Esnek İmalat Sistemlerinde Atama ve Çizelgeleme Problemleri

Proje No: 110M492

Yard. Doç.Dr. Selin ÖZPEYNİRCİ

HAZİRAN 2013
İZMİR

ÖNSÖZ

Esnek imalat sistemleri, bilgisayar nümerik kontrollü (CNC) makineler ile özdevimsel malzeme taşımanın bağlandığı sistemlerdir. Bu sistemler, transfer hatlarının verimliliği ile atölye imalatının esnekliğini birleştirip orta-hacim ve orta-çeşitlilikteki ürünlerin toplu üretimine en iyi şekilde ulaşmaktadır. Esnek imalat sistemleri, endüstride son yıllarda üretimin çok daha verimli bir şekilde gerçekleştirilmesinde etkili olmuştur. CNC makinelerinde uygun makine uçlarının yerleştirilmesiyle çok çeşitli işlemler yapılabilmektedir. Bu özellikleri ve yüksek yatırım gereksinimleri sebebiyle çok dikkatli planlama gerektirmekte ve literatürde ve endüstride büyük ilgi görmektedir. Esnek imalat sistemlerinde, parça seçimi, operasyon ataması, makine yüklenmesi ve makine ucu dağılımı gibi çeşitli problemler vardır.

Proje, Türkiye Bilimsel ve Teknolojik Araştırma Kurumu tarafından desteklenmiştir.

İÇİNDEKİLER

TABLO LİSTESİ

ŞEKİL LİSTESİ

ÖZET

Bu projede, esnek imalat sistemlerinde karşılaşılan birbirleriyle ilişkili önemli problemler üzerinde çalışılmıştır. Birincisi, işlerin ve işlerin gerçekleştirilmesi için gereken makine uçlarının makinelere atanması ve çizelgelenmesi problemidir. Amaç en son işin bitirilme zamanının en aza indirilmesidir. Bu problemde makine ucu değişimi göz ardı edilmiştir. İkinci problem olan işlerin gruplanması probleminde ise işler makinelerin makine ucu haznesi kapasiteleri göz önünde bulundurularak gruplara ayrılmıştır. Bu problemde gruplar arasında makine ucu değişimi yapıldığı varsayılmaktadır. Makine ucu değişimi uzun bir süre gerektirdiği ve işlerin gerçekleştirilmesinde önemli bir yer tuttuğu için, grup sayısı ve dolayısıyla makine ucu değişimi sayısı en aza indirilmeye çalışılmıştır. Son olarak, işlerin ve makine uçlarının makinelere atanması, işlerin çizelgelenmesi ve makine uçlarının değiştirilmesi problemleri birlikte çalışılmıştır. Bu problemde de amaç, son işin tamamlanma zamanının en aza indirilmesidir.

Yukarıda tanımlanan problemlerin her biri polinom zamanlı olmayan-zor problemlerdir. Birinci problem az sayıda araştırmacı tarafından problemi kolaylaştıran varsayımlarla çalışılmıştır. İkinci problem pek çok araştırmacı tarafından çalşılmış bir problemdir. Bizim bu projede bu problemi ele alma nedenimiz, birinci problemle bir arada düşünüldüğünde üçüncü probleme ulaşılmasıdır. Son problem ise literatürde daha önce çalışılmamıştır. Bu proje ile literatürdeki bu boşluğu doldurmak amaçlanmaktadır.

Her problem için matematiksel model yazılmış, kesin veya sezgisel çözüm yöntemleri geliştirilmiş ve bu yöntemler örnek problemler üzerinde denenmiştir. Deney sonuçları, geliştirilen yöntemlerin kısa sürelerde oldukça iyi sonuçlar verdiğini göstermektedir.

Anahtar Kelimeler: Esnek İmalat Sistemleri, Çizelgeleme, Makine Ucu Değişimi

ABSTRACT

In this project, three related and important problems that arise in flexible manufacturing systems are studied. First one is the problem of assigning jobs and their required tools to the machines and scheduling. Objective is to minimize makespan. In this problem, tool switching is ignored. In the second problem, which is job grouping problem, the jobs are assigned to groups so that the tools required in each group does not exceed the tool magazine capacity. In this problem, it is assumed that tool switching is performed between the groups. Since tool switching requires a significant amount of time and is important to process the jobs, minimizing the number of groups, hence the number of tool switches is aimed. Lastly, assignment of jobs and tools to machines, scheduling and tool switching problems are considered together. The objective is again minimizing the makespan.

Every problem defined above is an NP-hard problem. The first problem is studied by a few number of researchers with simplifying assumptions. The second problem is widely studied. The reason we study this problem is that combined with the first problem, it gives the third problem. The last problem is not studied in the literature previously. With this project, we aim to fill this gap in the scheduling literature.

For each problem, the mathematical model is written, exact or heuristic algorithms are developed and applied on example problem instances. Experimental results show that the proposed methods give good results in short time.

Keywords: Flexible Manufacturing Systems, Scheduling, Tool Switching

1. GİRİŞ

Esnek imalat sistemleri, bilgisayar nümerik kontrollü (CNC) makineler ile özdevimsel malzeme taşımanın bağlandığı sistemlerdir. Bu sistemler, transfer hatlarının verimliliği ile atölye imalatının esnekliğini birleştirip orta-hacim ve orta-çeşitlilikteki ürünlerin toplu üretimine en iyi şekilde ulaşmaktadır. Esnek imalat sistemleri, endüstride son yıllarda üretimin çok daha verimli bir şekilde gerçekleştirilmesinde etkili olmuştur. CNC makinelerinde uygun makine uçlarının yerleştirilmesiyle çok çeşitli işlemler yapılabilmektedir. Bu özellikleri ve yüksek yatırım gereksinimleri sebebiyle çok dikkatli planlama gerektirmekte ve literatürde ve endüstride büyük ilgi görmektedir. (STECKE, 1983) esnek imalat sistemlerinde karşılaşılabilecek ve çözüm bekleyen beş önemli problem tanımlamıştır: 1) aynı anda değişik makinelerde üretilecek parçaların seçimi, 2) aynı operasyonları işleyebilecek makinelerin gruplanması, 3) parçaların üretim oranlarının belirlenmesi, 4) bağlama düzeni ve palet ihtiyaçlarının belirlenmesi, ve 5) makine uçlarının ve operasyonların makinelere atanması.

Esnek imalat sistemlerinin asıl amacı operasyon atama ve makine ucu değişimleri yoluyla esneklik ve makine kapasitelerinin verimli kullanımını sağlamaktır. Operasyon atama problemi, operasyonların CNC makinelerine teknolojik kısıtlar gözönünde bulundurularak belli bir amaca ulaşmak için veya bir performans kriterini eniyilemek amacıyla atanmasıdır.

Esnek imalat sistemlerinde makine ucu ataması bir başka önemli problem olarak karşımıza çıkar. Tüm operasyonların yapılması için çok sayıda makine ucuna ihtiyaç vardır fakat makine haznelerinin kapasiteleri sınırlıdır ve ekonomik kısıtlardan dolayı sistemde belirli sayıda makine ucu bulundurulmaktadır.

Bu projede, öncelikle yukarıda belirtilen operasyon ve makine ucu atama problemleriyle birlikte operasyonların çizelgelenmesi problemi üzerinde çalışılmıştır. Problemin çözümü için üç farklı matematiksel model yazılmıştır. NP-zor bir problem olduğu için büyük boyutlu örneklerde en iyi çözüme ulaşılamamıştır. Problemin iki parçaya ayrılması ile bir ayrıştırma sezgiseli geliştirilmiştir. Ayrıca çizelgeleme problemlerinde iyi performans gösterdiği bilinen tabu arama algoritması ile eniyi çözüme oldukça yakın sonuçlar elde edilmiştir. Son olarak kısıt programlama modelleri yazılmış ve küçük ve orta boyuttaki örneklerde kısa sürede en iyi çözüme ulaşılmıştır.

Projenin ikinci aşamasında işlerin gruplanması problemi üzerinde çalışılmıştır. Bu problemde işler gerektirdikleri makine ucu sayısı makine ucu haznesini aşmayacak şekilde gruplanmakta ve gruplar arasında makine ucu değişimi yapıldığı varsayılmaktadır. Amacımız grup sayısını, dolayısıyla da makine ucu değişimi sayısını en aza indirmektir. Bu problem için bir matematiksel model yazılmış, bir sezgizsel yöntem ve bu yöntemin sonuçlarını iyileştirmek için de bir arama algoritması geliştirilmiştir.

Son olarak, işlerin ve makine uçlarının atanması, çizelgelenmesi ve makine ucu değişimi problemlerinin hepsi bir arada ele alınmıştır. Bu problem için matematiksel model ve kısıt programlama modeli yazılmıştır. Ayrıca problemin ayrıştırılmasına dayalı bir sezgisel geliştirilmiştir.

Bu proje raporu 6 bölümden oluşmaktadır. İkinci bölümde işlenilen problemlerin literatür taraması verilmiştir. Sonraki üç bölümde sırasıyla işlerin ve makine uçlarının çizelgelenmesi, işlerin gruplanması ve çizelgeleme ve makine ucu değişimi problemleri ele alınmıştır. Son bölümde ise sonuç ve gelecek araştırma konuları verilmiştir.

2. LİTERATÜR TARAMASI

Esnek imalat sistemleri üzerine genel bir tanım ve literatür araştırması içeren çalışmalardan (GRAY vd., 1993) esnek imalat sistemlerinde makine ucu yönetiminin önemi ve zorluğunu tartışmışlardır. (CRAMA, 1997) esnek imalat sistmelerinde karşılaşılan problemleri tanımlamış ve çözüm yöntemleri üzerine yapılan çalışmaları özetlemiştir. (BLAZEWICZ ve FINKE, 1994) esnek imalat sistemlerinde kaynak yönetimi ve çizelgeleme problemleri üzerine bir literatür çalışması yapmışlardır.

Literatürdeki benzer çalışmaları birkaç başlık altında toplamak mümkündür. Aşağıda ilgili çalışmalar farklı başlıklar altında verilmiştir.

2.1. İşlerin Gruplanması Problemi

Birinci gruptaki çalışmalar işleri gruplayarak, her grup arasındaki makine ucu değiştirme aşamalarının sayısını en aza indirmeyi amaçlamışlardır. Bu problem üzerindeki çalışmalar amaç fonksiyonlarına göre iki gruba ayrılabilir: işleri gruplayarak, her grup arasındaki makine ucu değiştirme aşamalarının sayısını en aza indirmek ve değiştirilen makine ucu sayısını en aza indirmek. Birinci gruptaki çalışmalarda makine ucu değiştirme süresinin sabit ve değiştirilen makine ucunun sayısından bağımsız olduğu varsayılmaktadır. İkinci gruptaki çalışmalarda ise makine ucu değiştirme süresinin değiştirilen makine ucu sayısı ile doğru orantılı olarak arttığı varsayılmaktadır. (CRAMA, 1997) özdevimsel imalat sistemlerinde karşılaşılan problemleri ve çözüm yöntemlerini içeren çalışmasında makine ucu değişimi problemlerine de yer vermiştir.

(TANG ve DENARDO, 1988b) değiştirilen makine ucu sayısından bağımsız olarak değiştirme zamanının sabit olduğunu varsaymış ve en az sayıda grup elde edecek şekilde ayırmışlardır. En iyi çözüme alt ve üst sınırlar geliştirmişler, daha sonra bu sınırları dal-sınır algoritması içinde kullanmışlardır. (SONG ve HWANG, 2002) aynı problem üzerinde makine ucu taşıyıcısının hareket sıklığını en aza indirmeyi amaçlayarak çalışmışlardır ve belirli bir iş sırası en iyi makine ucu kuralını geliştirmişlerdir. (DENİZEL, 2003) gerekli makine haznesi sayısını en aza indirmeyi amaçlamıştır. Lagrangean ayrışımı yöntemi ile alt ve üst sınırlar geliştirmiş, bu sınırları dal-sınır algoritmasında kullanmıştır. (KONAK vd., 2008) tek

makinenin olduğu bir esnek imalat sisteminde makine ucu değiştirme aşamalarının sayısını en aza indirmek amacıyla tabu arama yöntemi kullanarak iki algoritma geliştirmişlerdir. (KONAK ve KULTUREL-KONAK, 2007) aynı problemin çözümü için karınca kolonisi yöntemi ile bir algoritma geliştirmiştir.

Makine ucu değiştirme sayısını en aza indirmeyi amaçlayan çalışmaların başında (TANG and DENARDO, 1988a) gelmektedir. İş sırasının sabit olduğu özel durum için en iyi çözümü bulan bir algoritma geliştirmişlerdir. İşlerin sıralanması problemi için de sezgisel yöntemler önermişlerdir. (CRAMA vd., 1994) makine ucu değiştirme sayısını en aza indirerek iş ve makine ucu yükleme çizelgelemesi üzerinde çalışmışlardır. Problemin hesaplama karmaşıklığı üzerine sonuçlar elde etmişler ve sezgisel yöntemler geliştirmişlerdir. (BARD, 1988) aynı problemi doğrusal olmayan tamsayılı program olarak modellemiş ve çözümü için sezgisel yöntem önermiştir.

2.2. Operasyon Atama Problemi

Operasyonların makinelere atanması problemi üzerine çalışmalardan (TOKTAY ve UZSOY, 1998), problemi en büyük akış problemi şeklinde formüle etmiş ve problemin NP-zor olduğunu göstermişlerdir. Benzer bir çalışmada, (AKÇALI vd., 2005) bir iş için kullanılabilecek makine ucu sayısının ve bir makine üzerinde gerçekleştirilebilecek kurulumların sınırlı olduğunu varsaymışlardır. Her iki çalışmada da problemin çözümü için sezgisel algoritmalar önerilmiştir. (SHANKER ve SRINIVASULU, 1989) toplam iş yükünü ençoklamak amacıyla bir grup işin seçilip makinelere atanması problemi için karışık tamsayılı programlama modeli ve iki aşamalı bir dal-sınır algoritması geliştirmiştir. (SWARNKAR ve TIWARI, 2004), makine yüklemesi problemi için tabu arama ve benzetimli tavlama algoritmalarından faydalandıkları hibrid bir algoritma önermiştir.

2.3. Operasyon ve Makine Ucu Atama Problemi

Diğer bir grup çalışmada operasyonlar ile birlikte gerekli makine uçlarının da makinelere atanması problemi üzerinde çalışılmıştır. Fakat literatürdeki bu çalışmalar iki problemi ayrı ayrı ele almışlardır. Genel yaklaşım öncelikle operasyon atamalarını yapıp veya daha önce yapıldığını varsayıp sonraki aşamada makine uçlarını yerleştirmek şeklindedir.

(D'ALFONSO ve VENTURA, 1995) makinelerin sınırlı sayıda makine ucu haznesi olduğu ve makine uçlarının birden fazla hazne gerektirdiği durumda makineler arasındaki taşımaları en aza indirmeyi amaçlamışlardır. Lagrange gevşetim yöntemi ve çizge teorisi tabanlı bir sezgisel yöntem kullanarak çözüm getirmişlerdir. (VENTURA vd., 1988) makine ucu yükleme problemini atama problemi olarak ifade etmişlerdir.

Operasyonların ve gerekli makine uçlarının makinelere atanması problemi birlikte yalnızca proje yürütücüsü tarafından çalışılmıştır. (BİLGİN, 2004), (ÖZPEYNİRCİ, 2007), (BİLGİN ve AZİZOĞLU, 2006, 2009), (ÖZPEYNİRCİ ve AZİZOĞLU, 2009a, 2009b, 2009c, 2010) problemin matematiksel modelini yazmış, NP-zor olduğunu göstermiş, kesin çözümü için dal-sınır algoritması önermiştir. En iyi sonuca yakın çözümler bulan sezgisel yöntemler geliştirmek için demet biçimli arama algoritması ve tabu arama yöntemlerinden faydalanmışlardır. Lagrange gevşetim yöntemi ile alt ve üst sınırlar geliştirmişlerdir.

2.4. Operasyon Atama ve Çizelgeleme Problemi

Operasyonların makinelere atanması yanında sıralamasının belirlenmesi de çok önemli ve zor bir problemdir. Sıralama ve çizelgeleme problemleri literatürde oldukça geniş bir alana yayılan, hem akademik çalışmalarda hem de pratik uygulamalarda ilgi çeken bir konudur. (PINEDO, 2002) çizelgeleme konusunda oldukça önemli bir kaynaktır.

Literatürde esnek imalat sistemlerinde operasyon ve makine ucu atama ve çizelgeleme problemlerini birlikte içeren az sayıda çalışma vardır. Problemi genel yapısıyla ele alan bir çalışma olarak (KASHYAP ve KHATOR, 1996) makine ucu paylaşımı için simülasyon sonuçlarını sunmaktadır. Kullandıkları performans ölçütlerinden birisi de bizim çalışmamızdaki gibi son işin tamamlanma zamanıdır.

Projede bizim üzerinde çalıştığımız problem işlerin ve gerekli makine uçlarının makinelere atanması ve çizelgelemelerinin yapılmasıdır. Literatürde bu üç problemi ele alan ve aynı anda çözen çalışmalar çok az sayıdadır. Projede tanımlanan haliyle problem üzerinde daha önce yapılmış bir çalışma ise bulunmamaktadır.

Çalışmaların çoğunluğu problemleri hiyerarşik bir yapıda ele almıştır. Bu çalışmalardan (AVCI ve AKTURK, 1996) öncelikle makine uçlarının makinelere atandığı, sonra makine ucu haznelerinin düzenlendiği ve en son operasyon sıralamasının yapıldığı hiyerarşik bir algoritma geliştirmiştir. (GAMILA ve MOTAVALLI, 2003) yükleme ve rotalama problemlerini tam sayılı programlama modeli yardımıyla çözüp sonraki aşamada bu modelin sonucunu kullanıp detaylı çizelgelemeyi sezgisel yöntemle yapmışlardır. (PERSI vd., 1999) probleme hiyerarşik bir yaklaşım öneren bir diğer çalışmadır. Dört aşamalı çözüm yönteminde sırasıyla yığınlama (batching), yığın çizelgeleme, yığın bağlama (batch linking) ve her yığın içinde parçaları çizelgeleme problemlerini ele almışlardır.

Çizelgeleme ve makine ucu atama problemlerini beraber ele alan az sayıda çalışmadan biri (LEE vd., 2003) de iki aşamalı bir sezgisel önermişlerdir. Birinci aşamada aralarında öndelik ilişkisi olan işler için olası sıralamalar belirlenirken ikinci aşamada bu sıralamalar arasından makine ucu bekleme zamanını en aza indiren seçenek bulunmaktadır. (AKTURK ve OZKAN, 2001) makine uçlarının atanması ve işlerin sıralanması problemi üzerinde çalışmışlardır. Bu çalışmada, makinelerin kesme ve besleme hızlarının da işlere göre ayarlanması gerekmektedir ve işler arasında her seferinde bir makine ucunun değiştirilebildiği varsayılmıştır. Problemin çözümü için üç aşamalı ayrışımlı bir algoritma önermişlerdir. (AKTURK vd., 2003) bir CNC makinesinde işlerin sıralaması problemi üzerinde çalışmışlardır. Kesme takımı yıpranmakta ve değişime ihtiyaç duyulmaktadır. Amaç işlerin toplam bitirilme zamanlarının en aza indirilmesidir. Küçük problemler için kesin çözümler bulunmuş, ayrıca sezgisel yöntemler önerilmiştir. Kesici takım değiştirme zamanları kısa olduğunda en kısa işlem süresi (SPT) algoritmasının iyi sonuç verdiği, değişim zamanları uzadıkça diğer sezgisellerin sonuçlarının daha iyi olduğu gözlenmiştir. (ROH ve KIM, 1997) esnek imalat sistemlerinde makine ucu taşımayı gerekli makine uçlarınının bir makineden başka bir makineye veya makine ucu alanına bir taşıma sistemi ile transferi olarak tanımlamışlardır. Yükleme ve çizelgeleme problemleri üzerinde aynı anda çalışmış ve üç sezgisel yaklaşım önermişlerdir. (KUMAR N ve SRIDHARAN, 2007) çok makineli esnek imalat sistemlerinde makine ucu paylaşma problemi üzerinde çalışmışlardır. Üç senaryo üzerinde durmuşlar ve farklı çizelgeleme kuralları kullanarak benzetim analizi yapmışlardır.

İşlerin makinelere atanmasında kaynak kısıtlarını göz önünde bulunduran çalışmalar bizim problemimizle benzerlik göstermektedir. Makine uçlarını kaynak olduğunu ve sayılarının

kısıtlı olduğunu düşünürsek, bu konudaki çalışmalardan da faydalanmak mümkündür. (VENTURA ve KIM, 2003) işlerin ek kaynak gerektirdiği paralel makinelerde çizelgeleme problemi üzerinde çalışmıştır. Kaynak sayısının bir veya daha fazla olduğu iki problem için model ve çözüm yöntemi geliştirmişlerdir. Kaynak sayısının bir olduğu durum bizim problemimize yakındır, ancak yazarlar her işin süresinin bir birim olduğunu varsaymışlardır. (KELLERER ve STRUSEVICH, 2004) aynı problem için değişik durumları ve zorluklarını tartışmışlardır. Bir kaynağın olduğu durum bizim problemimize benzemekte ancak yazarlar her işin yapılacağı makinenin önceden belli olduğunu varsaymaktadır. (AGNETIS vd., 1997) iki makine için iş sıralaması verildiğinde makine ucu çizelgelemesinin kolayca çözülebildiğini göstermiş ve değişik ayrışım stratejilerini karşılaştırmışlardır. (RUPE ve KUO, 1997) işlerin sıralamasının belli olduğu durumlar için makine ucu değiştirme problemi üzerinde çalışmışlardır.

Projede çalışılan problemin gerçek hayatta da geniş uygulama alanları mevcuttur. Baskılı devre kartı ve yarıiletken üretimi bu alanlara iki örnektir. (CRAMA vd., 1997) baskılı devre kartlarının montajı üzerine bir vaka analizini sunmaktadır. (ÇATAY vd., 2003) ise yarıiletken üretiminde makine ucu planlaması üzerine çalışmışlardır. Projede çalıştığımız konuyu tam olarak ele alan bir çalışma bulunmadığı için sanayi uygulaması da literatürde yer almamaktadır.

2.5. Çizelgeleme ve Makine Ucu Değişimi Problemi

Son olarak, literatürde çizelgeleme ve makine ucu değişimi problemlerini tek makine üzerinde inceleyen çalışmalar bulunmaktadır. (CRAMA vd., 1994) problemin ayrıntılı bir incelemesini ve çözüm yöntemlerini sunmuştur. (LAPORTE vd., 2004) bir dal ve kesim, bir de dal ve sınır algoritması önermişlerdir. (ECKER ve GUPTA, 2005) makine ucu değişimlerinden kaynaklanan gecikmeleri en aza indirmeyi amaçlamış ve polinom zamanlı bir algoritma geliştirmişlerdir. (KARAKAYALI ve AZİZOĞLU, 2006)'nın amacı ise toplam akış zamanını enazlamaktır. Problemin NP-zor olduğunu göstermiş ve bir dal ve sınır algoritması geliştirmişlerdir. (TZUR ve ALTMAN, 2004) çizelgelemeye ek olarak makine ucu haznesi ataması problemini de ele almış, bir matematiksel model ve bir sezgisel algoritma sunmuşlardır. (HOP, 2005) tekdüze makine ucu büyüklüğü ve tekdüze makine ucu haznesi

kapasitesi kısıtlarını gevşetmiş, kısmi ve tam iş bölünmesi durumları için çözüm yöntemleri önermiştir.

Bu projede, makine ucu değişimi ve çizelgeleme problemleri birden fazla makine bulunan üretim ortamları için çalışılmıştır.

Bu projede, kısıtlı sayıda makine ucu haznesine sahip m tane makineye n tane işin ve bu işlerin gerektirdiği makine uçlarının atanması problemi çalışılmıştır. t çeşit makine ucu olduğu ve her çeşit makine ucundan sistemde r_k (k=1,2,…,t) tane mevcut bulunduğu varsayılmaktadır. Makine ucu haznesi kapasitelerini ve mevcut makine ucu sayılarını aşmadan işler gruplanarak makinelere atanmış ve her grup arasında makine ucu değişimi yapılmıştır. Problemin tamamına bakıldığında proje çalışması yapısı, amaç fonksiyonu ve çözüm yöntemleriyle literatürde daha önce çalışılmış problemlerden farklı bir yapıdadır.

Esnek imalat sistemlerinin literatürde ve sanayideki önemi, gerektirdiği maddi yatırımın yüksekliği göz önünde bulundurulduğunda, bu sistemlerdeki planlamanın hassas bir şekilde yapılmasının gerekliliği ortaya çıkacaktır. Çözümlerde yapılacak en ufak bir iyileştirme, sistemin verimliliği üzerinde büyük etkiye sahiptir. Yukarıda belirtildiği gibi, literatürde bu problemler genellikle hiyerarşik olarak ele alınmış, birisinin çözümü elde edildikten sonra bu çözüm için diğer problemlere en iyi sonuçlar aranmıştır. Fakat tüm sistemi eniyileme amacı taşıdığımızda problemi ayrıştırarak çözmek en iyi sonuçtan uzaklaşmamıza sebep olmaktadır. Önerilen proje ile önemli problemlerin bir arada çözülmeye çalışılması, literatürdeki bu konudaki boşluğun doldurulması hedeflenmektedir.

3. İŞLERİN VE MAKİNE UÇLARININ ÇİZELGELENMESİ PROBLEMİ

3.1. Problem Tanımı

Söz konusu problemde n iş ve bu işlerin yapılabileceği m makine mevcuttur. Her makine her işi yapabilir, fakat işlem süreleri makineden makineye değişiklik gösterebilir. i işinin j makinesindeki işlem süresi p_{ij} olarak alınmıştır. Ayrıca i işinin yapılabilmesi için $l(i)$ kümesinde yeralan makine uçlarının makineye yüklenmiş olması gerekmektedir. t çeşit makine ucu vardır ve k makine ucu çeşidinden sistemde $r(k)$ adet bulunmaktadır. Her makine ucunun, makinenin makine ucu haznesinde bir yuva işgal ettiği ve bir iş tarafından gereken makine ucu sayısının makinenin haznesinden daha fazla olmadığı varsayılmaktadır.

Problem işleri ve gerekli makine uçlarını makinelere atamak ve çizelgelemeyi yapmaktır. Amacımız en son işin tamamlanma zamanını (makespan) en aza indirmektir.

Sözü geçen problem üzerinde çalışmamızın amacı esnek imalat sistemlerinde her biri çok büyük öneme sahip atama ve çizelgeleme problemlerinin yapısını incelemek, problemin geneli için etkin çözüm yöntem(ler)i geliştirmektir. Operasyonların makinelere atanması ve hangi sırayla işleneceklerine karar verilmesi üretim planlama literatüründe önemli bir yer tutmaktadır. Esnek imalat sistemlerinde operasyonların işlenmesi için makinelere yüklenmesi gereken makine uçlarının planlanması probleme ek bir zorluk getirmektedir. Bu problemlerin bir arada çözümü için yöntemler geliştirmek amaçlanmaktadır.

Literatürde esnek imalat sistemleri, çizelgeleme ve makine ucu kararları sık çalışılmış konulardır ancak bu problemlerin hepsi birlikte projemizdeki gibi ele alınmamıştır. Erişilmek istenen sonuç gerçek hayatta sıkça karşılaşılan bu problemi çözmek için yöntem(ler) geliştirmek ve çizelgeleme literatürüne katkıda bulunmaktır. Bu proje ile alınacak sonuçların bu konuda teorik ve uygulama alanlarında çalışan diğer araştırmacılara ışık tutması hedeflenmektedir.

3.2. Matematiksel Modeller

Bu bölümde işlerin ve makine uçlarının makinelere atanması ve çizelgelenmesi problemi için geliştirilen matematiksel modeller sunulmaktadır.

3.2.1. Matematikse1 Model 1 (MIP1)

İndeksler

i, q: iş indeksi, $i, q = 1, 2, \ldots, n$

j: makine indeksi, $j = 1, 2, \ldots, m$

k: makine ucu çeşidi indeksi, $k = 1, 2, \ldots, t$

h_k: k çeşit makine ucuna ait kopya numarası, $h_k = 1, 2, \ldots, r_k$

Parametreler

p_{ij}: i işinin j makinesinde işlenme süresi

r_k: k çeşit makine ucuna ait kopya sayısı

$l(i)$: i işi için gereken makine uçlarının kümesi

Karar Değişkenleri

Cmax: son işin tamamlanma zamanı

C_i: i işinin tamamlanma zamanı

S_{ij}: i işinin j makinesinde başlama zamanı

X_{ij}: 1, eğer i işi j makinesinde işleniyorsa; 0, diğer durumda

$Z_{ih_k j}$: 1, eğer k makine ucunun h kopyası i işini j makinesinde işlemek için kullanılıyorsa; 0, diğer durumda

Y_{iqj}: 1, eğer i işi j makinesinde q işinden önce işleniyorsa; 0, diğer durumda

A_{iqh_k}: 1, eğer i ve q işleri k makine ucunun h kopyasını kullanıyorsa; 0, diğer durumda

b_{iq}: 1, eğer i ve q işleri aynı makine ucu kopyasını kullanıyorsa ve i işi q işinden önce yapılıyorsa

Matematiksel modelimizin amacı son işin tamamlanma zamanını enazlamaktır.

Enazla *Cmax*

Modelin kısıtları da aşağıda açıklamalarıyla verilmiştir.

- Her işin tamamlanma zamanı son işin tamamlanma zamanından küçük olmalıdır.

  $Cmax \geq C_i \qquad \forall i$

- İşlerin tamamlanma zamanı başlama zamanı ve işleme süresinin toplamıdır.

$$C_i = \sum_{j=1}^{m} \left(S_{ij} + p_{ij} X_{ij} \right) \qquad \forall i$$

- İşlerin atanmadıkları makinelerde başlama zamanı sıfırdır.

$$S_{ij} \leq M(X_{ij}) \qquad \forall i,j$$

- Bir iş bir makineye ancak gerekli makine uçları yüklendiyse atanabilir.

$$X_{ij} \leq \sum_{h_k=1}^{r_k} Z_{ih_k j} \qquad \forall i,j,k \in l(i)$$

- Bir işin yalnızca bir makinede işlenebilir.

$$\sum_{j=1}^{m} X_{ij} = 1 \qquad \forall i$$

- Eğer iki iş aynı makinede işleniyorsa, sonraki iş önceki iş bittikten sonra başlar.

$$S_{ij} \geq S_{qj} + p_{qj} - M(Y_{iqj}) - M(1 - X_{qj}) - M(1 - X_{ij}) \qquad \forall i \neq q,j$$

$$S_{qj} \geq S_{ij} + p_{ij} - M(1 - Y_{iqj}) \qquad \forall i \neq q,j$$

- İki iş arasında öncelik sonralık ilişkisi eğer aynı makineye atandılarsa bulunur.

$$X_{ij} + X_{qj} \geq 2\,(Y_{iqj} + Y_{qij}) \qquad \forall i \neq q,j$$

$$X_{ij} + X_{qj} \leq Y_{iqj} + Y_{qij} + 1 \qquad \forall i \neq q,j$$

- Eğer i ve q işleri aynı makine ucu kopyasını kullanıyorsa, A_{iqh_k} 1 değerini alır.

$$\sum_{j=1}^{m} Z_{ih_k j} + \sum_{j=1}^{m} Z_{qh_k j} \geq 2\left(A_{iqh_k} \right) \qquad \forall i<q,h_k | k \in l(i) \cap l(\mathrm{q})$$

$$\sum_{j=1}^{m} Z_{ih_k j} + \sum_{j=1}^{m} Z_{qh_k j} \leq A_{iqh_k} + 1 \qquad \forall i<q,h_k | k \in l(i) \cap l(\mathrm{q})$$

- Aynı makine ucu kopyasını kullanan iki iş arasında öncelik-sonralık ilişkisi vardır.

$$\sum_{h_k=1}^{r_k} A_{iqh_k} \leq M(b_{iq} + b_{qi}) \qquad \forall i<q,k \in l(i) \cap l(\mathrm{q})$$

$$b_{iq} + b_{qi} \leq 1 \qquad \forall i<q$$

$$\sum_{j=1}^{m} S_{ij} \geq \sum_{j=1}^{m} \left(S_{qj} + p_{qj} X_{qj} \right) - M(1 - b_{qi}) \qquad \forall i \neq q$$

$$\sum_{j=1}^{m} S_{qj} \geq \sum_{j=1}^{m} \left(S_{ij} + p_{ij} X_{ij} \right) - M(1 - b_{iq}) \qquad \forall i \neq q$$

- Bir iş sadece bir makine ucu kopyası kullanır.

$$\sum_{j=1}^{m} \sum_{h_k=1}^{r_k} Z_{ih_k j} \leq 1 \qquad \forall i,k \in l(i)$$

- Karar değişkenlerinin alabileceği değerler

$Cmax \geq 0$

$C_i \geq 0$	$\forall i$
$S_{ij} \geq 0$	$\forall i,j$
$X_{ij} \in \{0,1\}$	$\forall i,j$
$Z_{ih_k j} \in \{0,1\}$	$\forall i,j,k \in l(i)$
$Y_{iqj} \in \{0,1\}$	$\forall i \neq q,j$
$A_{iqh_k} \in \{0,1\}$	$\forall i<q,k \in l(i) \cap l(\mathrm{q})$
$b_{iq} \in \{0,1\}$	$\forall i \neq q$

3.2.2. Matematiksel Model 2 (MIP2)

Bu bölümde, Matematiksel Model 1'e göre daha az değişken ve kısıt içeren daha gelişmiş bir model açıklanmaktadır.

İndeksler

i, q: iş indeksi, *i, q* = 1, 2, …, *n*

j: makine indeksi, *j* = 1, 2, …, *m*

k: makine ucu çeşidi indeksi, *k* = 1, 2, …, *t*

h_k: *k* çeşit makine ucuna ait kopya numarası, h_k = 1, 2, …, r_k

Parametreler

p_{ij}: *i* işinin *j* makinesinde işlenme süresi

r_k: *k* çeşit makine ucuna ait kopya sayısı

l(*i*): *i* işi için gereken makine uçlarının kümesi

ST: Tek kopyalı makine uçlarını gerektiren iş ikilileri kümesi

MT: Birden fazla kopyalı makine uçlarını gerektiren iş ikilileri kümesi

Karar Değişkenleri

Cmax: son işin tamamlanma zamanı

C_i: *i* işinin tamamlanma zamanı

S_{ij}: *i* işinin *j* makinesinde başlama zamanı

X_{ij}: 1, eğer *i* işi *j* makinesinde işleniyorsa; 0, diğer durumda

Y_{iqj}: 1, eğer *i* işi *j* makinesinde *q* işinden önce işleniyorsa; 0, diğer durumda

W_{ih_k} : 1, eğer *k* makine ucunun *h* kopyası *i* işini işlemek için kullanılıyorsa; 0, diğer durumda

U_{iq}: 1, eğer *i* ve *q* işleri tek kopyası olan bir makine ucunu kullanıyorsa ve *i* işi *q* işinden önce yapılıyorsa; 0, diğer durumda

V_{iq}: 1, eğer *i* ve *q* işleri birden fazla kopyası olan bir makine ucunu kullanıyorsa ve *i* işi *q* işinden önce yapılıyorsa; 0, diğer durumda

Matematiksel modelimizin amacı son işin tamamlanma zamanını enazlamaktır.

Enazla *Cmax*

Modelin kısıtları da aşağıda açıklamalarıyla verilmiştir.

- Her işin tamamlanma zamanı son işin tamamlanma zamanından küçük olmalıdır.

  $Cmax \geq C_i \qquad \forall i$

- İşlerin tamamlanma zamanı başlama zamanı ve işleme süresinin toplamıdır.

  $C_i = \sum_j \left(S_{ij} + p_{ij} X_{ij} \right) \qquad \forall i$

- İşlerin atanmadıkları makinelerde başlama zamanı sıfırdır.

  $S_{ij} \leq M(X_{ij}) \qquad \forall i,j$

- Bir iş yalnızca bir makinede işlenir.

  $\sum_j X_{ij} = 1 \qquad \forall i$

- Tek kopyalı bir makine ucunu kullanan iki iş aynı zamanda yapılamaz.

  $\sum_j S_{ij} \geq \sum_j \left(S_{qj} + p_{qj} X_{qj} \right) - M \left(1 - U_{qi} \right) \qquad \forall (i,q) \in ST$

  $\sum_j S_{qj} \geq \sum_j \left(S_{ij} + p_{ij} X_{ij} \right) - M \left(1 - U_{iq} \right) \qquad \forall (i,q) \in ST$

  $U_{iq} + U_{qi} = 1 \qquad \forall (i,q) \in ST$

- Birden fazla kopyası olan bir makine ucunu kullanan iki iş aynı zamanda yapılamaz.

  $\sum_j S_{qj} \geq \sum_j \left(S_{ij} + p_{ij} X_{ij} \right) - M \left(2 - \left(W_{ih_k} + W_{qh_k} \right) \right) - M \left(1 - V_{iq} \right)$

  $\forall (i,q) \in MT,\ k \in l(i) \cap l(\mathrm{q}),\ r_k > 1$

  $\sum_j S_{ij} \geq \sum_j \left(S_{qj} + p_{qj} X_{qj} \right) - M \left(2 - \left(W_{ih_k} + W_{qh_k} \right) \right) - M \left(1 - V_{qi} \right)$

  $\forall (i,q) \in MT,\ k \in l(i) \cap l(\mathrm{q}),\ r_k > 1$

  $V_{iq} + V_{qi} \leq 1 \qquad \forall (i,q) \in MT$

  $W_{ih_k} + W_{qh_k} \leq 2 \left(V_{iq} + V_{qi} \right) \qquad \forall (i,q) \in MT,\ k \in l(i) \cap l(\mathrm{q}),\ r_k > 1$

- Aynı makineye atanmış işlerin yapılma zamanları çakışamaz.

$$S_{ij} \geq S_{qj} + p_{qj} - M\left(Y_{iqj}\right) - M\left(1 - X_{qj}\right) - M\left(1 - X_{ij}\right) \quad \forall i \neq q$$

$$S_{qj} \geq S_{ij} + p_{ij} - M\left(1 - Y_{iqj}\right) \quad \forall i \neq q$$

$$X_{ij} + X_{qj} \geq 2\left(Y_{iqj} + Y_{qij}\right) \quad \forall i \neq q$$

$$X_{ij} + X_{qj} \geq Y_{iqj} + Y_{qij} + 1 \quad \forall i \neq q$$

- Gerekli makine uçları işlere atanmalıdır.

$$\sum_{h_k} W_{ih_k} = 1 \quad \forall i,\ k \in l(i)$$

- Karar değişkenlerinin alabileceği değerler.

$$Cmax \geq 0$$

$$C_i \geq 0 \quad \forall i$$

$$S_{ij} \geq 0 \quad \forall i,j$$

$$X_{ij} \in \{0,1\} \quad \forall i,j$$

$$W_{ih_k} \in \{0,1\} \quad \forall i,k \in l(i)$$

$$Y_{iqj} \in \{0,1\} \quad \forall i \neq q,j$$

$$U_{iq} \in \{0,1\} \quad \forall i,q$$

$$V_{iq} \in \{0,1\} \quad \forall i,q$$

Ikinci modelde sağlanan gelişme temelde karar değişkenleri ve kısıt sayısının azaltılması sayesindedir. $Z_{ih_k j}$ değişkenleri W_{ih_k} değişkenleri ile değiştirilmiştir. X_{ij} değişkenleri sayesinde işlerin makinelere atanması bilgisini edindiğimiz için makineler için ayrı bir indekse ihtiyaç duyulmamaktadır. Ayrıca, A_{iqh_k} değişkenleri yeni modelde kullanılmamaktadır çünkü aynı bilgiye W_{ih_k} ve W_{qh_k} değişkenleri ile ulaşılmaktadır. Sonuç olarak yeni sunulan matematiksel modelin ilk modele göre daha kısa ve etkili olduğu söylenebilir.

3.2.3. Matematiksel Model 2-Simetri Kırma

Bir makine ucunun kopyaları birbirine eşdeğerdir. Makine ucu kopyalarını değişik bir şekilde numaralandırıp çok sayıda alternatif çözüm elde edebiliriz. Bu bölümde bu simetriyi kırmak için modele yeni kısıtlar öneriyoruz. Kısıt programlamada simetri kırma konusunda geniş bir

literatür olduğu halde matematiksel programlamada simetri kırma üzerine sınırlı sayıda çalışma bulunmaktadır. (JANS, 2009) paralel özdeş makinelerde kafile büyüklendirme problemi üzerinde çalışmıştır. Matematiksel programlama formülasyonları önermiş ve paralel özdeş makinelerden kaynaklanan simetriyi kırmak için kısıtlar eklemiştir. Matematiksel programlamada simetri kırmak amacıyla kısıt programlamadaki yaklaşımlar kullanılabilir.

Tekrar modelleme: Simetri farklı karar değişkenleri kullanarak farklı formülasyonlarla engellenebilir. (DEGRAEVE v.d., 2002) moda endüstrisinde yerleşim probleminde eşdeğer kumaş kesme örnekleri sebebiyle oluşan simetriyi kırmak için alternatif formülasyonlar önermişlerdir.

Simetri kırma kısıtları: Simetri kırma amacıyla matematiksel modele yeni kısıtlar eklenebilir. Simetri kırma kısıtları (SHERALI ve SMITH, 2001) tarafından ağ tasarımı problemleri için ve (SHERALI v.d., 2003) tarafından tesis yerleşim problemi için önerilmiştir.

Dinamik simetri kırma: Arama algoritması sırasında simetriyi belirlemek ve engellemek de mümkündür. (MARGOT, 2002a ve 2002b) ikili tamsyılı programlama problemleri için dal-sınır algoritması sırasında simetriyi kontrol etme yöntemini uygulamıştır.

Matematiksel Model 2'de birden fazla kopyası olan makine uçları için simetri kırma kısıtları ekleyebiliriz. Tablo 1'de verilen küçük örnekte 4 numaralı makine ucundan bir kopya, diğerlerinden ikişer kopya olduğunu varsayalım. 5 numaralı makine ucu 1 ve 4 numaralı işler tarafından kullanılmaktadır. 1 numaralı işin 1 numaralı kopyayı ve 4 numaralı işin 2 numaralı kopyayı kullandığı çözüm (W_{151}=1 and W_{452}=1) ile 1 numaralı işin 2 numaralı kopyayı ve 4 numaralı işin 1 numaralı kopyayı kullandığı çözüm (W_{152}=1 and W_{451}=1) eşdeğerdir. Modeli, bu çözümlerden sadece birisini değerlendirmesi için zorlamaya çalışıyoruz. Bu amaçla en iyi çözümü etkilemeden aşağıdaki kısıtı modele yazabiliriz:

$$2^2W_{151} + 2^1W_{451} \leq 2^2W_{152} + 2^1W_{452}$$

Benzer şekilde, 2 numaralı makine ucu için aşağıdaki kısıt yazılabilir:

$$2^3W_{121} + 2^2W_{221} + 2^1W_{521} \leq 2^3W_{122} + 2^2W_{221}\ 2+\ 2^1W_{5212}$$

Diyelim ki N_k, k makine ucunu kullanan işlerin sayısı ve z_{fk}, k makine ucunu kullanan f'inci iş olsun (f = 1, ..., N_k). Örneğin, z_{13}=2, z_{23}=3 ve z_{33}=4 olur. Simetri kırma kısıtlarını aşağıdaki gibi genelleştirebiliriz:

$$\sum_{f=1}^{N_k} 2^{(N_k+1)-f} W_{z_{fk}k1} \leq \sum_{f=1}^{N_k} 2^{(N_k+1)-f} W_{z_{fk}k2} \qquad \forall k | r_k > 1$$

Tablo 1. Küçük bir örnek

İş	**Makine ucu**
1	1,2,4,5
2	1,2,3,4
3	1,3
4	1,3,5
5	1,2,4

3.2.4. Deneyler

Yukarıda açıklanan matematiksel modellerin karşılaştırılması amacıyla değişik boyutlarda bir grup problem çözülmüştür. Bu modeller sırasıyla MIP1, MIP2 ve MIP2-Sim olarak adlandırılacaktır. İş sayısı, n=8, 10 ve 15, makine sayısı, m=2 ve 3, makine ucu sayısı, t=5 ve 8 alınmıştır. $|l(i)|$, [2,5] aralığında, r_k, [1,2] aralığında ve p_{ij}, [25,150] aralığında kesikli birbiçimli dağılım kullanılarak türetilmiştir. $l(i)$ kümesindeki makine uçları rasgele türetilmiştir.

Deney setlerinin oluşturulması sırasında literatürde daha önce yapılan çalışmalardan ve proje yürütücüsünün yayınlarından faydalanılmıştır. Bu çalışmalar arasında (CHEN v.d., 1995), (TOKTAY ve UZSOY, 1998), (ÖZPEYNİRCİ ve AZİZOĞLU, 2009) yeralmaktadır. Kullanılan verilerde çözüm süresi ve kalitesi üzerinde etkisi olan ve problem büyüklüğünü belirleyen iş sayısı, makine sayısı ve makine ucu sayısı gibi parametrelerde değişik değerler denenmiştir. Çözülmesi kolay olan problemlerle başlanıp giderek zorlaştırılmış, modelin çözemediği problem büyüklüğünde durulmuştur. Diğer parametreler için de gerçeğe yakın değerler kullanılmaya çalışılmıştır. Örneğin işlem zamanlarının makinenin modeli ve yaşına göre değişebileceği varsayılarak daha genel bir durum elde edilmiştir.

Modeller IBM OPL 6.1.1 kullanılarak çözülmüştür. Çözüm süresi için 1 saatlik bir sınır belirlenmiş, bu sürede çözülemeyen problemler için algoritma durdurulmuştur. Her parametre kombinasyonu için 10 problem türetilmiştir. Tablo 2'de bir satte çözülebilen problem sayısı (Opt), çözülen problemlerin ortalama çözüm süreleri saniye olarak (Süre) ve bir saatte çözülemeyen problemler için ortalama yüzde aralık (Ara) verilmiştir.

Tablo 2. MIP1 ve MIP2 modellerinin karşılaştırması

	MIP1			**MIP2**		
(*n*,*m*,*t*)	**Opt**	**Süre (sn)**	**Ara (%)**	**Opt**	**Süre (sn)**	**Ara (%)**
(8, 2, 5)	10	247,64	-	10	141,95	-
(8, 2, 8)	10	61,89	-	10	14,63	-
(10, 2, 5)	4	1404,19	21,39	4	788,30	34,37
(10, 2, 8)	9	468,54	7,14	10	183,19	-
(10, 3, 5)	2	1732,45	29,79	3	1118,00	28,18
(10, 3, 8)	10	1536,33	-	10	120,35	-
(15, 2, 8)	0	-	45,08	0	-	52,34

Tablo 2'de görüldüğü gibi, MIP2, MIP1'in bulabildiği tüm en iyi çözümlere ulaşmış, ek olarak iki tane en iyi çözüm bulmuştur. Ayrıca MIP2'nin çözüm süresi MIP1'inkinden ortalama olarak %56,59 daha azdır. MIP2'nin daha iyi performans göstermesinin sebebi daha önce de belirtildiği gibi daha az sayıda karar değişkeni ve kısıta sahip olmasıdır. Ortalamada MIP2'nin %52,95 daha az sayıda karar değişkeni ve %36,43 daha az sayıda kısıtı vardır. Bu sebeple, MIP2'nin MIP1'e göre daha etkili bir matematiksel model olduğu sonucuna varabiliriz. Ancak her iki model de iş sayısının 15 veya daha fazla olduğu hiç bir problemi çözememektedir.

MIP2'ye simetri kırma kısıtlarını eklemenin etkisi Tablo 3'te özetlenmiştir. Makine ucu kopyalarından kaynaklanan simetriyi engelememiz sonucunda azalan alternatif sonuç sayesinde bulunan en iyi sonuç sayısı artmış ve çözüm süresi azalmıştır.

Tablo 3. MIP2 ve MIP2-Sim modellerinin karşılaştırması

	MIP2			MIP2-Sim		
(*n*,*m*,*t*)	**Opt**	**Süre (sn)**	**Ara (%)**	**Opt**	**Süre (sn)**	**Ara (%)**
(8, 2, 5)	10	141,95	-	10	9,22	-
(8, 2, 8)	10	14,63	-	10	5,90	-
(10, 2, 5)	4	788,30	34,37	10	328,15	-
(10, 2, 8)	10	183,19	-	10	78,35	-
(10, 3, 5)	3	1118,00	28,18	10	239,88	-
(10, 3, 8)	10	120,35	-	10	16,96	-
(15, 2, 8)	0	-	52,34	2	2021,78	30,75

Simetri kırma kısıtları sayesinde modelin performansı ciddi şekilde gelişmesine rağmen hala 15 veya daha fazla iş içeren problemler çözülememktedir. Daha kısa sürede çalışan ve en iyi çözüme yakın sonuçlar veren bir algoritma ihtiyacı çok açıktır. Bundan sonraki bölümlerde bu amaçla geliştirilen yöntemler açıklanacaktır.

3.2.5. Matematiksel Model 3 (MIP3)

Çizelgeleme problemlerinin modellenmesinde zaman-indeksli modelleme yöntemi de kullanılabilmektedir. Bu yöntemde, planlama ufku birim zamanlara bölünmekte ve işlerin işlenme zamanları bu birim zamanlara atanarak bulunmaktadır. (THÖRNBLAD vd, 2012) ve (THÖRNBLAD, 2011) çeşitli üretim planlama problemlerini zaman-indeksli modelleme yöntemiyle modellemişlerdir.

İşlerin ve makine uçlarının çizelgelenmesi problemi için zaman-indeksli modelleme yöntemiyle yeni bir matematiksel model yazılmıştır.

İndeksler

i, q: iş indeksi, $i, q = 1, 2, \ldots, n$

j: makine indeksi, $j = 1, 2, \ldots, m$

k: makine ucu çeşidi indeksi, $k = 1, 2, \ldots, t$

h_k: k çeşit makine ucuna ait kopya numarası, $h_k = 1, 2, \ldots, r_k$

u: zaman aralığı, u=1,2,…,T

Parametreler

p_{ij}: i işinin j makinesinde işlenme süresi

r_k: k çeşit makine ucuna ait kopya sayısı

$l(i)$: i işi için gereken makine uçlarının kümesi

Karar değişkenleri

Cmax: son işin tamamlanma zamanı

C_i: i işinin tamamlanma zamanı

S_{ij}: i işinin j makinesinde başlama zamanı

X_{ij}: 1, eğer i işi j makinesinde u zaman aralığının başında işlenmeya başlayacaksa; 0, diğer durumda

$Z_{ih_k j}$: 1, eğer k makine ucunun h kopyası i işini j makinesinde işlemek için kullanılıyorsa; 0, diğer durumda

Matematiksel modelimizin amacı son işin tamamlanma zamanını enazlamaktır.

Enazla *Cmax*

Modelin kısıtları da aşağıda açıklamalarıyla verilmiştir.

- i işinin j makinesinde başlama zamanı.

$$S_{ij} = \sum_{u=1}^{T} uX_{iju} \qquad \forall i,j$$

- Her işin tamamlanma zamanı son işin tamamlanma zamanından küçük olmalıdır.

$$Cmax \geq C_i \qquad \forall i$$

- Bir işin tamamlanma zamanı başlama zamanı ve işleme süresinin toplamına eşittir.

$$C_i = \sum_{j=1}^{m} \sum_{u=1}^{T} \left(S_{ij} + p_{ij} X_{iju} \right) \qquad \forall i$$

- Bir iş sadece bir defa işlenmelidir.

$$\sum_{j=1}^{m}\sum_{u=1}^{T} X_{iju} = 1 \qquad \forall i$$

- Bir iş T zamanında bitirilemeyecek bir şekilde çizelgelenemez.

$$X_{iju} = 0 \qquad \forall i,j,u=T\text{-}p_{ij}+1,\ldots,T$$

- Bir makine aynı anda en fazla bir iş yapabilir.

$$\sum_{i=1}^{n}\sum_{\mu=u-p_{ij}+1}^{u} X_{ij\mu} \le 1 \qquad \forall j,u$$

- Bir iş ihtiyaç duyduğu makine ucundan sadece bir kopya kullanır.

$$\sum_{h_k=1}^{r_k} Z_{ih_k} = 1 \qquad \forall i,k \in l(i)$$

- Bir makine ucu kopyası aynı anda sadece bir makine tarafından kullanılır.

$$\sum_{j=1}^{m}\sum_{\mu=u-p_{ij}+1}^{u} X_{ij\mu} + \sum_{j=1}^{m}\sum_{\mu=u-p_{qj}+1}^{u} X_{qj\mu} \le 1 - M\left(Z_{ih_k} + Z_{qh_k} - 2\right) \qquad \forall i<q,h,k \in l(i)\cap l(q),u$$

- Karar değişkenlerinin alabileceği değerler.

$$Cmax \ge 0$$

$$C_i \ge 0 \qquad \forall i$$

$$S_{ij} \ge 0 \qquad \forall i,j$$

$$X_{iju} \in \{0,1\} \qquad \forall i,j,u$$

$$Z_{ih_k} \in \{0,1\} \qquad \forall i,k \in l(i)$$

Zaman-indeksli modelleme ile yazılan matematiksel model çok daha az sayıda kısıt grubu içermektedir. Ancak *u* indeksi planlama ufkunun değeri *T* kadar değer alabilmektedir ve uzun bir planlama ufku olduğu zaman karar değişkenlerinin ve kısıtların sayısı çok artmaktadır.

3.2.6. Deneyler

Matematiksel Model 3 Bölüm 3.2.4'te açıklanan problemler üzerinde test edilmiştir. Sonuçlar Tablo 4'de verilmektedir.

Tablo 4. MIP3 deney sonuçları

(*n*,*m*,*t*)	Opt	Süre (sn)	Ara (%)
(8, 2, 5)	10	157,99	-
(8, 2, 8)	10	46,87	-
(10, 2, 5)	1	1176,03	23,69
(10, 2, 8)	9	834,70	16,58
(10, 3, 5)	0	-	32,41
(10, 3, 8)	6	955,92	16,03
(15, 2, 8)	0	-	51,81

Tablo 4'de görüldüğü gibi MIP3, MIP1 ve MIP2'ye göre daha kötü bir performans göstermektedir. MIP3'ün performansı planlama ufkunun uzunluğuna bağlı olduğu için problem parametrelerinin değerlerine göre farklı çözüm süreleri ve bulunan en iyi çözüm sayısı gözlenebilir.

3.3. Ayrıştırma Sezgiseli

Bu bölümde, problemin ayrıştırılmasına dayanan bir sezgisel sunulmaktadır. Bu yöntemde problem iki adımda çözülür: İlk olarak işleri makinelere makine ucu ihtiyaçlarını da gözeterek atayan ve makine yüklerini dengeleyen bir model çözülür. İlk modelde iş yükünü dengelemek ikinci modelde en son yapılan işin tamamlanma zamanını en azlamaya yardım eder. İkinci model de ilk modelin atamalarını kullanarak işlerin ve makine uçlarının çizelgelemesini yapar.

Amaç tüm makineler arasında en fazla iş yükünü en azlamaktır:

(D1) Enazla *Pmax*

Kısıtlar aşağıdaki gibidir:

- *Pmax* tüm makinelerdeki en fazla iş yüküdür.

$$Pmax \geq \sum_{i=1}^{n} p_{ij} X_{ij} \qquad \forall j$$

- Bir iş yalnızca bir makinede işlenir.

$$\sum_{j} X_{ij} = 1 \qquad \forall i$$

- Karar değişkenlerinin alabileceği değerler.

  $X_{ij} \in \{0,1\}$ $\forall i,j$

D1 modelinin çözümüne göre işlerin makinelere atanmasından sonra aşağıdaki problem çözülerek çizelgeleme yapılır. İşleri yapacak makineler bilindiği için, bu modelde bazı parametre ve karar değişkenlerinde değişiklik yapılmıştır.

Parametreler

X_{ij}: 1, eğer i işi j makinesinde işleniyorsa; 0, diğer durumda

p_i: i işinin işlenme süresi

Karar değişkenleri

S_i: i işinin başlama zamanı

D2 modelinin amacı en son yapılan işin tamamlanma zamanını en azlamaktır.

(D2) Enazla *Cmax*

Modelin kısıtları açıklamalarıyla birlikte aşağıda verilmiştir.

- Her işin tamamlanma zamanı son işin tamamlanma zamanından küçük olmalıdır.

  $Cmax \geq C_i$ $\forall i$

- İşlerin tamamlanma zamanı başlama zamanı ile işlenme süresinin toplamıdır.

  $C_i = S_i + p_i$ $\forall i$

- Her işin gerektirdiği makine uçlarının atandığı makineye yüklenmesini gerekmektedir.

  $\sum_{h_k=1}^{r_k} Z_{ih_k j} = 1$ $\forall i,j,k \in l(i) | X_{ij}=1$

- Eğer i işi bir makinede q işinden sonra geliyorsa, i işinin başlama zamanı q işinin tamamlanma zamanından büyük olmalıdır.

  $S_i \geq S_q + p_q - M(Y_{iqj})$ $\forall i \neq q,j | X_{ij}=1 \ \& \ X_{qj}=1$

  $S_q \geq S_i + p_j - M(1-Y_{iqj})$ $\forall i \neq q,j | X_{ij}=1 \ \& \ X_{qj}=1$

- Eğer iki iş aynı makineye atandıysa, bu durumda bu iki iş arasında öncelik-sonralık ilişkisi olacaktır (Y_{iqj} ve Y_{qij} değişkenlerinden birisi pozitif değer alacaktır).

  $Y_{iqj} + Y_{qij}=1$ $\forall i \neq q,j | X_{ij}=1 \ \& \ X_{qj}=1$

  $Y_{iqj} + Y_{qij}=0$ $\forall i \neq q,j | X_{ij} + X_{qj}<2$

- Aynı makine ucu kopyasını kullanan işlerin işlenme zamanı çakışamaz.

$$S_i \geq S_q + p_q - M(1-b_{qi}) \qquad \forall i \neq q$$

$$S_q \geq S_i + p_i - M(1-b_{iq}) \qquad \forall i \neq q$$

- Eğer *i* ve *q* işleri aynı makine ucu kopyasını kullanıyorsa, A_{iqh_k} 1 değerini alır.

$$\sum_{j=1}^{m} Z_{ih_k j} + \sum_{j=1}^{m} Z_{qh_k j} \geq 2\left(A_{iqh_k}\right) \qquad \forall i<q, h_k | k \in l(i) \cap l(\mathrm{q})$$

$$\sum_{j=1}^{m} Z_{ih_k j} + \sum_{j=1}^{m} Z_{qh_k j} \leq A_{iqh_k} + 1 \qquad \forall i<q, h_k | k \in l(i) \cap l(\mathrm{q})$$

- Aynı makine ucu kopyasını kullanan iki iş arasında öncelik-sonralık ilişkisi vardır.

$$\sum_{h_k=1}^{r_k} A_{iqh_k} \leq M(b_{iq} + b_{qi}) \qquad \forall i<q, k \in l(i) \cap l(\mathrm{q})$$

$$b_{iq} + b_{qi} \leq 1 \qquad \forall i<q$$

$$\sum_{j=1}^{m} S_{ij} \geq \sum_{j=1}^{m}\left(S_{qj} + p_{qj} X_{qj}\right) - M(1-b_{qi}) \qquad \forall i \neq q$$

$$\sum_{j=1}^{m} S_{qj} \geq \sum_{j=1}^{m}\left(S_{ij} + p_{ij} X_{ij}\right) - M(1-b_{iq}) \qquad \forall i \neq q$$

- Bir iş sadece bir makine ucu kopyası kullanır.

$$\sum_{j=1}^{m} \sum_{h_k=1}^{r_k} Z_{ih_k j} \leq 1 \qquad \forall i, k \in l(i)$$

- Karar değişkenlerinin alabileceği değerler.

$$S_i \geq 0 \qquad \forall i, j$$

$$Cmax \geq 0$$

$$C_i \geq 0 \qquad \forall i$$

$$Z_{ih_k j} \in \{0,1\} \qquad \forall i, j, k \in l(i)$$

$$Y_{iqj} \in \{0,1\} \qquad \forall i \neq q, j$$

$$A_{iqh_k} \in \{0,1\} \qquad \forall i<q, k \in l(i) \cap l(\mathrm{q})$$

$$b_{iq} \in \{0,1\} \qquad \forall i \neq q$$

3.3.1. Deneyler

Ayrıştırma sezgiseli Bölüm 3.2.4'de belirtilen problemlerde test edilmiştir ve Tablo 5'de ortalama sapmalar, ortalama çözüm süreleri ve sezgiselin kaç problemde en iyi sonucu

bulabildiği raporlanmıştır. En iyi çözümü bilinen problemler için en iyi sonuçtan sapmalar verilmiştir. En iyi sonucu bilinmeyenler içinse 1 saat sonunda CPLEX tarafından verilen üst sınırdan sapmalar raporlanmıştır.

Tablo 5. Ayrıştırma sezgiseli için deney sonuçları

Problem boyutu	**Opt**	**Ara (%)**	**Süre (sn)**
n=10, m=2, t=5	1	2,88	651,03
n=10, m=2, t=8	4	5,82	16,00
n=10, m=3, t=5	-	5,46	239,04
n=10, m=3, t=8	2	4,93	3,18
n=15, m=2, t=8[1]	-	0,89	1.191,17

D1 çok kısa sürede çözülebilen kolay bir problemdir. Fakat işlerin makinelere atanması belli olmasına rağmen, D2 zor bir problemdir. Bu sebepten D2 için de çözüm süresine 1 saatlik bir sınır konmuş ve sonuç elde edilemezse program durdurulmuştur.

Problemin ayrıştırılması sonucunda orijinal modele göre problemler daha kolay çözülebilmektedir. Fakat iş sayısı 15'e yükseldiğinde bazı probemler için D2'nin sonucuna ulaşılamamıştır. Diğer taraftan, makine ucu çeşidinin arttırılması çözüm süresinin azalmasına sebep olmaktadır.

Küçük problemler için en iyi çözümden veya üst sınırdan sapmalar makul miktardadır. Parametre değerlerindeki değişikliklerle çözüm kalitesi arasında herhangi bir bağlantı gözlenmemiştir. İş, makine veya makine ucu çeşidi sayısını arttırmak sezgiselin performansını ciddi miktarda etkilememektedir.

3.4. Tabu Arama Algoritması

Projenin bu kısmında tabu arama sezgiseli kullanılarak kısa sürede en iyi çözüme yakın sonuçlar elde etmek amaçlanmıştır. Tabu arama, kombinatoryel eniyileme problemlerine çözüm bulmak için (GLOVER, 1986) tarafından geliştirilmiş bir meta-sezgisel yöntemdir.

[1] 1 saat içinde 4 problemin çözümüne ulaşılabilmiştir.

Tabu arama algoritması bir olurlu sonuç ile başlayıp her iterasyonda mevcut çözümün komşularından birine geçer. Genellikle komşular arasında en iyi çözüm seçilir; ama bu çözüm bir öncekinden daha iyi olmak zorunda değildir. Bir çözümden diğerine geçiş yapıldıktan sonra bu geçiş bir süreliğine tabu olarak adlandırılır ve yasaklanır. Amaç, aynı çözümlerin tekrarlanmasını, diğer bir deyişle çözüm uzayının aynı bölümlerinin araştırılmasını önlemektir. Tabu olarak adlandırılan bir geçiş sadece o zamana kadar bulunan en iyi sonuç elde edildiyse yapılabilir.

Tabu arama yöntemi iki strateji kullanır: yoğunluk ve çeşitlendirme. Yoğunluk stratejisi sadece mevcut çözümün komşuları arasında arama yapar. Başlangıç komşuluğunda belirli bir sayıda çözüm incelendikte sonra algoritma çeşitlendirme stratejisi ile farklı bir komşuluktaki bir çözüme atlar. Sıklığa dayalı bellek yapısı yoğunluk aşamasında yapılan geçişleri tutar ve arama uzayının farklı bir bölümünde bir başlangıç çözümü bulunnmasını sağlar. Algoritma çeşitlendirme aşaması belirli bir sayıda tekrar ettikten sonra durdurulur. Tabu arama hakkında ayrıntılı bilgi (GLOVER ve LAGUNA, 1997)'de bulunabilir.

Tabu arama algoritması çalışmaları sırasında da öncelikle (BİLGİN ve AZİZOĞLU, 2009)'nun esnek imalat sistemlerinde operasyon atama ve kapasite paylaştırma problemi için geliştirdiği tabu arama algoritmasından faydalanılmıştır. (CAO vd., 2005) paralel makinelerde çizelgeleme problemi için bir tabu arama algoritması geliştirmiştir. Tabu arama algoritması geliştirilirken (ROH ve KIM, 1997)'in geliştirdiği sezgisellerden de faydalanılmıştır.

Bu çalışmada geliştirilen tabu arama algoritması aşağıda açıklanmıştır.

3.4.1. Çözüm Sunumu

Her bir makinedeki atanan işlerin sırası çözüm temsili olarak belirlenmiştir. Aşağıda örnek bir temsil verilmiştir:

Makine 1 : iş1 – iş5 – iş7
Makine 2 : iş2 – iş9 – iş10
Makine 3 : iş3 – iş4 – iş6

3.4.2. Kısıt Yönetimi

Yapılabilir olmayan çözümler gözardı edilmektedir. Her yeni çözümün değerlendirilmesi öncesinde *Feasibility_Check()* fonksiyonu çalıştırılmakta ve olumsuz sonuç alındığında çözüm elenmektedir.

3.4.3. Başlangıç Çözümü

Başlangıç çözümü bir sezgisel yardımıyla bulunmaktadır. Bu sezgisel işleri makinelere aşağıdaki öncelik kuralına göre atamaktadır. (ROH ve KIM, 1997)'de benzer bir öncelik kuralı kullanılmıştır.

$$\pi_{ij} = p_{ij} \times a_{ij} \times b_{ij}^2$$

π_{ij} : i işinin j makinesindeki öncelik değeri

a_{ij} : i işi j makinesine atandığında gereken ek makine ucu sayısı (elimizde bulunan kopyalar için)

b_{ij} : i işi j makinesine atandığında gereken ek makine ucu sayısı (elimizde bulunmayan kopyalar için)

Her makinede ilk sırada yapılacak işi belirlerken o makinede işlenme süresi en kısa olan iş seçilir. Daha sonraki işlerde π_{ij} değeri küçük olana öncelik verilmektedir. Sezgiselin adımları aşağıda açıklanmaktadır.

Adım 0. Her makinenin ilk sırasına o makinede işlenme süresi en kısa olan işi ata.
Adım 1. İlk olarak boş kalan makineyi bul. Bu makine j' olsun.
Adım 2. j' için öncelik değerlerini (π_{ij}) hesapla. En küçük öncelik değerine sahip iş i' olsun.
Adım 3. j' makinesinden i' işi için gerekmeyen makine uçlarını çıkar.
Adım 4. i' işi için gereken makine uçlarını j' makinesine yükle. Eğer gereken bir makine ucundan boşta kopya yoksa, serbest kalan kadar i' işinin başlamasını ertele.
Adım 5. Adım 1'e git.

3.4.4. Komşuluk Yapısı

İki farklı komşuluk tanımı yapılmıştır. Birinci komşuluk "karşılıklı yer değiştirme", ikincisi ise "araya ekleme" olarak tanımlanmıştır.

(1) Karşılıklı Yer Değiştirme :

Birinci işten başlanarak, işlerin farklı makinede olup olmadığı bakılmadan işler yer değiştirilir.

(2) Araya Ekleme :

Her iş mevcut bulunduğu yerden çıkarılır ve farklı sıraya eklenir. Yine bu komşulukta da farklı makinede olup olmadığına bakılmaz.

Her yinelemede, iki komşuluktan çıkan atamalar karşılaştırılır ve daha iyi olanı seçilir. Seçilen sonuca göre tabu süreleri değiştirilir. Komşuluktan çıkan atamalar olumsuz ise değerlendirilmez.

3.4.5. Yoğunluk Aşaması

Arama sadece mevcut çözümün komşularında yapılır. En iyi iyileştirmeyi veren sonuç eğer tabu listesinde değilse aday çözüm olarak seçilir.

3.4.6. Çeşitlendirme Aşaması

Arama, sıklığa dayalı bellek yapısı göz önünde bulundurularak gezilmeyen bölgelere gidilmek için yapılır. Bu aşamada algoritma farklı bir başlangıç çözümü ile başlar. Başlangıç çözümünün belirlenmesinde sıklığa dayalı bellek yapısı temeldir.

3.4.7. Sıklığa Dayalı Bellek Yapısı

Bellek yapısında, i işinin j makinesine atanma sayısı tutulur. Bir iş makineye atandığı zaman atanma sayısı bir arttırılır, $f_{ij} = f_{ij} + 1$. Bu bellek yapısı sayesinde farklı bir başlangıç çözümü kolayca elde edilir.

3.4.8. Sonlandırma

Yineleme sayısı önceden belirlenen seviyeye ulaştığı zaman algoritma sona erer.

3.4.9. Deneyler

Tabu Arama algoritması bölüm 3.2.4'de belirtilen problemlerde test edilmiştir. Tabu arama parametreleri aşağıdaki gibi belirlenmiştir:

- Tabu Süresi: tabu süresi $\sqrt{iş\ sayısı}$ olarak belirlenmiştir.
- İç Döngü: iç döngü için yineleme sayısı 100 olarak belirlenmiştir.
- Dış Döngü: dış döngü için yineleme sayısı 30 olarak belirlenmiştir.

Tabu Arama algoritması Microsoft Visual Studio 2008 C dilinde kodlanmıştır.

Başlangıç çözümü ve tabu arama algoritması için elde edilen sonuçlar Tablo 6'da verilmiştir. Başlangıç çözümü çok kısa sürede bulunduğu için çözüm zamanı tabloda verilmemiştir. En iyi çözümü bilinen problemler için en iyi çözümden sapmalar verilmiştir. En iyi çözümü bilinmeyen problemler içinse CPLEX tarafından 1 saat sonunda verilen üst sınırdan sapma raporlanmıştır.

Tablo 6. Başlangıç çözümü ve tabu arama algoritması deney sonuçları

	Başlangıç Çözümü		**Tabu Arama**		
Problem boyutu	**Opt**	**Ara (%)**	**Opt**	**Ara (%)**	**Süre (sn)**
n=8, *m*=2, *t*=5	1	10,07	9	0,31	107,06
n=8, *m*=2, *t*=8	-	22,71	7	3,19	112,51
n=10, *m*=2, *t*=5	-	13,58	1	0,00	208,91
n=10, *m*=2, *t*=8	-	24,34	3	3,47	191,47
n=10, *m*=3, *t*=5	-	27,99	0	0,00	291,09
n=10, *m*=3, *t*=8	-	36,59	2	2,26	229,99
n=15, *m*=2, *t*=8	-	15,13	0	2,98	635,58

Yukarıdaki tabloda başlangıç çözümünün en iyi çözümden oldukça uzak olduğu görülmektedir. Tabu arama algoritması başlangıç çözümünü iyileştirip en iyiye yakın sonuçlar elde etmektedir. Çözüm kalitesinde problem büyüklüğüne bağlı olarak anlamlı bir değişim görülmemektedir. Ancak artan iş sayısı ile çok sayıda komşu sonuç oluştuğu için çözüm süresi artmaktadır.

Tabu arama algoritmasının sağlamlığını ölçmek amacıyla en iyi çözümle olan yüzde farkı değerlerine t-test uyguladık. Bu amaçla 1-α=0.95 güvenle aşağıdaki hipotezi tasarladık:

$H_0 : \mu \leq 0.02$

$H_1 : \mu \geq 0.02$.

En iyi çözümünü bildiğimiz 36 problem bulunmaktadır. t istatistiğinin değeri aşağıdaki eşitlikle bulunmuştur:

$$t = \frac{\bar{d} - 0.02}{S / \sqrt{n}}$$

d ve *S* örneklemin ortalama ve standart sapmasıdır. Bu değerler *d*=0,02536 ve *S*= 0,04331 olarak hesaplanmış ve t istatistiğini değeri de MINITAB 15.1.1.0 kullanılarak *t*= 0,74 olarak bulunmuştur. $t_{0.05,35}$=1.69>0.74 olduğu için, sıfır hipotezi reddedilememektedir. Sonuç olarak, tabu arama algoritmasının bulduğu sonuçların en iyi çözümden farkının %2'den daha az olduğunu söyleyebiliriz.

3.5. Kısıt Programlama Yaklaşımı

Kısıt programlama yaklaşımı yapay zeka konusunun bir alt dalıdır ve kombinatoryel problemin özelliklerini kısıt tatmin probleminin özelliklerine çevirir. Kısıt Programlama yaklaşımı özellikle çizelgeleme problemlerinin modellenmesinde ve çözülmesinde tercih edilen bir yöntem olmuştur. Bu yaklaşımın tercih edilmesinin en önemli sebebi etkin kısıt-tabanlı çizelgeleme algoritmalarıdır. Çizelgeleme problemlerinin kısıt programlama ile modellenmesinde iki temel konsept vardır; aktivite ve kaynak. Aktivite, bir olayı temsil etmek için kullanılır. Temsil ettiği olaya ait başlangıç zamanı, süre ve bitiş zamanı olarak üç farklı bilgiyi de tutar. Burada bitiş zamanı, başlangıç zamanı ve olayın süresinin toplamına eşittir. Kaynak ise, başlama ve bitiş zamanları kesişmeyecek ve bitiş zamanı gelmeden araya girilemeyecek işler kümesini modeller. Kaynak bir aktiviteyi işlemeye başladığı zaman, aktivite bitene kadar ne durdurulabilir ne de değiştirilebilir. Birçok kısıt programlama

dillerinde, yukarıda tanımlanan konseptler bulunmaktadır fakat gösterimleri farklılık gösterebilir. Bu projede, kısıt programlama yaklaşımı için IBM ILOG CP Optimizer 2.3 yazılımı kullanılmıştır. Bu yazılımda, birçok kısıt programlama algoritmaları ve modelleme için gerekli kaynak kısıtları tanımlanmış şekilde bulunmaktadır.

3.5.1. Literatürede Kısıt Programlama

Kombinatoryel problemlerin özellikle çizelgeleme problemlerinin modellenebilmesi ve çözülebilmesi için birçok kısıt ve belirli yapılar için algoritmalar geliştirilmiştir. (BAPTISTE ve LE PAPE, 1993) teorik ve deneysel olarak alternatif kaynaklı çizelgeleme için kısıt propagasyon tekniklerini araştırmışlardır. Çizelgeleme problemleri için kısıt programlama etkili bir araştırma alanıdır. Farklı problemlerin çalışılması dışında yazılım olarak veya çözüm yöntemi olarak da birçok çalışma mevcuttur. (LABORIE vd., 2009) kısıt programlamada kullanılabilecek yeni aralık bazlı modelleme tekniklerini araştırmışlardır. (MERCIER ve VAN HENTENRYCK, 2008) "edge finding" algoritmasını kümülatif çizelgelemeler üzerinde çalışmışlardır. (BAPTISTE ve LE PAPE, 1996) "edge finding" kısıt propagasyon algoritmasını alternatif ve kümülatif çizelgelemeler için geliştirmişlerdir. (GENT ve SMITHL, 2000) kısıt programlamada simetri kırma konusu üzerinde çalışmışlardır. Bu konu, özellikle arama uzayını daraltmada etkilidir. (VILIM, 2004) ise tekli kaynak kısıtları için filtreleme algoritmaları üzerinde çalışmıştır. (CASEAU ve LABURTHE, 1995) iş aralıkları ile alternatif kaynaklar üzerinde çalışma yapmışlardır. (FRISCH vd., 2006) sözlüksel sıralama kısıtı için propagasyon algoritması geliştirmiştir. Ek olarak, (FOCACCI ve MILANO, 2001) modellerde oluşan simetrileri elemek için global kesik yapısı geliştirmiştir. (FLENER vd., 2002) matris modellerde satır ve sütun simetrilerini kırmak için yöntem üzerinde çalışmışlardır.

Model bazlı araştırmalarda ise, (AGGOUN ve BELDICEANU, 1993) kompleks çizelgeleme ve yerleştirme problemleri çözebilmek için CHIP kısıt programlama platformunu geliştirmişlerdir. (COLOMBANI, 1996) siparişe göre (job-shop) üretim problemleri için kısıt programlama ile etkili ve pratik yaklaşım geliştirmiştir. (NUJITEN ve AARTS, 1996) birden fazla kapasiteli siparişe göre çizelgeleme problemleri için kısıt tatmin yönteminin etkisini araştırmışlardır. (LE PAPE vd., 2001) kısıt bazlı çizelgeleme çalışmalarında çizelgeleme problemlerine kısıt programlama uygulamaları üzerinde çalışmışlardır.

3.5.2. Kısıt Programlama Modelleri

Kısıt programlama modellerinde aktivite karar değişkenini tanımlamak için "*Activity*" komutu kullanılır. Örneğin, "*Activity a*" aktivite karar değişkeni olarak "*a*"nın tanımlanmasını sağlar. Bu aktivitenin yukarıda bahsedilen başlangıç zamanı, süresi ve bitiş zamanına sırasıyla; *StartOf(a), DurationOf(a)* ve *EndOf(a)* komutları girilerek ulaşılabilir. Eğer *a* aktivitesi *b* aktivitesinden önce geliyorsa (öncelik ilişkisi bulunuyorsa), bu durumda "*Before"* kısıtı; "*a Before b*" şeklinde kullanılır. Başlama ve bitiş zamanları kesişmeyecek ve bitiş zamanı gelmeden araya girilemeyecek işler kümesini modelleyen kaynak için "*UnaryResource*" komutu kullanılır. Örneğin, "*M"* böyle bir kaynak ise "*UnaryResource M*" şeklinde tanımlanır. Bazı kaynakların ise belirli bir kapasitesi olabilir. Bu kapasitenin anlamı ise bu kaynak farklı aktiviteler tarafından kapasiteyi geçmeyecek şekilde aynı anda kullanılabilir. Bu tür kaynaklara kümülatif kaynaklar denir. Örneğin, "*R"* kümülatif bir kaynak ise ve kapasitesi de "*c"* ise "*DiscreteResource R(capacity(c))*" şeklinde gösterilir. Çizelgeleme problemlerinde bir başka karşılaşılan durum ise, birden fazla eş kaynakların bulunduğu ve işlerin bu kaynaklardan sadece bir tanesine atanması durumudur. Bu çeşit kaynaklara ise alternatif kaynaklar denir. Örneğin, "*L"*, aktivite "*a*"nın kullanabileceği özdeş kaynakların listesi ise; "*Alternative(a,L)*" global kısıtı ile "*a*" değişkenin bu özdeş kaynaklardan bir tanesini seçmesi sağlanır. Ek olarak, atandığı kaynağa göre aktivitenin süresi değişiklik gösterebilir. Bu durum ise "*Alternative(a,L,P)*" kısıtı ile modele yansıtılır. Global kısıtın "*P*" parametresi, "*a*" aktivitesinin "*L*" listesindeki kaynaklara göre işlenme süresini tutmaktadır. "*a*" aktivitesinin "*M*" kaynağına atanıp atanmadığı "*hasActivity(M,a)*" kısıtı ile gösterilebilir. Eğer atanmışsa kısıt 1, atanmamışsa 0 değerini döndürür. Yukarıda bahsedilen global kısıtlar ve gösterimler bütün kısıt programlama yazılımlarında farklı gösterimlerle bulunmaktadır. Kısıt programlama algoritmaları aktivite karar değişkenin en erken başlama ve en geç bitme sınırlarını azaltarak çözüm bulmayı hedeflemektedir. Aktivitelerin aralık şeklinde gösterimi bazlı birçok etkili propagasyon algoritmaları geliştirilmiştir. Bu projede üç tane farklı kısıt programlama modeli geliştirilmiştir. Aşağıda bu modeller herhangi bir yazılıma bağlı kalmaksızın gösterilmiştir.

3.5.3. Makinelerin Kaynak Şeklinde Modellenmesi

Problemimizde, işleri aktivite olarak makineleri ise tekli kaynak şeklinde değerlendirebiliriz. Her iş, tekli kaynak kümesinden sadece bir makineye atanabilir. Kısıt programlama modeli

Şekil 1'de gösterilmiştir. Bu modele "CP1" ismi verilmiştir. CP1 modelinde, her iş bir aktivite olarak tanımlanmıştır ve (2)'de gösterilmiştir. Her bir makine ise ayrık alternatif kaynaklar olarak tanımlanmıştır ve (3)'de gösterilmiştir. *Alternative* global kısıtı ile her bir aktivite için atanabilecekleri kaynak setleri tanımlanmış ve kaynak-bağımlı işlem süreleri de gösterilmiştir. Bu kısıt ile her bir aktivite sadece bir kaynağa atanmıştır ve (5)'de gösterilmektedir. Amaç fonksiyonu ise son biten aktivitenin tamamlanma süresini en aza indirgemeye çalışmaktadır. Kısıt programlama (6)'da kullanılan *S* kümesi ise aynı makine ucu kopyalarını kullanan işleri göstermektedir. Eğer böyle işler varsa o işlerden bir tanesi diğerinden daha önce gelmelidir. *S* kümesi;

$$S = \left\{ \langle i, j, k \rangle \mid i \neq j,\ k \in l(i) \cap l(j),\ i \in \{1,\ldots,n\},\ j \in \{1,\ldots,n\} \right\}$$
where $\langle i, j, k \rangle \in S$ iff job i and job j require the same tool k

Amaç fonksiyonu:

(1) Min. $\max\{\text{endOf}(J_1),\ldots,\text{endOf}(J_n)\}$

Karar değişkenleri:

(2) Activity $J_i, \quad \forall i \in 1,\ldots,n$

(3) UnaryResource $M_j, \quad \forall j \in 1,\ldots,m$

(4) $W_{ik} \in \{1,\ldots,r_k\}, \quad \forall i \in \{1,\ldots,n\},\ \forall k \in \{1,\ldots,t\}$

Kısıtlar:

(5) $\text{Alternative}\left(J_i, [M_1,\ldots,M_m], [p_{i1},\ldots p_{im}]\right), \quad \forall i \in \{1,\ldots,n\}$

(6) $W_{ik} = W_{qk} \rightarrow \left(J_i \text{ Before } J_q \vee J_q \text{Before } J_i\right), \forall \langle i, q, k \rangle \in S$

Şekil 1. CP1 kısıt programlama modeli

3.5.4. Makinelerin ve Makine Uçlarının Kaynak Şeklinde Modellenmesi

Bir önceki kısıt programlama modelinde (CP1), sadece makineler kaynak şeklinde modellenmiştir. Fakat, makine uçlarının kaynak şeklinde modellenmemesinden dolayı bu kısıtlar mantıksal kısıtlar formatında modellenmiştir. Mantıksal kısıtlar da kısıt programlama algoritmalarının çözüm bulmasını zorlaştırır. Bir önceki modeli daha etkili hale getirmek için makine uçlarının da kaynak şeklinde modellenmesi düşünülmüştür. Her bir makine ucunun, her bir kopyası ayrı kaynak gibi düşünülmüştür. Bu şekilde, her iş ihtiyacı olan makine ucunun herhangi bir kopyasına atanabilmektedir. Bu modelde, global kısıtların tüm algoritmik gücü kullanılmaktadır. Model Şekil 2'de verilmiştir.

Amaç fonksiyonu:

(1) Min. $\max\{\text{endOf}(J_1),..., \text{endOf}(J_n)\}$

Karar değişkenleri:

(2) Activity $J_i, \quad \forall i \in 1,\ldots,n$

(3) UnaryResource $M_j, \quad \forall j \in 1,\ldots,m$

(4) UnaryResource $T_{kc}, \quad \forall k \in 1,\ldots,t, \ \forall c \in \{1,\ldots,r_k\}$

Kısıtlar:

(5) Alternative $\left(J_i,[M_1,\ldots,M_m],[p_{i1},\ldots p_{im}]\right), \quad \forall i \in \{1,\ldots,n\}$

(6) Alternative $\left(J_i,\left[T_{k1},\ldots,T_{kr_k}\right]\right), \quad \forall i \in \{1,\ldots,n\}, \ \forall k \in l(i)$

Şekil 2. CP2 kısıt programlama modeli

3.5.5. Makine Ucu ve Makine Ucu Kopya Simetrisini Kırma

Aynı tip makine ucu kopyalarının çözüm açısından bir farkı yoktur. Fakat, çözüm aramada fazladan çaba sarfedilmektedir. Bu nedenle, bu tip simetrileri kırmanın; çözüm zamanına ve kalitesine belirgin oranda katkı sağlaması beklenmektedir. Kısıt programlamada iki tip simetri vardır; çözüm simetrisi ve problem simetrisi. Çözüm simetrisi, karar değişkeni-değer ikililerinin yer değiştirmesi ile çözüm setini oluşturmasıdır. Problem simetrisi ise yine karar değişkeni-değer ikililerinin yer değiştirmesi ile kısıtların hala sağlanmasıdır. Bu iki tip simetrinin özel durumlarına değişken ve değer simetrisi denir. Değişken simetrisi, değişken kümesindeki değişkenlerin serbest bir şekilde yerlerinin değiştirilmesine; değer simetrisi ise atanabilecek değer kümesindeki değerlerin serbest bir şekilde yerlerinin değiştirilmesine denir. Problemlerdeki simetrileri kırmak için farklı yöntemler geliştirilmiştir; tekrardan modelleme, statik simetri kırma ve dinamik simetri kırma bu yöntemlerden bazılarıdır. Bu çalışmada, oluşan makine ucu ve kopyalarının simetrisini kırmak için iki yöntem kullanılmıştır. CP1 modelinde kullanılan ilk yöntemde, yeniden modelleme ile oluşan simetriler, kolon simetrilerine çevrilmiş daha sonra da statik olarak eklenen simetri kırma kısıtı (sözlüksel sıralama kısıtı) ile sonuç alınmaya çalışılmıştır. Bunu kullanabilmek için her bir makine ucu çeşidi için makine ucu çeşidi ve kopya numarası ile indekslenmiş M_k ikili karar değişkenin matrisi tanımlanır. Örneğin; $M_k[i,c] = 1$ i işine, k makine ucu çeşidinin c kopya numarası atanmış demektir. M karar değişkenini, W karar değişkeni ile ilişkilendirebilmek için kanal kısıtı kullanılmıştır.

$$W_{ik} = c \leftrightarrow M_k[i,c] = 1$$

Kanal kısıtı ile M karar değişkeni matrisinin kolonları farklı makine ucu kopyalarına denk gelmektedir. Sözlüksel sıralama kısıtı M karar değişkeni ile birlikte simetriyi kırmak için kullanılabilir. CP2 modelinde de aynı yöntem kullanılmıştır fakat W karar değişkenleri olmadığından kanal kısıtı, T değişkeni ile tanımlanmıştır.

$$\text{hasActivity}(T_{kc}, i) \leftrightarrow M_k[i,c] = 1$$

Simetri kısıtı eklenmiş modellere sırasıyla, CP1lex ve CP2lex isimleri verilmiştir. İkinci yöntemde ise, problem yeniden modellenmiş ve bir önceki modelde oluşan simetriler yeniden oluşmamıştır. Yeni modelde, tekil kaynak yerine kümülatif kaynak kullanılmıştır. Bu şekilde,

her makine ucu-kopya için yaratılan kaynaklar yerine tek kaynak yaratılmış ve bu şekilde simetrinin oluşması engellenmiştir. Bu modele CP3 ismi verilmiştir. Model Şekil 3'de verilmiştir. (4) no'lu kısıt, her bir makine ucu için yaratılan kümülatif kaynaktır ve kapasitesi de makine ucu kopya sayısı kadardır. Kısıt (6) ise her iş için gereken makine ucundan bir kopya atanması sağlanmaktadır.

Amaç fonksiyonu:

(1) Min. max{endOf(J_1),..., endOf(J_n)}

Karar değişkenleri:

(2) Activity $J_i, \quad \forall i \in 1,\ldots,n$

(3) UnaryResource $M_j, \quad \forall j \in 1,\ldots,m$

(4) DiscreteResource $T_k\left(capacity\left(r_k\right)\right), \quad \forall k \in 1,\ldots,t$

Kısıtlar:

(5) Alternative$\left(J_i,\left[M_1,\ldots,M_m\right],\left[p_{i1},\ldots p_{im}\right]\right), \quad \forall i \in \left\{1,\ldots,n\right\}$

(6) Alternative$\left(J_i,1,T_k\right), \quad \forall i \in \left\{1,\ldots,n\right\}, \ \forall k \in l\left(i\right)$

Şekil 3. CP3 kısıt programlama modeli

3.5.6. Deneyler

Bu bölümde, geliştirilen beş kısıt programlama modelinin karşılaştırılması yapılacaktır. Deney setinde, iş sayısı; 8, 10, 15, 17 ve 20, makine sayısı; 2 ve 3, makine ucu çeşidi sayısı 5 ve 8 olarak belirlenmiştir. $l(i)$ seti 2 ve 5 aralığında, r_k seti ise 1 ve 2 tekdüze dağılımdan oluşturulmuştur. İşlem süreleri ise 25 ve 150 aralığından yine tekdüze dağılımından

oluşturulmuştur. Her bir deney seti için 10 tane örnek oluşturulmuştur. Örnekler, IBM OPL 6.1.1 yazılımı ile AMD II P820, 1.8 GHz ve 3.74 GB RAM özellikli bilgisayarda çalıştırılmıştır. Her bir örnek için zaman limitinin üst sınırı bir saat olarak belirlenmiştir. Arama stratejisi olarak tekrardan başlama seçeneği kullanılmıştır. Modeller ilk önce kendi içlerinde simetri kısıtı eklenmeden önce ve sonra şeklinde karşılaştırılmıştır. Tablo 7'de CP1 ve CP1lex karşılaştırması verilmiştir.

Tablo 7. CP1 ve CP1lex modellerinin performans karşılaştırması

	CP1			CP1lex		
<iş, makine, makine ucu >	Optimal Çözüm	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı	Optimal Çözüm	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı
<8, 2, 5>	0	10/10	-	0	10/10	-
<8, 2, 8>	2	10/10	734	2	10/10	734
<10, 2, 5>	0	10/10	-	0	10/10	-
<10, 2, 8>	0	10/10	-	0	10/10	-
<10, 3, 5>	0	8/10	-	0	10/10	-
<10, 3, 8>	0	9/10	-	0	10/10	-
<15, 2, 8>	0	10/10	-	0	10/10	-
<17, 2, 5>	0	10/10	-	0	10/10	-
<17, 2, 8>	0	10/10	-	0	10/10	-
<17, 3, 5>	0	6/10	-	0	10/10	-
<17, 3, 8>	0	8/10	-	0	10/10	-
<17, 4, 5>	0	8/10	-	0	10/10	-
<17, 4, 8>	0	1/10	-	0	10/10	-
<20, 2, 5>	0	10/10	-	0	10/10	-
<20, 2, 8>	0	10/10	-	0	10/10	-
<20, 3, 5>	0	5/10	-	0	10/10	-
<20, 3, 8>	0	3/10	-	0	10/10	-

Tablo 7'ye göre, CP1 modeli CP1lex modeli ile aynı örneklerde en iyi çözüme ulaşmaktadır. Fakat, CP1lex modeli CP1 modelinin çözüm bulamadığı örneklerde de çözüm bulmaktadır.

Ek olarak, CP1lex modeli örneklerin %65'inde daha iyi sonuçlar bulmakta ve %10,32 oranında amaç fonksiyonunu iyileştirmektedir. Bu karşılaştırma ile mantıksal kısıtlarla oluşturulan modellerde simetri kırmanın etkisi açıkça görülmektedir.

Tablo 8'de CP2 modeli ile CP2lex modelinin karşılaştırılması verilmiştir. Tablo 8'e göre, global kısıtlarla oluşturulan CP2 ve CP2lex modellerinde, simetri kırmanın etkisi ters yönde görülmüştür. Simetri kırma herhangi bir iyileştirmede bulunmamış aksine ekstra hesaplama işlemlerine neden olmuştur. Bunun sonucunda, çözüm zamanları artmıştır. İki model de aynı örneklerde en iyi çözümü bulmaktadır. Fakat, CP2 modelinin çözüm zamanı yaklaşık %53 oranında CP2lex modelinden daha kısadır. Tablo 2 ve 3'den çıkarılacak sonuçlar CP1lex modelinin CP1 modelinden, CP2 modelinin de CP2lex modelinden daha iyi olduğudur.

Tablo 8. CP2 ve CP2lex modellerinin performans karşılaştırması

	CP2			CP2lex		
<iş, makine, makine ucu >	Optimal Çözüm	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı	Optimal Çözüm	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı
<8, 2, 5>	6	10/10	2,90	6	10/10	3,76
<8, 2, 8>	4	10/10	3,82	4	10/10	4,47
<10, 2, 5>	9	10/10	3,00	9	10/10	4,30
<10, 2, 8>	4	10/10	3,37	4	10/10	4,14
<10, 3, 5>	8	10/10	2,50	8	10/10	3,58
<10, 3, 8>	6	10/10	2,88	6	10/10	5,10
<15, 2, 8>	3	10/10	2,54	3	10/10	11,22
<17, 2, 5>	6	10/10	0,93	6	10/10	4,66
<17, 2, 8>	3	10/10	5,27	3	10/10	6,21
<17, 3, 5>	6	10/10	3,35	6	10/10	3,93
<17, 3, 8>	5	10/10	4,71	5	10/10	8,00
<17, 4, 5>	7	10/10	3,32	7	10/10	4,21
<17, 4, 8>	7	10/10	4,07	7	10/10	7,06
<20, 2, 5>	2	10/10	3,26	2	10/10	4,61
<20, 2, 8>	2	10/10	11,39	2	10/10	48,51
<20, 3, 5>	5	10/10	5,03	5	10/10	3,90
<20, 3, 8>	5	10/10	1,58	5	10/10	5,87

Tablo 9. CP2 ve CP1lex modellerinin performans karşılaştırması

	CP2			CP1lex		
<iş, makine, makine ucu >	Optimal Çözüm	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı	Optimal Çözüm	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı
<8, 2, 5>	6	10/10	2,90	0	10/10	-
<8, 2, 8>	4	10/10	3,82	2	10/10	734
<10, 2, 5>	9	10/10	3,00	0	10/10	-
<10, 2, 8>	4	10/10	3,37	0	10/10	-
<10, 3, 5>	8	10/10	2,50	0	10/10	-
<10, 3, 8>	6	10/10	2,88	0	10/10	-
<15, 2, 8>	3	10/10	2,54	0	10/10	-
<17, 2, 5>	6	10/10	0,93	0	10/10	-
<17, 2, 8>	3	10/10	5,27	0	10/10	-
<17, 3, 5>	6	10/10	3,35	0	10/10	-
<17, 3, 8>	5	10/10	4,71	0	10/10	-
<17, 4, 5>	7	10/10	3,32	0	10/10	-
<17, 4, 8>	7	10/10	4,07	0	10/10	-
<20, 2, 5>	2	10/10	3,26	0	10/10	-
<20, 2, 8>	2	10/10	11,39	0	10/10	-
<20, 3, 5>	5	10/10	5,03	0	10/10	-
<20, 3, 8>	5	10/10	1,58	0	10/10	-

Tablo 9'a göre, CP2 modeli belirgin bir şekilde CP1lex modelinden daha iyi performans göstermektedir. CP2 modelinin ortalama çözüm zamanı yaklaşık olarak 3 katı daha kısa, amaç fonksiyonu ise %0,10 daha iyidir. Bunun nedeni olarak, global kısıtlarla oluşturulan modelin, mantıksal kısıtlarla oluşturulan modele göre çözüm algoritmalarındaki etkisi olarak açıklanabilir. Son karşılaştırma CP2 ve CP3 modelleri arasında yapılmıştır.

Tablo 10. CP2 ve CP3 modellerinin performans karşılaştırması

	CP2			CP3		
<iş, makine, makine ucu >	Optimal Çözüm	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı	Optimal Çözüm	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı
<8, 2, 5>	6	10/10	2,90	6	10/10	2,57
<8, 2, 8>	4	10/10	3,82	4	10/10	2,50
<10, 2, 5>	9	10/10	3,00	9	10/10	3,27
<10, 2, 8>	4	10/10	3,37	4	10/10	2,80
<10, 3, 5>	8	10/10	2,50	8	10/10	2,87
<10, 3, 8>	6	10/10	2,88	6	10/10	2,60
<15, 2, 8>	3	10/10	2,54	3	10/10	3,65
<17, 2, 5>	6	10/10	0,93	6	10/10	20,9
<17, 2, 8>	3	10/10	5,27	3	10/10	5,99
<17, 3, 5>	6	10/10	3,35	6	10/10	6,19
<17, 3, 8>	5	10/10	4,71	5	10/10	6,31
<17, 4, 5>	7	10/10	3,32	7	10/10	5,80
<17, 4, 8>	7	10/10	4,07	7	10/10	6,53
<20, 2, 5>	2	10/10	3,26	2	10/10	3,09
<20, 2, 8>	2	10/10	11,39	2	10/10	48,5
<20, 3, 5>	5	10/10	5,03	5	10/10	5,31
<20, 3, 8>	5	10/10	1,58	5	10/10	6,38

Tablo 10'a göre, CP2 modeli CP3 modeli ile aynı en iyi çözümleri bulsa bile çözüm zamanları bakımından daha iyi performans göstermektedir. Bunun nedeni ise, global kümülatif kısıtının, global alternatif kısıtına göre algoritmadaki çözüme ulaşmadaki etkisinin daha az oluşudur. Sonuç olarak, geliştirilen beş kısıt programlama modeli arasından en iyi model CP2 modelidir.

4. İŞLERİN GRUPLANMASI PROBLEMİ

Projenin bu aşamasında makine ucu değişimi için harcanan zaman ön plana geçmektedir. İşler, makinelerin makine ucu haznesi kapasiteleri göz önünde bulundurularak gruplanacak, her grup arasında gerekli makine uçlarının değişimleri yapılacaktır. Makine ucu değişimi uzun bir süre gerektirdiği ve işlerin gerçekleştirilmesinde önemli bir yer tuttuğu için, değişim sayısının en aza indirilmesi amaçlanmaktadır.

Söz konusu problemin en temel halinde bir makinede yapılması gereken n iş mevcuttur. t çeşit makine ucu vardır ve i işinin yapılabilmesi için $l(i)$ kümesinde yeralan makine uçlarının makineye yüklenmiş olması gerekmektedir. Her makine ucunun, makinenin makine ucu haznesinde bir yuva işgal ettiği ve bir iş tarafından gereken makine ucu sayısının makinenin haznesinden daha fazla olmadığı varsayılmaktadır. Makine ucu haznesi c adet makine ucu alabilmektedir.

Problem makine ucu değişimi aşamalarını en aza indirecek şekilde işleri gruplamaktır.

4.1. Matematiksel Model

İndeksler

i: iş indeksi, $i = 1, 2, \ldots, n$

k: makine ucu çeşidi indeksi, $k = 1, 2, \ldots, t$

g: grup indeksi, $g = 1, 2, ..., s$

Parametreler

$l(i)$: i işi için gereken makine uçlarının kümesi

c: makine ucu haznesi kapasitesi

Karar Değişkenleri

X_{ig}: 1, eğer i işi g grubuna atandıysa; 0, diğer durumda

Y_{kg}: 1, eğer k makine ucu g grubuna atandıysa; 0, diğer durumda

Z_g: 1, eğer g grubu oluşturulduysa; 0, diğer durumda

Amaç fonksiyonu iş gruplarının sayısını en aza indirmeyi amaçlamaktadır. Bu sayede gruplar arasında yapılacak makine ucu değişimi aşamaları da en azlanacaktır.

Enazla $\sum_g Z_g$

Kısıtlar ise açıklamalarıyla birlikte aşağıda verilmiştir:

- Her iş sadece bir gruba atanabilir.

  $\sum_g X_{ig} = 1 \qquad \forall i$

- Bir iş bir gruba sadece grubun oluşturulması durumunda atanabilir.

  $Z_g \geq X_{ig} \qquad \forall i, g$

- Bir iş bir gruba gerekli makine uçları da atandıysa atanabilir.

  $X_{ig} \leq Y_{kg} \qquad \forall\ i, g, k \in l(i)$

- Makine ucu haznesi kapasitesinin aşılmamalıdır.

  $\sum_k Y_{kg} \leq c \qquad \forall g$

- Karar değişkenlerinin alabileceği değerler.

  $X_{ig} \in \{0,1\} \qquad \forall i,g$

  $Y_{kg} \in \{0,1\} \qquad \forall k,g$

  $Z_g \in \{0,1\} \qquad \forall g$

Yukarıda belirtilen problemin NP-zor olduğu (KONAK vd., 2008)'de gösterilmiştir.

4.1.1. Deneyler

Yazılan modelin doğrulanması amacıyla değişik boyutlarda bir grup problem çözülmüştür. Kullanılan problem parametreleri Tablo 11'de verilmektedir. $|l(i)|$ değerleri kesikli birbiçimli dağılım kullanılarak türetilmiştir. $l(i)$ kümesindeki makine uçları rasgele türetilmiştir.

Tablo 11. Problem parametreleri

Set	***n***	***t***	***|l(i)|***	***c***
1	10	8	U[1,2]	3
2	10	16	U[1,4]	6
3	15	8	U[1,2]	3
4	15	16	U[1,4]	6

Model Visual C++ kullanılarak kodlanmış ve CPLEX11.2 ile Pentium 4, 3 GHz, 1.96 GB RAM bilgisayarla çözülmüştür. Her parametre kombinasyonu için 10 problem türetilmiştir ve sonuçlar Tablo 12'de verilmiştir.

Tablo 12. En iyi çözümün deney sonuçları

Problem seti	**Grup sayısı**	**Çözüm süresi (sn.)**		**Problem seti**	**Grup sayısı**	**Çözüm süresi (sn.)**
1-1	3	6,74		3-1	4	7,31
1-2	3	4,93		3-2	3	6,73
1-3	3	5,02		3-3	3	7,45
1-4	3	4,89		3-4	4	9,35
1-5	3	5,06		3-5	4	6,02
1-6	2	4,45		3-6	3	7,48
1-7	4	5,17		3-7	3	4,77
1-8	3	4,58		3-8	4	6,82
1-9	3	4,81		3-9	4	6,27
1-10	2	4,19		3-10	3	6,62
2-1	3	23,62		4-1	5	217,93
2-2	3	5,31		4-2	4	9,54
2-3	4	5,11		4-3	5	131,03
2-4	4	11,15		4-4	4	20,00
2-5	3	5,37		4-5	3	7,10
2-6	3	4,77		4-6	4	45,23
2-7	3	5,87		4-7	4	32,58
2-8	3	4,83		4-8	5	11,63
2-9	3	5,99		4-9	5	73,47
2-10	4	4,64		4-10	4	29,06

Tablo 12'de görüldüğü gibi matematiksel model küçük problemler için kısa sürelerde en iyi sonuca ulaşmaktadır. Fakat problem boyutu arttıkça çözüm süresi artmaktadır.

4.2. Sezgisel Yöntem

İşlerin gruplanması problemi NP-zor bir problem olduğu için büyük boyutlardaki problemlerde yapılabilir çözümlere ulaşmak amacıyla sezgisel bir yöntem geliştirilmiştir. Geliştirilen yöntemde öncelikle her iş ikilisi için ihtiyaç duydukları ortak makine ucu sayıları hesaplanır. Sonraki adımda en fazla sayıda ortak makine ucuna sahip iş ikilisi seçilir. Toplam makine ucu sayısı makine haznesinin kapasitesini aşmıyorsa bu ikili bir grup oluşturur ve bu işleri yapmak için gereken makine uçları da gruba atanır. Grubu oluşturan işlerle en fazla ortak makine ucuna sahip iş seçilip gruba eklenir. Bu aşamada da makine ucu haznesinin kapasitesinin aşılıp aşılmadığı kontrol edilir. Makine haznesi dolduğunda, yeni makine ucuna ihtiyaç duymadan gruba atayabileceğimiz işler varsa gruba eklenir. Mevcut gruba daha fazla iş ekleyemedğimizde yeni bir grup açılır ve prosedür tekrarlanır.

Sezgisel yöntemde kullanılan parametreler aşağıda açıklanmıştır.

D: herhangi bir gruba atanmamış işler kümesi
U_{ij}: D kümesindeki i ve j işleri arasındaki ortak makine ucu sayısı
$l(i)$: i işi için gereken makine ucu kümesi
c: makine ucu haznesinin kapasitesi
w_g: g grubundaki işler için gereken makine ucu kümesi
A_g: g grubuna atanan işler kümesi

Sezgisel yöntemin aşamaları aşağıda verilmiştir.

0. Tüm i ve j ikilileri için U_{ij} değerlerini bul. g=1, A_g=∅ ∀g, D={1, 2, ..., n} ve w_g=∅ ∀g.

1. Ençok $\{U_{ij}\}= U_{i'j'}$ koşulunu sağlayan i've j' işlerini bul.

1.1. Eğer tüm $U_{i'j'}$ değerleri 0 ise D kümesinde en fazla sayıda makine ucu gerektiren bir p işi bul. p işini D kümesinden çıkar ve A_g kümesine ekle. $w_g=l(p)$. 2. adıma git.

1.2. Eğer $(|l(i')\cup l(j')|) >$ c ise $U_{i'j'}$ değerini -1'e eşitle ve 1. adımı tekrarla.

1.3. Eğer $(|l(i')\cup l(j')|) \leq$ c ise i've j' işlerini g grubuna ata. i' ve j' işlerini D kümesinden çıkar ve A_g kümesine ekle. $w_g=l(i')\cup l(j')$.

1.4. Eğer $|w_g|<c$ ise 2. adıma git. Eğer $|w_g|=c$ ise 4. adıma git.

2. D kümesinden öyle bir k işi bul ki, A_g kümesindeki işlerle ortak makine ucu sayısı en çok olsun.

2.1. D kümesinde A_g kümesindeki işlerle ortak makine ucu olan iş yoksa 3. adıma git.

2.2. Eğer $|w_g \cup l(k)| > c$ ise k işini gözardı ederek 2. adımı tekrarla.

2.3. Eğer $|w_g \cup l(k)| \leq c$ ise k işini g grubuna ata. k işini D kümesinden çıkar ve A_g kümesine ekle. $w_g = w_g \cup l(k)$.

2.4. Eğer $|w_g| < c$ ise 2. adımı tekrarla. Eğer $|w_g| = c$ ise 4. adıma git.

3. Makinedeki boş makine ucu haznesi kapasitesini, $c - |w_g|$, bul. D kümesindeki işlerden gerektirdiği makine ucu sayısı boş makine ucu haznesi kapasitesini geçmeyen işlerden bir q işi bul. q işini D kümesinden çıkar ve A_g kümesine ekle. $w_g = w_g \cup l(q)$. 2. adımı tekrarla. Böyle bir q işi bulamazsan 5. adıma git.

4. $l(h) \subseteq w_g$ koşulunu sağlayan bir h işi bul. h işini g grubuna ata. h işini D kümesinden çıkar ve A_g kümesine ekle. g grubuna atanabilecek başka bir iş kalmayana kadar devam et. 5. adıma git.

5. $D = \varnothing$ ise dur. Diğer durumda U_{ij} değerlerini güncelle, $g = g+1$ ve 1. adıma git.

4.2.1. Deneyler

Geliştirilen sezgisel yöntemin performansının değerlendirilmesi amacıyla değişik boyutlarda problemler çözülmüştür. İlk gruptaki problemler küçük boyuttaki setlerdir ve Tablo 13'deki parametreler kullanılarak oluşturulmuştur. $|l(i)|$ değerleri kesikli birbiçimli dağılım kullanılarak türetilmiştir. $l(i)$ kümesindeki makine uçları rasgele türetilmiştir. Her parametre kombinasyonu için 10 problem türetilmiştir.

Tablo 13. İş gruplama küçük problem parametreleri

Set	***n***	***t***	***|l(i)|***	***c***
1	10	8	U[1,2]	3
2	10	16	U[1,4]	6
3	15	8	U[1,2]	3
4	15	16	U[1,4]	6

Deneylerde kullanılan diğer bir grup problem ise (KONAK vd., 2008)'de kullanılan problem setleridir. Bu problemler daha büyük boyutlarda setlerdir ve parametreleri Tablo 14'de verilmiştir. Her parametre kombinasyonu için 30 problem türetmişlerdir.

Tablo 14. (KONAK vd., 2008) problem parametreleri

Set	***n***	***t***	***c***
L-I	40	20	15
L-II	50	25	20
L-III	60	30	25
VL-I	120	30	20
VL-II	120	60	20
VL-III	150	30	20
VL-IV	150	60	20
VL-V	180	30	20
VL-VI	180	60	20
VL-VII	210	30	20
VL-VIII	210	60	20

Model Visual C++ kullanılarak kodlanmış ve AMD Phenom(tm) II P820, 1.80 GHz, 4 GB RAM bilgisayarla çözülmüştür.

Tablo 15'te küçük problemler için en iyi çözüm ve sezgisel yöntemle bulunan grup sayıları verilmiştir. Çözüm sürelerinin tamamı 1 saniyenin altında olduğu için raporlanmamıştır. 40 problemin 33'ünde (%82,5) sezgisel yöntem en iyi çözüme ulaşmıştır. Geriye kalan 7 problemde (%17,5) en iyi çözümden bir fazla grup oluşturulmuştur. Miyop olarak adlandırılan bir tür algoritma olmasına rağmen küçük problemler için alınan sonuçlar oldukça iyidir.

Tablo 15. Küçük problemler için en iyi çözüm ve sezgisel yöntemle bulunan grup sayıları

Problem seti	En iyi çözüm	Sezgisel yöntem	Problem seti	En iyi çözüm	Sezgisel yöntem
1-1	3	3	3-1	4	4
1-2	3	3	3-2	3	3
1-3	3	3	3-3	3	3
1-4	3	3	3-4	4	4
1-5	3	3	3-5	4	4
1-6	2	3	3-6	3	4
1-7	4	4	3-7	3	3
1-8	3	3	3-8	4	5
1-9	3	3	3-9	4	4
1-10	2	2	3-10	3	3
2-1	3	3	4-1	5	6
2-2	3	3	4-2	4	4
2-3	4	4	4-3	5	5
2-4	4	4	4-4	4	4
2-5	3	3	4-5	3	4
2-6	3	3	4-6	4	4
2-7	3	3	4-7	4	5
2-8	3	3	4-8	5	5
2-9	3	3	4-9	5	5
2-10	4	4	4-10	4	5

Sezgisel algoritmanın büyük problemlerdeki performansını ölçmek amacıyla (KONAK vd., 2008)'in geliştirdiği problemlerde yapılan deneylerin sonuçları Tablo 16'da verilmiştir. L-I, L-II ve L-III setleri için verilen sonuçlarda ilk sütunlar (KONAK vd., 2008) tarafından raporlanan sonuçları göstermektedir. L-I grubundaki problemler için en iyi çözüm veya bulunan en iyi olurlu çözüm ile parantez içinde aralık verilmiştir. L-II ve L-III için geliştirdikleri tabu arama algoritmalarının 10 replikasyon sonucunda bulduğu en iyi sonuç verilmiştir.

Tablo 16'daki sonuçlarda görüldüğü gibi geliştirilen sezgisel algoritma büyük problemlerde daha zayıf bir performans göstermektedir. Bu, hızlı çalışan miyop bir algoritma olmasından dolayı beklenen bir sonuçtur.

Tablo 16. Büyük problemler (KONAK vd., 2008) deney sonuçları

	L-I		L-II		L-III	
Problem seti	**En iyi çözüm**	**Sezgisel yöntem**	**Tabu Search**	**Sezgisel yöntem**	**Tabu Search**	**Sezgisel yöntem**
1	9(%44)	10	9	13	10	14
2	8(%35)	12	16	17	9	10
3	3	4	9	11	8	13
4	9(%44)	10	2	2	5	6
5	6	7	11	12	13	16
6	7(%26)	8	12	15	12	15
7	-	11	8	10	14	16
8	7(%28)	9	8	11	10	14
9	8(%39)	10	10	12	12	15
10	9(%42)	10	13	17	10	13
11	10(%50)	11	8	12	14	18
12	10(%59)	10	10	12	8	10
13	8(%50)	11	4	5	12	20
14	-	12	17	19	13	16
15	12(%58)	13	9	15	11	13
16	4	5	7	8	9	13
17	5	6	13	14	16	18
18	11(%58)	12	11	14	4	5
19	6	8	14	17	15	16
20	8(%37)	9	10	11	8	12
21	3	3	8	10	12	16
22	9(%35)	9	4	4	12	17
23	-	14	9	12	7	9
24	6	7	10	13	3	4
25	9(%53)	12	13	16	17	18
26	9(%44)	11	7	8	5	7
27	10(%50)	12	14	16	8	11
28	-	15	5	6	11	13
29	9(%44)	11	4	5	19	23
30	8(%37)	8	11	12	8	10

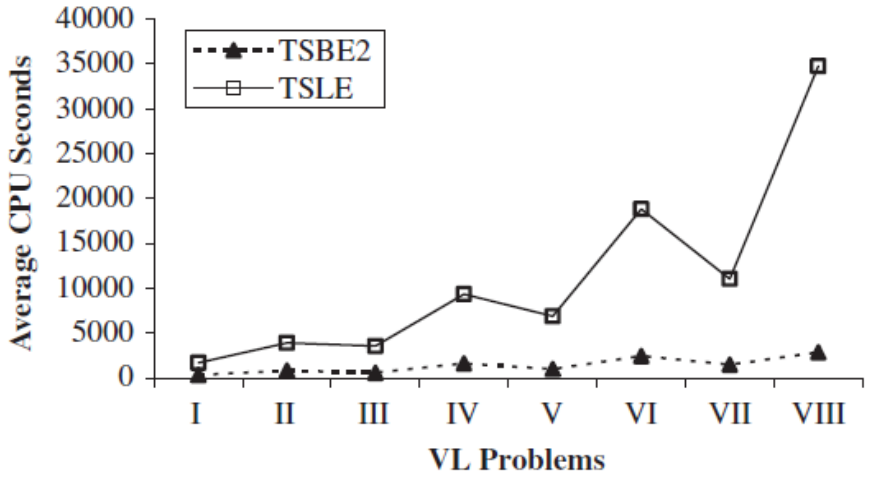
(KONAK vd., 2008) tarafından geliştirilen en büyük problem grubu (VL-I)-(VL-VIII) için de deneyler yapılmıştır. (KONAK vd., 2008)'de bu problemler için iki tabu arama algoritması (TSBE2 ve TSLE) tarafından bulunan ortalama makine ucu değişimi sayısı verilmiştir. Tablo 17'de bu değerler ile geliştirilen sezgisel algoritmanın bulduğu ortalama makine ucu değişimi sayıları verilmiştir. Bu problemler için en iyi çözüme ulaşılamamıştır.

Tablo 17. Çok büyük problemler (KONAK vd., 2008) deney sonuçları

Problem seti	**TSBE2**	**TSLE**	**Sezgisel yöntem**
VL-I	32,4	28,3	35,07
VL-II	61,8	55,8	52,53
VL-III	37,3	34,6	41,17
VL-IV	80,9	71,4	64,9
VL-V	43,3	40,6	49,86
VL-VI	99,9	85,7	77,8
VL-VII	53,5	46,4	56,33
VL-VIII	118,6	100,2	89,9

Çözüm süresi çok büyük problemlerde bile 1 saniyenin altında kalmaktadır. Bu sebeple tabloda verilmemiştir. Ortalama grup sayıları 8 setin 4'ünde her iki tabu arama algoritmasından daha düşük, diğerlerinde de ikisinden fazladır. (KONAK vd., 2008)'den alınan Şekil 4'de geliştirdikleri tabu arama algoritmalarının çok büyük problemler için ortalama çözüm süreleri verilmiştir. Sezgisel algoritmanın çok daha kısa sürede oldukça iyi sonuçlar aldığı görünmektedir.

Sezgisel algoritmanın çok büyük problemlerde, büyük problemlere göre daha iyi performans gösterdiği görülmektedir. Bunun bir sebebi büyük problemlerde tabu arama deneylerinde bulunan en iyi sonuçla, çok büyük problemlerde ise ortalama sonuçla karşılaştırma yapılmasıdır. Diğer bir sebep ise tabu arama algoritmalarının performanslarının problem boyutu büyüdükçe daha fazla düşmesi olabilir.



Şekil 4. (KONAK vd., 2008)'in tabu arama algoritmalarının çok büyük problemler için ortalama çözüm süreleri

4.3. Arama Algoritması

Sezgisel yöntem ile alınan sonuçların iyileştirilmesi amacıyla bir arama algoritması geliştirilmiştir. Algoritma sezgisel yöntemin bulduğu sonucu başlangıç çözümü olarak alıp komşuluk yapısı içindeki çözümleri kontrol ederek daha iyi bir çözüme ulaşmaya çalışmaktadır. Algoritmada bir çözümün komşusu işlerden birini ait olduğu gruptan alıp gerektirdiği makine uçlarıyla birlikte başka bir gruba yerleştirmekle bulunuyor. Bu değişim sadece değiştirilen işin yeni grubunda makine ucu haznesi kapasitesi aşılmazsa gerçekleştiriliyor, diğer bir deyişle sadece olurlu sonuçlara izin veriliyor. Başlangıç çözümünün komşuları arasında daha az gruba sahip bir çözüm bulursak bu çözümü yeni çözüm kabul edip onun komşuları arasında daha iyi bir çözüm arayışına başlıyoruz. Başlangıç çözümünün komşuları arasında daha az gruba sahip bir çözüm yoksa komşularının komşularını kontrol ediyoruz. Bu şekilde bir ağaç yapısı elde ediyoruz. Ağaç boyutunu sınırlandırmak amacıyla en fazla 3 seviye gidip daha iyi bir çözüm bulamazsak elde ettiğimiz komşu çözümlerden bir tanesini yeni çözüm kabul ederek yeni bir ağaç oluşturuyoruz. Sezgisel yöntemin sonucunda gruplara atanan ortalama iş sayısına bakarak en fazla analiz edilecek seviye sayısı 3 olarak belirlenmiştir. Arama algoritmasının çalışma süresine veya analiz edilen yeni çözüm sayısına üst sınır koyarak bu sınıra ulaşıldığında algoritmayı sonlandırıyoruz. Şekil 5'de arama algoritmasının ağaç yapısı verilmiştir.

Arama algoritmasının adımları aşağıda verilmiştir.

0. Bölüm 4.2'de verilen sezgisel yöntemi kullanarak bir olurlu çözüm elde et. Bu çözüme S diyelim. h=1.

1. i=h için i işini ait olduğu gruptan çıkarıp gerektirdiği makine uçları ile birlikte başka bir gruba yerleştir.

1.1 Eğer $i \in A_g$ ise $A_g \rightarrow A_g \backslash \{i\}$, $w_g \rightarrow w_g \backslash \{k | k \in l(i), k \notin l(q)\ \forall q \in A_g\}$.

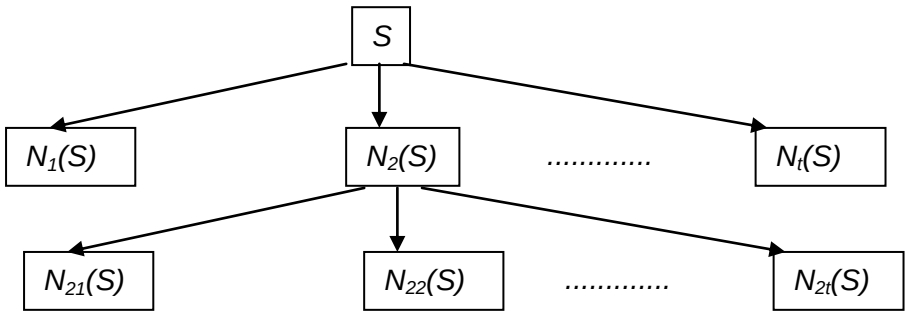
1.2 $b \neq g$ için, $A_b \rightarrow A_b \cup \{i\}$, $w_b \rightarrow w_g \cup l(i)$. Eğer $|w_b| \leq c$ ise bu çözümü $N_1(S)$ olarak kaydet.

1.3 $N_1(S)$ S'ten daha az grup içeriyorsa S=$N_1(S)$ ve 1. Adıma git. $N_1(S)$ ve S aynı sayıda grup içeriyorsa başka bir b grubu için 1.1'i uygula. Grup sayısı azalmazsa tüm gruplar için bu adımı tekrarla ve bulduğun olurlu çözümleri $N_2(S)$, $N_3(S)$.... olarak kaydet.

1.4 h=n ise 2. Adıma git. h<n ise h=h+1 ve 1.1'e git.

2. S'in komşu çözümlerinde grup sayısı azalmadıysa S'in komşularının komşularını bul ve 1. Adımı tekrarla. Aşağıdaki gibi bir ağaç yapısı elde et. Grup sayısının daha az olduğu bir çözüm bulduğunda o çözümü S olarak al ve 1. Adıma git. Ağacın ikinci seviyesinde hiç bir çözümde grup sayısı azalmazsa 3. Adıma git.

3. Ağacın üçüncü seviyesini de oluştur. $N_{11}(S)$, $N_{12}(S)$,.... çözümlerinin komşularını araştır. Grup sayısının azaldığı bir çözüm bulduğunda bu çözümü S olarak al ve 1. Adıma git. Üçüncü seviyede de grup sayısı azalmazsa şu ana kadar bulduğun çözümlerden birisini rasgele seç, bu çözümü S olarak al ve 1. Adıma git.



Şekil 5. Arama algoritması ağaç yapısı

4.3.1. Deneyler

Arama algoritmasının performansını değerlendirmek amacıyla Bölüm 4.2.1'de verilen problemler üzerinde deneyler yapılmıştır. Küçük problemlerde sezgisel yöntemin bulduğu sonuçlar arama algoritması tarafından iyileştirilememiştir. Bu problem grubunda sezgisel yöntem oldukça iyi performans göstermiş ve büyük oranda en iyi sonuca ulaşmıştır. Tablo 18'de büyük problemlerde (KONAK vd., 2008)'in sonuçları ve arama algoritmasının verdiği sonuçlar raporlanmıştır. Çözüm süreleri 1 saniyenin altında olduğu için tabloda verilmemiştir. Tablo 19'da ise çok büyük problemlerde (KONAK vd., 2008)'in ve arama algoritmasının ortalama sonuçları ve arama algoritmasının ortalama çözüm süreleri verilmiştir.

Tablo 18. Büyük problemler (KONAK vd., 2008) arama algoritması deney sonuçları

	L-I		L-II		L-III	
Problem seti	**En iyi çözüm**	**Arama algoritması**	**Tabu Search**	**Arama algoritması**	**Tabu Search**	**Arama algoritması**
1	9(%44)	10	9	11	10	14
2	8(%35)	11	16	17	9	10
3	3	3	9	11	8	10
4	9(%44)	10	2	2	5	6
5	6	7	11	12	13	15
6	7(%26)	8	12	14	12	15
7	-	11	8	10	14	16
8	7(%28)	9	8	10	10	12
9	8(%39)	10	10	12	12	14
10	9(%42)	9	13	15	10	12
11	10(%50)	11	8	10	14	17
12	10(%59)	10	10	11	8	10
13	8(%50)	10	4	5	12	19
14	-	11	17	18	13	14
15	12(%58)	13	9	13	11	13
16	4	5	7	7	9	12
17	5	6	13	13	16	17
18	11(%58)	10	11	12	4	5
19	6	7	14	17	15	16
20	8(%37)	8	10	11	8	11
21	3	3	8	10	12	14
22	9(%35)	9	4	4	12	15
23	-	14	9	10	7	8
24	6	7	10	12	3	4
25	9(%53)	11	13	16	17	17
26	9(%44)	11	7	7	5	7
27	10(%50)	11	14	15	8	10
28	-	14	5	6	11	12
29	9(%44)	10	4	5	19	23
30	8(%37)	8	11	12	8	10

Tablo 19. Çok büyük problemler (KONAK vd., 2008) arama algoritması deney sonuçları

Problem seti	TSBE2	TSLE	Arama algoritması	Çözüm süresi (sn)
VL-I	32,4	28,3	34,6	2,70
VL-II	61,8	55,8	52,5	2,47
VL-III	37,3	34,6	40,6	5,35
VL-IV	80,9	71,4	64,7	5,13
VL-V	43,3	40,6	-	-
VL-VI	99,9	85,7	-	-
VL-VII	53,5	46,4	-	-
VL-VIII	118,6	100,2	-	-

Arama algoritması büyük ve çok büyük problemlerde sezgisel çözümün bulduğu sonuç üzerinden az da olsa gelişme kaydetmiştir. Çok büyük problemlerde ilk dört sette çözüm sürelerinin oldukça kısa olduğu görülmektedir. Son dört sette ağaç boyutu çok büyüdüğü için çözüm alınamamaktadır. (KONAK vd., 2008) tarafından geliştirilen tabu arama algoritması ile karşılaştırışdığında bazı setlerde daha iyi, bazılarında daha kötü sonuç verdiği görülmektedir. Ancak çözüm süresi açısından arama algoritmasının çok daha hızlı olduğunu söyleyebiliriz.

4.4. Kısıt Programlama Yaklaşımı

Kısıt programlama yaklaşımı yapay zeka konusunun alt dalıdır ve kombinatoryel problemin özelliklerini kısıt tatmin probleminin özelliklerine çevirir. Kısıt Programlama yaklaşımı özellikle çizelgeleme problemlerinin modellenmesinde ve çözülmesinde tercih edilen bir yöntem olmuştur. Bu yaklaşımın tercih edilmesinin en önemli sebebi etkin kısıt-tabanlı çizelgeleme algoritmalarının çok kısa sürede en iyi çözüme ulaşıp, o çözümün en iyi olduğunu kanıtlamasıdır. Çizelgeleme problemleri dışında da kısıt programlama algoritmalarından ve modelleme tekniklerinden yararlanılır. Fakat diğer problem çeşitlerindeki karşılaşılan sorun, en iyi çözümü çok kısa bir sürede bulmasına karşın o çözümün en iyi olduğunu kanıtlayamamasıdır. Bu projede, kısıt programlama yaklaşımı için

IBM ILOG CP Optimizer 2.3 yazılımı kullanılmıştır. Bu yazılımda, birçok kısıt programlama algoritmaları ve modelleme için gerekli kaynak kısıtları tanımlanmış şekilde bulunmaktadır.

Literatürde, çizelgeleme problemleri ile ilgili birçok kısıt programlama çalışmaları bulunmasına rağmen diğer problem çeşitleri ile ilgili çok çalışma bulunmamaktadır. İlgili çalışmalar genellikle probleme özgü kısıt programlama algoritmaları geliştirmek şeklindedir.

4.4.1. Kısıt Programlama Modelleri

Bu bölümde incelenen problem için iki farklı kısıt programlama modeli geliştirilmiştir. Birinci kısıt programlama modelinde, matematiksel modelde olan set sayısını azaltmak hedeflenmiştir. Bu hedef, 0-1 karar değişkeninin tanım kümesini, açılabilecek grup seti şeklinde tanımlanarak gerçekleştirilmiştir. Bu modelde, kısıt programlama tekniklerinden "*channeling*" yöntemi kullanılmıştır. "*Channeling*" yöntemi, bir karar değişkeninin, diğer bir karar değişkeninde set (indis) olarak kullanılması olarak tanımlanmaktadır. Herhangi bir gruba atanan işler ve o işlerin işlenebilmesi için gerekli makine ucu seti arasındaki kısıtlar ise mantıksal kısıtlar yardımıyla oluşturulmuştur. Bu model JG-CP1 olarak isimlendirilmiş ve Şekil 6'da verilmiştir. İncelenen problemde, işlerin hangi numaralı gruba atandıklarının çözüm açısından bir farkı yoktur. Problemin amacı açılan grup sayısını enazlamaktır. Herhangi bir işin hangi numaralı gruba atanacağı kararı, problemde simetriye sebep olmaktadır ve çözüm kalitesini kötü yönde etkilemesi beklenmektedir. Kısıt programlamada iki tip simetri vardır; çözüm simetrisi ve problem simetrisi. Çözüm simetrisi, karar değişkeni-değer ikililerinin yer değiştirmesi ile çözüm setini oluşturmasıdır. Problem simetrisi ise yine karar değişkeni-değer ikililerinin yer değiştirmesi ile kısıtların hala sağlanmasıdır. Bu iki tip simetrinin özel durumlarına değişken ve değer simetrisi denir. Değişken simetrisi, değişken kümesindeki değişkenlerin serbest bir şekilde yerlerinin değiştirilmesine; Değer simetrisi ise atanabilecek değer kümesindeki değerlerin serbest bir şekilde yerlerinin değiştirilmesine denir. Problemlerdeki simetrileri kırmak için farklı yöntemler geliştirilmiştir; tekrardan modelleme, statik simetri kırma ve dinamik simetri kırma bu yöntemlerden bazılarıdır. Bu bölümde, ikinci kısıt programlama modelinde, statik simetri kırma kısıtı olan sözlüksel sıralama kısıtı (*lexicographic*) ile sonuç alınmaya çalışılmıştır. Model JG-CP2 olarak isimlendirilmiş ve Şekil 7'de verilmiştir.

Amaç fonksiyonu:

(7) Min. $\sum_{g \in G} Z_g$

Karar değişkenleri:

(8) $Z_g \in \{0,1\} \quad \forall g \in 1,\ldots,G$

(9) $Y_{kg} \in \{0,1\}, \quad \forall k \in 1,\ldots,K, \forall g \in 1,\ldots,G$

(10) $X_j \in \{1,\ldots,G\}, \quad \forall j \in 1,\ldots,N$

Kısıtlar:

(11) $Z_{X_i} = 1 \quad \forall i \in 1,\ldots,N$

(12) $(X_i = g) \rightarrow (Y_{kg} = 1) \quad \forall g \in 1,\ldots,G, \forall i \in 1,\ldots,N, \forall k \in 1,\ldots,K \mid k \in l($

(13) $\sum_{k \in K} Y_{kg} \leq c \quad \forall g \in 1,\ldots,G$

Şekil 6. JG-CP1 kısıt programlama modeli

Amaç fonksiyonu:

(1) Min. $\sum_{g \in G} Z_g$

Karar değişkenleri:

(2) $Z_g \in \{0,1\} \quad \forall g \in 1,\ldots,G$

(3) $Y_{kg} \in \{0,1\}, \quad \forall k \in 1,\ldots,K, \forall g \in 1,\ldots,G$

(4) $X_{jg} \in \{0,1\}, \quad \forall j \in 1,\ldots,N, \forall g \in 1,\ldots,G$

Kısıtlar:

(5) $Z_g \leq 1, \quad \forall g \in 1,\ldots,G$

(6) $Z_g \geq X_{ig} \quad \forall g \in 1,\ldots,G, \forall i \in 1,\ldots,N$

(7) $\sum_{g \in G} X_{ig} = 1 \quad \forall i \in 1,\ldots,N$

(8) $X_{ig} \leq Y_{kg} \quad \forall g \in 1,\ldots,G, \forall i \in 1,\ldots,N, \forall k \in 1,\ldots,K \mid k \in l(i)$

(9) $\sum_{k \in K} Y_{kg} \leq c, \quad \forall g \in 1,\ldots,G$

(10) $lex\left(\sum_{g=1}^{G-1} X_{ig}, \sum_{g=1}^{G-1} X_{ig+1}\right) \quad \forall i \in 1,\ldots,N$

Şekil 7. JG-CP2 kısıt programlama modeli

4.4.2. Deneyler

Bu bölümde, geliştirilen iki kısıt programlama modelinin karşılaştırılması yapılacaktır. Deney setinde, iş sayısı; 10 ve 15, makine ucu çeşidi sayısı 8 ve 16 olarak belirlenmiştir. $l(i)$ seti 1 ve 4 aralığında, r_k seti 1 ve 2, makine ucu hazne kapasite seti ise 3 ve 6 tekdüze dağılımdan oluşturulmuştur. Her bir deney seti için 10 tane örnek oluşturulmuştur. Örnekler, IBM OPL 6.1.1 yazılımı ile AMD II P820, 1.8 GHz ve 3.74 GB RAM özellikli bilgisayarda çalıştırılmıştır. Her bir örnek için zaman limitinin üst sınırı bir saat olarak belirlenmiştir. Arama stratejisi olarak tekrardan başlama seçeneği kullanılmıştır. Tablo 20'de CP1 ve CP2 karşılaştırması verilmiştir.

Tablo 20. JG-CP1 ve JG-CP2 modellerinin performans karşılaştırması

	CP1			CP2		
<iş, makine ucu, makine hazne kapasitesi >	Optimal Çözüm	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı	Optimal Çözüm	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı
<10, 8, 3>	10	10	3,49	10	10	3,03
<10, 16, 6>	10	10	5,47	10	10	4,38
<15, 8, 3>	10	10	15,28	10	10	18,62
<15, 16, 6>	10	10	1063	10	10	1309

Tablo 20'ye göre, CP1 ve CP2 modelleri tüm setlerde en iyi çözüme ulaşmaktadır. İki model arasında hangisinin daha iyi olduğu konusunda belirgin bir yargıya varılamamaktadır. Modeller birbirlerini tamamlayan niteliktedir. İki model de incelendiği zaman görülmektedir ki set(indis) sayıları azaltılmaya çalışıldığı zaman simetri kırma kısıtı kullanılamamakta, simetri kırma kısıtı kullanıldığında ise set sayıları azaltılamamaktadır. İş ve makine ucu sayıları arttığında, en iyi çözüm çok kısa bir sürede bulunmasına rağmen en iyi çözüm kanıtlaması yaklaşık olarak çözüm zamanın 95%'ini almaktadır. Problem boyutu büyüdükçe çözüm süresi artmaktadır.

Daha büyük problem setlerinde, iki kısıt programlama yaklaşımının da çok kısa sürede en iyi çözüme ulaşması fakat çözümün en iyi olduğunun kanıtlanmasının çok uzun sürmesi beklenmektedir. Bunun nedeni de, çizelgeleme problemleri haricinde kısıt programlama yaklaşımı dahilinde etkin algoritmaların geliştirilmemiş olmasıdır. Bu sorun, probleme özgü kısıt programlama çözüm yaklaşımları veya kısıt programlama yaklaşımlarının çok kısa bir sürede iyi bir çözüm elde ettiği düşünüldüğünde melez çözüm yaklaşımları (Çözümün en iyi olduğu da başka yöntemlerle sağlanması şeklinde) geliştirilerek giderilebilir.

5. İŞLERİN VE MAKİNE UÇLARININ ÇİZELGELENMESİ VE MAKİNE UCU DEĞİŞİMİ PROBLEMİ

5.1. Problem Tanımı

Bu problemde, işlerin ve makine uçlarının makinelere atanması ve çizelgelenmesi problemine ek olarak makine ucu değişimi de göz önüne alınmıştır. Problemde m paralel makine ve bu makinelerde işlenmesi gereken n iş bulunmaktadır. İşler arasında öncelik ilişkisi bulunmamaktadır. Her iş bir makineye atanmalıdır ve işlerin başladıktan sonra bölünmesine izin verilmemektedir. Her makine her işi yapabilmektedir, fakat makinelerin yaşı ve kapasitelerine bağlı olarak işlem süreleri değişebilmektedir. p_{ij}, i işinin j makinesindeki işlem süresidir. i işinin yapılabilmesi için $l(i)$ kümesindeki makine uçlarının makineye yüklenmiş olması gerekmektedir. Sistemde t çeşit makine ucu vardır ve k çeşit makine ucundan r_k kopya bulunmaktadır. Her makinenin haznesi c makine ucu alabilmektedir.

Ekonomik kısıtlardan dolayı elde bulunan makine ucu kopyası sayısı makine sayısından az olabilmektedir. Ayrıca makinelerin hazne kapasitesi tüm makine ucu çeşitlerini alamamaktadır. Bu kısıtlar makine ucu değişimini gerekli kılmaktadır. Bir işin gerektirdiği makine uçları makineye yüklenmemişse başka bir makineden veya takımhaneden alınmalıdır. Eğer başka bir makinede veya takımhanede gereken makine ucundan yoksa, bu makine ucunu kullanan iş bitene kadar değişim ertelenmelidir. Ayrıca yeni yüklenecek makine ucu için makinenin haznesinde yer yoksa gereksiz makine uçlarının çıkarılması gerekmektedir. Makine ucu değişiminin önemli bir zaman gerektirdiği ve değiştirilen makine çeşidi ve sayısından bağımsız olduğu varsayılmaktadır.

Makine ucu kullanımıyla ilgili diğer varsayımlar aşağıda verilmiştir:

- Makine uçları makineler arasında paylaşılamaz; işlem sırasında makine uçları değiştirilemez.
- Makine hazneleri başlangıçta boştur.
- Her makine ucu haznede bir birim yer kaplar.
- Işlem sırasında makine uçları kırılmaz.
- Bir iş için gereken makine ucu sayısı makine haznesinin kapasitesini aşmaz.

Problem, işlerin ve gerekli makine uçlarının son işin tamamlanma zamanını en azlayacak şekilde çizelgelenmesidir.

Çizelgeleme ve makine ucu değişimi üzerinde çalışmamızın amacı, teoride ve uygulamada önemli bir yere sahip çizelgeleme, makine ucu atama ve makine ucu değişimi problemlerini birlikte ele almak, böylece sistemin bütün olarak iyileştirilmesini sağlamaktır.

5.2. Matematiksel Model

Bu bölümde çizelgeleme ve makine ucu değişimi problemi için geliştirilen matematiksel model açıklanmıştır. Modelde kullanılan parametreler aşağıdaki gibidir:

$l(i)$: i işi için gereken makine uçları kümesi

p_{ij}: i işinin j makinesindeki işlem süresi

ts: makine ucu değişimi süresi

c: makine ucu haznesinin kapasitesi

Karar değişkenleri:

X_{ig}: 1, eğer i işi g grubuna atandıysa; 0, diğer durumda

$W_{h_k g}$: 1, eğer k makine ucunun h kopyası g grubundaki işleri işlemek için kullanılıyorsa; 0, diğer durumda

Y_{gbj}: 1, eğer g grubu b grubundan önce j makinesinde işleniyorsa; 0, diğer durumda

Z_{gj}: 1, eğer g grubu j makinesinde işleniyorsa; 0, diğer durumda

V_{gb}: 1, eğer g grubu ve b grubu aynı makine ucunu kullanıyorsa ve g grubu b grubundan önce işleniyorsa; 0, diğer durumda

C_g: g grubunun tamamlanma zamanı

S_{gj}: g grubunun j makinesinde başlama zamanı

pt_{gj}: g grubunun j makinesindeki toplam işlerm süresi

$Cmax$: son işin tamamlanma zamanı (makespan)

Matematiksel modelimizin amacı son işin tamamlanma zamanını enazlamaktır.

Min $Cmax$

Modelin kısıtları da aşağıda açıklamalarıyla verilmiştir.

- Her iş bir gruba atanmalıdır.

$$\sum_{g \in G} X_{ig} = 1 \quad \forall i$$

- Makine ucu haznesinin kapasitesi aşılamaz.

$$\sum_{h_k} \sum_{k \in K} W_{h_k g} \leq c \quad \forall g$$

- Bir gruba atanan işlerin gerektirdiği makine uçları makineye yüklenmelidir.

$$X_{ig} \leq \sum_{h_k} W_{h_k g} \quad \forall i, g, k \in l(i)$$

- Son işin tamamlanma zamanı tüm grupların tamamlanma zamanının en büyüğüdür.

$$\text{Cmax} \geq C_g \quad \forall g$$

- Bir grubun bir makinedeki toplam işlem zamanı, o gruba atanan işlerin o makinedeki işlem sürelerinin toplamına eşittir.

$$pt_{gj} \geq \sum_i p_{ij} \left(X_{ig} + Z_{gj} - 1 \right) \quad \forall g, j$$

- Bir grubun tamamlanma zamanı, makine ucu değişimi zamanı, başlama zamanı ve işlem sürelerinin toplamıdır.

$$C_g = ts + \sum_j \left(S_{gj} + pt_{gj} \right) \quad \forall g$$

- Aynı makineye atanan grupların işlem zamanı çakışamaz.

$$S_{gj} \geq C_b - M\left(Y_{gbj}\right) - M\left(1 - Z_{bj}\right) - M\left(1 - Z_{gj}\right) \quad \forall g \neq b, j$$

$$Z_{bj} + Z_{gj} \geq 2\left(Y_{bgj} + Y_{gbj}\right) \quad \forall g \neq b, j$$

$$Z_{bj} + Z_{gj} \leq Y_{bgj} + Y_{gbj} + 1 \quad \forall g \neq b, j$$

- Bir grubun bir makinede başlama zamanı, eğer o makineye atandıysa pozitif bir değer alabilir.

$$S_{gj} \leq M\left(Z_{gj}\right) \quad \forall g, j$$

- Her grup en fazla bir makineye atanmalıdır. (Açılacak grup sayısı başta bilinmediği için hiç bir işin yer almadığı gruplar makinelere atanmayacaktır.)

$$\sum_j Z_{gj} \leq 1 \quad \forall g$$

- Bir gruba iş atandıysa, o grup açılmalı ve bir makineye atanmalıdır.

$$\sum_j Z_{gj} \geq X_{ig} \quad \forall i, g$$

- Bir gruba hiç bir iş atanmadıysa, o grup hiç bir makineye atanmamalıdır.

$$Z_{gj} \le \sum_i X_{ig} \quad \forall g, j$$

- Aynı makine ucu kopyasını kullanan grupların işlem zamanları çakışamaz.

$$\sum_j S_{bj} \ge C_g - M\left(2-\left(W_{h_k g} + W_{h_k b}\right)\right) - M\left(1-V_{gb}\right) \quad \forall g, b$$

$$V_{gb} + V_{bg} \le 1 \quad \forall g, b$$

$$W_{h_k g} + W_{h_k b} \le 2\left(V_{bg} + V_{gb}\right) \quad \forall g, b$$

- Değişkenlerin alabileceği değerler:

$$X_{ig} \in \{0,1\} \quad \forall i, g$$

$$W_{h_k g} \in \{0,1\} \quad \forall h, k, g$$

$$Y_{gbj} \in \{0,1\} \quad \forall g, b, j$$

$$Z_{gj} \in \{0,1\} \quad \forall g, j$$

$$V_{gb} \in \{0,1\} \quad \forall g, b$$

$$C_g \ge 0 \quad \forall g$$

$$S_{gj} \ge 0 \quad \forall g, j$$

$$pt_{gj} \ge 0 \quad \forall g, j$$

Yukarıda verilen matematiksel model en küçük boyuttaki problemlerde bile bir saat içinde sonucu bulamamıştır. Problemin zorluğu göz önünde bulundurulduğunda bu beklenen bir sonuçtur.

5.2.1. Deneyler

Bu bölümde geliştirilen matematiksel modelin performans ölçümü yapılacaktır. Deney setinde iş sayısı; 8, 10 ve 15, makine sayısı; 2 ve 3, makine ucu çeşidi sayısı 5 ve 8 olarak belirlenmiştir. $l(i)$ setindeki makine ucu sayısı 1 ve 4 aralığında, r_k 1 ve 2, makine ucu hazne kapasitesi ise 4 ve 6 tekdüze dağılımdan oluşturulmuştur. Makine ucu değişim süresi de 20 olarak belirlenmiştir. Her bir deney seti için 10 tane örnek oluşturulmuştur. Örnekler, IBM OPL 6.1.1 yazılımı ile AMD II P820, 1.8 GHz ve 3.74 GB RAM özellikli bilgisayarda

çalıştırılmıştır. Her bir örnek için zaman limitinin üst sınırı bir saat olarak belirlenmiştir. Tablo 21'de geliştirilen matematiksel modelin performans değerleri verilmiştir.

Tablo 21. Çoklu makinelerde matematiksel model performansı

<iş, makine sayısı, makine ucu, makine hazne kapasitesi >	Optimal Çözüm	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı	Ort. En iyi Çözümden Sapma Yüzdesi
<8, 2, 5, 4>	1	10/10	1380	77,97
<8, 2, 8, 6>	0	10/10	-	85,21
<10, 2, 5, 4>	0	10/10	-	90,17
<10, 2, 8, 6>	0	10/10	-	85,47
<10, 3, 5, 4>	0	10/10	-	90,35
<10, 3, 8, 6>	0	10/10	-	86,20
<15, 2, 8, 6>	0	10/10	-	93,32

Tablo 21'e göre, matematiksel model sadece bir setin tek örneğinde optimal çözüme ulaşabilmektedir. Üst sınır bulduğu çözümlerde ise en iyi çözümden sapma yüzdesi ise oldukça yüksektir (%86,95). Matematiksel modelin çoklu makine ortamında sadece bir tane örnekte en iyi çözüme ulaşmasından dolayı tek makineli problem setleri yaratılmıştır. Bu setlerde, iş sayısı; 5, 8, 10 ve 15, makine ucu çeşidi sayısı; 5 ve 8, makine ucu çeşidi sayısı 5 ve 8 olarak belirlenmiştir. $l(i)$ setindeki makine ucu sayısı 1 ve 4 aralığında, r_k 1, makine ucu hazne kapasitesi ise 4 ve 6 tekdüze dağılımdan oluşturulmuştur. Makine ucu değişim süresi de 20 olarak belirlenmiştir. Tablo 22'de matematiksel modelin tek makineli ortamdaki performansını görebilirsiniz.

Tablo 22. Tekli makinede matematiksel model performansı

<iş, makine sayısı, makine ucu, makine hazne kapasitesi >	Optimal Çözüm	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı	Ort. En iyi Çözümden Sapma Yüzdesi
<5,5,4>	10	10/10	7,49	
<5,8,6>	10	10/10	7,04	
<8, 5, 4>	6	10/10	1305,86	41,67
<8, 8, 6>	5	10/10	2234,49	51,99
<10, 5, 4>	0	10/10		74,62
<10, 8, 6>	1	10/10	1968,23	69,36
<15, 8, 6>	0	10/10		90,39

Tek makineli problemde, en küçük iş sayılı setlerde geliştirilen matematiksel model her örnek için optimal çözümü bulmuştur. Fakat, iş sayısı arttıkça Tablo 22'den de görülebileceği gibi bulunan optimal çözüm sayısı azalmakta ve çözüm zamanı artmaktadır. Tek makineli problem setlerinde dahi, iş sayısı 8 ve üzeri sayıya çıktığı zaman, matematiksel modelin, her set için, en iyi çözüme ulaşma yüzdesi düşmektedir.

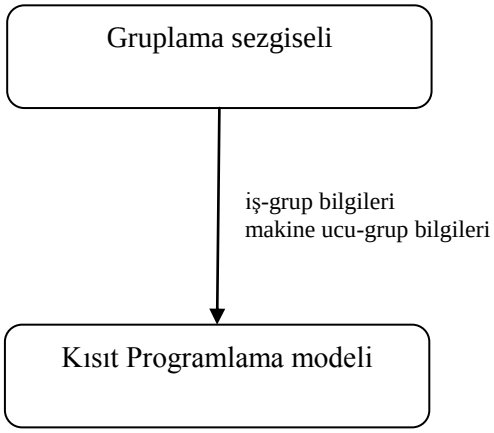
5.3. Ayrıştırma Sezgiseli

Çizelgeleme ve makine ucu atama probleminin zorluğu daha önceki bölümlerde belirtilmiştir. Problem, makine ucu değişimi ile birlikte ele alındığı zaman daha da zorlaşmıştır. 5.1'de de bahsedildiği gibi geliştirilen matematiksel model en küçük boyuttaki problemleri dahi çözememektedir. Fakat, çizelgeleme ve makine ucu atama problemi için önceden geliştirilen etkin çözüm yöntemleri bulunmaktadır (kısıt programlama ve tabu arama algoritması). Ayrıca, işlerin gereken makine ucu setlerine ve makine hazne kapasitesine göre makine ucu değişimlerinin en aza indirgenmesi amaçlanarak gruplanması için sezgisel bir yöntem geliştirilmiştir (gruplama sezgiseli). Problemin zorluğundan dolayı makine ucu değişimi ve makine ucu atama-çizelgeleme problemlerinin ayrıştırılması bir çözüm yöntemi olarak düşünülmüştür. Gruplama sezgiseli uygulanarak, işlerin ve makine uçlarının, makine ucu değişimlerini en aza indirgeyecek şekilde gruplanması sağlanmıştır. Oluşan her bir grup, bir iş gibi düşünüldüğünde ve makine ucu değişim zamanları her bir grubun işlenme sürelerine eklendiğinde, problem makine ucu atama ve çizelgeleme problemine dönüşmektedir. Bu problem için de etkin kısıt programlama ve tabu arama algoritması mevcuttur. Bu amaçla iki

tane farklı çözüm yöntemleri içeren ayrıştırma sezgiselleri geliştirilmiştir. Ayrıştırma sezgiseli 1'de makine ucu atama ve çizelgeleme problemi için kısıt programlama önerilirken ayrıştırma sezgiseli 2'de tabu arama algoritması önerilmiştir. 5.3.1 ve 5.3.2 'de önerilen ayrıştırma sezgiselleri detaylı bir şekilde açıklanmıştır.

5.3.1. Gruplama Sezgiseli ve Kısıt Programlama Modeli (Ayrıştırma sezgiseli 1)

Problemde, makine ucu değişimleri göz ardı edildiği zaman, problem çizelgeleme ve makine ucu atama problemine dönüşmektedir. Çizelgeleme ve makine ucu atama problemi için yazılan kısıt programlama modellerinin çok iyi sonuçlar verdiği raporlanmıştır. Makine ucu değişim durumlarının, kısıt programlama modeline global kısıtlar tanımlanarak eklenmesi oldukça zordur. Bu nedenle ilk çözüm yöntemi olarak ayrıştırma sezgiseli düşünülmüştür. İşlerin, işlenebilmeleri için gerekli makine ucu setlerine göre gruplama sezgiseli geliştirilmiştir. Geliştirilen sezgiselin sonucunda, hangi işlerin hangi gruplara hangi makine ucu kopyaları ile birlikte atandığı bilgisi yer almaktadır. Her grup işin işlenmesinin ardından makine ucu değişim süresi geçtikten sonra sıradaki grubun işlenmesine başlanmaktadır. Her problem için sezgiselin sonucundaki bilgiler kullanılarak, kısıt programlama modeli için uygun veri setleri oluşturulmuştur. Geliştirilen sezgiselin sonucu, çizelgeleme ve makine ucu için geliştirilen kısıt programlama modelin için girdi olarak kullanılmıştır. Bu yaklaşımda, kısıt programlama yöntemindeki, çizelgeleme problemlerine özgü geliştirilen kısıt propagasyonu algoritmalarının gücünden yararlanmak hedeflenmiştir. Ayrıştırma sezgiseli 1'in akışı aşağıda verilmiştir.



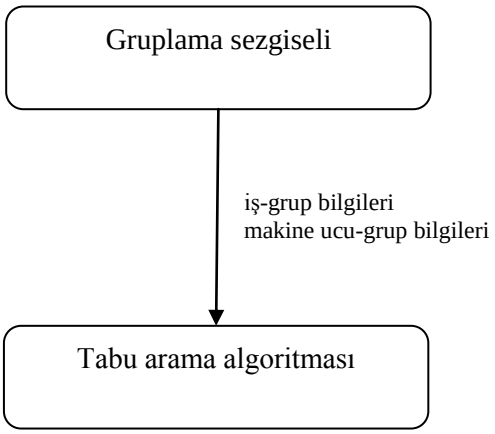
Şekil 8. Ayrıştırma sezgiseli 1'in akışı

Şekil 8'den de görülebileceği gibi gruplama sezgiselinin sonucunda elde edilen iş-grup atama bilgileri ve makine ucu-grup atama bilgileri, kısıt programlama modeli için girdi olarak kullanılmaktadır. Kısıt programlama modelinde her bir grup, bir iş olarak düşünülmüştür. Makine ucu değişimi durumlarının, ayrıştırma sezgiseli kullanılarak, kısıt programlama modelinde dikkate alınmasına gerek kalmamıştır. Bunun nedeni de, gruplama sezgiseli ile oluşan grupların kendi içlerinde makine ucu değişimlerine gerek olmamasıdır. Ayrıca, makine hazne kapasiteleri de gruplama sezgiselinde dikkate alındığından, kısıt programlama modelinde göz ardı edilmiştir. Fakat, her gruptan sonra makine ucu değişim süresine ihtiyaç vardır. Bu problem de, sezgiselden elde edilen grup işleme sürelerine makine ucu değişim sürelerinin eklenmesiyle çözülmüştür.

Makine ucu değişimlerinin, kısıt programlama yöntemine eklenmemesiyle, global kısıtlarının propagasyon algoritmalarının tüm gücünden yararlanılmıştır. Sezgisel ve kısıt programlama yöntemlerinin seçilmesinin bir diğer nedeni de, iki yöntemin de çözüm sürelerinin çok kısa olması ve dolayısıyla toplam çözüm süresinin kısa olmasıdır.

5.3.2. Gruplama Sezgiseli ve Tabu Arama Algoritması (Ayrıştırma sezgiseli 2)

Ayrıştırma sezgiseli 1 yönteminde olduğu gibi makine ucu değişimlerinin, işlerin gereken makine ucu setlerine göre, gruplama sezgiseli uygulayarak gruplanmasıyla ikinci aşamada kullanılacak olan yöntemde göz önünde bulundurulmasına ihtiyaç olmamıştır. Makine ucu değişimleri göz ardı edildiği zaman 5.3.1'de de anlatıldığı gibi problem, çizelgeleme ve makine ucu atama problemine dönüşmektedir. Birinci dönem raporunda bu problem için geliştirilen tabu arama algoritması raporlanmıştır. Tabu arama algoritması, çizelgeleme ve makine ucu atama problemi için oldukça iyi sonuçlar vermiştir. Bu nedenle, ayrıştırma sezgiseli 2 yönteminin çizelgeleme bölümü için tabu arama algoritması alternatif bir yöntem olarak belirlenmiştir. İşlerin gruplanması, gruplama sezgiseli uygulanarak elde edilmiştir. 5.3.1'de olduğu gibi her bir grup birer iş gibi düşünülerek, tabu arama algoritmasına girdi olarak verilmiştir. Yine bu yöntemde de, her bir grubun toplam işlenme zamanlarına makine ucu değişim süreleri eklenmiştir. Bu şekilde, makine ucu değişimi tabu arama algoritmasına yansıtılmıştır. Ayrıştırma sezgiseli 2'nin akışı aşağıda verilmiştir.



Şekil 9. Ayrıştırma sezgiseli 2'nin akışı

İş-grup atama ve makine ucu-grup atama bilgileri, gruplama sezgiselinden elde edilerek tabu arama algoritması için girdi olarak kullanılmıştır. Daha önce de belirtildiği gibi, makine ucu değişimleri düşünülmeden, tabu arama algoritması çizelgeleme için kullanılmıştır. Ayrıştırma sezgiseli 1 yöntemi'nde de bahsedildiği gibi makine ucu değişimleri ve makine hazne kapasiteleri, gruplama sezgiselinde göz önünde bulundurulduğundan tabu arama algoritmasında da dikkate alınmamıştır.

Ayrıştırma sezgiseli 2 yöntemini oluşturan iki sezgiselin de (gruplama sezgiseli ve tabu arama algoritması) çok kısa sürede çözüme ulaştıklarından, toplam çözüm zamanının da kısa olması beklenmektedir.

5.3.3. Deneyler

Bu bölümde geliştirilen iki ayrıştırma sezgiselinin karşılaştırılması yapılacaktır. Deney seti olarak, 5.2.1 'de oluşturulan set kullanılmıştır. Örnekler, IBM OPL 6.1.1 ve MICROSOFT VISUAL STUDIO yazılımları ile AMD II P820, 1.8 GHz ve 3.74 GB RAM özellikli bilgisayarda çalıştırılmıştır. Her bir örnek için zaman limitinin üst sınırı bir saat olarak belirlenmiştir. Kısıt programlama modeli için arama stratejisi tekrardan başlama seçeneği kullanılmıştır. Tablo 23 ve 24'te arama ayrıştırma sezgiselleri 1 ve 2'nin karşılaştırılması verilmiştir.

Tablo 23. Çoklu makine ortamında ayrıştırma sezgiseli 1

<iş, makine sayısı, makine ucu, makine hazne kapasitesi >	Optimal Çözüm	Matematiksel Modelden Çözümden Sapma Yüzdesi	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı
<8, 2, 5, 4>	0	17,48	10/10	3,42
<8, 2, 8, 6>	0	41,59	10/10	2,84
<10, 2, 5, 4>	0	22,14	10/10	3,93
<10, 2, 8, 6>	0	67,62	10/10	3,20
<10, 3, 5, 4>	0	35,01	10/10	3,02
<10, 3, 8, 6>	0	86,47	10/10	3,13
<15, 2, 8, 6>	0	51,86	10/10	3,47

Tablo 24. Çoklu makine ortamında ayrıştırma sezgiseli 2

<iş, makine sayısı, makine ucu, makine hazne kapasitesi >	Optimal Çözüm	Matematiksel Modelden Çözümden Sapma Yüzdesi	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı
<8, 2, 5, 4>	0	17,48	10/10	0,16
<8, 2, 8, 6>	0	41,59	10/10	0,12
<10, 2, 5, 4>	0	23,12	10/10	0,22
<10, 2, 8, 6>	0	69,62	10/10	0,18
<10, 3, 5, 4>	0	35,01	10/10	0,23
<10, 3, 8, 6>	0	86,47	10/10	0,18
<15, 2, 8, 6>	0	51,86	10/10	0,33

Tablo 23 ve 24'e göre, geliştirilen ayrıştırma sezgiselleri, matematiksel model performansı ile karşılaştırıldığı zaman hiçbir problem setin en iyi çözüme ulaşamamışlardır. Çözümlerden sapma yüzdelerine bakıldığı zaman, kısıt programlama kullanılan ayrıştırma sezgiselinin 10 iş ve 2 makine olan setlerde daha iyi sonuçlar verdiği görülmüştür. Ayrıştırma yaklaşımında, problem iki farklı bağımsız alt problem olarak ele alındığı için, en iyi çözümden olan sapmaların büyük olması beklenmektedir.

Problem setleri tek makineye indirgendiğinde, işlerin gruplanması elde edilince çizelgeleme problemini çözmeye gerek yoktur, çünkü tek makinede son işin tamamlanma zamanı her

çizelge için aynıdır. Gruplar arasında makine ucu değişimi zamanı ile işlerin yapılma zamanı toplamı bize amaç fonksiyonunun değerini verecektir.

Tablo 25. Tekli makine ortamında ayrıştırma sezgiseli 1

<iş, makine sayısı, makine ucu, makine hazne kapasitesi >	Optimal Çözüm	Matematiksel Modelden Çözümden Sapma Yüzdesi	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı
<5, 5, 4>	10	0	10/10	3,64
<5, 8, 6>	9	0,4	10/10	7,04
<8, 5, 4>	10	0	10/10	3,85
<8, 8, 6>	10	0,57	10/10	3,86
<10, 5, 4>	10	0	10/10	3,84
<10, 8, 6>	9	0,21	10/10	3,79
<15, 8, 6>	10	0	10/10	4,40

Tablo 26. Tekli makine ortamında ayrıştırma sezgiseli 2

<iş, makine ucu, makine hazne kapasitesi >	Optimal Çözüm	Matematiksel Modelden Çözümden Sapma Yüzdesi	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı
<5, 5, 4>	10	0	10/10	0,24
<5, 8, 6>	9	0,4	10/10	0,18
<8, 5, 4>	10	0	10/10	0,17
<8, 8, 6>	10	0,57	10/10	0,20
<10, 5, 4>	10	0	10/10	0,13
<10, 8, 6>	9	0,21	10/10	0,25
<15, 8, 6>	10	0	10/10	0,35

Problem tek makineye indirgendiği zaman, ayrıştırma sezgiselinin neredeyse çoğu problem setinde en iyi çözüme ulaştığı Tablo 25 ve 26'dan görülebilir. Sadece iki tane örnek problemde, ayrıştırma sezgiselleri en iyi çözüme ulaşamamıştır. Ayrıca, matematiksel modelden sapma yüzdeleri de oldukça düşüktür. Tablo 25 ve 26'ya bakıldığında, iki ayrıştırma sezgiselinin de aynı sonuçları verdiği görülebilir. Bunun nedeni, iki ayrıştırma sezgiseli için de gruplama sezgisel sonuçlarının aynı olması ve tek makinede çizelgelenmesidir. Tek makinede, işleri işlenme sırası değişse bile işlerin işlenmesi için geçen toplam sürenin değişmesi beklenmez.

5.4. Kısıt Programlama Modeli

Probleme çözüm yaklaşımı olarak kısıt programlama yöntemini kullanılmıştır. Kısıt programlama yaklaşımı genel olarak çizelgeleme problemleri için oldukça iyi sonuç vermektedir. Operasyon atama ve çizelgeleme probleminde bunu gözlemleme fırsatımız olmuştur. Literatürde benzer çalışmalar yer almaktadır. Örneğin (ZEBALLOS, 2010) ve (ZEBALLOS vd., 2010) çizelgeleme ve makine ucu atama problemleri için bir kısıt programlama yaklaşımı önermişlerdir. Makine ucu yıpranmalarını da ele almışlar fakat makine ucu değişimini göz önünde bulundurmamışlardır. Bu iki çalışma ve projenin ikinci döneminde yazılan kısıt programlama modeline ek olarak makine ucu değişimi için kısıtlar eklenecektir.

5.4.1. Parallel Makineler için Kısıt Programlama Modeli

Çizelgeleme ve makine ucu değişimi problemi için geliştirilen kısıt programlama modelinde global kısıtların yanı sıra mantıksal kısıtlar da kullanılmıştır. Özellikle makine ucu değişim durumlarının modellenmesi için ekstra karar değişkenlerinin tanımlanmasına gerek duyulmuştur. Çizelgeleme bölümü için önceden geliştirilen modeldeki çizelgeleme problemlerine özgü kısıtlar ve karar değişkenleri de kullanılmıştır. Fakat, amaç fonksiyonu önceki kısıt programlama modeline göre farklı tanımlanmıştır. En son işin işlenme süresini en azlamanın yanı sıra makine ucu değişimlerinin de enazlanması gerekmektedir. Kısıt programlama yönteminde, çizelgeleme problemlerine özgü tanımlanan aktivite karar değişkeninin işlenme zamanları değişken olarak tanımlanamamaktadır. Makine ucu değişim zamanı bir aktivitenin süresine eklenemez bu nedenle propagasyon algoritmalarından da yararlanmak için makine ucu değişim zamanları ekstra bir karar değişkeni ile hesaplanmaktadır.

Problemde, işleri aktivite olarak makine ve her bir makine ucunun her bir kopyasını da kaynak şeklinde değerlendirilmektedir. Bu şekilde, problemin çizelgeleme bölümü için global kısıtların tüm algoritmik gücünden yararlanılmaktadır. Makine ucu değişimlerini modelleyebilmek için ise her makinedeki iş ve her makine ucu kopyasının atandığı makinelerin sıraları bilgilerini elde etmek gerekmektedir. Bu nedenle, ekstra karar

değişkenlerinin tanımlanması kaçınılmazdır çünkü bu bilgilere kısıt programlamaya özgü karar değişkenlerinden ulaşılamamaktadır.

Kısıt programlama modelinde, önceki modellerden farklı olarak "*count*" global kısıtı kullanılmıştır. "*count*" kısıtı, girdi olarak verilen karar değişkeni dizininin içinden, ikinci girdi olarak verilen karar değişkeni sayısı kadar değişkenin üçüncü girdi olarak verilen değere eşit olmasını sağlar. Ayrıca modelde, kullanılan yazılıma özgü iki tane fonksiyon da kullanılmıştır. "*maxl*" fonksiyonu verilen dizindeki maksimum değeri döndürürken, "*minl*" fonksiyonu da minimum değeri döndürür. Bu problemde de kısıt programlama yaklaşımı için IBM ILOG CP Optimizer 2.3 yazılımı kullanılmıştır. Kısıt programlama modelinde kullanılan parametreler aşağıdaki gibidir:

$l(i)$: i işi için gereken makine uçları kümesi
p_{ij}: i işinin j makinesindeki işlem süresi
ts: makine ucu değişimi süresi
cap: makine ucu haznesinin kapasitesi

Karar değişkenleri:

Activity J_i,	Her bir işin aktivite olarak tanımlanmasını sağlar
UnaryResource M_j,	Her bir makinenin tekli kaynak şeklinde tanımlanmasını sağlar
UnaryResource T_{kc},	Her bir makine ucu kopyasının tekli kaynak olarak tanımlanmasını sağlar
$Position_{ij} \in \{0,...,n\}$	Her işin atandığı makinelerdeki sıra numarasını gösterir. İşin atanmadığı makinede sıra numarası "0" dır.
$State_{skj} \in \{0,1\}$	1, eğer j makinesinde, s sıra numarasında, k makine ucu kopyası yüklenmişse
$Switching_{ij} \in \{0,1\}$	1, eğer j makinesinin s sırasında makine ucu değişimi

	gerçekleşmişse
$Situation_{skj} \in \{0,1,2\}$	Eğer j makinesinde, s sıra numarasında, k makine ucu kopyası yüklendiği durumda, değer 1'e eşit ise, makine ucu değişimi olacaktır.
$ToolPos_{ki} \in \{0,...,n\}$	k makine ucu kopyasının i işine hangi sırada yüklendiğini göstermektedir.
$Load_j$	Her makineye atanan toplam iş sayısı
$X_i \in \{1,...,m\}$	i işinin hangi makineye atandığını gösterir
$TotalSwitch$	Toplam makine ucu değişim zamanı

Kısıt programlama modelimizin amacı son işin tamamlanma zamanı ve toplam makine ucu değişimi için harcanan zamanı en azlamaktır.

$$Min.\ \max\left\{\text{endOf}\left(J_1\right),\ldots,\text{endOf}\left(J_n\right)\right\}+TotalSwitch$$

Modelin kısıtları aşağıda açıklamalarıyla birlikte verilmiştir.

- Toplam makine ucu değişim zamanı, her sıradaki makine ucu değişim zamanlarının toplamına eşittir.

  $$TotalSwitch \geq \sum_i Switching_{Position_{i,x_i},x_i} ts$$

- Eğer farklı iki iş aynı makine ucu kopyasını kullanıyorlarsa ve işin tamamlanma zamanı diğer işin başlangıç zamanına eşit veya küçükse, makine ucu kopya sıra numarasında o işin numarası diğer işin numarasından küçük olmalıdır.

  $$State_{Position_{i,x_i},k,x_i} = 1 \wedge State_{Position_{q,x_q},k,x_q} = 1 \wedge endOf\left(J_i\right) < startOf\left(J_q\right) \rightarrow ToolPos_{k,i} < ToolPos_{k,q}$$
  $$\forall i,q,\ \forall k \mid l\left(i\right) = l\left(q\right) = k$$

- Aynı makine ucu kopyasının art arda yüklendiği işler farklı makinelerde ise kopyayı farklı makineye yüklemek için makine ucu değişimi gerekir.

  $$ToolPos_{k,i} = s \wedge ToolPos_{k,q} = s+1 \wedge X_i \neq X_q \rightarrow Switching_{position_{q,x_q},x_q} = 1\ \ \forall i,q,s \mid s < n\ \ \forall k$$

- Aktivite karar değişkeninin atandığı makine ile tanımlanan ekstra atama karar değişkeninin atandığı makineler aynı olmalıdır.

  $J_i = j \rightarrow X_i = j \quad \forall i, j$

- Her bir makinenin yükü, o makineye atanan işlerin sayısına eşit olmalıdır.

  $\sum_i \left(J_i = j \right) = Load_j \quad \forall j$

- Her işe bir sıra numarası atanmalıdır. Her makine için iş sayısı kadar sıra numarası tanımlanmıştır. Ancak paralel makineler olduğundan, bir makineye iş sayısından az sayıda iş atanabilir. Bu nedenle her sıra numarası en fazla bir işe atanmalıdır.

  $count\left(all(i)\ Position_{i,j}, s \right) \leq 1 \quad \forall j, s$

- İşlerin *s* sırasına atanmadığını göstermek için "0" değeri kullanılmıştır. Bu nedenle, en fazla iş sayısı kadar işler atanmadığı makinelerde "0" sıra numarası değerini alır.

  $count\left(all(i)\ Position_{i,j}, 0 \right) \leq n \quad \forall j$

- Eğer farklı iki iş aynı makineye atandıysa ve birinin bitiş zamanı diğerinin başlangıç zamanına eşit veya küçükse, o makinede işin sıra numarası diğer işin sıra numarasından küçük olmalıdır.

  $X_i = j \wedge X_q = j \wedge endOf\left(J_i \right) \leq startOf\left(J_q \right) \rightarrow Position_{i,x_i} < Position_{q,x_q} \quad \forall i, q, j \mid i \neq q$

- Boş olmayan her sıraya, atanan işin gerektirdiği makine ucu çeşidinin yalnızca tek bir kopyası atanmalıdır.

  $\sum_{c \in r_k} State_{Position_{i,x_i},c,x_i} = 1 \quad \forall i, k \mid k \in l\left(i \right)$

- Herhangi bir iş atanmış sıraya atanan farklı makine ucu sayısı makine haznesini geçemez.

  $\sum_k State_{Position_{i,x_i},k,x_i} \leq cap \quad \forall i$

- Art arda sıra numaralarındaki makine ucu değişim durumları *Situation* karar değişkenine eşitlenir. Eğer değer "0" veya "2" ise yüklenen makine uçlarında değişiklik olmamıştır. Değer "1" ise bir önceki sırada olmayan bir makine ucu yüklenmiş veya çıkartılmıştır.

$$maxl\left(State_{Position_{i,x_i},k,x_i}, State_{Position_{i,x_i}+1,k,x_i}\right) - minl\left(State_{Position_{i,x_i},k,x_i}, State_{Position_{i,x_i}+1,k,x_i}\right) = Situation_{Position_{i,x_i}+1,k,x_i}$$
$$\forall i,k$$

- Her bir iş mutlaka bir makineye atanmalıdır.

$$Alternative\left(J_i,[M_1,\ldots,M_m],[p_{i1},\ldots,p_{im}]\right), \quad \forall i \in \{1,\ldots,n\}$$

- Her işin işlenebilmesi için gerektirdiği makine ucu kopyalarından bir tanesine mutlaka atanmalıdır.

$$Alternative\left(J_i,\left[T_{k1},\ldots,T_{k,r_k}\right]\right), \quad \forall i \in \{1,\ldots,n\}, \forall k \in l(i)$$

Yukarıda verilen kısıtlarda, X_i ve $Position_{ij}$ karar değişkenlerinin başka bir karar değişkeninde indis olarak kullanılmasına "*channeling*" adı verilir. *Channeling*, kısıt programlama yaklaşımında problemin boyutunu azaltmaya yönelik olan bir yöntemdir. Geliştirilen kısıt programlama modeli çizelgeleme ve makine ucu değişimi problemini bütünüyle ele almaktadır. Fakat, mantıksal ve ikili karar değişkenlerin kullanım gereksiniminden dolayı kısıt propagasyon algoritmalarının etkili bir şekilde çalışması beklenmemektedir. Ayrıca, mantıksal kısıtlarda ikiden fazla farklı ifadeyi önlemek amacıyla bazı ekstra karar değişkenleri tanımlanmıştır. Bunun da amacı, kısıt propagasyon algoritmalarının daha iyi çalışmasıdır.

5.4.2. Tek Makine için Kısıt Programlama Modeli

Çizelgeleme ve makine ucu değişim problemini tek makineye indirgediğimiz zaman 5.4.1'de geliştirilen kısıt programlama yaklaşımından biraz farklı şekilde modellenebilmektedir. Bir önceki modelde yer alan makine bilgisine artık ihtiyaç yoktur. Her iş aynı makinede işleneceği için her iş birbirinden farklı sıra numarasını atanacaktır. Bir önceki modelde sıra numarası ataması için "*count*" global kısıtı kullanılmıştır çünkü işlerin atanmadığı makinelerdeki sıra numarası "0" değerini alacaktır. Fakat bu modelde, her iş aynı makineye atanacağı ve sıra numaralarının da birbirinden farklı olacağı için "*alldifferent*" global kısıtı

kullanılmıştır. "*count*" global kısıtının kullanılmamasının nedeni, birden fazla aynı sıra numarası değerini alacak işin olmamasıdır. "*alldifferent*" global kısıtı, verilen karar değişkeni dizinindeki değişkenlere atanan değerlerin birbirinden farklı olmasını sağlar. Ayrıca bu modelde X_i karar değişkenine de ihtiyaç yoktur çünkü problemde sadece tek makine vardır. Bu problemde de kısıt programlama yaklaşımı için IBM ILOG CP Optimizer 2.3 yazılımı kullanılmıştır. Kısıt programlama modelinde kullanılan parametreler aşağıdaki gibidir:

$l(i)$: i işi için gereken makine uçları kümesi
p_{ij}: i işinin j makinesindeki işlem süresi
ts: makine ucu değişimi süresi
cap: makine ucu haznesinin kapasitesi
Karar değişkenleri:

Activity J_i,	Her bir işin aktivite olarak tanımlanmasını sağlar
UnaryResource M_j,	Her bir makinenin tekli kaynak şeklinde tanımlanmasını sağlar
UnaryResource T_{kc},	Her bir makine ucu kopyasının tekli kaynak olarak tanımlanmasını sağlar
$Position_i \in \{0,...,n\}$	Her işin atandığı sıra numarasını gösterir.
$State_{sk} \in \{0,1\}$	1, eğer s sıra numarasında, k makine ucu kopyası yüklenmişse
$Switching_i \in \{0,1\}$	1, eğer s sırasında makine ucu değişimi gerçekleşmişse
$Situation_{sk} \in \{0,1,2\}$	Eğer s sıra numarasında, k makine ucu kopyası yüklendiği durumda, değer 1'e eşit ise, makine ucu değişimi olacaktır.
$TotalSwitch$	Toplam makine ucu değişim zamanı

Kısıt programlama modelimizin amacı son işin tamamlanma zamanı ve toplam makine ucu değişimi için harcanan zamanı en azlamaktır.

$$Min.\ \max\left\{\text{endOf}\left(J_1\right),\ldots,\text{endOf}\left(J_n\right)\right\}+TotalSwitch$$

Modelin kısıtları aşağıda açıklamalarıyla birlikte verilmiştir.

- Toplam makine ucu değişim zamanı, makine ucu değişim zamanlarının toplamına eşittir.

  $$TotalSwitch \geq \sum_i Switching_{Position_i} ts$$

- Her iş farklı bir sıra numarasına atanmalıdır

  $$alldifferent\left(all(i)\ Position_i\right)$$

- Farklı iki işin birinin bitiş zamanı diğerinin başlangıç zamanına eşit veya küçükse, makinede işin sıra numarası diğer işin sıra numarasından küçük olmalıdır.

  $$endOf\left(J_i\right) \leq startOf\left(J_q\right) \rightarrow Position_i < Position_q \quad \forall i,q \mid i \neq q$$

- Boş olmayan her sıraya, atanan işin gerektirdiği makine ucu çeşidinin yalnızca tek bir kopyası atanmalıdır.

  $$\sum_{c \in r_k} State_{Position_i,c} = 1 \quad \forall i,k \mid k \in l\left(i\right)$$

- Herhangi bir iş atanmış sıraya atanan farklı makine ucu sayısı makine haznesini geçemez.

  $$\sum_k State_{Position_i,k} \leq cap \quad \forall i$$

- Art arda sıra numaralarındaki makine ucu değişim durumları *Situation* karar değişkenine eşitlenir. Eğer değer "0" veya "2" ise yüklenen makine uçlarında değişiklik olmamıştır. Değer "1" ise bir önceki sırada olmayan bir makine ucu yüklenmiş veya çıkartılmıştır.

  $$maxl\left(State_{Position_i,k}, State_{Position_i+1,k}\right) - minl\left(State_{Position_i,k}, State_{Position_i+1,k}\right) = Situation_{Position_i+1,k} \quad \forall i,k$$

- Her bir iş mutlaka bir makineye atanmalıdır.

  $$Alternative\left(J_i,\left[M_1,\ldots,M_m\right],\left[p_{i1},\ldots,p_{im}\right]\right), \quad \forall i \in \left\{1,\ldots,n\right\}$$

- Her işin işlenebilmesi için gerektirdiği makine ucu kopyalarından bir tanesine mutlaka atanmalıdır.

$$Alternative\left(J_i, \left[T_{k1}, \ldots, T_{k,r_k}\right]\right), \quad \forall i \in \{1, \ldots, n\}, \forall k \in l(i)$$

5.4.3. Deneyler

Kısıt programlama modellerinin performanslarını ölçmek amacıyla 5.2.1'de verilen setler üzerinde deneyler yapılmıştır.tablo 27 ve 28'de sonuçlar verilmektedir.

Tablo 27. Çoklu makine ortamında kısıt programlama modeli

<iş, makine sayısı, makine ucu, makine hazne kapasitesi >	Optimal Çözüm	Matematiksel Modelden Çözümden Sapma Yüzdesi	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı
<8, 2, 5, 4>	0	-2,67	10/10	
<8, 2, 8, 6>	0	0,92	10/10	
<10, 2, 5, 4>	0	1,47	10/10	
<10, 2, 8, 6>	0	6,78	10/10	
<10, 3, 5, 4>	0	-3,54	10/10	
<10, 3, 8, 6>	0	-8,54	10/10	
<15, 2, 8, 6>	0	32,21	10/10	

Geliştirilen kısıt programlama modeli hiçbir problem setinde optimal çözüme ulaşamamıştır. Fakat, üç problem setinde, kısıt programlama modelinin bulduğu sonuçlar ortalamada matematiksel modele göre daha iyidir. Tablo 27'den de görülebileceği gibi iş sayısı 15 olduğu zaman, matematiksel model kısıt programlamaya göre daha iyi bir performans sergilemiştir.

Tablo 28. Tekli makine ortamında kısıt programlama modeli

<iş, makine ucu, makine hazne kapasitesi >	Optimal Çözüm	Matematiksel Modelden Çözümden Sapma Yüzdesi	Çözüm Alınma Sayısı	Ortalama Çözüm Zamanı
<5, 5, 4>	10	0	10/10	4,75
<5, 8, 6>	10	0	10/10	565,6
<8, 5, 4>	10	0	10/10	1305,9
<8, 8, 6>	10	0	10/10	2234,5
<10, 5, 4>	10	0	10/10	985,89
<10, 8, 6>	10	0	10/10	1360,58
<15, 8, 6>	0	3,21	10/10	0,35

Tekli makine için geliştirilen kısıt programlama modeli daha etkili olmuştur. Bunun da nedeni paralel makinelerden ortaya çıkan bazı durumların modele eklenmesine gerek duyulmamasıdır. Çözüm zamanının, matematiksel modelin çözüm zamanına göre uzun olmasına karşın, matematiksel modelin optimal çözümü bulamadığı birçok problem setinde kısıt programlama modeli en iyi çözüme ulaşmıştır. Ancak, çoklu makine problem setinde olduğu gibi iş sayısı 15 olduğunda, iki modelin verilen zaman içerisinde buldukları çözümler karşılaştırıldığı zaman (Tablo 22 ve 28) ortalamada matematiksel modelin daha iyi sonuçlar bulduğu gözlemlenmiştir.

İşlerin ve makine uçlarının çizelgelenmesi ve makine ucu değişimi problemi için matematiksel model, iki farklı ayrıştırma sezgiseli ve kısıt programlama modelleri geliştirilmiştir. Makine ucu değişim durumlarının probleme getirdiği zorluk nedeniyle kısıt programlama modelinin sadece tek makineli setlerde en iyi çözüme ulaşması beklenen bir durumdur. Ancak, çoklu makine için geliştirilen kısıt programlama modelinde, fazla mantıksal kısıtlar olmasına rağmen ortalamada üç problem setinde matematiksel modelden daha iyi sonuçlar vermiştir. Geliştirilen ayrıştırma sezgisellerinde de, çizelgeleme ve makine ucu atama probleminde kullanılan çözüm yöntemlerinin gücünden faydalanılmak istenmiştir.

6. SONUÇ

Bu projede esnek imalat sistemlerinde atama ve çizelgeleme problemleri üzerinde çalışılmıştır ve bu problemler üç ana başlık altında incelenmiştir.

Problemlerden ilki işlerin ve makine uçlarının makinelere atanması ve çizelgelenmesi problemidir. Bu problem için matematiksel modeller ve kısıt programlama yoluyla en iyi sonuç aranmış, ayrıştırma sezgiseli ve tabu arama algoritması ile de büyük problemler için olurlu sonuçlar elde edilmiştir.

İkinci problemde işlerin toplam gereken makine ucu sayısı makine haznesini geçmeyecek şekilde gruplanması amaçlanmıştır. Son problemde işlerin ve makine uçlarının makinelere atanması, çizelgelenmesi ve makine ucu değişimi problemleri birleştirilmiştir. Bu problemler için de matematiksel model, kısıt programlama ve sezgiseller yardımıyla sonuçlara ulaşılmıştır.

Birinci ve üçüncü problemler daha önce tanımlanan şekliyle literatürde çalışılmamıştır. Bu projede esnek imalat sistemleri literatürüne katkıda bulunmak, yeni problemler tanımlamak ve daha önce denenmemiş çözüm yöntemleri geliştirmek amaçlanmıştır. Matematiksel model ve kısıt programlama yaklaşımları büyük problemlerde sonuç vermemektedir. Problemlerin zorluğu düşünüldüğünde bu, beklenen bir durumdur. Sezgisel yöntemler yardımıyla problemlere sonuçlar bulunmuş ve bu sonuçların en iyi çözüme yakın olduğu gösterilmiştir.

Makine ucu değiştirme problemi için geliştirilen kısıt programlama modeli ileri teknikler yardımıyla iyileştirilebilir.

Bu projede, makine ucu değişimlerinin değiştirilen makine ucu sayısı ve çeşidinden bağımsız olduğu varsayılmıştır. Bu sebeple, makine ucu değişimi sayısı enazlanmaya çalışılmıştır. Gelecek çalışma konuları arasında makine ucu değişimi süresini en aza indirmek düşünülebilir. Bu problemde makine ucu değişimi süresinin değiştirilen makine ucu sayısına göre farklılık gösterdiği durum incelenebilir.

REFERANSLAR

AGGOUN A., Beldiceanu N., Extending chip in order to solve complex scheduling and placement problems, *Mathematical and Computer Modeling*, 17(7), 57 – 73, (1993).

AGNETIS A., Alfieri A., Brandimarte P., Prinsecchi P., Joint job/tool scheduling in a flexible manufacturing cell with no on-board tool magazine, *Computer Integrated Manufacturing Systems*, 10, 61-68, (1997).

AKÇALI E., Üngör A., Uzsoy, R., Short-term capacity allocation problem with tool and setup constraints, *Naval Research Logistics*, 52, 754-764, (2005).

AKTURK M.S., Ghosh J.B., Gunes E.D., Scheduling with tool changes to minimize total completion time: A study of heuristics and their performance, *Naval Research Logistics*, 50, 15-30, (2003).

AKTURK M.S., Ozkan S., Integrated scheduling and tool management in flexible manufacturing systems, *International Journal of Production Research*, 39, 2697-2722, (2001).

AVCI S., Akturk M.S., Tool magazine arrangement and operations sequencing on CNC machines, *Computers and Operations Research*, 23, 1069-1081, (1996).

BAPTISTE Ph., Le Pape C., A theoretical and experimental comparison of constraint propagation techniques for disjunctive scheduling, Proceedings of 14th International Joint Conference on Artificial Intelligence, 2, 600-606, (1993).

BAPTISTE Ph., Le Pape C., Edge-finding constraint propagation algorithms for disjunctive and cumulative scheduling, Proceedings of the 15th Workshop of the U.K. Planning Special Interest Group, (1996).

BARD J.F., A heuristic for minimizing the number of tool switches on a flexible machine, *IIE Transactions*, 20, 382-391, (1988).

BİLGİN S., *A capacity allocation problem in flexible manufacturing systems*, (Yüksek Lisans Tezi), Orta Doğu Teknik Üniversitesi Endüstri Mühendisliği Bölümü, (2004).

BİLGİN S., Azizoğlu M., Capacity and tool allocation problem in flexible manufacturing systems, *Journal of the Operational Research Society*, 57, 670-681, (2006).

BİLGİN S., Azizoğlu M., Operation assignment and capacity allocation problem in automated manufacturing systems, *Computers and Industrial Engineering*, 56, 662-676, (2009).

BLAZEWICZ J., Finke G., Scheduling with resource management in manufacturing systems, *European Journal of Operational Research*, 76, 1-14, (1994).

CAO D., Chen M., Wan G., Parallel machine selection and job scheduling to minimize machine cost and job tardiness, *Computers & Operations Research*, 32, 1995-2012, (2005).

CASEAU Y., Laburthe F., Disjunctive scheduling with task intervals, LIENS Technical Report, 95-25, The MIT press, (1995).

COLOMBANI Y., Constraint Programming: an efficient and practical approach to solving the job-shop problem, Proceeding 2nd International Conference on Principles and Practice of Constraint Programming, 149 – 163, (1996).

CRAMA Y., Kolen A.W.J., Oerlemans A.G., Spieksma F.C.R., Minimizing the number of tool switches on a flexible machine, *The Internatinal Journal of Flexible Manufacturing Systems*, 6, 33-54, (1994).

CRAMA Y., Combinatorial optimization models for production scheduling in automated manufacturing systems, *European Journal of Operational Research*, 99, 136-153, (1997).

CRAMA Y., Flippo O.E., van de Klundert J., Spieksma F.C.R., The assembly of printed circuit boards: A case with multiple machines and multiple board types, *European Journal of Operational Research*, 98, 457-472, (1997).

ÇATAY B., Erengüç Ş.S., Vakharia A.J., Tool capacity planning in semiconductor manufacturing, *Computers and Operations Research*, 30, 1349-1366, (2003).

CHEN, F.F., Ker, J-I., Kleawpatinon, K., An effective part-selection model for production planning of flexible manufacturing systems, *International Journal of Production Research*, 33, 2671-2683, (1995).

D'ALFONSO T.H., Ventura J.A., Assignment of tools to machines in a flexible manufacturing system, *European Journal of Operational Research*, 81, 115-133, (1995).

DEGRAEVE Z., Gochet W., Jans R., Alternative formulations for a layout problem in the fashion industry, *European Journal of Operational Research*, 143, 80-93, (2002).

DENİZEL M., Minimization of the number of tool magazine setups on automated machines: a Lagrangean decomposition approach, *Operations Research*, 51, 309-320, (2003).

ECKER K.H., Gupta J.N.D., Scheduling tasks on a flexible manufacturing machine to minimize tool changing delays, *European Journal of Operational Research*, 164, 627-638, (2005).

FLENER P., Frisch A. M., Hnich B., Kiziltan Z., Miguel I., Pearson J., Walsh T., Breaking row and column symmetries in matrix models, 8th International Conference on Principles and Practice of Constraint Programming, 2470, 462 – 476, (2002).

FOCACCI F., Milano M., Global cut framework for removing symmetries, 17th International Conference on Principles and Practice of Constraint Programming, 2239, 77 – 92, (2001).

FRISCH A. M., Hnich B., Kiziltan Z., Miguel I., Walsh T., Propagation algorithms for lexicographic ordering constraints, *Artificial Intelligence*, 170(10), 333 – 347, (2006).

GAMILA M.A., Motavalli S., Modeling technique for loading and scheduling problems in FMS, *Robotics and Computer Integrated Manufacturing*, 19, 45-54, (2003).

GENT I. P., Smithl B. M., Symmetry breaking in constraint programming, 14th European Conference on Artificial Intelligence, 599-603, (2000).

GLOVER F., Future paths for integer programming and linkage to artificial intelligence, *Computers & Operations Research*, 13, 533-549, (1986).

GLOVER F., Laguna, M., *Tabu search*, Kluwer Academic Publishers, (1997).

GRAY A.E., Seidmann A., Stecke K.E., A synthesis of decision models for tool management in automated manufacturing, *Management Science*, 39, 549-565, (1993).

HOP N.V., The tool-switching problem with magazine capacity and tool size constraints, *IIE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 35, 617-628, (2005).

JANS R., Solving lot sizing problems on parallel identical machines using symmetry-breaking constraints, *INFORMS Journal on Computing*, 21, 123-136, (2009).

KARAKAYALI İ., Azizoğlu M., Minimizing total flow time on a single flexible machine, *International Journal of Flexible Manufacturing Systems*, 18, 55-73, (2006).

KASHYAP A.S., Khator S.K., Analysis of tool sharing in an FMS: a simulation study, *Computers and Industrial Engineering*, 30, 137-145, (1996).

KELLERER H., Strusevich V.A., Scheduling problems for parallel dedicated machines under multiple resource constraints, *Discrete Applied Mathematics*, 133, 45-68, (2004).

KONAK A., Kulturel-Konak S., An ant colony optimization approach to the minimum tool switching instant problem in flexible manufacturing system, Proceedings of IEEE Symposium on Computational Intelligence in Scheduling, 43-48, (2007).

KONAK A., Kulturel-Konak S., Azizoğlu M., Minimizing the number of tool switching instants in Flexible Manufacturing Systems, *International Journal of Production Economics*, 116, 298-307, (2008).

KUMAR N S., Sridharan R., Simulation modeling and analysis of tool sharing and part scheduling decisions in single-stage multimachine flexible manufacturing systems, *Robotics and Computer-Integrated Manufacturing*, 23, 361-370, (2007).

LABORIE P., Rogerie J., Shaw P., Vilim P., Reasoning with conditional time-intervals part ii: An algebraical model for resources, Proceedings of the 22th International FLAIRS Conference, (2009).

LAPORTE G., Salazar-Gonzalez J.J., Semet F., Exact algorithms for the job sequencing and tool switching algorithm, *IIE Transactions*, 36, 37-45, (2004).

LE PAPE C., Baptiste Ph., Nuijten W., *Constraint-based scheduling: applying constraint programming to scheduling problems*, Kluwer Academic Publishers (2001).

LEE C.S., Kim S.S., Choi J.S., Operation sequence and tool selection in flexible manufacturing systems under dynamic tool allocation, *Computers and Industrial Engineering*, 45, 61-73, (2003).

MARGOT F., Pruning by isomorphism in branch-and-cut, *Mathematical Programming Series B*, 94, 71-90, (2002a).

MARGOT F., Exploiting orbits in symmetric ILP, *Mathematical Programming Series B*, 98, 3-21, (2002b).

MERCIER L., Van Hentenryck P., Edge finding for cumulative scheduling, *INFORMS Journal on Computing*, 20(1), 143-153, (2008).

NUJITEN W. P. M., Aarts E. H. L. A., A computational study of constraint satisfaction for multiple-capacitated job-shop scheduling, *European Journal of Operational Research*, 90, 269 – 284, (1996).

ÖZPEYNİRCİ S., *Allocation and tooling decisions in flexible manufacturing systems*, (Doktora Tezi), Orta Doğu Teknik Üniversitesi Endüstri Mühendisliği Bölümü, (2007).

ÖZPEYNİRCİ S., Azizoğlu M., Bounding approaches for operation assignment and capacity allocation problem in flexible manufacturing systems, *Computers and Operations Research*, 36, 2531-2540, (2009a).

ÖZPEYNİRCİ S.B.**,** Azizoğlu M., Beam search algorithm for capacity allocation problem in flexible manufacturing systems, *Computers and Industrial Engineering*, 56, 1464-1473, (2009b).

ÖZPEYNİRCİ S., Azizoğlu M., Capacity allocation problem in flexible manufacturing systems: Branch and bound based approaches, *International Journal of Production Research*, 47, 5941-5958, (2009c).

ÖZPEYNİRCİ S., Azizoğlu M., A Lagrangean relaxation based approach for the capacity allocation problem in flexible manufacturing systems, *Journal of the Operational Research Society*, 61, 872-877, (2010).

PERSI P., Ukovich W., Pesenti R., Nicolich M., A hierarchic approach to production planning and scheduling of a flexible manufacturing system, *Robotics and Computer Integrated Manufacturing*, 15, 373-385, (1999).

PINEDO M., *Scheduling Theory, Algorithms and Systems*, Prentice Hall, New Jersey, (2002).

ROH H. -K., Kim Y. -D., Due-date based loading and scheduling methods for a flexible manufacturing system with an automatic tool transporter, *International Journal of Production Research*, 35, 2989-3003, (1997).

RUPE J., Kuo W., Solutions to a modified tool loading problem for a single FMM, *International Journal of Production Research*, 35, 2253-2268, (1997).

SHANKER K., Srinivasulu A., Some solution methodologies for loading problems in a flexible manufacturing system, *International Journal of Production Research*, 27, 1019-1034, (1989).

SHERALI H.D., Smith J.C., Improving discrete model representations via symmetry considerations, *Management Science*, 47, 1396-1407, (2001).

SHERALI H.D., Fraticelli B.M.P., Meller R.D., Enhanced model formulation for optimal facility layout, *Operations Research*, 51, 629-644, (2003).

SONG C.Y., Hwang H., Optimal tooling policy for a tool switching problem of a flexible machine with an automatic tool transporter, *International Journal of Production Research*, 40, 873-883, (2002).

STECKE K.E., Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems, *Management Science*, 29, 273-288, (1983).

SWARNKAR R., Tiwari M.K., Modeling machine loading problem of FMSs and its solution methodology using a hybrid tabu search and simulated annealing-based heuristic approach, *Robotics and Computer-Integrated Manufacturing*, 20, 199-209, (2004).

TANG C.S., Denardo E.V., Models arising from a flexible manufacturing machine, Part I: Minimization of the number of tool switches, *Operations Research*, 36, 767-777, (1988a).

TANG C.S., Denardo E.V., Models arising from a flexible manufacturing machine, Part II: Minimization of the number of switching instances, *Operations Research*, 36, 778-784, (1988b).

THÖRNBLAD, K., Almgren T., Patriksson M., Strömberg A.-B., Mathematical optimization of a flexible job shop problem including preventive maintenance and availability of fixtures, Proceedings of the 4th World P&OM Conference / 19th International Annual EurOMA Conference, Amsterdam, Netherlands, (2012).

THÖRNBLAD, K., *On the Optimization of Schedules of a Multitask Production Cell*, (Licentiate thesis), Department of Mathematical Sciences, Chalmers University of Technology and the University of Gothenburg, (2011).

TOKTAY, L. B., Uzsoy R., A capacity allocation problem with integer side constraints, *European Journal of Operational Research*, 109, 170-182, (1998).

TZUR M., Altman A., Minimization of tool switches for a flexible manufacturing machine with slot assignment of different tool sizes, *IIE Transactions*, 36, 95-110, (2004).

VENTURA J.A., Chen F.F., Leonard M.S., Loading tools to machines in flexible manufacturing systems, *Computers and Industrial Engineering*, 15, 223-230, (1988).

VENTURA J.A., Kim D., Parallel machine scheduling with earliness-tardiness penalties and additional resource constraints, *Computers & Operations Research*, 30, 1945-1958, (2003).

VILIM P., O(n log n) filtering algorithms for unary resource constraint, Proceedings of CP-AI-OR, 3011, 335-347, (2004).

ZEBALLOS L.J., A constraint programming approach to tool allocation and production scheduling in flexible manufacturing systems, *Robotics and Computer-Integrated Manufacturing*, 26, 725-743, (2010).

ZEBALLOS L.J., Quiroga O.D., Henning G.P., A constraint programming model for the scheduling of flexible manufacturing systems with machine and tool limitations, *Engineering Applications of Artificial Intelligence*, 23, 229-248, (2010).

TÜBİTAK
PROJE ÖZET BİLGİ FORMU

Proje Yürütücüsü:	Yrd. Doç. Dr. SELİN ÖZPEYNİRCİ
Proje No:	110M492
Proje Başlığı:	Esnek İmalat Sistemlerinde Atama Ve Çizelgeleme Problemleri
Proje Türü:	Kariyer
Proje Süresi:	24
Araştırmacılar:	
Danışmanlar:	
Projenin Yürütüldüğü Kuruluş ve Adresi:	İZMİR EKONOMİ Ü. BİLGİSAYAR BİLİMLERİ F. ENDÜSTRİ SİSTEMLERİ MÜHENDİSLİĞİ B.
Projenin Başlangıç ve Bitiş Tarihleri:	15/04/2011 - 15/04/2013
Onaylanan Bütçe:	43520.0
Harcanan Bütçe:	33536.89
Öz:	Bu projede, esnek imalat sistemlerinde karşılaşılan birbirleriyle ilişkili önemli problemler üzerinde çalışılmıştır. Birincisi, işlerin ve işlerin gerçekleştirilmesi için gereken makine uçlarının makinelere atanması ve çizelgelenmesi problemidir. Amaç en son işin bitirilme zamanının en aza indirilmesidir. Bu problemde makine ucu değişimi göz ardı edilmiştir. İkinci problem olan işlerin gruplanması probleminde ise işler makinelerin makine ucu haznesi kapasiteleri göz önünde bulundurularak gruplara ayrılmıştır. Bu problemde gruplar arasında makine ucu değişimi yapıldığı varsayılmaktadır. Makine ucu değişimi uzun bir süre gerektirdiği ve işlerin gerçekleştirilmesinde önemli bir yer tuttuğu için, grup sayısı ve dolayısıyla makine ucu değişimi sayısı en aza indirilmeye çalışılmıştır. Son olarak, işlerin ve makine uçlarının makinelere atanması, işlerin çizelgelenmesi ve makine uçlarının değiştirilmesi problemleri birlikte çalışılmıştır. Bu problemde de amaç, son işin tamamlanma zamanının en aza indirilmesidir. Yukarıda tanımlanan problemlerin her biri polinom zamanlı olmayan-zor problemlerdir. Birinci problem az sayıda araştırmacı tarafından problemi kolaylaştıran varsayımlarla çalışılmıştır. İkinci problem pek çok araştırmacı tarafından çalşılmış bir problemdir. Bizim bu projede bu problemi ele alma nedenimiz, birinci problemle bir arada düşünüldüğünde üçüncü probleme ulaşılmasıdır. Son problem ise literatürde daha önce çalışılmamıştır. Bu proje ile literatürdeki bu boşluğu doldurmak amaçlanmaktadır. Her problem için matematiksel model yazılmış, kesin veya sezgisel çözüm yöntemleri geliştirilmiş ve bu yöntemler örnek problemler üzerinde denenmiştir. Deney sonuçları, geliştirilen yöntemlerin kısa sürelerde oldukça iyi sonuçlar verdiğini göstermektedir.
Anahtar Kelimeler:	Esnek İmalat Sistemleri, Çizelgeleme, Makine Ucu Değişimi
Fikri Ürün Bildirim Formu Sunuldu Mu?:	Hayır
Projeden Yapılan Yayınlar:	1- Esnek İmalat Sistemlerinde Çizelgeleme ve Makine Ucu Atama Problemi için Kısıt Programlama Yaklaşımı (Bildiri), 2- Mathematical modelling and constraint programming approaches for operation assignment and tool loading problems in flexible manufacturing systems (Bildiri), 3- Scheduling with Tool Switching in Flexible Manufacturing Systems (Bildiri)