constraints”, as this equation is refered most commonly throughtout.

In our chance-constrained model, we can now replace the original service level constraints with the new formulation in Eq. 20. As a consequence, the service level constraints are now expressed only in terms of the order-up-to-levels. Therefore, the expectation operator can be safely applied to the closing-inventory-levels, $\{P_t\}$, and to the stochastic demands, $\{d_t\}$, since these variables are only affecting the objective function in which we are minimizing an expected value. In what follows, the expected value of P_t and d_t are denoted by $\tilde{P}_t$ and $\tilde{d}_t$, respectively.

We can now express the whole model in terms of a new set of decision variables R_t, $t = 1, \ldots, N$. If there is no replenishment scheduled for period t, that is if $\delta_t = 0$, then R_t must be equal to the expected closing-inventory-position in period $t-1$, that is $R_t = \tilde{P}_{t-1}$. If there is a review T_i in period t, R_t is equal to the order-up-to-position R_{T_i} for this review. Therefore, the desired order-up-to-positions, $\{R_{T_i}\}$, as required for the solution to the problem, are those values of R_t, for which $\delta_t = 1$.

The complete model under the replenishment cycle policy is then:

$$\min \;\; E\{TC\} = \sum_{t=1}^{N} \Big(h \cdot \tilde{P}_t + a \cdot \delta_t \Big) \tag{21}$$

subject to,

Eq. 20 (SL Constraints)

$$R_t > \tilde{P}_{t-1} \Rightarrow \delta_t = 1 \qquad t = 1, \ldots, N \tag{22}$$

$$R_t \geq \tilde{P}_{t-1} \qquad t = 1, \ldots, N \tag{23}$$

$$R_t = \tilde{P}_t + \tilde{d}_t \qquad t = 1, \ldots, N \tag{24}$$

$$R_t \geq 0, \quad \tilde{P}_t \geq 0, \quad \delta_t \in \{0, 1\} \qquad t = 1, \ldots, N, \tag{25}$$

where $\{T_1, \ldots, T_m\} = \{t \in \{1, \ldots, N\} | \delta_t = 1\}$.

It should be noted that the domain of each $\tilde{P}_t$ variable — as in the zero lead time case (see Tarim and Smith [41]) — is limited. In fact, since the period demand variance is additive, the uncertainty can only increase in the length of a replenishment cycle. Therefore the longer a cycle is, the higher the inventory levels that are required to achieve a certain service level. It directly follows that a single replenishment covering the whole planning horizon will provide upper bounds for the expected period closing-inventory-positions throughout the horizon.

4.1. An example

We assume an initial null inventory level and a normally distributed demand with a coefficient of variation $\sigma_t/\tilde{d}_t = 0.3$ for each period $t \in \{1, \ldots, 5\}$. The expected values for the demand in each period are: {36, 28, 42, 33, 30}. The other parameters are $a = 1$, $h = 1$, $\alpha =$

Policy cost: 356					
Period (t)	1	2	3	4	5
$\tilde{d}_t$	36	28	42	33	30
R_t	125	124	129	87	55
δ_t	1	1	1	1	1
Shortage probability	–	–	5%	5%	5%

Table 1: A replenishment plan.

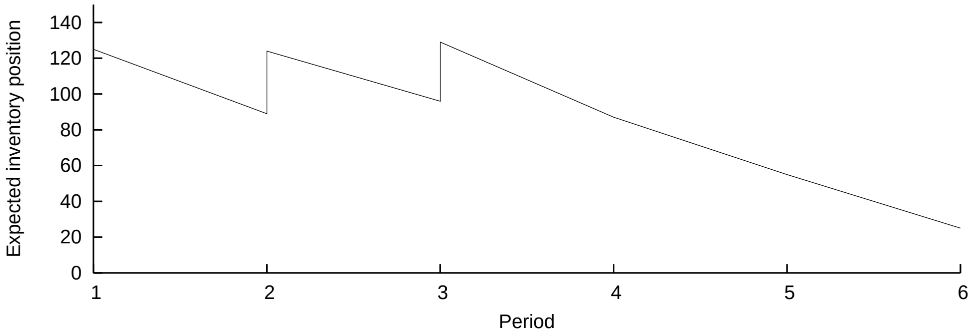


Figure 1: A graphical illustration of the replenishment plan in Table 1.

$0.95(z_{\alpha=0.95} = 1.645)$. We consider that every period i in the planning horizon has following lead time probability mass function $f_t(k) = \{0.3(0), 0.2(1), 0.5(2)\}$. This implies that we receive an order placed in period i after $t \in \{0, \ldots, 2\}$ periods with a given probability (0 periods: 30%; 1 period: 20%; 2 periods: 50%). It is obvious that in this case, we will always receive the order within 2 periods, after it is placed. In Table 1, we show the optimal solution. The optimal replenishment plan is also illustrated in Fig. 1. We now show, through Eq. 20 (SL Constraints), that the order-up-to-positions in this example satisfy every service level constraint in the model. We assume that for the first 2 periods, no service level constraint is enforced, since it is not possible to control the inventory in the first 2 periods. Therefore, we enforce the required service level on period 3, 4 and 5 (that is Eq. 20 or SL Constraints) for $t = 3, \ldots, N$. Let us verify that the given order-up-to-levels satisfy this condition for these three periods. Since we know the probability mass function $f_t(\cdot)$ for each period t in the planning horizon, we can compute the probability $Pr(\omega_t)$ for each scenario $\omega_t \in \Omega_t$. We thus have four of these scenarios for each period $t \in \{3, \ldots, N\}$, as we are placing an order in each period.

- S_1, $Pr\{S_1\} = 0.15$; in this scenario for period t, we receive all the former orders

- S_2, $Pr\{S_2\} = 0.35$; in this scenario for period t, we do not receive the last order placed in period t
- S_3, $Pr\{S_3\} = 0.35$; in this scenario for period t, we do not receive the last two orders placed in period t and $t-1$
- S_4, $Pr\{S_4\} = 0.15$; in this scenario for period t, we do not receive the order placed in period $t-1$, and we observe order-crossover.

In the described scenarios, every possible configuration is considered, without loss of generality. In fact, if some of the configurations are unrealistic (for instance, if we assume that order-crossover may not take place) we just need to set the probability of the respective scenario to zero. Now, it is possible to write SL Constraints (Eq. 20) for each period $t \in \{3, \dots, N\}$. For period 3,

$$\begin{aligned} Pr\{S_1\} \cdot G\left(\frac{129-42}{0.3\sqrt{42^2}}\right) + Pr\{S_2\} \cdot G\left(\frac{124-(28+42)}{0.3\sqrt{28^2+42^2}}\right) + \\ Pr\{S_3\} \cdot G\left(\frac{125-(36+28+42)}{0.3\sqrt{36^2+28^2+42^2}}\right) + \\ Pr\{S_4\} \cdot G\left(\frac{125+(129-124)-(36+42)}{0.3\sqrt{36^2+42^2}}\right) = 94.60 \cong 95, \end{aligned} \tag{26}$$

where $G(\cdot)$ is the standard normal distribution function with zero mean and unit standard deviation. This implies that the combined effect of order delivery delays in our policy, under any possible scenario results in a stock-out probability of exactly 95% for period 3. A similar reasoning can be applied to verify that the given solution satisfies the required service level for period 4 and 5.

5. Heuristic Method I

In this section, we introduce a first heuristic method, named Heuristic I or, shortly, H1, for computing near optimal replenishment cycle policy parameters under non-stationary stochastic demand and lead time. The key intuition behind this heuristic strategy consists in noticing that when **all the replenishment decisions** have been fixed, then SL Constraints (Eq. 20) can be used to check if in a given period the required service level constraint is met. If the service level is not met, the gradient function is able to indicate which order-up-to-position should be increased in order to achieve the maximum service level improvement for that period. This method employs one of the efficient techniques proposed in the literature such as the one in Tarim and Smith [41].

$$\min \; E\{TC\} = \sum_{t=1}^{N} \left(a\delta_t + h\tilde{I}_t \right) \tag{27}$$

subject to, for $t = 1 \dots N$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} \geq 0 \tag{28}$$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} > 0 \Rightarrow \delta_t = 1 \tag{29}$$

$$\tilde{I}_t \geq b\left(\max_{j \in \{1,\dots,t\}} j \cdot \delta_j, t \right) \tag{30}$$

$$\tilde{I}_t \in \mathbb{Z}^+ \cup \{0\}, \quad \delta_t \in \{0,1\}, \tag{31}$$

where $b(i,j) = G^{-1}_{d_i+d_{i+1}+\dots+d_j}(\alpha) - \sum_{k=i}^{j} \tilde{d}_k$, and $G^{-1}_{d_i+d_{i+1}+\dots+d_j}(\cdot)$ denotes the inverse cumulative distribution function of $d_i + d_{i+1} + \dots + d_j$.

In Tarim and Smith's model (Eqs. 27–31), each decision variable $\tilde{I}_t$, represents the expected inventory level at the end of period t. Each $\tilde{d}_t$ represents the expected value of the demand in a given period t, according to its probability density function $g_t(d_t)$. The binary decision variables δ_t state whether a replenishment is fixed for period t ($\delta_t = 1$) or not ($\delta_t = 0$). The objective function (27) minimizes the expected total cost over the given planning horizon. The two terms that contribute to the expected total cost are ordering costs and inventory holding costs. Constraint (28) enforces a no-buy-back condition, which means that received goods cannot be returned to the supplier. As a consequence of this, the expected inventory level at the end of period t must be no less than the expected inventory level at the end of period $t-1$ minus the expected demand in period t. Constraint (29) expresses the replenishment condition. We have a replenishment if the expected inventory level at the end of period t is greater than the expected inventory level at the end of period $t-1$ minus the expected demand in period t. This means that we receive some extra goods as a consequence of an order. Constraint (30) enforces the required service level α. This is done by specifying the minimum expected closing-inventory-level (or "buffer stock") required for each period t in order to assure that, at the end of each time period, the probability that the net inventory is not negative is at least α. These buffer stocks, which are stored in matrix $b(\cdot,\cdot)$, are pre-computed following the approach suggested in Tarim and Kingsman [38].

The buffer stocks mentioned in the previous paragraph refer to the case in which no lead time is considered and where every order is delivered immediately. These buffer stocks are typically lower than those required to provide the required service level α, when a stochastic delivery lead time is considered. Our strategy, in this first heuristic, consists in iteratively adjusting the buffer stocks in the matrix in order to increase the service level in each period till the required service level α is met in every period of the planning horizon. The local search procedure to compute policy parameters under stochastic lead time is shown in Algorithm 1.

Algorithm 1: `Heuristic Method I`

```
    input  : $d_1, \ldots, d_N$; $a$; $h$; $\alpha$
    output: a replenishment plan
1  begin
2      for each period $i$ in $1, \ldots, N$ do
3          for each period $j$ in $i, \ldots, N$ do
4              $b(i,j) \leftarrow G^{-1}_{d_i+d_{i+1}+\ldots+d_j}(\alpha) - \sum_{k=i}^{j} \tilde{d}_k$
5      Solve the model in Eqs. 27–31 with input $d_1, \ldots, d_N$, $a$, $h$, $\alpha$, and buffer matrix
       $b(\cdot,\cdot)$;
6      By using SL Constraints (Eq. 20), check if the solution found provides the
       required service level $\alpha$ at the end of each period;
7      while the current solution does not provide service level $\alpha$ do
8          Let $\mathcal{R}$ be the set of consecutive replenishment cycles in the solution;
9          for each replenishment cycle $\mathcal{R}(i,j)$ in $\mathcal{R}$ do
10             for each period $t$ in $\mathcal{R}(i,j)$ do
11                 Let $\mathcal{P}$ be the set of former cycles influencing the service level in period $t$
                   according to Eq. 20;
12                 while the service level in period $t$ is less than $\alpha$ do
13                     For each cycle $\mathcal{R}(m,n) \in \mathcal{P} \bigcup \{\mathcal{R}(i,j)\}$ obtain the respective
                       minimum allowed order-up-to-position $\underline{R}_m = \tilde{I}_n + \sum_{i=m}^{n} \tilde{d}_i$;
14                     Let $\mathcal{R}(m,n) \in \mathcal{P} \bigcup \{\mathcal{R}(i,j)\}$ be the cycle for which a unit increment
                       in $\underline{R}_m$ produces the highest service level improvement;
15                     $b(m,n) \leftarrow \tilde{I}_n + 1$;
16                     $\tilde{I}_n \leftarrow \tilde{I}_n + 1$;
17         Solve the model in Eqs. 27–31 with input $d_1, \ldots, d_N$, $a$, $h$, $\alpha$, and modified
           buffer matrix $b(\cdot,\cdot)$;
18         By using SL Constraints (Eq. 20), check if the solution found provides the
           required service level $\alpha$ at the end of each period;
19     return the current replenishment plan;
20 end
```

The method initially solves the model in Eqs. 27–31 (Algorithm 1, line 5), with buffer

Period (t)	1	2	3	4	5
$\tilde{d}_t$	36	28	42	33	30
R_t	54	42	63	49	45
δ_t	1	1	1	1	1
Shortage probability	–	–	78%	78%	77%

Table 2: No lead time solution.

stocks set as in the no lead time case. This gives a replenishment plan, that is assignments for decision variables δ_t and a (possibly) infeasible set of assignments for the respective order-up-to-positions. A replenishment plan is made of a number of replenishment cycles. A replenishment cycle $\mathcal{R}(i, j)$ is the set of periods that are located between two consecutive replenishment periods i and $j + 1$. If this assignment is infeasible (Algorithm 1, line 6) with respect to the lead time and the service level considered, using SL Constraints (Eq. 20), we consider sequentially (Algorithm 1, line 10) each period in every replenishment cycle scheduled and we increase the buffer stocks (Algorithm 1, line 15) of replenishment cycles affecting the service level in the current period (Algorithm 1, line 11), according to SL Constraints, until the required service level α is met for that specific period (Algorithm 1, line 12). Buffer stocks are increased according to a rule that increments at each step the buffer stock that produces the highest service level improvement for the period considered (Algorithm 1, line 14). This process is iterated by solving again the model in Eqs. 27–31 using this modified buffer stock matrix (Algorithm 1, line 17), until the model directly produces a feasible solution (Algorithm 1, line 18). We now provide a simple example to illustrate this procedure.

We present the same example proposed in Section 4.1. By disregarding the information on the stochastic lead time, we use the relevant data in the model shown in Eqs. 27–31. By solving this model, we obtain the solution shown in Table 2. Considering the stochastic lead time and by using SL Constraints (Eq. 20), it is possible to compute the service level provided in period 3, 4 and 5 for this solution (These service levels are also shown in Table 2). Clearly, these service levels are not higher than the required minimum service level α. Therefore, using SL Constraints we modify the buffer stock matrix $b(\cdot, \cdot)$ in such a way as to increase the relevant buffers and thus obtain a feasible solution. For instance, we increase the buffer stock level $b(1, 1)$ of replenishment cycle $\mathcal{R}(1, 1)$ from 18 to 102, the buffer stock level $b(2, 2)$ of replenishment cycle $\mathcal{R}(2, 2)$ from 14 to 106, the buffer stock level $b(3, 3)$ of

Period (t)	1	2	3	4	5
$\tilde{d}_t$	36	28	42	33	30
R_t	138	102	74	83	50
δ_t	1	0	0	1	1
Shortage probability	–	–	5%	35%	48%

Table 3: Solution with increased buffers.

Policy cost: 397					
Period (t)	1	2	3	4	5
$\tilde{d}_t$	36	28	42	33	30
R_t	138	134	136	94	61
δ_t	1	1	1	0	0
Shortage probability	–	–	2%	2%	5%

Table 4: Feasible solution.

replenishment cycle $\mathcal{R}(3,3)$ from 21 to 94, and the buffer stock level $b(4,4)$ of replenishment cycle $\mathcal{R}(4,4)$ from 16 to 50. This is based on the greedy rule discussed in Algorithm 1, and further based on the SL Constraints. We solve the model again using the modified buffer stock matrix $b(\cdot,\cdot)$. Since some of the buffers are increased, it is not anymore optimal to schedule a replenishment in each period, as noticed in the solution obtained using this new modified buffer stock matrix (Table 3). The service level provided in period 3 is now sufficiently high, but those provided in period 4 and 5 is not. We shall therefore iterate the process until we eventually converge to the feasible solution presented in Table 4. This heuristic is about 11% more costly than the optimal replenishment strategy.

6. Heuristic method II

The Heuristic Method I presented in the former section typically converges to a good solution in a few iterations, but often it may not produce solutions that are sufficiently close to the optimal. In order to produce higher quality solutions, we discuss here a different strategy that employs a Constraint Based Local Search approach. We name this second heuristic Heuristic II or H2.

6.1. Constraint Reasoning

Constraint Programming (see Apt [2]) is a declarative programming paradigm in which relations between decision variables are stated in the form of constraints. Informally speaking, constraints specify the properties of a solution to be found. The constraints used in constraint programming are of various kinds: logic constraints (i.e. "x or y is true", where x and y are boolean decision variables), linear constraints, and *global constraints* (Régin [27]). A global constraint captures a relation among a non-fixed number of variables. One of the most well known global constraints is the **alldiff** constraint (Régin [26]), that can be enforced on a certain set of decision variables in order to guarantee that no two variables are assigned the same value.

With each constraint, CP associates a *filtering algorithm* able to remove provably infeasible or suboptimal values from the domains of the decision variables that are constrained and, therefore, to enforce some degree of *consistency* (see Rossi et al. [29]). These filtering algorithms are repeatedly called until no more values are pruned. This process is called *constraint propagation*.

In addition to constraints and filtering algorithms, constraint solvers also feature some sort of *heuristic search engine* (e.g. a backtracking algorithm). During the search, the constraint solver exploits filtering algorithms in order to proactively prune part of the search space that cannot lead to a feasible or to an optimal solution.

6.2. Local Search

A *neighborhood structure* is a function $\mathcal{N} : \mathcal{S} \to 2^{\mathcal{S}}$ that assigns to every solution $s \in \mathcal{S}$, a set of neighbors $\mathcal{N}(s) \subseteq \mathcal{S}$. $\mathcal{N}(s)$ called the neighborhood of s. Without loss of generality, we here restrict the discussion to minimization problems. A *locally minimal solution* (or local minimum) with respect to a neighborhood structure $\mathcal{N}$ is a solution $\widehat{s}$ such that $\forall s \in \mathcal{N}(\widehat{s}) : f(\widehat{s}) \leq f(s)$. We call $\widehat{s}$ a strict local minimal solution if $\forall s \in \mathcal{N}(\widehat{s}) : f(\widehat{s}) < f(s)$. Local search (LS) algorithms for COPs start from some initial solution and iteratively try to replace the current solution by a better solution in an appropriately defined neighborhood of the current solution. In this process, it is extremely important to achieve a proper balance between *diversification* and *intensification* of the search. The term diversification generally refers to the exploration of the search space, whereas the term intensification refers to the exploitation of the accumulated search experience. Among the most popular local search strategies we recall the *Iterative Improvement*, or *Hill Climbing*, in which each move is only performed if the resulting solution is better than the current solution and the algorithm stops as soon as it finds a local minimum. *Tabu Search* is a more advanced strategy, in fact it is among the most cited and used. Tabu search explicitly uses the history of the search, both to escape from local minima and to implement an explorative strategy. *Iterated Local Search* and *Variable Neighborhood Search* constitute other examples of local search strategies. For a comprehensive survey on local search and metaheuristic strategies, the reader may refer to Blum and Roli [8]. In what follows, we will employ a strategy known as *coordinate descent*. Coordinate descent algorithms, sometimes called one-at-a-time, minimize (maximize) a given function by minimizing (maximizing) it over a single variable while holding all other variables constant. There are two approaches: cyclic algorithms that cycle through all of the variables; and greedy algorithms that choose the variable that reduces the cost by the largest amount in each iteration. In the coordinate descent strategy discussed in the next section, an algorithm belonging to this second "greedy" class will be employed.

6.3. The Approach

The key intuition behind the second heuristic strategy slightly differs from that of the first heuristic. In this heuristic, we emphasize that when **some replenishment decisions** have been fixed, SL Constraints (Eq. 20) can be used to check if in a given period, the required service level constraint is met. If the service level is not met, a gradient function indicates which order-up-to-position should be increased in order to achieve the maximum service level improvement in such a period. We shall now provide a simple example to clarify how this procedure works.

We consider again the example proposed in Section 4.1. In Table 5, we show a possible partial replenishment plan that schedules orders in period 1, 2, and 3; the remaining replenishment decisions are not yet fixed. Clearly, the order-up-to-level in each period $t = 1, \dots, 3$ will be at least as high as those required to provide the service level α, when the lead time is 0. Therefore, a good starting configuration for the order-up-to-level is 54, 42, and 63 respectively for R_1, R_2, and R_3. As it is easy to observe using SL Constraints, it is now possible to compute the service level provided in period 3.

Period (t)	1	2	3	4	5
$\tilde{d}_t$	36	28	42	33	30
R_t	54	42	63	–	–
δ_t	1	1	1	–	–
Shortage probability	–	–	78%	–	–

Table 5: Partial assignment.

Policy cost: 366					
Period (t)	1	2	3	4	5
$\tilde{d}_t$	36	28	42	33	30
R_t	131	128	130	88	55
δ_t	1	1	1	1	1
Shortage probability	–	–	3%	3%	5%

Table 6: Full assignment.

$$\begin{aligned} Pr\{S_1\} \cdot G\left(\frac{63-42}{0.3\sqrt{42^2}}\right) + Pr\{S_2\} \cdot G\left(\frac{42-(28+42)}{0.3\sqrt{28^2+42^2}}\right) &+ \\ Pr\{S_3\} \cdot G\left(\frac{54-(36+28+42)}{0.3\sqrt{36^2+28^2+42^2}}\right) &+ \\ Pr\{S_4\} \cdot G\left(\frac{54+(63-42)-(36+42)}{0.3\sqrt{36^2+42^2}}\right) = 0.2192 \cong 0.22 & \end{aligned} \tag{32}$$

The service level provided (about 0.22) is not sufficient to satisfy SL Constraints (Eq. 20) in period 3. In order to decide which order-up-to-position to increase, we analyze the behavior of the service level at period 3, when R_1, R_2, and R_3 are increased respectively. If we increase R_1 by one unit, the service level at period 3 becomes 0.2229; if we increase R_2 by one unit, the service level at period 3 becomes 0.2175; finally, if we increase R_3 by one unit, the service level at period 3 becomes 0.2239 It follows that, an increase of one unit for R_3 achieves the maximum service level improvement. We proceed in a similar fashion by increasing at each step the order-up-to-position R_t that produces the maximum increase in the service level provided, until SL Constraints are satisfied for the period of interest. The reader may be easily convinced that, when we consider period 3, this process terminates after a few steps, when $R_1 = 131$, $R_2 = 95$, and $R_3 = 75$. We then proceed and repeat the same process, assuming that the ordering decisions are all fixed and that an order is scheduled in every period. In this case, we consider period 4 and period 5 sequentially. The final solution produced by this approach is shown in Table 6.

We next describe the complete approach. Our technique exploits the model presented in

Eq. 21 - 25. This model is implemented within Choco 1.2 (Laburthe et al. [22]), an open source Constraint Programming solver developed in Java. The variable selection heuristic branches first on decision variables δ_t. These variables are selected according to their natural order, that is $\{\delta_1, \ldots, \delta_N\}$. The value selection heuristic selects values in increasing order. SL Constraints (Eq. 20) cannot be directly implemented as such, and therefore are replaced, in our Constraint Programming model, by a global constraint able to dynamically compute the required order-up-to-level for a given partial replenishment plan (that is, a partial assignment for decision variables δ_t). As discussed, the order-up-to-levels are computed using the gradient-based local search approach shown in the former example, which in practice follows a *coordinate descent* strategy. A pseudo-code describing the propagation logic of this constraint is presented below in Algorithm 2.

```
Algorithm 2: Heuristic Method II - Propagation
input  : a partial assignment for decision variables $\delta_t$, $t = 1, \ldots, N$,
         the expected closing-inventory-positions $\tilde{P}_t$, $t = 1, \ldots, N$,
         the service level $\alpha$
begin
    Let $\mathcal{R}$ be the set of consecutive replenishment cycles identified by the partial
    assignment for decision variables $\delta_t$;
    for each replenishment cycle $\mathcal{R}(i,j)$ in $\mathcal{R}$ do
        for each period $t$ in $\mathcal{R}(i,j)$ do
            Let $\mathcal{P}$ be the set of former cycles influencing the service level in period $t$
            according to SL Constraint (Eq. 20);
            while the service level in period $t$ is less than $\alpha$ do
                For each cycle $\mathcal{R}(m,n) \in \mathcal{R} \bigcup \{\mathcal{R}(i,j)\}$ obtain the respective minimum
                allowed order-up-to-position $\underline{R}_m$ as $Inf\{Dom(\tilde{P}_m)\} + \tilde{d}_m$;
                Let $\mathcal{R}(m,n) \in \mathcal{R} \bigcup \{\mathcal{R}(i,j)\}$ be the cycle for which a unit increment in
                $\underline{R}_m$ produces the highest service level improvement;
                $Inf\{Dom(\tilde{P}_m)\} \leftarrow Inf\{Dom(\tilde{P}_m)\} + 1$;
end
```

The propagation logic described in Algorithm 2 is triggered each and every time, during the search, a replenishment decision is fixed. Recall that our search strategy branches first on decision variables δ_t, according to their natural sequence, so that at each node of the search tree we have a set of consecutive replenishment decisions $\{\delta_1, \ldots, \delta_t\}$ that have been assigned. This implies, that at each node of the search tree, we will also have a set of consecutive replenishment cycles, $\mathcal{R} \equiv \{\mathcal{R}(1,i),\ldots,\mathcal{R}(j,t)\}$, that are uniquely identified by the current partial assignment (Algorithm 2, line 2). For each cycle, we consider each and every period (Algorithm 2, line 4). By using SL Constraints (Eq. 20) and more specifically condition 10, we can easily identify which former cycles (and order-up-to-positions) are affecting the service level in the period under consideration (Algorithm 2, line 5). We consider each of these order-up-to-positions, say R_m, and observe the behavior of the service level when the minimum value allowed for it, $\underline{R}_m = Inf\{Dom(\tilde{P}_m)\} + \tilde{d}_m$, (Algorithm 2, line 7) is increased by one unit. That is, when the minimum value (Inf) in the domain (Dom) of $\tilde{P}_m$ plus $\tilde{d}_m$, is increased by one unit. Once the order-up-to-position R_m that produces the maximum service level improvement for the period considered is identified (Algorithm 2, line 8), we restrict the domain of the corresponding expected closing inventory position $\tilde{P}_m$ by removing the value $Inf\{Dom(\tilde{P}_m)\}$ (Algorithm 2, line 9). This process eventually produces (by subsequent greedy improvements), a set of order-up-to-positions that meet the required service level.

Also this second approach is clearly heuristic, since the order-up-to-levels are adjusted by using “local” moves that aim to locally maximize the improvement in the service level provided at a given period. It is also an example of a hybrid method that employs local search, at each node of the search tree, within the propagation logic of a global constraint. Nevertheless, this approach does perform better than the previous one in terms of quality of the solutions produced. For instance, with respect to the example presented in Section 4.1, this second heuristic approach produces a solution with cost 366, that is only 2.8% more costly than the optimal solution.

It is worth recalling that, as discussed in Section 1, the strategy described in this section introduces local search techniques within a constructive global search algorithm, i.e. the Constraint Programming solver. More specifically, a *coordinate descent* heuristic is employed in order to heuristically propagate the service level constraints within a Constraint Programming model. In contrast, the technique discussed in Section 5 exploits the local search engine to “guide” the search, by restructuring the buffer stock matrix, while Constraint Programming is used to explore promising neighborhood, i.e. to find the optimal solution with respect to a given buffer stock matrix.

7. Computational Results

In the illustrative example provided in the previous section, it was shown that Heuristic I and Heuristic II are 11% and 3% more costly than the optimal solution. In this section, we aim to further analyze both the effectiveness and the efficiency of these two heuristics. More specifically, we consider a large set of instances and we will investigate, for each of the two heuristics, how close the solution produced is to the optimal solution obtained with the complete approach in Rossi et al. [31] and how long the heuristic search runs in order to produce the approximate solution. In addition, we investigate how well the two heuristics scale, when the instances become intractable for the complete approach.

We consider four patterns for the expected value of the demand in each period of the planning horizon. These patterns resemble the structure of the experiments proposed in Berry [6] and comprise a constant level, a life-cycle trend, a sinusoidal change, and a very erratic pattern (Fig. 2). The demand is normally distributed, in each period, about the forecast value. We consider two possible values for the coefficient of variation, $c_v \in \{0.2, 0.3\}$. c_v shows the effect of the size of random variation in demand about the mean. Recall that $\sigma_t = \tilde{d}_t c_v$, where σ_t is the standard deviation of the demand in period t and $\tilde{d}_t$ is the expected value of the demand in period t. The holding cost h is assumed to be fixed to 1 for all the instances, while the ordering cost a takes values in the set $\{100, 175, 250\}$. We consider two possible service levels $\alpha \in \{0.85, 0.95\}$. In our experiments, we consider five possible discrete probability density functions for the stochastic lead time. These are shown in Fig. 3. The lead time may therefore take one of the values shown in the x-axis with the probability indicated on the y-axis. It should be noted that it is not relevant in this work to compare the performance of our heuristics for a deterministic lead time. In fact, efficient complete solution methods have been proposed in the literature [37] for the case in which the lead time is absent. These methods, according to the discussion in [36], can be directly applied to solve instances in which the lead time is deterministic.

All the experiments were performed on an Intel® Pentium®4 3.66 Ghz with 2 Gb of RAM. Heuristic I has been implemented using ILOG OPL Studio 3.7 [1] and Solver 6.0 interfaced with a Java routine for updating the buffer stock matrix. Heuristic II has been implemented on the top of Choco 1.2 (Laburthe et al. [22]).

The CP model repeatedly solved by Heuristic I has a very compact size, in fact it comprises only $2N$ variables and $3N$ constraints, as discussed in [41]. Several efficient approaches

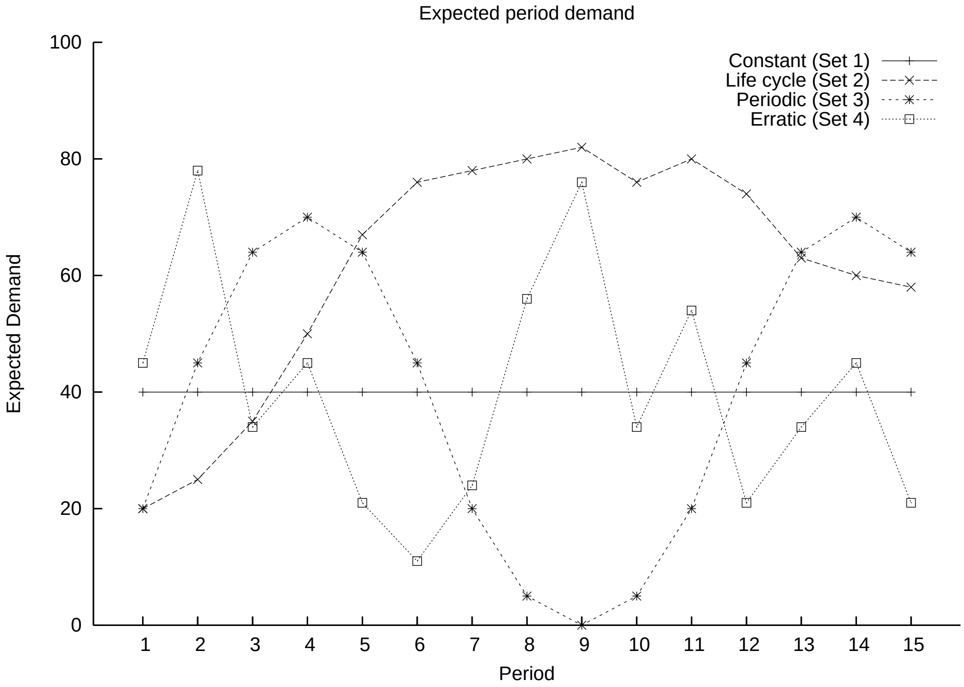


Figure 2: The mean demand patterns over time as in Berry [6].

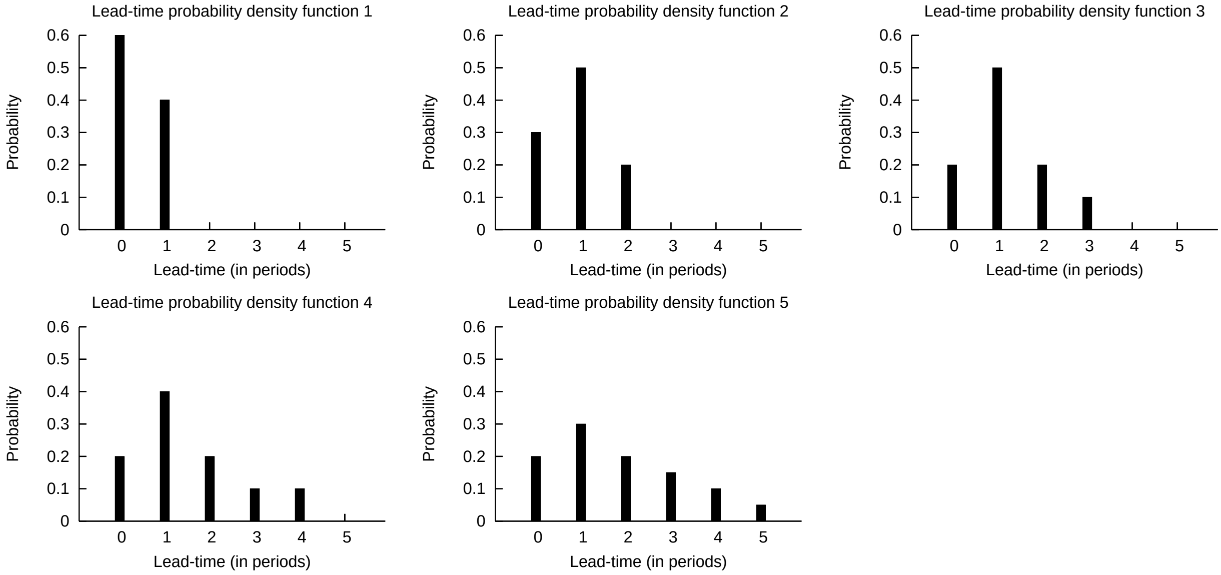


Figure 3: The lead time probability density functions.

[41, 37, 32] exist for solving such a model in fraction of a second for planning horizons comprising hundreds of periods. Both the complete approach in Rossi et al. [31] and Heuristic II operate on CP models that comprise $3N$ variables and $3N + 1$ constraints. Therefore all these models remain quite compact as the lead time and the planning horizon length increase.

In order to assess the quality of the solutions produced by the two heuristics, we first consider small instances over a 6-period planning horizon. We also limit the maximum length of the stochastic lead time, so that the resulting instances are tractable for the complete approach, which can therefore prove optimality in a reasonable time. In order to do so, we only consider, in Fig. 2, the expected demand in periods $\{1, \ldots, 6\}$, and, in Fig. 3, the lead time probability density functions (LT) 1, 2, and 3. By varying the model parameters (a, c_v, α, etc.) as discussed in the previous paragraphs, we obtain a total of 144 instances.

The results obtained by running the exact approach in Rossi et al. [31] over the test bed are shown in Table 9 and Table 10.[2] The exact approach often required a significant amount of time in order to solve these instances to optimality. More precisely, in the worst case it ran for 105 hours before finding the optimal solution. On an average, the optimal solution was found in 2 hours. For half of the instances, the optimal solution was found in less than 45 seconds; approximately 80% of the instances required less than 30 minutes to be solved; and about 90% of the instances required less than 2 hours.

In Table 11, we compare the cost of the solutions found by the two heuristics to the optimal ones. For convenience, results are also summarized in Table 7. Heuristic I found the optimal solution only for 2.7% of the instances, while Heuristic II succeeded in finding the optimal solution for about 25% of the instances. For half of the instances, the cost overhead incurred by Heuristic I was below 3.1%, while that incurred by Heuristic II was significantly lower, being only 0.54%. In Table 7 we provide further statistics, namely, the 80% and the 90% percentile. In the worst case, Heuristic I produced a solution 22% costlier than the optimal one, while Heuristic II produced a solution that is 8.5% more costly. On an average, the solution found by Heuristic I was 3.8% expensive than the optimal one, while Heuristic II produced a solution that was 0.56% more expensive than the optimal. It should be noted that, since none of the two heuristics fully dominates the other in terms of solution quality, if they are used in conjunction (Table 7, column "H1 $\bigoplus$ H2") the overall performance improves. In fact, the average cost overhead decreases to 0.48%, the maximum cost overhead is halved

[2] Note that Tables 9,. . .,15 are presented in the Appendix

	Cost Overhead			
Instances	H1	H2	H1 $\bigoplus$ H2	Exact - 10 secs
2.7%	0	0	0	0
25%	< 0.26%	0	0	0
50%	< 3.1%	< 0.54%	< 0.29%	0
80%	< 7%	< 1%	< 0.75%	< 8.5%
90%	< 10%	< 2%	< 1%	< 16%
100%	< 22%	< 8.5%	< 4.27%	< 45.7%

Table 7: Statistics on the cost overhead, over a 6-period planning horizon, incurred by Heuristic I (H1), Heuristic II (H2), a strategy that combines H1 and H2 (H1 $\bigoplus$ H2), and a strategy that limits to 10 seconds the run time of the exact approach (Exact - 10 secs). The cost overhead is expressed in percentage of the optimal policy cost.

to 4.27%, and also the other quantiles, as shown in Table 7, significantly improve.

In Table 12, we present the run-times for the two heuristics. It is easy to observe that for all the instances Heuristic II was faster than Heuristic I. Nevertheless, both the heuristics did not require more than a few seconds to solve any of the instances. More precisely, the maximum run time observed is 8.5 seconds.

It might be argued that the exact approach in Rossi et al. [31] may converge quickly to good solutions and therefore be used as a heuristic approach by simply limiting the run time and collecting the best solution found within the given time limit. Since the maximum run time observed for our heuristic methods over the given test bed is 8.5 seconds, we allocated a run time of 10 seconds to the exact approach and observed the cost difference between the optimal solution produced by the exact approach without a time limit and the best solution that this approach could find in the given time limit of 10 seconds. In Table 13 we present cost differences, in percentage of optimal costs. For convenience, we also summarized the results in Table 7, column "Exact - 10 secs". The exact approach with a 10 seconds limit was able to reach the optimal solution for more than 50% of the instances. Nevertheless, the performance of this heuristic strategy quickly deteriorates when we consider the 30 most difficult instances. For 20% of the instances, the error exceeded 8.5%. For 10% of the instances, the error exceeded 16%. In the worst case, the error reached 45.7%. On an average, the solution found by this strategy was 4.8% more expensive than the optimal one. As shown, both Heuristic I and II produced better average and worst case results. In the light of these results, it is clear that imposing a time limit to the exact approach in Rossi et al. [31] does not constitute a viable heuristic strategy. It is should be also noted that longer planning horizons exacerbate the poor performance of this heuristic.

In order to assess the scalability of the two heuristics, we now consider the full demand patterns in Fig. 2, therefore we now run tests over a 15-period planning horizon. In addition, we also consider all the 5 possible discrete probability density functions shown in Fig. 3 (therefore LT ranges now in $\{1, \ldots, 5\}$) and we vary the remaining model parameters (a, h, c_v and α) as discussed before. By doing so, we obtain a total of 240 instances. In Table 14 and Table 15 we compare, respectively, the run times and the cost of the solutions found by the two heuristics for this new set of instances. Since these instances are intractable for the exact approach, we will only compare the two heuristics among each other.

Over the 240 instances considered, Heuristic I produced the best solution only 25% of the times. In all the other cases, Heuristic II found a better solution (75% of the instances). In the worst case, Heuristic I produced a solution that was 14.6% more costly than the one obtained by using Heuristic II. Heuristic II, in contrast, produced a solution that was only 3.5% more costly than the one found by Heuristic I, in the worst case. On an average, for the instances in which Heuristic I could not produce the best solution, the solution produced was 3.66% more costly than that produced by Heuristic II; for those instances for which Heuristic II could not find the best solution, the solution found by this heuristic was only 0.80% more costly than that produced by Heuristic I.

From this comparison, and from the previous discussion, it is possible to observe that Heuristic II typically performs better than Heuristic I in terms of quality of the solution produced. Nevertheless, in terms of run times (Table 8), the picture is different. In fact, Heuristic I maintains good performances over all the instances in the 15-period planning horizon test bed. More specifically, the average run time for Heuristic I was 30 seconds, in contrast to an average run time of about 16 minutes for Heuristic II. In the worst case, Heuristic I took less than 3 minutes to complete the search, while Heuristic II completed the search in 1 hour and 10 minutes. About 50% of the instances could be solved by Heuristic I in less than 25 seconds, in contrast to the 11 minutes required by Heuristic II. Furthermore, we observed that 80% of the instances required less than 40 seconds in order to be solved by using Heuristic I, in contrast to the 30 minutes required by Heuristic II. Finally, the 90% quantile was less than a minute for Heuristic I and about 35 minutes for Heuristic II.

	Run time	
Instances	H1	H2
25%	0	0
50%	< 25 seconds	< 11 minutes
80%	< 40 seconds	< 30 minutes
90%	< 1 minute	< 35 minutes
100%	< 3 minutes	< 1 hour and 10 minutes

Table 8: Statistics on the run times incurred by Heuristic I (H1) and Heuristic II (H2), over a 15-period planning horizon.

8. Summary and Conclusions

We addressed the computation of near-optimal replenishment cycle policy parameters under non-stationary stochastic demand, stochastic supplier lead time and service-level constraints. Two hybrid heuristic algorithms that blend Constraint Programming and Local Search were proposed.

Firstly, we compared these two heuristics for small instances against an exact method. In our experiments, Heuristic I was within 3.8% of the optimal, while Heuristic II was within 0.56% of the optimal. In addition, when used in conjunction, the two heuristics were within 0.48% of the optimal. In terms of computational time, the average run time for Heuristic I, Heuristic II and the Optimal solution is 2.96 seconds, 0.75 seconds and 2 hours respectively. The results proved that Heuristic II typically performs better than Heuristic I in terms of quality of the solutions produced, by achieving a very little cost overhead.

Secondly, the two heuristics were tested and compared with each other, in terms of both solution quality and run time over a set of larger instances that are intractable for the exact approach. In terms of solution quality, Heuristic II performed better and was on average 3% better than the performance of Heuristic I, while the average run times for Heuristic I and Heuristic II were 30 seconds and 16 minutes respectively. The results confirmed that Heuristic I typically produces lower quality solutions than Heuristic II, but also that Heuristic I runs much faster than Heuristic II.

We can conclude that, if run time is a critical aspect, Heuristic I may be a viable choice, while if the quality of the solution produced is a major concern, then Heuristic II (or a combination of the two heuristics) should be chosen, as it achieves high quality solutions and is orders of magnitude faster than the exact approach.

References

[1] *OPL Studio 3.7 Users Manual.* ILOG, Inc., Incline Village, NV, 2007.

[2] K. Apt. *Principles of Constraint Programming.* Cambridge University Press, Cambridge, UK, 2003.

[3] M. Z. Babaï, A. Syntetos, Y. Z. Dallery, and K. Nikolopoulos. Dynamic re-order point inventory control with lead-time uncertainty: analysis and empirical investigation. *International Journal of Production Research*, 47(9):2461–2483, 2009.

[4] B. De Backer, V. Furnon, P. Shaw, P. Kilby, and P. Prosser. Solving vehicle routing problems using constraint programming and metaheuristics. *Journal of Heuristics*, 6(4):501–523, 2000.

[5] S. Bashyam and M.C. Fu. Optimization of (s,S) inventory systems with random lead times and a service level constraint. *Management Science*, 44(12):243–256, 1998.

[6] W. L. Berry. Lot sizing procedures for requirements planning systems: A framework for analysis. *Production and Inventory Management Journal*, 13(2):19–34, 1972.

[7] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming.* Springer Verlag, New York, 1997.

[8] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, September 2003.

[9] J. H. Bookbinder and J. Y. Tan. Strategies for the probabilistic lot-sizing problem with service-level constraints. *Management Science*, 34(9):1096–1108, 1988.

[10] A. Cesta, G. Cortellessa, A. Oddi, N. Policella, and A. Susi. A constraint-based architecture for flexible support to activity scheduling. In *AI*IA 01: Proceedings of the 7th Congress of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence*, pages 369–381, London, UK, 2001. Springer-Verlag.

[11] A. Charnes and W. W. Cooper. Chance-constrainted programming. *Management Science*, 6(1):73–79, 1959.

[12] A. G. de Kok. Basics of inventory management: part 2 The (R,S)-model. Research memorandum, FEW 521, 1991. Department of Economics, Tilburg University, Tilburg, The Netherlands.

[13] T. de Kok and K. Inderfurth. Nervousness in inventory management: Comparison of basic control rules. *European Journal of Operational Research*, 103:55–82, 1997.

[14] G. D. Eppen and R. K. Martin. Determining safety stock in the presence of stochastic lead time and demand. *Management Science*, 34(11):1380–1390, 1988.

[15] F. Focacci, F. Laburthe, and A. Lodi. Local Search and Constraint Programming. *In F. Glover and G. Kochenberger, editors, Handbook of Metaheuristics, volume 57 of International Series in Operations Research and Management Science. Kluwer Academic Publishers, Norwell, MA*, 2002.

[16] F. Focacci, A. Lodi, and M. Milano. Cost-based domain filtering. In *Proceedings of the 5th International Conference on the Principles and Practice of Constraint Programming*, pages 189–203. Springer Verlag, 1999. Lecture Notes in Computer Science No. 1713.

[17] J. C. Hayya, U. Bagchi, J. G. Kim, and D. Sun. On static stochastic order crossover. *International Journal of Production Economics*, 114(1):404–413, July 2008.

[18] J. C. Hayya, S. H. Xu, R. V. Ramasesh, and X. X. He. Order crossover in inventory systems. *Stochastic Models*, 11(2):279–309, 1995.

[19] Gerald Heisig. *Planning Stability in Material Requirements Planning Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[20] J. A. Hunt. Balancing accuracy and simplicity in determining reorder points. *Management Science*, 12(4):B94–B103, 1965.

[21] R. S. Kaplan. A dynamic inventory model with stochastic lead times. *Management Science*, 16(7):491–507, 1970.

[22] F. Laburthe and the OCRE project team. Choco: Implementing a cp kernel. Technical report, Bouygues e-Lab, France, 1994.

[23] C. Nevison and M. Burstein. The dynamic lot-size model with stochastic lead-times. *Management Science*, 30(1):100–109, 1984.

[24] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, August 1999.

[25] G. Pesant and M. Gendreau. A view of local search in constraint programming. In Eugene C. Freuder, editor, *Proceedings of the Second International Conference on Principles and Practice of Constraint Programming, Cambridge, Massachusetts, USA, August 19-22, 1996*, volume 1118 of *Lecture Notes in Computer Science*, pages 353–366. Springer, 1996.

[26] J.-C. Régin. A filtering algorithm for constraints of difference in csps. In *Proceedings of the twelfth national conference on Artificial intelligence (vol. 1), Seattle, Washigton*, pages 362–367. American Association for Artificial Intelligence, 1994.

[27] J.-C Régin. *Global Constraints and Filtering Algorithms.* in Constraints and Integer Programming Combined, Kluwer, M. Milano editor, 2003.

[28] J. Riezebos. Inventory order crossovers. *International Journal of Production Economics*, 104(2):666–675, December 2006.

[29] F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.

[30] R. Rossi, S. A. Tarim, B. Hnich, and S. Prestwich. A global chance-constraint for stochastic inventory systems under service level constraints. *Constraints*, 13(4):490–517, 2008.

[31] R. Rossi, S. A. Tarim, B. Hnich, and S. Prestwich. Computing the non-stationary replenishment cycle inventory policy under stochastic supplier lead-times. *International Journal of Production Economics*, 2010. forthcoming.

[32] R. Rossi, S. A. Tarim, B. Hnich, and S. Prestwich. A state space augmentation algorithm for the replenishment cycle inventory policy. *International Journal of Production Economics*, 2010. forthcoming.

[33] E. A. Silver, D. F. Pyke, and R. Peterson. *Inventory Management and Production Planning and Scheduling*. John-Wiley and Sons, New York, 1998.

[34] T. W. Speh and G. Wagenheim. Demand and lead-time uncertainty: The impacts of physical distribution performance and management. *Journal of Business Logistics*, 1(1):95–113, 1978.

[35] C. S. Tang. Perpectives in supply chain risk management. *International Journal of Production Economics*, 103:451–488, 2006.

[36] S. A. Tarim. *Dynamic Lotsizing Models for Stochastic Demand in Single and Multi-Echelon Inventory Systems*. PhD thesis, Lancaster University, 1996.

[37] S. A. Tarim, B. Hnich, R. Rossi, and S. Prestwich. Cost-based filtering techniques for stochastic inventory control under service level constraints. *Constraints*, 14(2):137–176, 2009.

[38] S. A. Tarim and B. G. Kingsman. The stochastic dynamic production/inventory lot-sizing problem with service-level constraints. *International Journal of Production Economics*, 88(1):105–119, 2004.

[39] S. A. Tarim and B. G. Kingsman. Modelling and Computing (R^n,S^n) Policies for Inventory Systems with Non-Stationary Stochastic Demand. *European Journal of Operational Research*, 174(1):581–599, 2006.

[40] S. A. Tarim, S. Manandhar, and T. Walsh. Stochastic constraint programming: A scenario-based approach. *Constraints*, 11(1):53–80, 2006.

[41] S. A. Tarim and B. Smith. Constraint Programming for Computing Non-Stationary (R,S) Inventory Policies. *European Journal of Operational Research*, 189(3):1004–1021, 2008.

[42] H. Tempelmeier. On the stochastic uncapacitated dynamic single-item lotsizing problem with service level constraints. *European Journal of Operational Research*, 181(1):184–194, 2007.

[43] D. C. Whybark and J. G. Williams. Material requirements planning under uncertainty. *Decision Science*, 7(4):595–606, 1976.

[44] P. Zipkin. Stochastic leadtimes in continuous-time inventory models. *Naval Research Logistics Quarterly*, 33(4):763–774, 1986.

		$a = 100$				$a = 175$				$a = 250$			
		$c_v = 0.2$		$c_v = 0.3$		$c_v = 0.2$		$c_v = 0.3$		$c_v = 0.2$		$c_v = 0.3$	
Set	LT	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$
1	1	620	682	640	728	770	833	797	893	920	983	947	1043
	2	677	764	686	826	834	914	836	976	970	1042	986	1126
	3	703	820	738	869	853	967	888	1020	970	1042	1030	1138
2	1	715	783	745	847	895	973	937	1051	1045	1123	1087	1201
	2	803	961	838	1028	1028	1131	1057	1215	1187	1281	1207	1365
	3	903	1025	953	1103	1057	1243	1105	1317	1207	1383	1255	1467
3	1	772	854	798	924	966	1052	1016	1148	1116	1202	1166	1298
	2	836	1032	886	1098	1061	1182	1100	1280	1214	1332	1250	1430
	3	889	1087	950	1162	1065	1278	1137	1362	1215	1378	1287	1504
4	1	530	590	560	650	680	740	710	800	808	886	860	950
	2	576	662	588	716	726	811	738	866	808	886	874	994
	3	545	678	600	734	695	811	750	884	808	886	874	994

Table 9: Exact approach, optimal policy costs, 6-period planning horizon.

		$a = 100$				$a = 175$				$a = 250$			
		$c_v = 0.2$		$c_v = 0.3$		$c_v = 0.2$		$c_v = 0.3$		$c_v = 0.2$		$c_v = 0.3$	
Set	LT	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$
1	1	20	46	15	47	0.98	2.5	0.86	3.7	0.39	0.72	0.52	1.2
	2	210	582	142	793	4.4	16	3.2	24	0.66	1.9	0.80	4.5
	3	3188	15570	4258	18496	27	243	35	318	0.66	2.8	2.0	19
2	1	210	583	212	934	7.2	20	9.4	37	1.1	2.9	1.4	4.8
	2	1611	14409	1993	23781	150	521	148	935	16	43	13	76
	3	46082	88934	62789	132963	794	16092	1224	24845	49	537	82	943
3	1	571	1800	436	2413	28	69	39	153	3.3	8.3	4.3	16
	2	1866	16488	2486	20712	340	1176	379	2016	17	115	14	234
	3	70805	216835	104318	380934	2630	49724	6767	79378	110	1261	208	5472
4	1	3.6	6.1	3.1	9.5	0.42	1.2	0.69	1.9	0.19	0.53	0.44	1.1
	2	52	211	32	292	1.4	12	1.2	15	0.19	0.53	0.49	1.4
	3	90	2684	304	4139	2.0	40	7.7	90	0.20	0.53	0.47	1.4

Table 10: Exact approach, run times (secs), 6-period planning horizon.

		$a = 100$								$a = 175$								$a = 250$							
		$c_v = 0.2$				$c_v = 0.3$				$c_v = 0.2$				$c_v = 0.3$				$c_v = 0.2$				$c_v = 0.3$			
		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$	
Set	LT	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2
1	1	3.4	0.48	0.44	0	3.1	0.63	1.4	0.55	3.4	0.39	0.12	0.36	4.6	0.38	0.11	0.67	2.7	0.33	0.10	0.31	4	0.32	0	0.58
	2	5.0	0.44	0.52	0.79	5.4	0.87	0.85	0.97	3.2	0.48	0.44	0.66	4.4	0.72	0.82	0.82	4.2	0	0.19	0	4	0.61	0	0.71
	3	22	0	0.12	3.3	8.0	0.54	0.12	4.1	5.0	0	0.21	0	4.6	0.45	0.10	4.2	7.6	0	0.19	0	5	0	0.09	0
2	1	4.2	0.56	0.77	0.51	6.7	0.27	2.7	0.71	4.9	0.67	0.31	0.31	7.5	0.64	0.10	0.29	4.2	0.57	0.27	0.27	6	0.55	0.08	0.25
	2	14	0.50	1.2	0.94	14	1.1	1.6	0.58	5.6	0.39	0.27	0.18	6.0	0.85	0.25	0.49	4.1	0.17	0.23	0.16	5	0.75	0.22	0.44
	3	2.3	0	6.9	0.68	4.0	0.10	7.7	1.1	13	0	0.16	0.80	7.1	0	0	1.2	4.8	0	0.14	0	6	0	0.07	1.1
3	1	2.8	0.78	0.35	0.70	3.9	0.75	0.22	0.65	5.1	0.62	1.2	0.57	3.6	0.30	0.26	0.52	4.4	0.54	0.17	0.50	5	0.26	0.23	0.46
	2	15	2.0	0.29	0.78	8.9	2.6	2.3	0.82	6.4	1.0	0.25	0.68	4.7	0.91	0.47	0.31	5.4	0.66	0.23	0.60	4	0.80	0.28	0.28
	3	13	4.3	4.0	6.0	10	2.4	4.5	6.3	8.8	3.5	0.16	1.9	8.9	0	0.22	1.8	7.8	3.0	0.15	0	7	0	0.07	0
4	1	5.7	0.38	0.34	1.0	9.3	1.1	1.2	1.2	4.4	0.29	0.27	0.81	6.6	0.85	0.12	1.0	6.4	0	0.56	0	5	0.70	0.11	0.84
	2	5.2	0.69	0.76	1.2	9.4	1.0	0.42	1.1	4.1	0.55	0.49	0	6.1	0.81	0.35	0.92	10	0	0.45	0	7	0	0.10	0
	3	21	0	1.2	8.6	9.5	0	0.82	8.2	10	0	0.37	0	10	0	0.68	4.0	10	0	0.34	0	12	0	0.10	0

Table 11: Additional cost, in % of the cost of the optimal policy, incurred if Heuristic I (H1) or Heuristic II (H2) are used, 6-period planning horizon.

		$a = 100$								$a = 175$								$a = 250$							
		$c_v = 0.2$				$c_v = 0.3$				$c_v = 0.2$				$c_v = 0.3$				$c_v = 0.2$				$c_v = 0.3$			
		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$	
Set	LT	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2
1	1	3.5	0.72	2.4	0.16	1.2	0.14	2.8	0.16	1.2	0.11	1.7	0.13	0.88	0.34	1.8	0.20	1.3	0.39	1.2	0.36	0.91	0.69	1.4	0.14
	2	3.9	0.72	3.5	0.33	3.0	0.56	5.2	0.41	1.8	0.19	2.4	0.27	2.3	0.45	1.5	0.27	1.6	0.30	1.8	0.70	2.0	1.0	1.8	0.67
	3	3.4	0.36	4.4	1.3	3.5	1.2	2.2	1.1	2.3	0.27	2.2	1.8	2.8	0.49	2.1	1.27	1.5	0.70	1.4	0.69	1.3	0.77	1.2	1.5
2	1	2.8	0.45	3.2	0.17	2.4	0.20	3.2	0.20	1.8	0.17	2.4	0.55	2.4	0.17	2.6	0.20	1.8	0.52	2.1	0.44	2.0	0.16	2.5	0.50
	2	3.8	0.89	2.9	0.66	4.7	0.77	3.2	1.5	3.5	0.70	3.4	0.95	3.6	0.38	3.4	1.1	2.2	0.83	2.2	0.94	2.3	0.39	2.6	1.0
	3	3.3	0.83	3.7	2.0	4.4	1.4	4.3	1.7	3.1	1.5	2.7	2.5	3.8	1.5	3.4	1.6	3.0	0.97	2.0	0.88	2.7	1.5	3.1	1.3
3	1	2.7	0.53	3.8	0.78	2.4	0.16	4.4	0.20	3.4	1.2	3.7	0.58	1.5	0.55	3.6	0.70	2.3	0.41	2.0	0.31	2.4	0.47	4.6	0.34
	2	4.3	0.80	6.4	1.7	3.2	0.88	5.7	1.0	3.8	1.1	4.3	0.67	2.9	0.78	3.9	1.2	4.1	0.69	2.5	0.91	3.0	0.61	4.1	0.94
	3	7.7	1.1	5.5	2.4	6.3	1.4	5.5	3.4	4.0	2.6	4.5	1.6	4.8	1.4	3.5	1.1	3.7	1.2	2.1	1.3	4.0	0.36	5.6	1.3
4	1	2.5	0.33	2.6	0.39	2.8	0.33	3.0	0.38	2.4	0.16	2.1	0.31	1.3	0.23	2.7	0.39	1.3	0.31	1.4	0.34	0.7	0.09	1.7	0.30
	2	3.2	0.59	4.0	0.74	3.9	0.70	3.0	0.58	2.7	0.16	2.9	0.50	1.7	0.72	3.9	0.70	2.0	0.50	2.1	0.55	2.1	0.36	2.3	0.17
	3	3.8	0.49	8.5	0.61	3.2	0.63	4.8	2.3	2.6	0.16	2.7	1.1	4.0	0.55	2.9	1.2	1.4	0.16	1.3	0.78	1.8	0.45	3.3	0.47

Table 12: Heuristic I (H1) and Heuristic II (H2) run times (secs), 6-period planning horizon.

		$a = 100$				$a = 175$				$a = 250$			
		$c_v = 0.2$		$c_v = 0.3$		$c_v = 0.2$		$c_v = 0.3$		$c_v = 0.2$		$c_v = 0.3$	
Set	LT	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$	$\alpha = 0.85$	$\alpha = 0.95$
1	1	0	0	0	2.0	0	0	0	0	0	0	0	0
	2	1	2.3	0	0.85	0	0	0	0	0	0	0	0
	3	12	8.7	6.7	10	0.59	0	0	4.2	0	0	0	0
2	1	4.2	5.1	5.6	6.3	0	0	0	0	0	0	0	0
	2	42	28	45	31	1.3	15	22	17	0	2.4	0	1.6
	3	26	20	28	22	15	5.2	17	8.4	7.1	0	9.2	2.4
3	1	5.7	7	8.5	8	0	1.1	0	0	0	0	0	0
	2	12	17	7	17	2.8	3.9	0	11	0	3.4	0	1.6
	3	25	12	24	14	12	1.9	12	4.9	2.5	0	5.6	0
4	1	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	2.9	0	0	0	0	0	0	0	0
	3	11	0	5	4.5	0	0	0	0	0	0	0	0

Table 13: Exact approach with a run time limited to 10 seconds. Additional cost, in % of the cost of the optimal policy.

		$a = 100$								$a = 175$								$a = 250$							
		$c_v = 0.2$				$c_v = 0.3$				$c_v = 0.2$				$c_v = 0.3$				$c_v = 0.2$				$c_v = 0.3$			
		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$	
Set	LT	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2
1	1	1.5	0	0.5	0	1.7	0	0	0.21	0.68	0	0	0.45	1.2	0	0	0.59	1.4	0	0	0.50	1.7	0	0	0.54
	2	2.0	0	0	0.18	1.1	0	0.04	0	1.3	0	0	0.84	1.5	0	0	0.91	1.5	0	0	0.37	2.2	0	0	0.85
	3	1.6	0	1.4	0	2.7	0	0	0.12	1.5	0	0	0	0.76	0	0	0.71	2.2	0	0	0.93	1.4	0	0	0.97
	4	9.8	0	3.4	0	11	0	2.6	0	11	0	1.0	0	5.0	0	2.3	0	2.6	0	1.3	0	1.6	0	0.33	0
	5	18	0	6.1	0	5.1	0	2.1	0	16	0	2.4	0	9.9	0	0	0.94	8.9	0	1.3	0	9.6	0	0.53	0
2	1	0.9	0	0	0.04	1.6	0	0	0	1.3	0	0	0.43	0.82	0	0	0.5	0.88	0	0	0.09	1.1	0	0	0.49
	2	6.2	0	0.5	0	4.0	0	0.9	0	3	0	0	0.48	2.4	0	0	0	3.0	0	0	0.55	0.77	0	0	0.58
	3	5.4	0	4.3	0	1.0	0	1.1	0	1.0	0	2.5	0	2.0	0	0.65	0	1.9	0	2.1	0	1.5	0	0.44	0
	4	2.9	0	5.7	0	1.7	0	2.5	0	3.5	0	2.9	0	3.2	0	3.5	0	2.5	0	2.6	0	3.0	0	2.88	0
	5	9.4	0	6.9	0	1.1	0	0	3.3	14	0	0	0.29	0.09	0	0	0.19	8.1	0	1.6	0	3.7	0	0	0.84
3	1	3.0	0	0	0.44	4.0	0	0	0	2.4	0	0	0	4.8	0	0.25	0	4.5	0	0	0.51	5.3	0	0	0.73
	2	8.6	0	0.5	0	5.5	0	0.42	0	6.0	0	0	0.54	2.4	0	0	0	3.6	0	0	0.48	4.3	0	0.32	0
	3	8.1	0	0	1.4	8.5	0	0	0	6.2	1.1	0.45	0	6.1	0	0	3.5	5.2	0	1.7	0	5.9	0	0	3.4
	4	15	0	3.3	0	11	0	0	0	6.0	2.6	2.4	0	9.6	0	0	0.32	5.3	0	2.2	0	9.5	0	0	0.27
	5	6.4	0	0.8	0	1.8	0	0	0	14	0.73	1.6	0	4.3	0	0	2.8	7.5	0	1.5	0	4.1	0	0	1.4
4	1	2.7	0	0.0	0.65	3.5	0	0	0.82	0.82	0	0	0.47	2.1	0	0	0.71	2.4	0	0	0.20	3.0	0	0	0.61
	2	2.2	0	1.9	0	3.7	0	0.55	0	2.6	0	0	0.70	3.1	0	0	0.86	2.3	0	0	0.44	2.7	0	0	0.81
	3	2.2	0	0.3	0	3.7	0	1.7	0	2.9	0	0	1.3	3.7	0	0	0.65	3.2	0	0	0.88	3.8	0	0	0.58
	4	14	0	7.0	0	8.0	0	3.9	0	5.0	0	6.2	0	1.5	0	5.3	0	5.5	0	3.0	0	3.1	0	2.85	0
	5	8.7	0	4.6	0	11	0	3.0	0	5.6	0	3.2	0	6.1	0	0.64	0	1.2	0	0	0.03	2.8	0	0	0.03

Table 14: Cost comparison between Heuristic I (H1) and Heuristic II (H2). A value of 0 is associated with the heuristic that has found the best solution, the other value denotes the cost difference — in percentage of the best solution — achieved by the other heuristic, 15-period planning horizon.

		$a = 100$								$a = 175$								$a = 250$							
		$c_v = 0.2$				$c_v = 0.3$				$c_v = 0.2$				$c_v = 0.3$				$c_v = 0.2$				$c_v = 0.3$			
		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$		$\alpha = 0.85$		$\alpha = 0.95$	
Set	LT	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2	H1	H2
1	1	24	161	17	224	18	195	15	218	12	150	16	196	10	127	18	194	21	69	21	102	14	60	21	104
	2	26	446	33	420	14	311	28	404	23	327	30	421	12	275	27	405	28	210	36	360	19	153	30	355
	3	26	602	33	763	24	568	43	741	29	588	36	762	19	560	36	742	40	437	49	730	28	397	46	702
	4	36	1096	53	1294	46	1088	51	1516	39	1098	43	1296	19	1089	41	1439	43	979	56	1291	38	935	46	1431
	5	48	1874	48	2178	37	1902	66	2391	30	1870	42	2190	31	1909	47	2391	54	1646	50	2191	50	1678	44	2402
2	1	11	295	16	328	14	281	15	327	17	299	15	331	10	282	16	330	12	255	23	311	12	231	21	310
	2	25	520	29	669	18	472	28	648	25	524	35	674	18	480	29	653	24	523	29	673	16	473	37	651
	3	28	959	35	1345	24	924	51	1219	23	961	35	1272	25	930	35	1220	26	961	40	1272	25	925	53	1217
	4	32	1837	99	2105	35	1826	27	2443	36	1836	59	2099	32	1825	38	2449	31	1843	71	2103	36	1825	84	2447
	5	37	3129	48	3470	46	3223	36	4206	38	3122	46	3479	38	3207	73	4221	30	3123	66	3475	33	3213	67	4267
3	1	21	196	21	218	18	189	21	220	13	120	21	172	12	122	19	194	12	43	18	80	6	50	17	110
	2	35	359	32	442	27	342	39	440	28	323	26	438	19	318	25	439	20	178	24	308	17	165	22	357
	3	33	629	33	793	36	605	38	811	25	587	34	792	25	576	26	809	22	360	30	689	22	366	24	761
	4	36	1219	42	1303	32	1202	29	1549	24	1207	30	1304	32	1190	31	1551	21	883	20	1283	27	853	28	1519
	5	30	1800	31	2076	30	1915	34	2422	26	1794	20	2070	28	1906	23	2421	19	1559	17	2044	22	1697	19	2418
4	1	11	200	12	219	9	182	12	217	12	98	22	141	9	75	17	137	15	39	17	64	4	29	16	61
	2	18	343	22	400	17	322	27	387	26	313	29	401	21	283	26	385	20	153	24	276	18	128	22	270
	3	13	547	27	722	21	530	39	706	19	515	29	721	26	493	30	705	21	290	29	616	21	278	27	605
	4	65	1018	153	1347	23	992	65	1353	30	1016	46	1345	32	1045	48	1351	33	784	35	1289	25	706	39	1288
	5	25	1709	40	2067	22	1720	52	2126	139	1702	46	2069	47	1717	37	2124	28	1465	41	2055	36	1471	53	2121

Table 15: Heuristic I (H1) and Heuristic II (H2) run times (secs), 15-period planning horizon.

Contents lists available at ScienceDirect

Int. J. Production Economics

journal homepage: www.elsevier.com/locate/ijpe

Computing the non-stationary replenishment cycle inventory policy under stochastic supplier lead-times

Roberto Rossi [a,*], S. Armagan Tarim [b,1], Brahim Hnich [c,1], Steven Prestwich [d]

[a] Logistics, Decision and Information Sciences, Wageningen UR, Hollandseweg 1, 6706 KN, Wageningen, The Netherlands
[b] Department of Management, Hacettepe University, Turkey
[c] Faculty of Computer Science, Izmir University of Economics, Izmir, Turkey
[d] Cork Constraint Computation Centre, University College, Cork, Ireland

ARTICLE INFO

Article history:
Received 18 December 2007
Accepted 18 May 2010

Keywords:
Inventory control
Demand uncertainty
Supplier lead-time uncertainty
Stochastic constraint programming
Global chance-constraints

ABSTRACT

In this paper we address the general multi-period production/inventory problem with non-stationary stochastic demand and supplier lead-time under service level constraints. A replenishment cycle policy (R^n,S^n) is modeled, where R^n is the *n*th replenishment cycle length and S^n is the respective order-up-to-level. We propose a stochastic constraint programming approach for computing the optimal policy parameters. In order to do so, a dedicated global chance-constraint and the respective filtering algorithm that enforce the required service level are presented. Our numerical examples show that a stochastic supplier lead-time significantly affects policy parameters with respect to the case in which the lead-time is assumed to be deterministic or absent.

1. Introduction

An interesting class of production/inventory control problems is the one that considers the single location, single product case under stochastic demand. One of the well-known policies that can be adopted to control such a system is the “replenishment cycle policy’ (*R*,*S*). Under the non-stationary demand assumption this policy takes the form (R^n,S^n), where R^n denotes the length of the *n*th replenishment cycle, and S^n the order-up-to-level value for the *n*th replenishment. This easy to implement inventory control policy yields at most 2*N* policy parameters fixed at the beginning of an *N*-period planning horizon. For a discussion on inventory control policies see Silver et al. (1998). The replenishment cycle policy provides an effective means of damping the planning instability. Furthermore, it is particularly appealing when items are ordered from the same supplier or require resource sharing. In such a case all items in a coordinated group can be given the same replenishment period. Periodic review also allows a reasonable prediction of the level of the workload on the staff involved and is particularly suitable for advanced planning environments. For these reasons, as stated by Silver et al. (1998), (*R*,*S*) is a popular inventory policy. Due to its combinatorial nature, the computation of (R^n,S^n) policy parameters is known to be a difficult problem to solve to optimality. An early approach proposed by Bookbinder and Tan (1988) is based on a two-step heuristic method. Tarim and Kingsman (2004, 2006) and Tempelmeier (2007) propose a mathematical programming approach to compute policy parameters. Tarim and Smith (2008) give a computationally efficient constraint programming formulation. An exact formulation and a solution method are presented in Rossi et al. (2008).

All the above mentioned works assume either zero or a fixed (deterministic) supplier lead-time (i.e., replenishment lead-time). However, the lead-time uncertainty, which in various industries is an inherent part of the business environment, has a detrimental effect on inventory systems. For this reason, there is a vast inventory control literature analysing the impact of supplier lead-time uncertainty on the ordering policy (Whybark and Williams, 1976; Speh and Wagenheim, 1978; Nevison and Burstein, 1984). A comprehensive discussion on stochastic supplier lead-time in continuous-time inventory systems is presented in Zipkin (1986). Kaplan (1970) characterises the optimal policy for a dynamic inventory problem where the lead-time is a discrete random variable with known distribution and the demands in successive periods are assumed to form a stationary stochastic process. Since tracking all the outstanding orders through the use of dynamic programming requires a large multi-dimensional state vector, Kaplan assumes that orders do not cross in time and supplier lead time probabilities are independent of the size/number of outstanding orders (for details on order-crossover see Hayya et al., 1995).

[*] Corresponding author. Tel.: +31 317 482321; fax: +31 317 485646.
E-mail addresses: roberto.rossi@wur.nl (R. Rossi), armtar@yahoo.com (S.A. Tarim), brahim.hnich@ieu.edu.tr (B. Hnich), s.prestwich@4c.ucc.ie (S. Prestwich).

[1] The author is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027.

0925-5273/$ - see front matter © 2010 Published by Elsevier B.V.
doi:10.1016/j.ijpe.2010.05.011

The assumption that orders do not cross in time is valid for systems where the supplier production system has a single-server queue structure operating under a FIFO policy. Nevertheless, there are settings in which this assumption is not valid and orders cross in time. This has been recently investigated in Hayya et al. (2008), Bashyam and Fu (1998) and Riezebos (2006). As Riezebos underscores, the types of industries that have a higher probability of facing order crossovers are either located upstream in the supply chain, or use natural resources, or order strategic materials from multiple suppliers or from abroad. In a case study, he showed that the potential cost savings realized by taking order crossovers into account were approximately 30%. Unfortunately, he remarks, modern ERP systems are not able to handle order crossovers effectively.

In a recent work, Babaï et al. (2009) analyze a dynamic re-order point control policy for a single-stage, single-item inventory system with non-stationary demand and lead-time uncertainty. To the best of our knowledge, there is no complete or heuristic approach in the literature that addresses the computation of (R^n,S^n) policy parameters under stochastic supplier lead time and service level constraints. Computing optimal policy parameters under these assumptions is a hard problem from a computational point of view. We argue that incorporating both a non-stationary stochastic demand and a stochastic supplier lead time—without assuming that orders do not cross in time—in an optimization model is a relevant and novel contribution.

In this work, we propose a stochastic constraint programming (Walsh, 2002) model for computing optimal (R^n,S^n) policy parameters under service level constraints and stochastic supplier lead times. In stochastic constraint programming, complex non-linear relations among decision and stochastic variables—such as the chance-constraints that enforce the required service level—can be effectively modeled by means of *global chance-constraints* (Hnich et al., 2009). Examples of global chance-constraints applied to inventory control problems can be found in Rossi et al. (2008) and Tarim et al. (2009). Our model incorporates a dedicated global chance-constraint that enforces, for each replenishment cycle scheduled, the required non-stockout probability. The model is tested on a set of instances that are solved to optimality under a discrete stochastic supplier lead time with known distribution.

The paper is organized as follows. In Section 2 we provide the formal definition of the problem and we discuss the working assumptions. In Section 3 we provide a deterministic reformulation for the chance-constraints that enforce the required service level. In Section 4 we introduce stochastic constraint programming and we discuss how it is possible to embed the deterministic reformulation of the chance-constraints within a global chance-constraint. This global chance-constraint is then enforced in the stochastic constraint programming model for computing the optimal policy parameters. In Section 5 we present our computational experience on a set of instances. Finally, in Section 6, we draw conclusions.

2. Problem definition

We consider the uncapacitated, single location, single product inventory problem with a finite planning horizon of N periods and a demand d_t for each period $t \in \{1,\dots,N\}$, which is a random variable with probability density function $g_t(d_t)$. We assume that the demand occurs instantaneously at the beginning of each time period. The demand we consider is non-stationary, that is it can vary from period to period, and we also assume that demands in different periods are independent.

Following Eppen and Martin (1988), an order placed in period t will be received after l_t periods, where l_t is a discrete random variable with probability mass function $f_t(\cdot)$. This means that an order placed in period t will be received after k periods with probability $f_t(k)$. We shall assume that there is a maximum lead-time L for which $\sum_{k=0}^{L} f_t(k)=1$. Therefore the possible lead-time lengths are limited to $\Lambda=\{0,\dots,L\}$ and the probability mass function is defined on the finite set Λ. Note that lead-times are mutually independent and each of them is also independent of the respective order quantity.

A fixed delivery cost a is incurred for each order. A linear holding cost h is incurred for each unit of product carried in stock from one period to the next. Without loss of generality, we will adopt the following assumption that concerns the accounting of inventory holding costs: we will charge an inventory holding cost at the end of each period based on the current inventory position, rather than the current inventory level. This will reflect the fact that interests are charged not only on the actual amount of items in stock, but also on outstanding orders. Doing so often makes sense since companies may assess holding cost on their total invested capital and not simply on items in stock. A further and detailed justification for this can be found in Hunt (1965).

We assume that it is not possible to sell back excess items to the vendor at the end of a period and that negative orders are not allowed, so that if the actual stock exceeds the order-up-to-level for that review, this excess stock is carried forward and not returned to the supply source. However, such occurrences are regarded as rare events (see the discussion in Bookbinder and Tan, 1988 and Tarim and Kingsman, 2004) and accordingly the cost of carrying this excess stock and its effect on the service levels of subsequent periods are ignored.

As a service level constraint we require that, with a probability of at least a given value α, at the end of each period the net inventory will be non-negative. Our aim is to minimize the expected total cost, which is composed of ordering costs and holding costs, over the N-period planning horizon, satisfying the service level constraints by fixing the future replenishment periods and the corresponding order-up-to-levels at the beginning of the planning horizon.

The actual sequence of actions is adopted from Kaplan (1970). At the beginning of a period, the inventory on hand after all the demands from previous periods have been realized is known. Since we are assuming complete backlogging, this quantity may be negative. Also known are orders placed in previous periods which have not been delivered yet. On the basis of this information, an ordering decision is made for the current period. All the deliveries that are to be made during a period are assumed to be made immediately after this ordering decision and hence are on hand at the beginning of the period. In summary, there are three successive events at the beginning of each period. First, stock on hand and outstanding orders are determined. Second, an ordering decision is made on the basis of this information. Third, all supplier deliveries for the current period, possibly including the most recent orders, are received.

3. Non-stationary stochastic lead-time

Let us denote the inventory position (the total amount of stock on hand plus outstanding orders minus back-orders) at the end of period t as P_t. It directly follows that

$$P_t = I_t + \sum_{\{k|1\le k\le t, l_k+k>t\}} X_k, \qquad (1)$$

Please cite this article as: Rossi, R., et al., Computing the non-stationary replenishment cycle inventory policy under stochastic supplier lead-times. International Journal of Production Economics (2010), doi:10.1016/j.ijpe.2010.05.011

where I_t is the inventory level (stock on hand minus back-orders) at the end of period t, X_k is the size of the replenishment order placed in period k, $X_k \geq 0$ (received in period $k+l_k$), and it is assumed that I_0 equals the initial inventory.

The general chance-constrained programming model for the problem described in Section 2 is given below. The reader is referred to Bookbinder and Tan (1988) for the zero lead-time version of this problem.

$$\min \quad E\{TC\} = \int_{d_1} \dots \int_{d_N} \sum_{t=1}^{N} (a\delta_t + hP_t)$$

$$g_1(d_1)\dots g_N(d_N) d(d_1)\dots d(d_N), \tag{2}$$

subject to

$$\delta_t = \begin{cases} 1 & \text{if } X_t > 0, \\ 0 & \text{otherwise,} \end{cases} \quad t = 1,\dots,N, \tag{3}$$

$$P_t = I_0 + \sum_{k=1}^{t} (X_k - d_k) \quad t = 1,\dots,N, \tag{4}$$

$$\Pr\left\{ P_t \geq \sum_{\{k|1 \leq k \leq t, l_k > t-k\}} X_k \right\} \geq \alpha \quad t = L+1,\dots,N, \tag{5}$$

$$P_t \in \mathbb{R}, \quad X_t \geq 0, \ t = 1,\dots,N, \tag{6}$$

where we comply with the following notation:

$E\{.\}$ the expectation operator
TC total cost
d_t the demand in period t, a random variable with probability density function, $g_t\,(d_t)$
a the fixed ordering cost (incurred when an order is placed)
h the proportional stock holding cost
l_t the lead-time length of the order placed in period t, a discrete random variable with a probability mass function $f_t(\cdot)$
δ_t a $\{0,1\}$ variable that takes the value of 1 if a replenishment occurs in period t and 0 otherwise

the objective function (Eq. (2)) minimizes the expected total ordering and inventory holding cost. It should be noted that, by charging holding cost on the inventory position rather than on the inventory level, the objective function becomes particularly simple and it resembles the one employed when the lead time is zero. Eq. (3) states that if a replenishment occurs in period t—i.e. the order quantity X_t is greater than 0—then the corresponding indicator variable δ_t must take value 1. Eq. (4) enforces the inventory conservation constraint for each period t. This constraint is expressed in terms of the inventory position P_t. Eq. (5) enforces the required service level in each period t, and it is also expressed in terms of the inventory position P_t. Finally Eq. (6) states that the inventory position in each period may either be zero or take any positive/negative value (i.e. full backorders) and that the order quantity is forced to be greater or equal to 0.

Note that depending on the probabilities assigned to each lead time length by the probability mass function, it may not be possible, in general, to provide the required service level for some initial periods. Nevertheless, by reasoning on a worst case scenario, it will always be possible to provide the required service level α starting from period $L+1$. Hence, the service level constraints are enforced in periods $L+1,\dots,N$ (see Eq. (5)).

Consider a review schedule, which has m reviews over the N period planning horizon with orders placed at $T_1,T_2,\dots,T_m$, where $T_i < T_{i+1}$. In order to incorporate the "replenishment cycle policy" into this model, we express the whole model in terms of a new set of decision variables, R_{T_i}, $i=1,\dots,m$. Define

$$P_t = R_{T_i} - \sum_{k=T_i}^{t} d_k, \ T_i \leq t < T_{i+1}, \quad i = 1,\dots,m, \tag{7}$$

where R_{T_i} ("order-up-to-position") can be interpreted as the inventory position up to which inventory should be raised after placing an order at the ith review period T_i. By doing so, order quantities X_t have to be decided only after the demands in the former periods have been realized. Under such a policy the orders X_t are all equal to zero except at replenishment periods $T_1,T_2,\dots,T_m$.

The service level constraint has to be expressed as a relation between the order-up-to-positions such that the overall service level provided at the end of each period is at least α. In order to express this service level constraint we propose a scenario-based approach over the discrete random variables l_t, $t=1,\dots,N$. In a scenario-based approach (Birge and Louveaux, 1997; Tarim et al., 2006), a scenario tree is generated which incorporates all possible realisations of discrete random variables into the model explicitly, yielding a fully deterministic model under the non-anticipativity constraints.

In our problem we can divide random variables into two sets: the random variables $\{l_t|t=1,\dots,N\}$, which represent lead-times, and the random variables $\{d_t|t=1,\dots,N\}$, which represent demands. We deal with each set in a separate fashion, by employing a scenario-based approach for the l_t and a deterministic equivalent modeling approach for the d_t variables. This is possible since under a given scenario discrete random variables are treated as constants. The problem is then reduced to the general multi-period production/inventory problem with dynamic deterministic lead-times and stochastic demands. It should be noted that, although it has been assumed that the supplier lead-time is zero in Tarim and Kingsman (2004), it is possible to extend their model for the non-zero lead-time situation without any loss of generality when the lead time is deterministic and remains constant for each order. In the Appendix we show how to model the situation in which the lead time is deterministic and dynamic (i.e. it may take a different deterministic value in each period). This more general situation corresponds to what is observed within any given scenario.

A scenario ω_t is a possible lead-time realization for all the orders placed up to period t in a given review schedule. We denote the probability of a scenario ω_t as $\Pr\{\omega_t\}$. Let $l_{T_i}(\omega_t)$ be the realized lead-time in scenario ω_t for the order placed in period T_i, where $i=1,\dots,m$. Finally, let Ω_t be the set of all the possible scenarios ω_t. Note that $\sum_{\Omega_t} \Pr\{\omega_t\} = 1$ for all $t=1,\dots,N$. We define $T_{p(t)}$ as the latest review before period t in the planning horizon, for which we are sure that all the former orders, including the one placed in $T_{p(t)}$, have been delivered within period t. Under the assumption that the probability mass function $f_t(\cdot)$ is defined on a finite set Λ, the index $p(t)$ provides a bound for the scenario tree size. In fact if the possible lead-time lengths in Λ are $0,\dots,L$, the earliest order that is delivered in period t with probability 1 under every possible scenario ω_t is the latest placed in the span $1,\dots,t-L$. Therefore since each scenario ω_t identifies the orders that have been received before or in period t, it directly follows that the number of scenarios in the tree that is needed to compute the order-up-to-positions for periods $t-L,\dots,t$ under any possible review schedule is at most 2^L, when we place $L+1$ orders in periods $t-L,\ldots,t$, but it may be lower if fewer reviews are planned. In order to clarify this concept, a small numerical example is provided in the Appendix.

The service level constraint at period t is always a relation over at most $L+1$ decision variables R_{T_i} that represent the order-up-to-positions of the replenishment cycles covering the span $t-L,\ldots,t$. Let $p_\omega(t)$ be the value of $p(t)$ under a given scenario ω_t when a review schedule is considered. In order to satisfy the service level constraints in our original model, we require that the overall service level under all the possible scenarios for each set of at most $L+1$ decision variables is at least α or equivalently,

$$\sum_{\omega_t \in \Omega_t} \Pr\{\omega_t\} \cdot G_S\left(R_{T_{p_\omega(t)}} + \sum_{\{i|i > p_\omega(t), l_{T_i}(\omega_t) \le t-T_i\}} (R_{T_i} - R_{T_{i-1}})\right) \ge \alpha, \quad t = L+1,\ldots,N, \tag{8}$$

where $S = \sum_{k=T_{p_\omega(t)}}^{t} d_k - \sum_{\{i|i > p_\omega(t), l_{T_i}(\omega_t) \le t-T_i\}} (d_{T_{i-1}} + \cdots + d_{T_i-1})$, and $G_S(\cdot)$ is the cumulative distribution function of S. Further details on the derivation of Eq. (8) are provided in the Appendix.

As the reader may notice, the service level constraints (Eq. (8)) are now fully deterministic constraints expressed only in terms of the order-up-to-positions, R_{T_i}. This makes it possible to replace throughout the rest of the model the P_t variables with their expected values $\tilde{P}_t$, as originally proposed in Bookbinder and Tan (1988), since these affect only the objective function in which we are considering expected values.

We can now express the whole model in terms of the new set of decision variables R_t, $t=1,\ldots,N$. If there is no replenishment scheduled for period t, that is if $\delta_t = 0$, then R_t must be equal to the expected closing-inventory-position in period $t-1$, that is $R_t = \tilde{P}_{t-1}$. If there is a review T_i in period t, R_t is simply the order-up-to-position, R_{T_i}, for this review. Therefore, the set of the desired order-up-to-positions, $\{R_{T_i}|i=1,\ldots,m\}$, as required for the solution to the problem, comprises those values of R_t for which $\delta_t = 1$.

Hence, the complete deterministic equivalent model under the replenishment cycle policy can be expressed as

$$\min \quad E\{TC\} = \sum_{t=1}^{N} (a\delta_t + h\tilde{P}_t) \tag{9}$$

subject to

$$\delta_t = 0 \Rightarrow R_t = \tilde{P}_{t-1} \quad t=1,\ldots,N, \tag{10}$$

$$R_t \ge \tilde{P}_{t-1} \quad t=1,\ldots,N, \tag{11}$$

$$R_t = \tilde{P}_t + \tilde{d}_t \quad t=1,\ldots,N, \tag{12}$$

Eq. (8)(service level constraints,

$$R_t \ge 0, \ \tilde{P}_t \ge 0, \ \delta_t \in \{0,1\} \ \ t=1,\ldots,N, \tag{13}$$

where $\{T_1,\ldots,T_m\} = \{t \in \{1,\ldots,N\} | \delta_t = 1\}$.

The model neatly resembles the original stochastic programming formulation. The reader can easily notice that, while the objective function and the remaining constraints in the model are now deterministic and linear—thus they can be easily modeled by means of existing mathematical programming packages—Eq. (8) is deterministic but non-linear and it cannot be implemented in a straightforward manner by using existing solvers. For this reason, in the following section, we will introduce a stochastic constraint programming formulation that we will employ to solve the above model.

4. A stochastic constraint programming approach

In this section, we aim to propose a stochastic constraint programming approach for modeling and solving the model discussed in the previous section. Firstly, we introduce the key concepts in constraint programming and stochastic constraint programming, the extension of constraint programming that deals with problems of decision making under uncertainty. Secondly, we introduce our stochastic constraint programming model.

4.1. Constraint reasoning

Constraint programming (CP) (Apt, 2003) is a declarative programming paradigm in which relations between decision variables are stated in the form of constraints. Informally speaking, constraints specify the properties of a solution to be found. The constraints used in constraint programming are of various kinds: logic constraints (i.e. "*x* or *y* is true", where *x* and *y* are boolean decision variables), linear constraints, and *global constraints* (Régin, 2003). A global constraint captures a relation among a non-fixed number of variables. One of the most well known global constraints is the **alldiff** constraint (Régin, 1994), that can be enforced on a certain set of decision variables in order to guarantee that no two variables are assigned the same value.

With each constraint, CP associates a *filtering algorithm* able to remove provably infeasible or suboptimal values from the domains of the decision variables that are constrained and, therefore, to enforce some degree of *consistency* (see Rossi et al., 2006). These filtering algorithms are repeatedly called until no more values are pruned. This process is called *constraint propagation*. In addition to constraints and filtering algorithms, constraint solvers also feature some sort of heuristic *search engine* (e.g. a backtracking algorithm). During the search, the constraint solver exploits filtering algorithms in order to proactively prune parts of the search space that cannot lead to a feasible or to an optimal solution.

Stochastic constraint programming (SCP) was first introduced in Walsh (2002) in order to model combinatorial decision problems involving uncertainty and probability. According to Walsh, SCP combines together the best features of CP (i.e. global constraints, search heuristics, filtering strategies, etc.) and of stochastic programming (Kall and Wallace, 1994) (i.e. stochastic variables, chance-constraints, etc.). In addition to decision variables, SCP features stochastic variables. Furthermore, in SCP it is possible to capture complex non-linear relations among decision and stochastic variables by means of *global chance-constraints* (Rossi et al., 2008; Hnich et al., 2009). Similarly to global constraints, global chance-constraints incorporate efficient strategies for performing logical inference on these relations during the search in order to enforce some degree of consistency through constraint propagation.

In what follows we will introduce an SCP model for computing (R^n,S^n) policy parameters under non-stationary stochastic demand, lead time, and service level constraints. In order to capture the service level constraints, a dedicated global chance-constraint and the respective propagation logic are introduced and incorporated in the SCP model.

Please cite this article as: Rossi, R., et al., Computing the non-stationary replenishment cycle inventory policy under stochastic supplier lead-times. International Journal of Production Economics (2010), doi:10.1016/j.ijpe.2010.05.011

4.2. A stochastic constraint programming model

We now present an SCP formulation for computing (R^n,S^n) policy parameters under stochastic lead times. Results from Section 3 will be employed in the SCP formulation. More specifically, in order to model the service level constraint (Eq. (8)), a new global chance-constraint, serviceLevel(·), will be defined. Such a constraint is needed to dynamically compute the correct expected closing-inventory-positions $\{\tilde{P}_t | t=1,\dots,N\}$ on the basis of the current replenishment plan, that is $\{\delta_t | t=1,\dots,N\}$ assignments.

The SCP model that incorporates our dedicated global chance-constraint is therefore

$$\min \quad E\{TC\} = \sum_{t=1}^{N} (a \cdot \delta_t + h \cdot \tilde{P}_t) \tag{14}$$

subject to

$$\delta_t = 0 \Rightarrow \tilde{P}_t + \tilde{d}_t - \tilde{P}_{t-1} = 0 \quad t=1,\dots,N, \tag{15}$$

$$\tilde{P}_t + \tilde{d}_t - \tilde{P}_{t-1} \geq 0 \quad t=1,\dots,N, \tag{16}$$

$$serviceLevel(\delta_1,\dots,\delta_N,\tilde{P}_1,\dots,\tilde{P}_N,g_1(d_1),\dots,g_N(d_N),f(\cdot),\alpha), \tag{17}$$

$$\tilde{P}_t \geq 0, \quad \delta_t \in \{0,1\} \quad t=1,\dots,N. \tag{18}$$

It should be noted that the domain of each $\tilde{P}_t$ variable—as in the zero lead time case (see Tarim and Smith, 2008)—is limited. In fact, since the period demand variance is additive, the uncertainty can only increase in the length of a replenishment cycle. Therefore the longer a cycle is, the higher are the inventory levels that are required to achieve a certain service level. It directly follows that a single replenishment covering the whole planning horizon will provide upper bounds for the expected period closing-inventory-positions throughout the horizon.

We now describe the signature of the new constraint we have introduced. serviceLevel(·) describes a relation between all the decision variables in the model. It also accepts as parameters the distribution of the demand in each period t, $g(d_t)$; the probability mass function of the lead time $f(\cdot)$, which, without loss of generality, is here assumed to be the same for all the periods; and the required service level α.

A high-level pseudo-code for the propagation logic of serviceLevel(·) is presented in Algorithm 1. Note that to keep the description of the algorithm simple we assume here a stochastic lead time l with probability mass function $f(l)$ in every period. The maximum lead time length is L.

In order to propagate this constraint, we consider every set of consecutive replenishment cycles covering at least L+1 periods (that is the one of interest plus L former periods) and having the smallest possible cardinality in terms of replenishment cycle number (Algorithm 1, line 5). Obviously, to identify such a group of cycles, we have to wait until, during the search, a subset of consecutive δ_t variables is assigned (Algorithm 1, line 10). Then, in order to verify if the service level constraint is satisfied for the last period in this group, we check that for each replenishment cycle in the group identified at least one decision variable $\tilde{P}_t$ is assigned (Procedure checkBuffers, line 3 and line 22). If this is the case the partial policy for the span is completely defined and, by recalling that $R_t = \tilde{P}_t + \tilde{d}_t$, its feasibility can be checked by using the condition in Eq. (8) (Procedure checkBuffers, line 25). If the condition is not satisfied we backtrack (Procedure checkBuffers, line 26). Notice that such a condition involves for each period only a subset of all the decision variables in the model, which means that our constraint is able to detect infeasible partial assignments, i.e. *nogoods* (Rossi et al., 2006).

Finally, it should be emphasized that, during the search, any CP solver will be able to exploit constraint propagation and detect infeasible or suboptimal assignments with respect to the other constraints in the model. Furthermore, suboptimal solutions may be pruned by using dedicated *cost-based filtering* methods (Focacci et al., 1999; Tarim et al., 2009).

Algorithm 1. propagate

```
   input: $\delta_1,\dots,\delta_N$, $\tilde{P}_1,\dots,\tilde{P}_N$, $\alpha$, $d_1,\dots,d_N$, $l$, $L$, $N$
1  begin
2    cycles ← {};
3    pointer ← 1;
4    periods ← 0;
5    For each period $i$ in $2,\dots,N$ do
6      if $\delta_i$ is not assigned then
7        cycles ← {};
8        periods ← 0;
9        pointer = −1;
10     elseif $\delta_i$ is assigned to 1 then
11       if pointer ≠ −1 then
12         cycle ← a replenishment cycle over {pointer, ..., $i$−1};
13         add cycle to cycles;
14       if periods ≥ $L$ then
15         checkBuffers();
16
17       pointer ← $i$;
18       periods ← periods + 1;
19     else
20       periods ← periods + 1;
21
22   if pointer ≠ −1 then
23     cycle ← a replenishment cycle over {pointer, ..., $N$};
24     add cycle to cycles;
       if periods ≥ $L$ then
         checkBuffers();
25 end
```

Table 1
Optimal solution.

$E\{TC\}$: 356					
Period (t)	1	2	3	4	5
$\tilde{d}_t$	36	28	42	33	30
R_t	125	124	129	87	55
δ_t	1	1	1	1	1
Shortage probability	–	–	5%	5%	5%

Please cite this article as: Rossi, R., et al., Computing the non-stationary replenishment cycle inventory policy under stochastic supplier lead-times. International Journal of Production Economics (2010), doi:10.1016/j.ijpe.2010.05.011

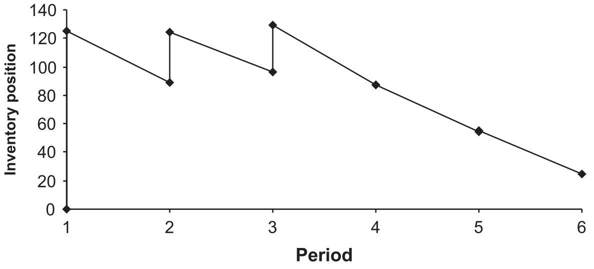


Fig. 1. Optimal policy under stochastic lead time, $f_t(k)$={0.3, 0.2, 0.5}.

4.2.1. An example

We assume an initial null inventory level and a normally distributed demand with a coefficient of variation $\sigma_t/\tilde{d}_t = 0.3$ for each period $t \in \{1,\ldots,5\}$. The expected values for the demand in each period are: {36, 28, 42, 33, 30}. The other parameters are a=1, h=1, $\alpha = 0.95$. We consider for every period t in the planning horizon the following lead time probability mass function $f_t(k)$={0.3(0), 0.2(1), 0.5(2)}, which means that we receive an order placed in period t after {0,…,2} periods with the given probability (0 periods: 30%; 1 period: 20%; 2 periods: 50%). It is obvious that in this case we will always receive the order at most after 2 periods. In Table 1 (Fig. 1) we show the optimal solution found by the SCP model. We now want to show that the order-up-to-positions—computed in this example by using Eq. (8)—satisfy every service level constraint in the model. We assume that for the first 2 periods no service level constraint is enforced, since it is not possible to fully control the inventory in the first 2 periods. Therefore we enforce the required service level on periods 3, 4 and 5, that is Eq. (8) for t=3,…,N. Let us verify that the given order-up-to levels satisfy this condition for each of these three periods. Since we know the probability mass function $f_t(\cdot)$ for each period t in the planning horizon we can easily compute the probability $\Pr(\omega_t)$ for each scenario $\omega_t \in \Omega_t$. We have four of these scenarios for each period $t \in \{3,\ldots,N\}$, since we are placing an order in every period:

- S_1, $\Pr\{S_1\} = 0.15 = (0.3+0.2)0.3$; in this scenario at period t all the orders placed are received. That is the order placed in period $t-1$ is received immediately (probability 0.3), or after one period (probability 0.2), while the order placed in period t is received immediately (probability 0.3)
- S_2, $\Pr\{S_2\} = 0.35 = (0.3+0.2)(0.2+0.5)$; in this scenario at period t we do not receive the last order placed in period t. That is the order placed in period $t-1$ is received immediately (probability 0.3), or after one period (probability 0.2), while the order placed in period t is not received immediately, therefore it is received after one period (probability 0.2), or after two periods (probability 0.5)
- S_3, $\Pr\{S_3\} = 0.35 = 0.5(0.2+0.5)$; in this scenario at period t we do not receive the last two orders placed in periods t and $t-1$. That is the order placed in period $t-1$ is received after two periods (probability 0.5), and the order placed in period t is not received immediately, therefore it is received after one period (probability 0.2), or after two periods (probability 0.5)
- S_4, $\Pr\{S_4\} = 0.15 = 0.5 \cdot 0.3$; in this scenario at period t we do not receive the order placed in period $t-1$ and we observe order-crossover. That is the order placed in period $t-1$ is received after two periods (probability 0.5), and the order placed in period t is received immediately (probability 0.3)

Procedure 1. checkBuffers

```
begin
  cycle ← the last element in cycles,
  a replenishment cycle over {i, . . . ,j};
  if no decision variable $\tilde{P}_i, \ldots, \tilde{P}_j$ is assigned then
    return;
  counter ← 1;
  For each period t covered by cycle do
    formerCycles ← cycles;
    remove cycle from formerCycles;
    coveredPeriods ← the number of periods
    covered by cycles in formerCycles;
    head ← first element in formerCycles;
    headLength ← periods covered by head;
    if counter < L then
    while coveredPeriods−headLength+counter ≥ L do
      remove head from formerCycles;
      head ← first element in formerCycles;
      headLength ← periods covered by head;
    else
      formerCycles ← {};
    condition ← true;
    For each cycle c in formerCycles do
      let {m, . . . ,n} be the periods covered by c;
      if no decision variable $\tilde{P}_m, \ldots, \tilde{P}_n$ is assigned then
        condition ← false;

    if condition then
      if Eq.(8) for period t n cycle
      and former replenishment
      cycles in formerCycles is not satisfied then
        backtrack();
    counter ← counter+1;
end
```

In the described scenarios every possible configuration is considered. We do this without any loss of generality. In fact if some of the configurations are unrealistic (for instance if we assume that order-crossover may not take place) we need only to set the probability of the respective scenario to zero. Now it is possible to write Eq. (8) for each period $t \in \{3,\ldots,N\}$. Consider period 3:

$$\Pr\{S_1\} \cdot G\left(\frac{129-42}{0.3\sqrt{42^2}}\right) + \Pr\{S_2\} \cdot G\left(\frac{124-(28+42)}{0.3\sqrt{28^2+42^2}}\right) + \Pr\{S_3\} \cdot G\left(\frac{125-(36+28+42)}{0.3\sqrt{36^2+28^2+42^2}}\right) + \Pr\{S_4\} \cdot G\left(\frac{125+(129-124)-(36+42)}{0.3\sqrt{36^2+42^2}}\right) = 94.60\% \cong 95\% \tag{19}$$

where $G(\cdot)$ is the standard normal distribution function. This means that the combined effect of order delivery delays in our policy, when all the possible scenarios are taken into account, gives a no-stock-out probability of about 95% for period 3. Similar reasoning can be employed to verify that the given solution satisfies the required service level also for period $t \in \{4,5\}$.

Please cite this article as: Rossi, R., et al., Computing the non-stationary replenishment cycle inventory policy under stochastic supplier lead-times. International Journal of Production Economics (2010), doi:10.1016/j.ijpe.2010.05.011

The reader may notice that, since we are placing an order in every period and since the lead time is at most of two periods, the service level in any given period is only influenced by the replenishment in such a period and by the last two replenishments. For instance, the service level in period 4 is only influenced by the order-up-to-position in periods 3 and 2. Let us consider the partial assignment in Table 2. The shortage probability in period 4 is greater than the required 5% therefore this partial assignment constitutes a *nogood*. As soon as our global chance-constraint detects this partial assignment during the search, it will immediately trigger a backtrack and it will prevent the CP solver from exploring any assignment that extends such a partial assignment.

5. Computational experience

In this section we solve to optimality an 8-period inventory problem under stochastic demand and lead time. Different lead time configurations are considered. The stochastic, deterministic and zero lead time cases are compared. As in the previous example we assume an initial null inventory level and a normally distributed demand with a coefficient of variation $\sigma_t/\tilde{d}_t=0.3$ for each period $t\in\{1,\ldots,8\}$. The expected value $\tilde{d}_t$ for the demand in each period $t=1,\ldots,N$ are listed in Table 3. The other parameters are $a=30$, $h=1$, $\alpha=0.95$. Initially, we consider the problem under stochastic demand and no lead time. An efficient CP approach to find policy parameters in this case was presented in Tarim and Smith (2008) and Tarim et al. (2009). Obviously our approach is general and can provide solutions for this case as well, although less efficiently. The optimal solution for the instance considered is presented in Fig. 2; details about the optimal policy are reported in Table 4. We observe five replenishment cycles; policy parameters are: cycle lengths =[1, 2, 1, 2, 2] and order-up-to-positions =[72, 42, 49, 65, 52]. The shortage probability is at most 5%, therefore the service level is met in every period. The $E\{TC\}$ is 303.

Table 2
A partial assignment and the respective shortage probability in period 4. The dashes, "–", are used to denote decision variables that have not been assigned yet.

$E\{TC\}$: 211 (lower bound)					
Period (t)	1	2	3	4	5
$\tilde{d}_t$	36	28	42	33	30
R_t	–	124	100	87	–
δ_t	–	1	1	1	–
Shortage probability				6%	

Table 3
Forecasts of period demands.

Period (t)	1	2	3	4	5	6	7	8
$\tilde{d}_t$	15	18	13	33	30	18	23	15

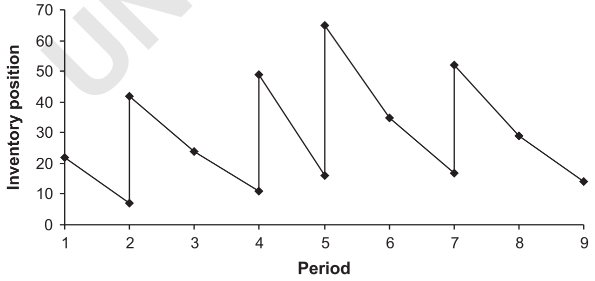


Fig. 2. Optimal policy under no lead time.

We now consider the same instance, but with a deterministic lead time of one period. The optimal solution is presented in Fig. 3; details about the optimal policy are reported in Table 5. We observe now only four replenishment cycles; policy parameters are: cycle lengths=[2, 1, 2, 3] and order-up-to-positions=[59, 64, 105, 72]. Again the shortage probability is at most 5% in every period, which means that the service level constraint is met. The $E\{TC\}$ is 456. Therefore we observe now an expected total cost that is 50.5% higher than the zero lead time case. The replenishment plan is significantly affected by the lead time both in term of replenishment cycle lengths and order-up-to-positions.

When a deterministic lead time of two periods is considered, as the reader may expect, we observe again higher costs and a different replenishment policy. The optimal solution is presented in Fig. 4; details about the optimal policy are reported in Table 6. The number of replenishment cycles is now again 5; policy parameters are: cycle lengths=[1, 1, 2, 1, 3] and order-up-to-positions=[59, 84, 119, 92, 72]. The service level constraint is met in every period. The $E\{TC\}$ is 602. This means that we observe a cost 98.6% and 32.0% higher than respectively the zero lead time case and the one period lead time case. The replenishment plan is again completely modified as a consequence of the lead time length.

We now concentrate on two instances where a stochastic lead time is considered and we compare results with the former cases. Firstly we analyze a stochastic lead time with probability mass function f_t(k)={0.2(0), 0.6(1), 0.2(2)}. That is an order is received

Table 4
Optimal policy under no lead time.

$E\{TC\}$: 303								
Period (t)	1	2	3	4	5	6	7	8
R_t	22	42	24	49	65	35	52	29
δ_t	1	1	0	1	1	0	1	0
Shortage probability	5%	0%	5%	5%	0%	5%	0%	5%

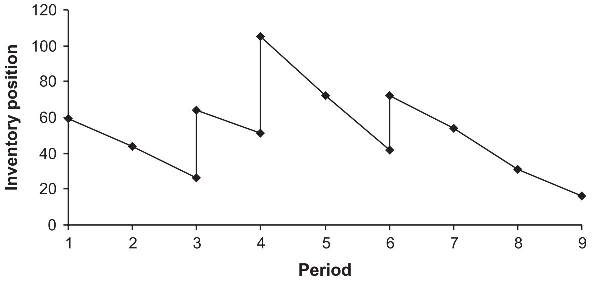


Fig. 3. Optimal policy under deterministic one period lead time.

Table 5
Optimal policy under deterministic one period lead time, notice that the service level in the first period can obviously not be controlled.

$E\{TC\}$: 456								
Period (t)	1	2	3	4	5	6	7	8
R_t	59	44	64	105	72	72	54	31
δ_t	1	0	1	1	0	1	0	0
Shortage probability	–	0%	5%	5%	0%	5%	0%	5%

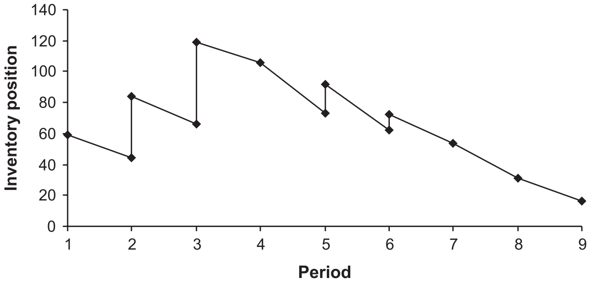


Fig. 4. Optimal policy under deterministic two-period lead time.

Table 6
Optimal policy under deterministic two-period lead time.

$E\{TC\}$: 602								
Period (t)	1	2	3	4	5	6	7	8
R_t	59	84	119	106	92	72	54	31
δ_t	1	1	1	0	1	1	0	0
Shortage probability	–	–	5%	5%	0%	5%	5%	5%

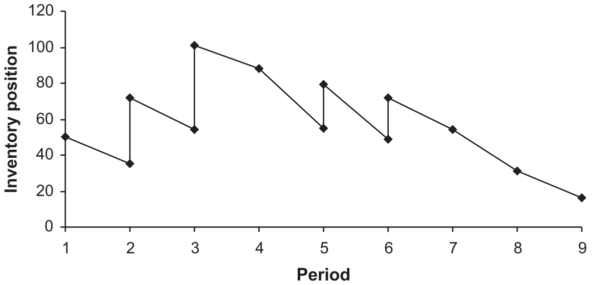


Fig. 5. Optimal policy under stochastic lead time, $f_t(k)=\{0.2(0), 0.6(1), 0.2(2)\}$.

Table 7
Optimal policy under stochastic lead time, $f_t(k)=\{0.2(0), 0.6(1), 0.2(2)\}$, in periods {1, 2} the inventory cannot be controlled.

$E\{TC\}$: 532								
Period (t)	1	2	3	4	5	6	7	8
R_t	50	72	101	88	79	72	54	31
δ_t	1	1	1	0	1	1	0	0
Shortage probability	–	–	5%	5%	3%	5%	5%	5%

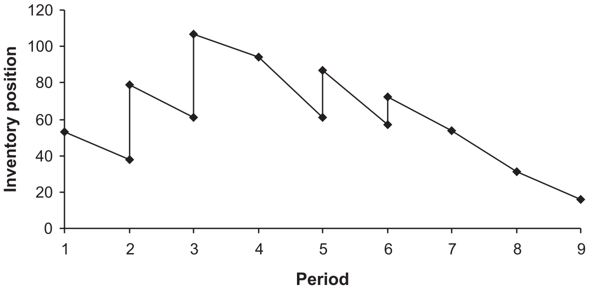


Fig. 6. Optimal policy under stochastic lead time, $f_i(t)=\{0.5(0), 0.0(1), 0.5(2)\}$.

Table 8
Optimal policy under stochastic lead time, $f_i(t)=\{0.5(0), 0.0(1), 0.5(2)\}$.

$E\{TC\}$: 562								
Period (t)	1	2	3	4	5	6	7	8
R_t	53	79	107	94	87	72	54	31
δ_t	1	1	1	0	1	1	0	0
Shortage probability	–	–	5%	5%	0%	5%	5%	5%

immediately with probability 0.2, after one period with probability 0.6, and after two periods with probability 0.2. The optimal solution is presented in Fig. 5; details about the optimal policy are reported in Table 7. The number of replenishment cycles is again 5, as in the two-period lead time case; policy parameters are: cycle lengths=[1, 1, 2, 1, 3] and order-up-to-positions=[50, 72, 101, 79, 72]. Therefore we see that the number and the length of replenishment cycles does not change from the deterministic two-period lead time case, although we observe lower order-up-to-positions as we may expect since the lead time is in average one period therefore lower than in the former case. Also the cost reflects this, in fact it is 11.6% lower than in the two-period deterministic lead time case. It should be noted that the uncertainty of the lead time plays a significant role, in fact although the average lead time is one period, the structure of the policy resembles much more the one under a two-period deterministic lead time than the one under a deterministic one period lead time. Moreover the expected total cost is 16.6% higher than in this latter case.

We finally consider a different probability mass function for the lead time: $f_t(k)=\{0.5(0), 0.0(1), 0.5(2)\}$, which means that we maintain the same average lead time of one period, but we increase its variance. The optimal solution is presented in Fig. 6; details about the optimal policy are reported in Table 8. The number of replenishment cycles is still 5; policy parameters are: cycle lengths=[1, 1, 2, 1, 3] and order-up-to-positions=[50, 72, 101, 79, 72]. Although the average lead time is still one period, order-up-to-positions are slightly higher than in the former case where the variance of the lead time was lower. Also the cost reflects this, in fact it is 5.6% higher than in the former case, but this is still lower than the expected total cost of the two-period deterministic lead time case.

To summarize, in our experiments we saw that supplier lead time uncertainty may significantly affect the structure of the optimal (R^n,S^n) policy. Computing optimal policy parameters constitutes a hard computational and theoretical challenge. Under different degrees of lead time uncertainty, when other input parameters for the problem remain fixed, order-up-to-positions and reorder points in the optimal policy change significantly. Deciding what the optimal decisions are for certain input parameters is a counterintuitive task. Our approach provides a systematic way to compute optimal policy parameters.

6. Conclusions

A novel approach for computing replenishment cycle policy parameters under non-stationary stochastic demand, stochastic lead time and service level constraints has been presented. The approach is based on SCP and it employs a dedicated global chance-constraint in order to enforce the required service level in each period. The assumptions under which we developed our approach for the stochastic lead time case proved to be less restrictive than those commonly adopted in the literature for complete methods. In particular we faced the problem of order-crossover, which is a very active research topic. Our approach merged well-known concepts such as deterministic equivalent

Please cite this article as: Rossi, R., et al., Computing the non-stationary replenishment cycle inventory policy under stochastic supplier lead-times. International Journal of Production Economics (2010), doi:10.1016/j.ijpe.2010.05.011

modeling of chance-constraints and scenario-based modeling. Our computational experience showed that a stochastic supplier lead time may significantly impact the structure and the cost of the optimal replenishment cycle policy with respect to the case in which the lead time is deterministic or absent. In our future research, we aim to develop dedicated cost-based filtering algorithms able to significantly speed up the search for the optimal policy parameters.

Appendix A

A.1. Deterministic equivalent non-linear formulation of the service level constraints

We discuss the main steps required to derive the deterministic equivalent non-linear formulation of the service level constraints (Eq. (8)).

To begin, we discuss how to obtain a deterministic equivalent formulation for the chance-constraints that enforce the required service level when the lead time in each period varies and assumes a given deterministic value. The same reasoning is then easily generalized to the case in which the lead time is stochastic and assumes a different distribution from period to period.

When a dynamic deterministic lead time $L_t \geq 0$ is considered in each period $t=1,\ldots,N$, an order placed in period t will be received only at period $t+L_t$, that is

$$I_t = I_0 + \sum_{\{k|k \geq 1, L_k + k \leq t\}} X_k - \sum_{k=1}^{t} d_k \quad t = 1,\ldots,N. \tag{20}$$

Let us recall that the inventory position, P_t, represents the total amount of inventory on-hand plus outstanding orders minus backorders at the end of period t. It directly follows that

$$P_t = I_t + \sum_{\{k|1 \leq k \leq t, L_k + k > t\}} X_k, \tag{21}$$

where we assume $P_0 = I_0$. It is easy, then, to reformulate the model using the inventory position.

Next, we modify the general stochastic programming formulation in order to incorporate the "replenishment cycle policy". Consider a review schedule, which has m reviews over the N period planning horizon with orders placed at $\{T_1, T_2, \ldots, T_m\}$, where $T_i > T_{i-1}$, $T_m \leq N - L_{T_m}$. For convenience, T_1 is defined as the start of the planning horizon and $T_{m+1} = N+1$ as the period immediately after the end of the planning horizon.[2] The associated inventory reviews will take place at the beginning of periods T_i, $i_1,\ldots,m$. In the replenishment cycle policy considered here, clearly the orders X_t are all equal to zero except at replenishment periods $T_1, T_2, \ldots, T_m$. The inventory level I_t carried from period t to period $t+1$ is the opening inventory plus any orders that have arrived up to and including period t less the total demand to date. Hence, the inventory balance equation becomes,

$$I_t = I_0 + \sum_{\{i|L_{T_i} + T_i \leq t\}} X_{T_i} - \sum_{k=1}^{t} d_k, \quad t = 1,\ldots,N. \tag{22}$$

Define $T_{p(t)}$ as the latest review before period t in the planning horizon, for which all the former orders, including the one placed in $T_{p(t)}$, are delivered within period t, therefore

$$p(t) = \max\{i | \forall j \in \{1,\ldots,i\}, T_j + L_{T_j} \leq t, \quad i = 1,\ldots,m\}, \tag{23}$$

for all $t=1,\ldots,N$. The inventory level I_t at the end of period t (Eq. (22)) can be expressed as

$$I_t = I_0 + \sum_{i=1}^{p(t)} X_{T_i} + \sum_{\{i|i > p(t), L_{T_i} + T_i \leq t\}} X_{T_i} - \sum_{k=1}^{t} d_k, \quad t = 1,\ldots,N. \tag{24}$$

We now want to reformulate the constraints of the chance-constrained model in terms of a new set of decision variables R_{T_i}, $i=1,\ldots,m$. Define,

$$P_t = R_{T_i} - \sum_{k=T_i}^{t} d_k, \quad T_i \leq t < T_{i+1}, \quad i = 1,\ldots,m, \tag{25}$$

where R_{T_i} can be interpreted as the inventory position up to which inventory should be raised after placing an order at the ith review period T_i. We can now express the whole model in terms of these new decision variables R_{T_i}. The new problem is to determine the number of reviews, m, the T_i, and the associated R_{T_i} for $i=1,\ldots,m$.

Let us now express Eq. (24) using R_{T_i} as decision variables

$$I_t = R_{T_{p(t)}} + \sum_{\{i|i > p(t), L_{T_i} + T_i \leq t\}} (R_{T_i} - R_{T_{i-1}} + d_{T_{i-1}} + \cdots + d_{T_i - 1}) - \sum_{k=T_{p(t)}}^{t} d_k, \quad t = 1,\ldots,N. \tag{26}$$

As mentioned earlier, α is the desired minimum probability that the net inventory level in any time period is non-negative. Depending on the values assigned to L_t it is obviously not possible to provide the required service level for some initial periods. In general, we provide the required service level α starting from the period t, for which the value $t+L_t$ is minimum. Let M be this period. Notice that, it will never be optimal to place any order in a period t such that $t+L_t > N$, since such an order will not be received within the given planning horizon.

By substituting I_t with the right hand term in Eq. (26) we obtain

$$G_S\left(R_{T_{p(t)}} + \sum_{\{i|i > p(t), L_{T_i} + T_i \leq t\}} (R_{T_i} - R_{T_{i-1}})\right) \geq \alpha, \quad t = M,\ldots,N. \tag{27}$$

where $S = \sum_{k=T_{p(t)}}^{t} d_k - \sum_{\{i|i > p(t), L_{T_i} + T_i \leq t\}} (d_{T_{i-1}} + \ldots + d_{T_i - 1})$, and $G_S(.)$ is the cumulative distribution function of S.

The service level constraints are now deterministic and they are expressed only in terms of the order-up-to-positions. Eq. (27) can be directly employed in order to obtain Eq. (8), under the original assumption that the lead time in each period $t \in \{1,\ldots,N\}$ is a discrete random variable l_t.

A.2. Scenarios for orders in the pipeline: a numerical example

Consider a planning horizon of $N=6$ periods. The probability mass function for the lead-time in each period $t=1,\ldots,6$ is $f_t(\cdot) = \{0(1/3), 1(1/3), 2(1/3)\}$, therefore an order will arrive immediately with probability 1/3, after one period with probability 1/3, and after 2 periods with probability 1/3. It follows that in our example $L=2$ and $f_t(\cdot)$ is defined on a finite set Λ that comprises 3 possible states. Let us now consider period $t=5$. Clearly, $T_{p(t)}=3$, in fact with probability 1.0 an order placed at period 3, as well as any other order placed at previous periods, is received by period 5. Under a review schedule that places an order in every period, there are $2^L=4$ possible scenarios for the remaining orders that have been delivered by period 5:

- S_1, $\Pr\{S_1\} = (1/3+1/3)1/3$; both the orders placed at periods 4 and 5 have been delivered by period 5;
- S_2, $\Pr\{S_2\} = (1/3+1/3)(1/3+1/3)$; the order placed at period 4 has been delivered by period 5, but not the one placed at period 5;

[2] The review schedule may be generalized to consider the case where $T_1 > 1$, if the opening inventory I_0 is sufficient to cover the immediate needs at the start of the planning horizon.

- S_3, $\Pr\{S_3\}=1/3\cdot1/3$; the order placed at period 5 has been delivered by period 5, but not the one placed at period 4;
- S_4, $\Pr\{S_3\}=1/3(1/3+1/3)$; the orders placed at periods 4 and 5 have not been delivered by period 5;

It is easy to see that under any other possible review schedule the number of scenarios to be considered for the orders that have been delivered by period 5 is less or equal to $2^L=4$. For instance, consider a review schedule in which orders are placed only in period 1, period 3, and period 5. In this case we only have 2 possible scenarios at period 5. As in the previous case, any order placed at period 3 or before will be received with probability 1.0 by period 5. No order is placed at period 4. The 2 scenarios for the remaining order are

- S_1, $\Pr\{S_1\}=1/3$; the order placed at period 5 has been delivered by period 5;
- S_2, $\Pr\{S_1\}=2/3$; the order placed at period 5 has not been delivered by period 5.

Q1

References

Apt, K., 2003. Principles of Constraint Programming. Cambridge University Press, Cambridge, UK.

Babaï, M.Z., Syntetos, A., Dallery, Y.Z., Nikolopoulos, K., 2009. Dynamic re-order point inventory control with lead-time uncertainty: analysis and empirical investigation. International Journal of Production Research 47 (9), 2461–2483.

Bashyam, S., Fu, M.C., 1998. Optimization of (s,S) inventory systems with random lead times and a service level constraint. Management Science 44 (12), 243–256.

Birge, J.R., Louveaux, F., 1997. Introduction to Stochastic Programming. Springer Verlag, New York.

Bookbinder, J.H., Tan, J.Y., 1988. Strategies for the probabilistic lot-sizing problem with service-level constraints. Management Science 34 (9), 1096–1108.

Eppen, G.D., Martin, R.K., 1988. Determining safety stock in the presence of stochastic lead time and demand. Management Science 34 (11), 1380–1390.

Focacci, F., Lodi, A., Milano, M, 1999. Cost-based domain filtering. In: Jaffar, J. (Ed.) Proceedings of Fifth International Conference on Principles and Practice of Constraint Programming—CP'99, Alexandria, Virginia, USA, October 11–14, Lecture Notes in Computer Science, Springer, vol. 1713, pp. 189–203.

Hayya, J.C., Bagchi, U., Kim, J.G., Sun, D., 2008. On static stochastic order crossover. International Journal of Production Economics 114 (1), 404–413.

Hayya, J.C., Xu, S.H., Ramasesh, R.V., He, X.X., 1995. Order crossover in inventory systems. Stochastic Models 11 (2), 279–309.

Hnich, B., Rossi, R., Tarim, S.A., Prestwich, S.D., 2009. Synthesizing filtering algorithms for global chance-constraints. In: Gent, I.P. (Ed.) Proceedings of 15th International Conference on Principles and Practice of Constraint Programming—CP 2009, Lisbon, Portugal, September 20–24, Lecture Notes in Computer Science, vol. 5732, Springer, pp. 439–453.

Hunt, J.A., 1965. Balancing accuracy and simplicity in determining reorder points. Management Science 12 (4), B94–B103.

Kall, P., Wallace, S.W., 1994. Stochastic Programming. Wiley.

Kaplan, R.S., 1970. A dynamic inventory model with stochastic lead times. Management Science 16 (7), 491–507.

Nevison, C., Burstein, M., 1984. The dynamic lot-size model with stochastic lead-times. Management Science 30 (1), 100–109.

Régin, J.-C., 1994. A filtering algorithm for constraints of difference in csps. In: Proceedings of the Twelfth National Conference on Artificial Intelligence, vol. 1, Seattle, Washington, American Association for Artificial Intelligence, pp. 362–367.

Régin, J.-C., 2003. Global constraints and filtering algorithms. In: Milano, M. (Ed.), Constraints and Integer Programming Combined. Kluwer.

Riezebos, J., 2006. Inventory order crossovers. International Journal of Production Economics 104 (2), 666–675.

Rossi, F., van Beek, P., Walsh, T., 2006. Handbook of constraint programming (foundations of artificial intelligence). Elsevier Science Inc., New York, NY, USA.

Rossi, R., Tarim, S.A., Hnich, B., Prestwich, S., 2008. A global chance-constraint for stochastic inventory systems under service level constraints. Constraints 13 (4), 490–517.

Silver, E.A., Pyke, D.F., Peterson, R., 1998. Inventory Management and Production Planning and Scheduling. Wiley, New York.

Speh, T.W., Wagenheim, G., 1978. Demand and lead-time uncertainty: the impacts of physical distribution performance and management. Journal of Business Logistics 1 (1), 95–113.

Tarim, S.A., Hnich, B., Rossi, R., Prestwich, S., 2009. Cost-based filtering techniques for stochastic inventory control under service level constraints. Constraints 14 (2), 137–176.

Tarim, S.A., Kingsman, B.G., 2004. The stochastic dynamic production/inventory lot-sizing problem with service-level constraints. International Journal of Production Economics 88 (1), 105–119.

Tarim, S.A., Kingsman, B.G., 2006. Modelling and computing (R^n,S^n) policies for inventory systems with non-stationary stochastic demand. European Journal of Operational Research 174 (1), 581–599.

Tarim, S.A., Manandhar, S., Walsh, T., 2006. Stochastic constraint programming: a scenario-based approach. Constraints 11 (1), 53–80.

Tarim, S.A., Smith, B., 2008. Constraint programming for computing non-stationary (R,S) inventory policies. European Journal of Operational Research 189 (3), 1004–1021.

Tempelmeier, H., 2007. On the stochastic uncapacitated dynamic single-item lotsizing problem with service level constraints. European Journal of Operational Research 181 (1), 184–194.

Walsh, T., 2002. Stochastic constraint programming. In: Proceedings of the 15th European Conference on Artificial Intelligence, ECAI'2002, Lyon, France, pp. 111–115.

Whybark, D.C., Williams, J.G., 1976. Material requirements planning under uncertainty. Decision Science 7 (4), 595–606.

Zipkin, P., 1986. Stochastic leadtimes in continuous-time inventory models. Naval Research Logistics Quarterly 33 (4), 763–774.

Please cite this article as: Rossi, R., et al., Computing the non-stationary replenishment cycle inventory policy under stochastic supplier lead-times. International Journal of Production Economics (2010), doi:10.1016/j.ijpe.2010.05.011

Neuroevolutionary Inventory Control in Multi-Echelon Systems*

S. D. Prestwich[1], S. A. Tarim[2], R. Rossi[3], and B. Hnich[4]

[1]Cork Constraint Computation Centre, Ireland
[2]Operations Management Division, Nottingham University Business School, Nottingham, UK
[3]Logistics, Decision and Information Sciences Group, Wageningen UR, the Netherlands
[4]Faculty of Computer Science, Izmir University of Economics, Turkey
s.prestwich@cs.ucc.ie, armtar@yahoo.com.tr,
roberto.rossi@wur.nl, brahim.hnich@ieu.edu.tr

Abstract. Stochastic inventory control in multi-echelon systems poses hard problems in optimisation under uncertainty. Stochastic programming can solve small instances optimally, and approximately solve large instances via scenario reduction techniques, but it cannot handle arbitrary nonlinear constraints or other non-standard features. Simulation optimisation is an alternative approach that has recently been applied to such problems, using policies that require only a few decision variables to be determined. However, to find optimal or near-optimal solutions we must consider exponentially large scenario trees with a corresponding number of decision variables. We propose a neuroevolutionary approach: using an artificial neural network to approximate the scenario tree, and training the network by a simulation-based evolutionary algorithm. We show experimentally that this method can quickly find good plans.

1 Introduction

In the area of optimisation under uncertainty, one of the most mature fields is inventory control. This field has achieved excellent theoretical and practical results using techniques such as dynamic programming, but some problems are too large or complex to be solved by classical methods. Particularly hard are those involving *multi-echelon systems*, in which multiple stocking points form a supply chain. In such cases we may resort to simulation-based methods. Simulation alone can only evaluate a plan, but when combined with an optimisation algorithm it can be used to find near-optimal solutions (or plans). This approach is called *simulation optimisation* (SO) and has a growing literature in many fields including production scheduling, network design, financial planning, hospital administration, manufacturing design, waste management and distribution. It is a practical approach to optimisation under uncertainty that can handle problems containing features that make them difficult to model and solve by other methods: for example non-linear constraints and objective function, and demands that are correlated or have unusual probability distributions.

* B. Hnich is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027.

SO approaches to inventory control are typically based on policies known to be optimal in certain situations, involving a small number of reorder points and reorder quantities. For example in (s, S) policies whenever a stock level falls below s it is replenished up to S, while in (R, S) policies the stock level is checked at times specified by R, and if it falls below S then it is replenished up to S. SO can apply standard optimisation techniques such as genetic algorithms to these policies by assigning genes to reorder points and replenishment levels. In more complex situations involving constraints, multiple stocking points, etc, these policies may be suboptimal in terms of expected cost, though they can have other desirable properties such as improved planning stability. But a cost-optimal plan for a multi-stage problem with recourse must specify an order quantity in every possible scenario, so the plan must be represented via a *scenario tree*. The number of scenarios might be very large, or infinite in the case of continuous probability distributions, making the use of SO problematic. Scenario reduction techniques may be applied to approximate the scenario tree, but it might not always be possible to find a small representative set of scenarios.

An alternative form of approximation is to use an artificial neural network (ANN) to represent the policy. For example, the inputs to the ANN could be the current stock levels and time, and the outputs could be the recommended actions (whether or not to replenish and by how much). We must then train the ANN so that its recommendations correspond to a good plan. No training data is available for such a problem so the usual ANN backpropagation training algorithm cannot be applied. Instead we may use an evolutionary algorithm to train the network to minimise costs. This *neuroevolutionary* approach has been applied to control problems [8, 9, 21] and to playing strategies for games such as Backgammon [16] and Go [14], but it has not been extensively applied to inventory control. In this paper we apply neuroevolution to stochastic inventory control in multi-echelon systems. Section 2 presents our method, Section 3 evaluates the method experimentally, Section 4 surveys related work, and Section 5 concludes the paper.

2 A neuroevolutionary approach

To approximate the scenario tree, we construct a function whose input is a vector containing the time period and current inventory levels, and whose output is a vector of order quantities (which might be zero). We design the function automatically by simulation optimisation.

2.1 Scenario tree compression by neural network

An obvious choice for this function is an artificial neural network (ANN), which can approximate any function with arbitrary accuracy given a sufficient number of units. ANNs also come with a ready-made algorithm for optimisation: the well-known backpropagation algorithm. However, there is a problem with this approach: we do not have training data available (this also precludes the use of Support Vector Machines). To obtain training data we would have to solve a set of instances, and there is no known method for solving the harder instances to optimality. Instead we must use an ANN to

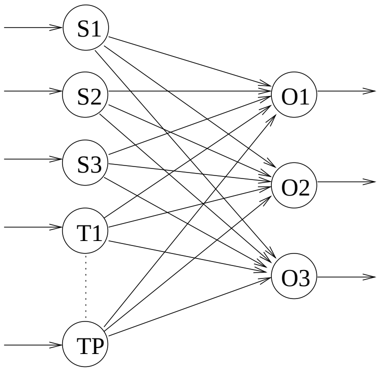


Fig. 1. The feedforward ANN used

solve a problem in *reinforcement learning*, in which we must choose its weights in order to maximise reward (in this case to minimise expected cost). Backpropagation cannot be used for this task, but we can instead use an evolutionary algorithm (EA) whose genes are the weights and whose fitness function is minus the cost. This neuroevolutionary approach has been applied to control problems and game learning.

In our experiments we began with a standard three-layer feedforward ANN, which is a universal function approximator: it can approximate any function to arbitrary accuracy given a sufficient number of hidden units. We tried different numbers of hidden units, including multiple hidden layers, with different transfer functions in all the units (including sigmoids, limiter functions and polynomial expressions), and with two alternative representations of time period t: as an integer $t = 1 \dots P$ and using the well-known *unary encoding* which is often used to represent symbolic ANN inputs and gave better results here. In the unary encoding we associate a binary variable with each period, and period t is represented by a vector $(0_1, \dots, 0_{t-1}, 1, 0_{t+1}, \dots, 0_P)$. Surprisingly, we obtained best results using an extremely simple network, with no hidden layer and the identity transfer function $f(x) = x$. No bias term is needed because the unary encoding already provides a time-dependent bias.

The ANN corresponding to three stocking points is shown in Figure 1, where Si denotes the i^{th} stock level, Oi the i^{th} order level, and Tj the j^{th} binary variable in the unary time encoding. All units use the identity transfer function. Each arrowed line connecting two units in the diagram has an associated weight, so the ANN has $K(P + K)$ weights, where K is the number of stocking points. This ANN represents a simple set of affine relationships

$$O_j = \sum_i S_i w_{ij} + w_{tj}$$

where w_{ij} is the ANN weight between stock level S_i and order level O_j, and w_{tj} is the ANN weight between time t and order level O_j. (An affine transformation is a linear transformation followed by a translation.) One would not expect this to yield an efficient or even a sensible policy, but our policy is not yet complete as we have not handled the problem constraints.

2.2 Constraint handling

The ANN forms only part of the policy. We also need a way of handling the constraints of the problem, which forbid (i) negative orders (corresponding to selling unused stock back to the supplier), and (ii) negative stock levels. We will train the ANN by an EA and there are several ways of handling constraints in EAs. We use a *decoder* which transforms the (possibly infeasible) ANN solution into one that violates no constraints. Decoders are a way of finding feasible solutions from chromosomes that represent infeasible states. They are problem-specific and ours works as follows. Suppose at period t we have stock levels S_i and the ANN suggests ordering quantities O_i. We modify each quantity O_i by

$$O_i \leftarrow \max(O_i, 0)$$

to avoid violating constraints of type (i). Then for any stocking point i that supplies a set of stocking points X_i we modify its order level O_i by

$$O_i \leftarrow \max\left(O_i, \left(\sum_{j \in X_i} O_j\right) - S_i\right)$$

This ensures that each supplier orders sufficient stock to fulfil its deliveries, and avoids violating constraints of type (ii). The policy is now the composition of the ANN and the decoder, which transforms the affine function of the ANN into a continuous piecewise affine function.

Note that we must modify the order levels of the stocking points earlier in the supply chain first. This is always possible if the supply chain is in the form of a directed acyclic graph. If lateral transshipments are allowed (orders between stocking points at the same level) or if there are constraints on order sizes or storage capacities then the decoder must be modified; we leave this issue for future work.

2.3 The evolutionary algorithm

To train the ANN we use an EA. There are many such algorithms in the literature, and we now describe our choice and the design decisions behind it. Firstly, we decided not to use genetic recombination. When training an ANN by EA one can encounter the well-known *competing conventions* problem (see [20] for example). This is caused by two forms of symmetry: an ANN's hidden units can be permuted without changing its output, and a hidden unit's weights can all be multiplied by -1 without changing its output. Thus if there are h hidden units then there are $2^h h!$ equivalent ANNs. Crossover is unlikely to give good results unless the parent chromosomes are aiming for symmetrically similar representations, though it is possible to design crossover operators that handle the symmetries [23]. This problem does not apply to our simple ANN because it has no hidden units, but in experiments crossover did not improve results so we do not use it.

We decided to use a $(\mu + 1)$-Evolution Strategy (ES) because it is almost exactly a steady-state genetic algorithm without crossover, and an efficient method for handling noise in the fitness function is known for a steady-state genetic algorithm (see below). However, we adapted it to a *cellular* ES, in which each chromosome is notionally placed in an artificial space and nearby chromosomes form its neighbourhood. Cellular algorithms can reduce premature convergence, which we found to be a problem with our initial standard ES. In our ES the population size is μ, at each iteration a new chromosome c' is created by mutating a randomly selected chromosome c, and if c' is fitter than the least-fit chromosome c^* in the neighbourhood of c then it replaces c^*, otherwise c' is discarded. We used a ring topology and define the neighbourhood of a chromosome to be its two adjacent chromosomes.

A common form of mutation adds normally distributed noise to each gene, but we use a method that gave better results in experiments. For each chromosome we generate two uniformly distributed random numbers, p in the range $(0, 1)$ and q in the range $(0, 0.5)$. Then for each allele in the chromosome, with probability p we change it, otherwise with probability $1 - p$ we leave it unchanged. If we do change it then with probability q we set it to 0, otherwise with probability $1 - q$ we add a random number with Cauchy distribution to it. *Cauchy mutation* has been shown to speed up EAs [24]. It can be computed as $s \tan(u)$ where u is a uniformly distributed random variable in the range $(-\pi, \pi)$ and s is a scale factor. For each chromosome we compute a random scale factor, itself with Cauchy distribution and fixed scale factor 100. Finally, if no allele was modified (which is possible for small p) then we modify one randomly selected allele as described. This rather complex mutation operator is designed to generate a variety of random moves, with different numbers of modified alleles and different scale factors. All chromosomes initially contain alleles generated randomly using the same Cauchy distribution. We do not use the well-known technique of self-adapting step sizes, because in a $(\mu + 1)$-ES offspring with reduced mutation variances are always preferred [2].

2.4 Handling uncertainty

When demand is probabilistic the fitness function of the EA is noisy. In such cases we must average costs over a number of simulations. In some previous SO approaches to inventory control, this problem was tackled by averaging costs over a small number of simulations because the simulations were computationally expensive: for example [13] use 3 samples. The standard deviation of the sample mean of a random variable with standard deviation σ is $\sigma/\sqrt{n}$ where n is the number of samples, so a large number of samples may be needed for very noisy fitness functions. Here we use smaller problems than those in [13] so we can afford to use a much larger number of simulations and obtain reliable cost estimates. To do this for every chromosome would be expensive but there are more efficient methods, and we use the *greedy averaged sampling* resampling scheme of [17]. This requires two parameters to be tuned by the user: U and S. On generating a new chromosome c it takes S samples to estimate its fitness before placing it into the population. It then selects another chromosome c' (which may be c) for *resampling*: another S samples are taken for c' and used to refine its fitness estimate. c' is the chromosome with highest fitness among those with fewer than U samples, so the function of U is to prevent any chromosome from being sampled more times than necessary. If all chromosomes in the population have been sampled U times then no resampling is performed. The algorithm is summarised in Figure 2.

```
train($\mu, S, U$)
  create ANN population of size $\mu$
  evaluate population using $S$ samples
  while not(termination condition)
    select a parent
    breed an offspring O by mutation
    evaluate O using $S$ samples
    if O fitter than locally least-fit chromosome L
      replace L by O
    select globally fittest chromosome F with #samples$<U$
    if F exists
      re-evaluate F using $S$ more samples
  return best chromosome found with #samples$\geq U$
```

Fig. 2. Cellular evolution strategy with resampling

The aim of this resampling method is to obtain chromosomes with good fitness averaged over many samples, while expending a smaller number of samples on less-promising chromosomes. In our experiments we set $U = 10000$ so that cost estimates are obtained over 10000 samples, but by setting $S = 1$ we only expend approximately 200 samples per chromosome on average (this number was found by experiment). As the population size is 50, and $50 \times 200 = 10000$, this implies that a typical chromosome uses little more than one sample before being rejected as unfit. Using small S also has an effect beyond reducing the average number of samples per chromosome: it encourages exploration by preserving less-fit chromosomes for longer. We found this to be a very beneficial effect.

Some points are glossed over in Figure 2 for the sake of readability. Firstly, if S is not a divisor of U then fewer than S samples are needed in the final resampling of any chromosome to bring its total to U. Secondly, the termination condition is unspecified, and we simply use a timeout. Thirdly, if no chromosome has U samples on termination then we must choose another chromosome to return. To avoid this, S should be assigned a sufficiently large value so that in experiments there is always a chromosome with U samples on termination. This value must be chosen by experimentation.

2.5 Discussion of the method

We refer to our method as NEMUE[1] (Neuro-Evolution for MUlti-Echelon systems). NEMUE is the result of many experiments with alternative versions. We experimented with an array of ANNs, one for each time period. This model has $12P$ weights and

[1] The “lady of the lake” in Arthurian legend.

clearly subsumes the model above: any plan that can be represented by that model can also be represented by this one. The results should therefore be at least as good, but in experiments they were significantly worse. We believe that the ANN array is simply harder to train than a single ANN.

We also tried a non-unary encoding of time, in which order levels are linear functions of stock levels and polynomial functions of time. Fixing the polynomial degree makes the size of the ANN independent of the number of time periods. Using a cubic function of time gave reasonable results but was inferior to the unary encoding.

We used a decoder to handle the problem constraints, but there are other ways of handling constraints in EAs. The simplest is to use a *penalty function* which adds a large artificial cost for each violated constraint. In our problem this forces the ANN to learn to order sufficient stock in order to avoid stockout. We tried a penalty function but it gave inferior results to the decoder.

3 Experiments

Ultimately we are interested in solving large, realistic inventory problems with multiple stocking points, stochastic lead times, correlated demands and other features that make classical approaches impractical. Unfortunately there are no known methods for solving such problems to optimality, so there is no way of evaluating our method. Instead we consider more modest problems to test the ability of NEMUE to find good plans.

Our benchmark problems have two multi-echelon topologies: *arborescent* and *serial*. In the arborescent case we have three stocking points A, B and C, with C supplying A and B, while in the serial case C supplies B which supplies A. In both cases we have linear holding costs, linear penalty costs, fixed ordering costs, and stationary probabilistic demands. The closing inventory levels for period t are $I_t^A = I_{t-1}^A + Q_t^A - d_t^A$, $I_t^B = I_{t-1}^B + Q_t^B - d_t^B$ and $I_t^C = I_{t-1}^C + Q_t^C - Q_t^A - Q_t^B$ where Q_t is the order placed in period t and d_t is the demand in period t. If $I_t < 0$ then the incurred cost is $-I_t.\pi$, otherwise it is $I_t.h$, where π is the penalty cost and h the holding cost. Suppliers are not allowed to run out of stock. We prepared 28 instances of both the arborescent and serial types, with various costs and 2–9 time periods, giving a total of 56 instances with a range of characteristics as follows. The holding costs for A, B and C are 4, 5 and 1 respectively for arborescent instances 1–14; 3, 2 and 1 for arborescent instances 15–28; and 3, 2 and 1 for all serial instances. For the arborescent instances the penalty costs for A and B are 12 and 25 respectively for instances 1–14; and 3 and 6 for instances 15–28. For all the serial instances the penalty cost for instance A is 12. The ordering costs for A, B and C are 150, 130 and 170 respectively for arborescent instances 1–14; 80, 75 and 100 for arborescent instances 15–28; and 75, 80 and 100 for all serial instances. For space reasons we do not specify the demands in detail, but we used 10 patterns for arborescent instances and 4 patterns for serial instances. In each period we specify a deterministic demand which is then multiplied by either $\frac{2}{3}$ with probability 0.25, 1 with probability 0.5, or $\frac{4}{3}$ with probability 0.25. Thus the number of possible scenarios is 3^P, giving 59,049 scenarios for the largest problems ($P = 10$).

We solved these problems in two ways: using Stochastic Programming (SP) [3] and NEMUE. SP is a field of Operations Research designed to solve optimisation problems under uncertainty via scenario reduction techniques: a representative subset of all possible scenarios is selected and used to generate a deterministic equivalent optimisation problem, which is then typically solved using integer linear programming. We use the SP results to evaluate the quality of plans found by NEMUE. The optimal replenishment plans are obtained using the following Stochastic Integer Programming model:

$$
\begin{array}{l}
\min \mathsf{E}[C] = \sum_{t=1}^{N} \sum_{p \in P} \left(a_p \delta_{pt} + h_p I_{pt}^{+} + \pi_p I_{pt}^{-} \right) \\
\text{s.t. } t = 1, \ldots, N \text{ and } p \in P \\
I_{pt} = I_{p,t-1} + Q_{pt} - Q_{P_p,t} - d_{pt} \\
I_{pt} = I_{p,t}^{+} - I_{p,t}^{-} \\
Q_{pt} \leq M \delta_{pt} \\
\delta_{pt} \in \{0,1\} \quad Q_{pt} \geq 0
\end{array}
$$

where

C : total holding and ordering/set-up cost of the system over N periods;
a : fixed ordering/set-up cost;
h : proportional inventory holding cost per period;
P : the set of all stocking points;
P_p : the set of stocking points supplied directly by the stocking point p;
d_{pt} : random demand at stocking point p, in period t;
δ_{pt} : a binary variable that takes the value of 1 if a replenishment occurs
: at stocking point p in period t and 0 otherwise;
I_{pt} : the inventory level at the end of period t at stocking point p;
Q_{pt} : the order quantity at the beginning of period t at stocking point p;

and I^{+} and I^{-} denote positive and negative closing inventory levels. Except for the lowest echelon stocking points, I^{-} is zero. M is some large positive number. In this stochastic model a *here-and-now* policy is adapted: all decision variables are set before observing the realisation of the random variables. The certainty equivalent model is obtained using the compiler described in [22] and solved with CPLEX 11.2.

Results comparing SP and NEMUE are shown in Table 1. All SP runs were terminated after one hour and all NEMUE results after 30 minutes on a 2.8 GHz Pentium (R) 4 with 512 RAM, each figure being the best of six five-minute runs. The NEMUE parameters used were $S = 1, U = 10000$ and $\mu = 50$. SP runs that were aborted because of memory problems are denoted by "—". (In the few cases that SP found and proved optimality, this sometimes took much less than one hour.) The columns marked "%opt" denote the optimality gap: a reported cost c and gap g means that SP proved that the optimal solution cannot have cost lower than $c' = c(100 - g)/100$ (this does not imply the existence of a solution with cost c'). In several cases NEMUE finds superior plans to those found by SP, showing that on larger instances SP fails to find optimal plans. In a few cases NEMUE appears to find plans that are slightly better than optimal: this is of course impossible, and is a consequence of the empirical nature of the data. In such cases we assume that NEMUE found an optimal plan.

SP was unable to find provably optimal plans for all but the smallest instances. We believe that for the medium-sized instances SP finds optimal plans but does not prove optimality before timeout. For the largest instances SP ran out of memory, though we

arborescent						serial					
		SP		NEMUE				SP		NEMUE	
#	periods	cost	%opt	cost	%opt	#	periods	cost	%opt	cost	%opt
1	4	2507	0	2573	2.6	1	4	995	0	993	0
2	5	3124	1.4	3180	3.1	2	5	1269	0.7	1298	2.9
3	6	3657	2.7	3775	5.7	3	6	1493	1.8	1491	1.7
4	7	4214	5.6	4250	6.4	4	7	1794	7.4	1797	7.6
5	8	4654	8.2	4722	9.5	5	8	2087	12.0	1987	7.6
6	9	5472	16.9	5164	11.9	6	9	2741	25.7	2295	11.3
7	10	—	?	5590	?	7	10	—	?	2603	?
8	4	2100	0	2169	3.2	8	4	1311	0.2	1306	0.0
9	5	2626	0.6	2722	4.1	9	5	1598	2.2	1594	2.0
10	6	3311	1.8	3409	4.6	10	6	1833	4.3	1832	4.2
11	7	4065	2.5	4153	4.6	11	7	2024	6.7	2024	6.7
12	8	4454	3.4	4542	5.3	12	8	2160	9.3	2142	8.5
13	9	5158	10.3	5115	9.5	13	9	2678	25.1	2264	11.4
14	10	—	?	5432	?	14	10	—	?	2407	?
15	4	1342	0.2	1340	0.1	15	4	1104	0	1104	0
16	5	1657	1.8	1671	2.6	16	5	1417	2.1	1423	2.5
17	6	1930	2.2	1938	2.6	17	6	1759	4.1	1763	4.3
18	7	2180	4.5	2192	5.0	18	7	2057	5.4	2055	5.3
19	8	2428	6.1	2393	4.7	19	8	2266	6.6	2258	6.3
20	9	2853	13.9	2617	6.1	20	9	2706	17.7	2479	10.2
21	10	—	?	2864	?	21	10	—	?	2627	?
22	4	1086	0	1096	0.9	22	4	828	0	828	0
23	5	1334	0.2	1330	0.0	23	5	931	0	934	0.3
24	6	1680	0.6	1677	0.4	24	6	1259	1.3	1265	1.8
25	7	2055	0.7	2051	0.5	25	7	1633	2.4	1639	2.8
26	8	2219	1.1	2220	1.1	26	8	1757	2.7	1766	3.2
27	9	2479	2.0	2531	4.0	27	9	1983	3.9	2000	4.7
28	10	—	?	2665	?	28	10	—	?	2150	?

Table 1. Experimental results

use the state-of-the-art CPLEX solver and a powerful machine (an Intel Core 2 Duo CPU E7200 with 2.53 GHz and 3GB of RAM). On the largest instances for which SP did not run out of memory, it was unable to prove optimality even within several days. Thus our benchmark problems straddle the borderline of solvability by classical methods.

Despite the simplicity of the policy and the large number of scenarios (at least on the larger instances) the NEMUE results are remarkably good. On 13 of the 28 arborescent instances and 19 of the 28 serial instances, NEMUE found plans that were at least as good as those found by SP. On the three serial instances for which SP found provably optimal plans, NEMUE found equally good plans. On most of the larger instances NEMUE found better plans than SP. These results show that: (i) a relatively simple, continuous, piecewise affine function can closely approximate a large policy tree for multi-echelon systems; (ii) such a function can be effectively represented by an affine function followed by a decoder function; (iii) the affine function can be learned in a reasonable time by evolutionary search; (iv) that our approach is more scalable than SP.

It is tempting to speculate that with improved heuristics and longer runtimes we might find optimal strategies for *all* instances. But there is no guarantee that all scenario trees can be well-approximated in this way, and in more extensive experiments on arborescent instance 1 (for example) we have been unable to find an optimal plan. Nevertheless, the results are very promising.

4 Related work

Though simulation was originally used only to evaluate solutions found by other means, the field of SO has recently become more popular — see the survey of [5]. SO may be *recursive* or *non-recursive*. In the non-recursive approach an approximate cost function is learned during a simulation phase, then this function is minimised using an optimisation algorithm during a second phase. NEMUE is an example of recursive SO in which simulation and optimisation alternate and inform each other.

A tutorial and survey of the application of SO to inventory control is given in a recent paper [12]. Relatively little work has been done on applying SO to multi-echelon systems, and we have been unable to find any other work on inventory control via neuroevolution, though several papers use EAs to evolve plans (for example [1, 13, 15, 18]). Another difference of NEMUE is that it aims to approximate optimal policy trees, whereas most SO methods aim to find parameters for special policies such as (s, S). A different way of using ANNs for inventory control is to solve a set of training instances by some other method, then train an ANN to learn how to find good solutions from new instances (for example [6]). But we then need another algorithm to solve the problems, which is the aim of NEMUE. A related approach to neuroevolution is *genetic programming*, in which an EA is used to evolve an algorithm for solving the problem, instead of an ANN. This approach has also been applied to inventory control [11].

Another interesting approach to sequential decision problems such as those in inventory control is the field variously referred to as *neuro-dynamic programming*, *temporal difference learning* and *approximate dynamic programming*. This blend of dynamic programming and simulation has been applied to many problems including inventory control: see for example [4, 7, 10, 19]. A drawback is that special techniques are needed to cope with the well-known "curse of dimensionality": the vast number of states that result from a simple discretisation of the continuum of states in these problems. In contrast, neuroevolution can directly handle a continuum of states.

5 Conclusion

We have proposed what seems to be the first neuroevolutionary method for approximating optimal plans in multi-echelon stochastic inventory control problems. Large or infinite scenario trees are approximated by a neural network, which is trained by an evolutionary algorithm with resampling, while problem constraints are handled by a decoder. Because the method is simulation-based and uses general-purpose techniques such as evolutionary algorithms and neural networks, it does not rely on special properties of the problem and can be applied to inventory problems with non-standard features. We showed experimentally that the method can find near-optimal solutions. In future work we will extend the method to handle problem features such as capacity constraints.

References

1. J. Arnold, P. Köchel. Evolutionary Optimisation of a Multi-Location Inventory Model With Lateral Transsshipments. *9th International Working Seminar on Production Economics*, Igls 1996, Preprints 2, pp. 401–412.
2. T. Bäck, F. Hoffmeister, H.-P. Schwefel. A Survey of Evolution Strategies. *4th International Conference on Genetic Algorithms*, 1991.
3. J. R. Birge, F. Louveaux. Introduction to Stochastic Programming. Springer, New York, 1997.
4. S. K. Chaharsooghi, J. Heydari, S. H. Zegordi. A Reinforcement Learning Model for Supply Chain Ordering Management: an Application to the Beer Game. *Decision Support Systems* 45(4):949–959, 2008.
5. M. C. Fu. Optimization for Simulation: Theory vs Practice. *INFORMS Journal of Computing* 14:192–215, 2002.
6. L. K. Gaafar, M. H. Choueiki. A Neural Network Model for Solving the Lot-Sizing Problem. *Omega* 28(2):175–184, 2000.
7. I. Giannoccaro, P. Pontrandolfo. Inventory Management in Supply Chains: a Reinforcement Learning Approach. *International Journal of Production Economics* 78(2):153–161, 2002.
8. F. Gomez, J. Schmidhuber, R. Miikkulainen. Efficient Non-Linear Control Through Neuroevolution. *Journal of Machine Learning Research* 9:937–965, 2008.
9. N. M. Hewahi. Engineering Industry Controllers Using Neuroevolution. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 19(1):49–57, 2005.
10. C. Jiang, Z. Shenga. Case-Based Reinforcement Learning for Dynamic Inventory Control in a Multi-Agent Supply-Chain System. *Expert Systems with Applications* 36(3 part 2):6520–6526, 2009.
11. P. Kleinau, U. W. Thonemann. Deriving Inventory-Control Policies With Genetic Programming. *OR Spectrum* 26(4):521–546, 2004.
12. P. Köchel. Simulation (Optimisation) in Inventory Theory. Tutorial, *8th ISIR Summer School on New and Classical Streams in Inventory Management: Advances in Research and Opening Frontiers*, 2007.

13. P. Köchel, U. Nieländer. Simulation-Based Optimisation of Multi-Echelon Inventory Systems. *International Journal of Production Economics* 1:503–513, 2005.
14. A. Lubberts, R. Miikkulainen. Co-Evolving a Go-Playing Neural Network. *Genetic and Evolutionary Computation Conference,* Kaufmann, 2001, pp. 14–19.
15. A. L. Olsen. An Evolutionary Algorithm for the Joint Replenishment of Inventory with Interdependent Ordering Costs. *Genetic and Evolutionary Computation Conference, Lecture Notes in Computer Science* vol. 2724, Springer, 2003, pp. 2416–2417.
16. J. B. Pollack, A. D. Blair. Co-Evolution in the Successful Learning of Backgammon Strategy. *Machine Learning* 32(3):225–240, 1998.
17. S. D. Prestwich, S. A. Tarim, R. Rossi, B. Hnich. A Steady-State Genetic Algorithm With Resampling for Noisy Inventory Control. *10th International Conference on Parallel Problem Solving From Nature, Lecture Notes in Computer Science* vol. 5199, Springer, 2008, pp. 559–568.
18. S. D. Prestwich, S. A. Tarim, R. Rossi, B. Hnich. A Cultural Algorithm for POMDPs from Stochastic Inventory Control. *5th International Workshop on Hybrid Metaheuristics, Lecture Notes in Computer Science* vol. 5296, Springer, 2008, pp. 16–28.
19. B. Van Roy, D. P. Bertsekas, Y. Lee, J. N. Tsitsiklis. A Neuro-Dynamic Programming Approach to Retailer Inventory Management. *Proceedings of the IEEE Conference on Decision and Control*, 1997.
20. J. Schaffer, D. Whitley, L. Eshelman. Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art. *International Workshop on Combinations of Genetic Algorithms and Neural Networks*, 1992, pp. 1–37.
21. K. O. Stanley, R. Miikkulainen. Evolving Neural Networks Through Augmenting Topologies. *Evolutionary Computation* 10(2):99–127, 2002.
22. S. A. Tarim, S. Manandhar, T. Walsh. Stochastic Constraint Programming: A Scenario-Based Approach. *Constraints* 11:53–80, 2006.
23. D. Thierens. Non-Redundant Genetic Coding of Neural Networks. *International Conference on Evolutionary Computation*, Nagoya, Japan, 1996, pp. 571–575.
24. X. Yao, Y. Liu, G. Lin. Evolutionary Programming Made Faster. *IEEE Transactions on Evolutionary Computation* 3(2):82–102, 1999.

TÜBİTAK

PROJE ÖZET BİLGİ FORMU

Proje No: 108K027

Proje Başlığı: Türkiye'de Perakende Tedarik Zinciri Envanter Sistemlerinin Belirsizlik Altında Yönetimi için Planlama Modelleri ve Çözüm Yöntemleri

Proje Yürütücüsü ve Araştırmacılar:

Prof. Dr. Ş. Armağan TARIM (Yürütücü)
Prof. Dr. Brahim HNICH (Araştırmacı)
Yrd. Doç. Dr. Ayşegül TAŞ (Araştırmacı)

Projenin Yürütüldüğü Kuruluş ve Adresi:

Hacettepe Üniversitesi, İşletme Bölümü, Beytepe Kampüsü, Ankara

Destekleyen Kuruluş(ların) Adı ve Adresi:

Yok

Projenin Başlangıç ve Bitiş Tarihleri:

01 Temmuz 2008 – 30 Haziran 2010

Öz

Tedarik zinciri yönetimi alanındaki araştırma faaliyetlerinin uygulamada sonuç alabilmesi için belirsizlik gibi iş dünyasının çevresini tanımlayan temel unsurları dikkate alması gereklidir. Bu proje perakende tedarik zincirleri için belirsizlik altında envanter planlaması sorununu ele almıştır. Talep belirsizliği altında beklenen maliyetleri azaltmaya dönük tedarik zinciri envanter politikaları geliştirilmiş, politika parametrelerinin hesaplanmasında kullanılacak yeni tedarik zinciri envanter modelleri kurulmuş ve kurulan modelleri etkin şekilde çözmek için algoritmalar geliştirilmiştir.

Anahtar Kelimeler:

Perakendecilik; Tedarik Zinciri Yönetimi; Envanter Yönetimi; Stokastik Modelleme

Fikri Ürün Bildirim Formu Sunuldu mu? □ □ **Gerekli Değil** ☒

Projeden Yapılan Yayınlar:

Uluslararası Dergiler/Kitap Bölümü:

"Finding Reliable Solutions: Event-Driven Probabilistic Constraint Programming," S. A. Tarim, B. Hnich, S. Prestwich and R. Rossi, Annals of Operations Research, 171, pp.77-99, 2009.

"A Note on Liu-Iwamura's Dependent-Chance Programming," R. Rossi, S. A. Tarim, B. Hnich, S. Prestwich, C. Guran, European Journal of Operational Research, 198, pp.983–986, 2009.

"A Multi-Objective Stochastic Programming Approach for Supply Chain Design Considering Risk," A. Azaron, K. N. Brown, S. A. Tarim, M. Modarres, International Journal of Production Economics, 116, pp.129-138, 2008.

"Scheduling Internal Audit Activities: A Stochastic Combinatorial Optimization Problem," R. Rossi, S. A. Tarim, B. Hnich, S. Prestwich, S. Karacaer, Journal of Combinatorial Optimization, 19, pp.325-349, 2010.

"A State Space Augmentation Algorithm for the Replenishment Cycle Inventory Policy," R. Rossi, S. A. Tarim, B. Hnich, S. Prestwich, International Journal of Production Economics, to appear, 2010.

“An Investigation of Setup Stability in Non-Stationary Stochastic Inventory Systems," O. A. Kilic, S. A. Tarim, International Journal of Production Economics, to appear, 2010.

"Computing the Non-Stationary Replenishment Cycle Inventory Policy under Stochastic Supplier Lead-Times," R. Rossi, S. A. Tarim, B. Hnich, S. Prestwich, International Journal of Production Economics, to appear, 2010.

Invited Paper: "A Survey on CP-AI-OR Hybrids for Decision Making under Uncertainty," B. Hnich, R. Rossi, S. A. Tarim, S. Prestwich, to appear, 2010.

"Cost-based Domain Filtering for Stochastic Constraint Programming," R. Rossi, S. A. Tarim, B. Hnich, S. Prestwich, Lecture Notes in Computer Science, Springer-Verlag, LNCS 5202, pp. 235-250, 2008.

"A Steady-State Genetic Algorithm With Resampling for Noisy Inventory Control," S. Prestwich, S. A. Tarim, R. Rossi and B. Hnich, Lecture Notes in Computer Science, Springer-Verlag, LNCS 5199, pp. 559-568, 2008.

"A Cultural Algorithm for POMDPs from Stochastic Inventory Control," S. D. Prestwich, S. A. Tarim, R. Rossi, B. Hnich, Lecture Notes in Computer Science, Springer-Verlag, LNCS 5296, pp. 16-28, 2008.

“The Cost Of Using Stationary Inventory Policies When Demand Is Non-Stationary,” H.

Tunç, O. A. Kılıç, S. A. Tarim, B. Ekşioğlu, Omega'da hakem sürecinde (1. raund).

"Constraint-based Local Search for Inventory Control under Stochastic Demand and Lead time," R. Rossi, S. A. Tarim, R. Bollapragada, INFORMS Journal on Computing'te hakem sürecinde (2. raund).

"Neuroevolutionary Inventory Control in Multi-Echelon Systems," S. D. Prestwich, S. A. Tarim, R. Rossi, B. Hnich, Lecture Notes in Artificial Intelligence, Springer-Verlag, LNAI 5783, pp.402-413, 2009.

Uluslararası Bildiriler:

"Neuroevolutionary Inventory Control in Multi-Echelon Systems", S. Prestwich, S.A. Tarim, R. Rossi, B. Hnich, 1st International Conference on Algorithmic Decision Theory (ADT-09) October 21-23, 2009 Venice, Italy.

"Synthesizing Filtering Algorithms for Global Chance-Constraints", B. Hnich, R. Rossi, S.A. Tarim, S. Prestwich, 5th International Conference on Principles and Practice of Constraint Programming (CP-09) September 21-24, 2009, Lisbon, Portugal.

"Evolving Parameterised Policies for Stochastic Constraint Programming", S. Prestwich, S.A. Tarim, R. Rossi, B. Hnich, 15th International Conference on Principles and practice of Constraint Programming (CP-09) September 21-24, 2009, Lisbon, Portugal.

"A Relaxation Approach for Solving the Stochastic Lot Sizing Problem with Service Level Constraints" S.A. Tarim, U. Ozen, M.K. Dogru, R. Rossi, INFORMS Annual Meeting, October 11-14, 2009, San Diego, CA.

"Boosting Partial Symmetry Breaking by Local Search", S. Prestwich, B. Hnich, H. Simonis, R. Rossi and S.A. Tarim, 9th International Workshop on Symmetry and Constraint Satisfaction Problems (SYMCON 2009), September 21-24, 2009, Lisbon, Portugal, in conjunction with CP 2009.

"Synthesizing Filtering Algorithms in Stochastic Constraint Programming", B. Hnich, R. Rossi, S.A. Tarim, S. Prestwich, 40th Annual Conference of the Italian Operational Research Society (AIRO 2009) September 8-11, 2009, Siena, Italy.

"Stochastic Constraint Programming by Neurevolution with Filtering," S. Prestwich, S. A. Tarim, R. Rossi, B. Hnich, 7th International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) techniques in Constraint Programming, Bologna, Italy, June 14-18, 2010.