# Synthesizing Filtering Algorithms for Global Chance-Constraints*

Brahim Hnich[1], Roberto Rossi[2], S. Armagan Tarim[3], and Steven Prestwich[4]

[1] Faculty of Computer Science, Izmir University of Economics, Turkey
brahim.hnich@ieu.edu.tr
[2] Logistics, Decision and Information Sciences, Wageningen UR, The Netherlands
roberto.rossi@wur.nl
[3] Operations Management Division, Nottingham University Business School, UK
armtar@yahoo.com
[4] Cork Constraint Computation Centre, University College Cork, Ireland
s.prestwich@4c.ucc.ie

**Abstract.** Stochastic Constraint Satisfaction Problems (SCSPs) are a powerful modeling framework for problems under uncertainty. To solve them is a P-Space task. The only solution approach to date compiles down SCSPs into classical CSPs. This allows the reuse of classical constraint solvers to solve SCSPs, but at the cost of increased space requirements and weak constraint propagation. This paper tries to overcome some of these drawbacks by automatically synthesizing filtering algorithms for global chance-constraints. These filtering algorithms are parameterized by propagators for the deterministic version of the chance-constraints. This approach allows the reuse of existing propagators in current constraint solvers and it enhances constraint propagation. Experiments show the benefits of this novel approach.

## 1 Introduction

Stochastic Constraint Satisfaction Problems (SCSPs) are a powerful modeling framework for problems under uncertainty. SCSPs were first introduced in [10] and further extended in [9] to permit multiple chance-constraints and a range of different objectives in order to model combinatorial problems under uncertainty. SCSP is a PSPACE-complete problem [10]. The approach in [9] compiles down SCSPs into deterministic equivalent CSPs. This makes it possible to reuse existing solvers, but at the cost of increased space requirements and of hindering constraint propagation. In this paper we overcome some of these drawbacks by automatically synthesizing filtering algorithms for global chance-constraints. These filtering algorithms are built around propagators for the deterministic version of the chance-constraints. Like the approach in [9], our approach reuses the propagators already available for classical CSPs. But, unlike [9], our approach uses fewer decision variables and strengthens constraint propagation. Our results

* Brahim Hnich is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027.

I.P. Gent (Ed.): CP 2009, LNCS 5732, pp. 439–453, 2009.

show that our approach is superior to the one in [9], since it achieves stronger pruning and therefore it proves to be more efficient in terms of run time and explored nodes.

The paper is structured as follows: in Section 2 we provide the relevant formal background; in Section 3 we discuss the structure of a SCSP solution; in Section 4 we describe the state-of-the-art approach to SCSPs; in Section 5 we discuss our novel approach; in Section 6 we present our computational experience; in Section 7 we provide a brief literature review; finally, in Section 8 we draw conclusions.

## 2 Formal Background

A Constraint Satisfaction Problem (CSP) consists of a set of variables, each with a finite domain of values, and a set of constraints specifying allowed combinations of values for some variables. A *solution* to a CSP is an assignment of variables to values in their respective domains such that all of the constraints are satisfied. Constraint solvers typically explore partial assignments enforcing a local consistency property. A constraint $c$ is *generalized arc consistent* ($GAC$) iff when a variable is assigned any of the values in its domain, there exist compatible values in the domains of all the other variables of $c$. In order to enforce a local consistency property on a constraint $c$ during search, we employ filtering algorithms that remove inconsistent values from the domains of the variables of $c$. These filtering algorithms are repeatedly called until no more values are pruned. This process is called *constraint propagation*.

An $m$-stage SCSP is defined as a 7-tuple $\langle V, S, D, P, C, \theta, L\rangle$, where $V$ is a set of decision variables and $S$ is a set of stochastic variables, $D$ is a function mapping each element of $V$ and each element of $S$ to a domain of potential values. In what follows, we assume that both decision and stochastic variable domains are finite. $P$ is a function mapping each element of $S$ to a probability distribution for its associated domain. $C$ is a set of chance-constraints over a non-empty subset of decision variables and a subset of stochastic variables. $\theta$ is a function mapping each chance-constraint $h \in C$ to $\theta_h$ which is a threshold value in the interval $(0, 1]$. $L = [\langle V_1, S_1\rangle, \ldots, \langle V_i, S_i\rangle, \ldots, \langle V_m, S_m\rangle]$ is a list of *decision stages* such that each $V_i \subseteq V$, each $S_i \subseteq S$, the $V_i$ form a partition of $V$, and the $S_i$ form a partition of $S$.

The solution of an $m$-stage SCSP is, in general, represented by means of a *policy tree* [9]. The arcs in such a policy tree represent values observed for stochastic variables whereas nodes at each level represent the decisions associated with the different stages. We call the policy tree of an $m$-stage SCSP that is a solution a *satisfying policy tree*.

## 3 Satisfying Policy Trees

In order to simplify the presentation, we assume without loss of generality, that each $V_i = \{x_i\}$ and each $S_i = \{s_i\}$ are singleton sets. All the results can be easily extended in order to consider $|V_i| > 1$ and $|S_i| > 1$. In fact, if $S_i$ comprises more