Constraints (2018) 23:196–209
https://doi.org/10.1007/s10601-018-9283-8

Modeling uncertainties with chance constraints

Imen Zghidi[1] · **Brahim Hnich**[2] · **Abdelwaheb Rebaï**[1]

Published online: 3 April 2018

Abstract Chance constraints are a major modeling tool for problems under uncertainty. We summarize the basic modeling ingredients of uncertain combinatorial problems and show how the Stochastic Constraint Satisfaction Problems formalism is able to support high-level declarative constructs that allow for ease of modeling of such problems in general. Then, we outline the different propagation methods for chance constraints. Finally, we identify some modeling subtleties that might arise when modeling with chance constraints.

Keywords Uncertainty · Chance constraints · Stochastic constraint satisfaction problems

1 Introduction

In most industrial contexts, decision makers cannot be certain of the future behavior of factors that will affect the outcome resulting from various options under consideration. Decision making under uncertainty is thus characterized by the necessity of making decisions based on incomplete information and without knowing their full effects until a later stage in the future. But, as values of more and more unknowns are unveiled, there may be opportunities for corrective recourse actions. Such problems appear in various fields

This article belongs to the Topical Collection: *20th Anniversary Issue*

✉ Imen Zghidi
zghidi.imen@gmail.com

Brahim Hnich
hnich.brahim@gmail.com

Abdelwaheb Rebaï
Abdelwaheb.Rebai@fsegs.rnu.tn

[1] MODILS Research Lab, FSEGS, Sfax University, Sfax, Tunisia

[2] CES, ENIS, Sfax University, Sfax, Tunisia

of application and present many interesting challenges from both a modeling and solving perspective.

The uncertainties in a problem have to be represented in a way that can properly be exploited when making decisions.

There have been many attempts to incorporate uncertainty within the Constraint Satisfaction Problem (CSP) formalism (e.g., [1, 5, 8, 9, 19]). But, in this paper, we focus on the Stochastic Constraint Satisfaction approach inspired by stochastic programming [4, 11] and chance-constrained programming [6].

Stochastic Constraint Satisfaction Problems (SCSPs) provide a powerful modeling framework for problems in which one is required to make decisions under uncertainty. In these stochastic problems, the uncertainty is modeled by using *random variables* to capture uncontrollable factors such as the customer demands, the processing times of machines, house prices, etc. These discrete random variables can take a set of possible different values, each with an associated probability. They are useful to model factors that fall outside the control of the decision maker who only knows the probability distribution function of these random variables. Such a distribution may be learned from statistical data or simply an educated guess. There are controllable variables on which one can decide, named *decision variables* which model the set of possible choices for the decisions. Finally, such problems comprise *chance constraints* of the form: $pr\{C\} \geq \beta$ where C is a *stochastic constraint* expressing the relationship between random and decision variables and $pr\{C\}$ is the *satisfaction probability* of this constraint that should be satisfied within a given threshold β since requiring C to hold almost surely in the presence of random variables is almost impossible.

Chance constraints are a key concept within SCSPs. This paper is not a survey on solution methods of SCSPs in general. Rather, the focus is on the role of chance constraints in the modeling of uncertainties and the various approaches that extend constraint programming to support and reason about them. A thorough review of the different solution methods of SCSPs is found in [9].

The rest of the paper is organized as follows. In Section 2, we study the basic modeling ingredients of uncertain problems and shown how the SCSP formalism is able to support the declarative of such problems in general in Section 3. Next, we outline the various propagation methods in Section 4. In Section 5, we identify some modeling subtleties arising when modeling uncertain problems with chance constraints and we conclude the paper in Section 6.

2 Modeling ingredients of uncertain problems

In this section, we present the modeling ingredients of uncertain problems.

2.1 Random variables

The uncertain elements in an uncertain problem can be modeled by using *random variables* to capture uncontrollable factors such as the customer demands, the processing times of machines, house prices, etc. These random variables can take a set of possible different values, each with an associated probability; they can be used to model factors that fall outside the control of the decision maker who only knows the probability distribution function of these random variables. Such a distribution may be learned from statistical data or simply an educated guess. The random variables can either be *discrete* or *continuous*.

There are also controllable variables on which one can decide, named *decision variables*; they allow to model the set of possible choices for the decisions to be made. It should be noted that we ”decide” what values the decision variables take while we can only ”observe” the outcome of random variables.

2.2 Chance constraint types

In addition, such problems may comprise *stochastic constraints* of the form $C_s(x, r)$ where x is a subset of decision variables and r is a non-empty subset of random variables. A stochastic constraint expresses the relationship between random and decision variables. An assignment $\bar{x}$ satisfies C_s iff for every observed values $\bar{r}$ of r, $C_s(\bar{x}, \bar{r})$ holds.

Stochastic constraints impose that a feasible decision must hold in whatever way the uncertainty may unfold. This might be a conservative approach and hinders finding feasible solutions in practice. One way to remedy this, for a given assignment $\bar{x}$, is to require that $C_s(\bar{x}, r)$ hold most of the time, rather than every time. That is, one would insist on decisions guaranteeing feasibility as much as possible. In such a case, we may tolerate the fact that for certain observed values $\bar{r}$ of r, $C_s(\bar{x}, \bar{r})$ does not hold but insist that the sum of the probabilities of cases where $C_s(\bar{x}, \bar{r})$ holds should be above a given threshold $\beta \in (0, 1]$. This can be quantified in terms of the *satisfaction probability* of $C_s(\bar{x}, r)$ as:

$$pr\{C_s(\bar{x}, r)\} = \sum_{\bar{r} \in support(r) | C_s(\bar{x}, \bar{r}) holds} pr\{\bar{r}\}$$

Requiring this probability to be equal to 1 can be modeled using stochastic constraints. But if, instead, we require that the satisfaction probability is above a given threshold β, we have *chance constraints* which are expressed as:

$$pr\{C_s(x, r)\} \geq \beta$$

Note that when $C_s(x, r)$ is a single constraint, we have what we call a *single chance constraint* or simply a chance constraint. But, when $C_s(x, r)$ is a conjunction of other stochastic constraints, we have what is referred to as a *joint chance constraint*.

2.3 Decision stages

Furthermore, the timing of observations versus decisions is specific to every uncertain problem but we need to effectively model the evolution of such information, i.e., the sequences of decisions and observations.

The question of which one comes first, the fixing of the decision or the making of an observation that unveils the uncertainty, is crucial. Both cases can arise in practice and are problem-specific, and in fact in many uncertain problems partial decisions and partial observations are made in *stages* and are interleaved with one another.

Despite the fact that we are dealing with many future uncertain elements, the emphasis should be on enhancing the decision that must be made in the present. Furthermore, decisions should be taken while considering not only the present knowledge but also by incorporating the new knowledge due to future observations in subsequent stages. A decision cannot properly be made in the present without considering the opportunities for correction at later stages. We refer to decisions made at later stages as *recourse decisions*. While taking recourse decisions, we should respond to observations of random variables that have become available since the initial decision.

It is assumed that the probability structure is unaffected by any of the decisions that are taken. That is the distributions of the random variables are fixed and unaffected by the decisions taken. The random variables, however, are not required to be independent of each other in the statistical sense.

Given a set V of decision variables and a set R of random variables, the *process* of alternating between decisions and observations can be stated as follows:

Time		
t_0	$V_0 \subseteq V$	set of the values chosen for the variables in V_0
t_1	$R_1 \subseteq R$	observation of the random variables in R_1
t_2	$V_1 \subseteq V$	set of the recourse values chosen for the variables in V_1
...		
t_{2i-1}	$R_i \subseteq R$	observation of the random variables in R_i
t_{2i}	$V_i \subseteq V$	set of the recourse values chosen for the variables in V_i
...		
t_{2k-1}	$R_k \subseteq R$	observation of the random variables in R_k
t_{2k}	$V_k \subseteq V$	set of the recourse values chosen for the variables in V_k

where the V_i's form a partition of V, and the R_i's form a partition of R.

At the beginning (at time t_0), we make some initial decisions followed by k stages. Each stage i is composed of some observations (at time t_{2i-1}) followed by some recourse actions (at time t_{2i}) that respond to the previous observations made at earlier stages. The number of stages k is known as the *finite horizon* of the uncertain problem. It is crucial to understand the evolution of information within this process. When we take the initial decisions V_0, nothing about the random variables in our process has been observed yet. However, when we make recourse decisions at stage i, we should make use of the observations of all random variables made at earlier stages $(R_1, \ldots, R_i)$. At this point in time, we have observed $R_1, \ldots, R_i$ and these become fixed data points whereas the corresponding distribution for $(R_{i+1}, \ldots, R_k)$ becomes the *conditional probability* distribution given observations of $R_1, \ldots,$ and R_i. Indeed, when making recourse decisions V_i, we have additional information available to us in the form of observations of previous random variables that are not longer random variables, but instead have fixed values.

Thus, if at the present (time t_0) we wish to decide on some recourse decisions at a later stage V_i, we need to compute a range of possible decisions, one for each possible outcome of the random variables $R_1, \ldots,$ and R_i. It is simply wrong to think of the values of the recourse decisions as a single value for each variable in V_i, instead we need to compute in advance exactly how we would respond to each outcome of the previous observations. The recourse decisions in stage i must respond to information that becomes available between the initial stage and stage i. Therefore, these recourse decisions have to be modeled as a function of that information instead of as a unique set of decisions to V_i.

With such a structure we cannot properly make decisions at a given stage without taking into account the recourse opportunities we might have at later stages. Recourse decisions are *non-anticipative* because they react to the past, but they cannot be based on knowing the future before it happens [4].

3 Stochastic constraint satisfaction problems

In this section, we present the SCSP formalism and detail its expressive power in modeling uncertain problems and identify utilizations of chance constraints in modeling uncertainties in different situations.

3.1 Modeling support for uncertain problems

SCSPs [10, 21, 22] provide a powerful modeling framework for problems in which one is required to make decisions under uncertainty. An m-stage SCSP is a 7-tuple $\langle V, R, D, P, C, \beta, L\rangle$, where V is a set of decision variables, R is a set of random variables, and D is a function mapping each element of V (respectively, R) to a domain (respectively, support) of potential values. In classical SCSPs both decision variable domains and random variable supports are assumed to be finite. P is a function mapping each element of S to a probability distribution for its associated support. C is a set of chance-constraints over a non-empty subset of decision variables and a subset of random variables. β is a function mapping each chance-constraint $h \in C$ to β_h which is a threshold value in the interval (0, 1]. $L = [\langle V_1, R_1\rangle, \ldots, \langle V_i, R_i\rangle, \ldots, \langle V_m, R_m\rangle]$ is a list of *decision stages* such that each V_i is a subset of V, each R_i is a subset of R, the V_i's form a partition of V, and the R_i's form a partition of R.

SCSPs are indeed expressive enough to allow the *declarative* modeling of the different requirements that might exist within an uncertain problem:

Decision and random variables: The decision and random variables are first-class citizens within the SCSP formalism. Decision variables represent the decisions one wishes to take whereas random variables model factors that fall outside the control of the decision maker who only knows the probability distribution function of these random variables.

Stochastic constraints: to express a stochastic constraint $C_s(x, r)$ where x is a subset of the decision variables and r is a non-empty subset of the random variables that should hold regardless of how the uncertainty will unfold, one should state the following:

$$pr\{C_s(x, r)\} \geq 1$$

Single chance constraints: to express a single chance constraint $C(x, r)$ where x is a subset of the decision variables and r is a non-empty subset of the random variables such that the satisfaction probability is beyond a given threshold β, one should state the following:

$$pr\{C_s(x, r)\} \geq \beta$$

Joint chance constraints: to express a joint chance constraint over $C_s^1(x_1, r_1)$, ..., $C_s^m(x_m, r_m)$ that should hold with a satisfaction probability beyond a given threshold β, one should state the following:

$$pr\{C_s^1(x_1, r_1) \wedge \ldots \wedge C_m(x_m, r_m)\} \geq \beta$$

Stages: the decision stages of SCSPs neatly model the problem-specific sequences of decisions and observations allowing for a declarative description of such evolution of information. It also allows for specifying which decisions are recourse decisions and at what stage.

3.2 Policy trees

The only remaining part is how to represent a solution to a SCSP that incorporates the process of alternating between decisions and observations while satisfying all chance constraints. To solve an m-stage SCSP, an assignment to the variables in V_1 must be found such that, given random values for R_1, assignments can be found for V_2 such that, given random values for R_2, ..., assignments can be found for V_m so that, given random values for R_m, the chance constraints are satisfied in the specified fraction of all possible scenarios. Under the assumption that random variable supports are finite, the solution of an m-stage SCSP is, in general, represented by means of a *policy tree* [21]. The arcs in such a policy tree represent values observed for random variables whereas nodes at each level represent the (recourse) decisions associated with the different stages. Each level of the tree represents a stage. The arc in a path from the root to a leaf node is a *scenario* that represents a specific scenario of how the future might unfold. We call the policy tree of an m-stage SCSP that is a solution a *satisfying policy tree*. In a satisfying policy tree, each chance constraint is satisfied in a number of scenarios. The sum of the probabilities of those scenarios is above the satisfaction probability threshold value of that chance constraint. Solving an m-stage SCSP is a computationally challenging task and it is in PSPACE [22], in general.

An m-stage SCOP is an m-stage SCSP with an additional objective function f over a non-empty subset of decision variables and a subset of random variables. An *optimal* solution to a maximization SCOP is a satisfying policy tree with a maximum expected value for f.

Example 1 As an example of a global chance constraint, let us consider an instance of the global chance constraint $sorted_<$ over decision variables $x \in \{1\}$, $y \in \{3\}$, and $z \in \{5\}$ as well as random variables $r_1 \in \{1(\frac{1}{3}), 2(\frac{1}{3}), 3(\frac{1}{3})\}$ and $r_2 \in \{3(\frac{1}{3}), 4(\frac{1}{3}), 5(\frac{1}{3})\}$ such that:

$$pr\{sorted_<(x, r_1, y, r_2, z)\} \geq \beta$$

where $\beta = \frac{1}{9}$

Assignment $\langle x = 1, y = 3, z = 5\rangle$ has a satisfying policy tree as shown in Fig. 1.

3.3 Different utilizations of chance constraints

We distinguish between different possible utilizations of the various types of chance constraints depending on the uncertainty context and modeling needs:

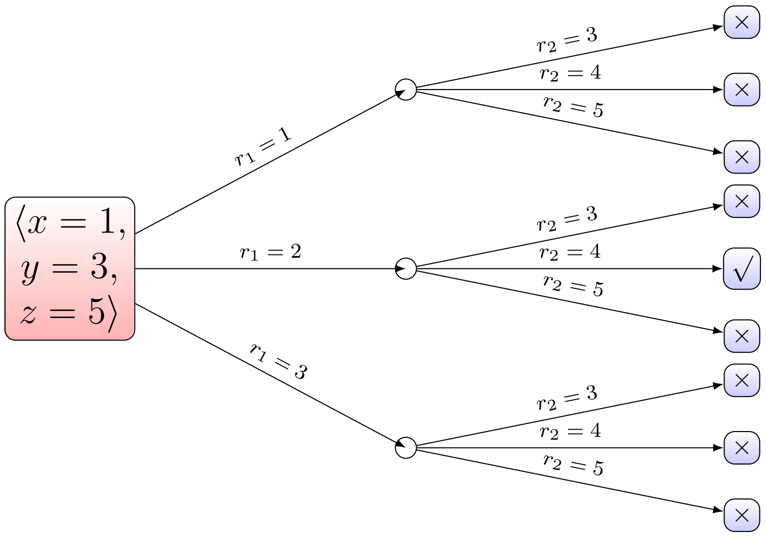

Fig. 1 Policy tree for the assignment $\langle x = 1, y = 3, z = 5\rangle$ wrt to chance constraint $sorted_<(x, r_1, y, r_2, z)$ for which the satisfaction probability is equal to $\frac{1}{9}$

- Finding feasible decisions prior to the observation of random parameters is very challenging. It is almost impossible to definitely exclude later constraint violation caused by realizations of random variables. For this reason, such constraint violation can be balanced afterwards by some recourse decisions taken in a second stage. For instance, making a recourse decision by buying a resource is an option for companies as a recourse to an excepted shortage. So in a sense, for some uncertain problems, one can deal with constraint violation by making use of the recourse decisions as long as it can be done and makes sense. Such a class of problems can then be effectively modeled as *m-stage SCSPs* that only comprise *stochastic constraints*.
- In some situations, however, the option of a recourse simply does not exist or cannot be easily modeled as such in any reasonable way. For example in an inventory problem with uncertain demands, one wishes to guarantee a certain level of service level by meeting a certain percentage of customer demands. Such situations cannot be modeled in a sensible way by using recourse actions. Instead, one would rather insist on decisions that guarantee the satisfaction probability above a certain threshold, i.e., as *chance constraints*. Such a class of problems can be effectively modeled as *single-stage SCSPs* that only comprise *chance constraints* in which one wishes to make decisions in an uncertain environment and wants to guarantee a certain level of stability despite the uncertainties.
- Obviously, there are situations which require a m-stage SCSP with stochastic and chance constraints such as the inventory models in [23].

4 Propagation of chance constraints

In constraint programming, the constraints are not restricted to a particular form but they can be linear, nonlinear, global, logical, or even symbolic. Hence, the forms of chance constraints support such a diversity as well. This expressive power, however, comes at cost since solving an m-stage SCSP is a computationally challenging task and it is in PSPACE in general [22]. However, as we show next, even if we restrict the form of chance constraints, they still remain difficult to reason about.

Consider the following linear chance constraint [18]:

$$pr\{\xi x_1 + x_2 \geq 7\} \geq \beta$$

where decision variables $x_1, x_2 \in \Re$, $\beta \in (0, 1]$, and ξ is uniformly distributed in [0, 1] with cumulative distribution:

$$F(t) = \begin{cases} 0 & \text{if } t \in (-\infty, 0) \\ t & \text{if } t \in [0, 1] \\ 1 & \text{otherwise} \end{cases}$$

let us denote by $S(\beta)$ be the set of feasible solutions, i.e.,

$$S(\beta) = \{(x_1, x_2) \in \Re^2 | pr\{\xi x_1 + x_2 \geq 7\} \geq \beta)\}$$

As shown in [18], $S(0.3)$ is a convex set whereas $S(0.7)$ is not. The feasible region defined by a chance constraint generally is not convex even if $\xi x_1 + x_2 \geq 7$ is convex for every possible realization of ξ. This simple example demonstrates that convexity of the feasible set of chance constraints is not guaranteed in general even when we have linear functions. Indeed, the convexity of chance constraints does not only depend on the convexity properties of the constraint function h but also of the distribution of the random parameters ζ. Only under very strict assumptions is the convexity guaranteed [18].

Constraint propagation techniques are inference methods that help reduce the original CSP to another which is smaller in size by pruning inconsistent values inferred by enforcing a local consistency property on the individual constraints. In what follows, we review the different approaches of propagating chance constraints.

4.1 Reformulation-based approaches

Since reasoning with chance constraints is computationally a challenging task, a naive approach for stochastic constraints is to get rid off the uncertainty by substituting each random variable r with some estimate $\tilde{r}$. The resulting is a deterministic constraint that ignores all other possible choices for r. Such a reformulation might be attractive from a computational perspective but is not effective in dealing with uncertainty. By considering only one choice for the random variable and ignoring the others, hedging of decisions becomes impossible. Hedging decisions require considering various possible outcomes of the uncertain elements in order to be able to protect the decision maker from more losses than she can afford by using only one estimate value of the random variables. That is, we are no more able come up with hedging decisions that balance the risks and reflect the realities by considering only one estimate value for r no matter how well selected the estimate might be.

A better way to allow for hedging is to reason about a chance constraint by means of *scenarios*. Each scenario is a particular representation of how the future might unfold with a certain probability. Each scenario is a possible realization of the random variables whose probability is the product of the respective probabilities of the random variables' values. We denote by Ω_h the set of all possible scenarios of a chance constraint $h : pr\{C(x, r)\} \geq \beta$. The probability of scenarios $\check{r} \in \Omega_h$ is denoted by $pr(\check{r})$. Note that the number of scenarios is exponential in the number of random variables. When we have k random variables, each with only 2 possible values, the number of scenarios is 2^k.

An equivalent scenario-based reformulation of chance constraint h that rewrites h as a conjunction of deterministic constraints is obtained as follows [20].

Step 1: Introduce for each scenario $\check{r} \in \Omega_h$ an auxiliary Boolean variable $B_{\check{r}}$.
Step 2: Introduce for each scenario $\check{r} \in \Omega_h$ the deterministic logical constraint:

$$C(x, \check{r}) \leftrightarrow B_{\check{r}}$$

Step 3: Introduce the following sum constraint:

$$\sum_{\check{r} \in \Omega_h} pr(\check{r}) \geq \beta$$

Such a scenario-based reformulation can provide an "approximate" propagation method for chance constraints in general. However, it may suffer from the following potential drawbacks:

– The number of auxiliary Boolean variables and logical constraints is exponential in the number of random variables which may lead to an inefficient propagation method. To avoid the exponential complexity, one could sample a limited number of scenarios. Sampling few scenarios, however, may lead to a bad approximation of the satisfaction probability unless the scenarios are sampled from the joint probability distribution;
– Most constraint solvers implement constraint reification for only some non-global constraints. Thus, when $C(x, \check{r})$ is a global constraint itself, we are forced to use a

decomposition instead which makes the logical constraints even weaker in terms of pruning; and

– When sampling is used, [24] introduced the notion of *(α, ϑ)-consistency* inspired by (α, ϑ)-solutions proposed in [17]. Such a notion formalizes the level of consistency achieved when sampling is used.

4.2 Consistency-based approaches

The authors in [10] extend the notion of Generalized Arc Consistency (GAC) for chance constraints. Since chance constraints arise more naturally in single-stage SCSPs, we state a simplified definition of GAC for single-stage chance constraints. The definition that takes into consideration multiple-stages is given in [10].

In [2], the authors studied the computational complexity of reasoning with global constraints and identified a number of crucial questions. At the core of all generic arc consistency algorithms is the question which is generally asked for all values one by one [2]:

Instance A constraint C, a domain D on $var(C)$ (i.e., the variables in the scope of constraint C), and a value v for variable $x \in var(C)$

Question Does value v for x have a support on C in D? I.e., does there exist an assignment that satisfies C in which $x = v$

We establish stochastic support for a given value for a *chance constraint* as follows:

Definition 1 Let a chance constraint $pr\{C_s(x, r)\} \geq \beta$ be given. A value for v in the domain of $x_i \in var(C_s)$ is GAC iff there exists an assignment $\bar{x}$ in which $x_i = v$ and

$$\sum_{\bar{r} \in support(r) | C_s(\bar{x}, \bar{r}) holds} pr(\bar{r}) \geq \beta$$

In other words, for single-stage chance constraints, a value v in the domain of $x_i \in var(C_s)$ is GAC iff there exits an assignment $\bar{x}$ in which $x_i = v$ which is a satisfying policy tree.

Definition 2 A chance constraint $pr\{C_s(x, r)\} \geq \beta$ is GAC iff every value in the domain of every variable in $var(C_s)$ is GAC.

In [10] the authors show that GAC on a chance constraint can be intractable even when maintaining GAC on the corresponding deterministic version of that constraint is tractable. In particular, the authors in [10] show that maintaining GAC on the global *alldifferent* chance constraint is NP-hard while maintaining GAC on the deterministic *alldifferent* constraint is polynomial [12].

There exist different approaches to build propagation algorithms for single-stage chance constraints in the literature:

Generic: The authors in [10] propose novel approximate generic propagation algorithms for *any* single-stage chance constraint by synthesizing filtering algorithms for chance-constraints by reusing the propagator of the deterministic version of the chance constraint. The generic filtering algorithm has been extended in two ways in order to obtain two incremental variations: a lightweight version as well as a memory-intensive one. Experiments on three benchmark problems (two stochastic constraint satisfaction problems and a stochastic constraint optimization one) have shown that the filtering algorithms in [10] outperform the reformulation-based approach in [21] in terms of pruning and runtime. Overall, the approach in [10] has the advantage of being generic and is superior to the reformulation approach in [21] but still has a huge space and time complexity.

Specialized Propagators: The works in [13–16] propose ad-hoc filtering strategies for handling specific chance-constraints.

In [7], an approximate flow-based filtering algorithm – based on specific bounding mechanisms computed by means of minimum-cost network flows– has been developed for the intractable single-stage weighted *alldifferent* chance constraint. The approach in [7] differs from [10] in that it does not explicitly represent the policy tree. It instead builds it during search for the particular chance constraint considered. By exploring the combinatorial structure of the weighted *alldifferent* chance constraint, the authors managed to generate sufficiently small sub-policy trees. Thus, achieving a more scalable propagator wrt approach in [10] or [21] but with the drawback of being less effective in terms of pruning.

Note that the above consistency definition for chance constraints assumes that all the random variables in R have *discrete finite* support. If at least one random variable in R has an *infinite* support (e.g. infinite discrete support and/or infinite continuous support), then for any assignment $\bar{x}$ the corresponding policy tree would compromise an infinite number of arcs. Indeed, the sum that tests whether a value v is GAC or not would be an infinite sum. Inspired by the concept of (α, ϑ)-solutions proposed in [17], the work in [24] introduced the notion of *(α, ϑ)-consistency* for chance constraints of single-stage SCSPs in which at least one random variable has an *infinite support*. In [24], the problem of enforcing (α, ϑ)-consistency is reformulated as a statistical inference problem about an unknown parameter. Two methods based on sampling for enforcing (α, ϑ)-consistency have been proposed: the first is based on confidence intervals whereas the second reformulates the problem using composite hypothesis testing. These methods have been validated empirically in [24]. However, none of these methods has been applied to a benchmark SCSP.

5 Discussion

Chance constraints provide a powerful modeling framework for expressing various relationships between random and decision variables. However, there are some subtleties worth noting when modeling with chance constraints that have not been addressed yet in the literature.

5.1 Choice of proper satisfaction probability thresholds

For many applications, single-stage SCSPs that only comprise chance constraints may seem appealing from a modeling perspective. However, there are some difficulties worth noting. Firstly, given a chance constraint $pr\{C_s\} \geq \beta$, an important question is what constitutes a good value for the satisfaction probability β. It should get closer to the value 1 as the seriousness of violating the constraint increases but there is no guidance or method of quantitatively assess the ”seriousness” of constraint violation.

Secondly, the issue of the interaction of multiple chance constraints arises since we are dealing with different satisfaction probabilities. Suppose we have two chance constraints h_1 and h_2:

$$h_1 : pr\{C_s^1\} \geq \beta_1 \wedge h_2 : pr\{C_s^2\} \geq \beta_2$$

We have two subtleties to pay attention to: (1) when a violation of either chance constraint would mean the violation of h_1 and h_2?; and (2) shouldn't we rather state the conjunction of two separate chance constraint as a *joint chance constraint* instead, as follows?

$$h : pr\{C_s^1 \wedge C_s^2\} \geq \beta$$

However, when using such a formulation, what would be an appropriate value of β and how to differentiate between the "seriousness" of constraint violation for each chance constraint?

5.2 Decomposition of chance constraints

Another modeling subtlety arises when we wish to decompose a global chance constraint. In most cases, a deterministic global constraint can be decomposed into a conjunction of more primitive constraints that are semantically equivalent. For instance, the *alldifferent* constraint can be decomposed into a clique of binary not-equals constraints. These two formulations are semantically equivalent but wrt to propagation or complexity one might be more effective than the other [3].

Let C be an n-ary deterministic global constraint that is semantically decomposable into:

$$C \Leftrightarrow \bigwedge_{i=1}^{m} C_i$$

Now, suppose that C is a chance constraint instead over a set of decision and discrete random variables. Then, the global chance constraint can be stated as follows:

$$pr\{C\} \geq \beta \Leftrightarrow pr\{\bigwedge_{i=1}^{m} C_i\} \geq \beta$$

But this reformulation just converted the *single* chance constraint into a *joint* chance constraint!

Furthermore, when in the deterministic case, we have:

$$C \Leftrightarrow \bigwedge_{i=1}^{m} C_i$$

in general and when $\beta < 1$, $pr\{C\} \geq \beta$ is *not* semantically equivalent to

$$\bigwedge_{i=1}^{m} pr\{C_i\} \geq \beta$$

Example 2 Consider the following instance of the *alldifferent* global chance constraint:

$$pr\{alldifferent(x, y, r)\} \geq \beta = \frac{1}{2}$$

where decision variable $x \in \{1\}$, decision variable $y \in \{2\}$, and discrete random variable $r \in \{1(\frac{1}{3}), 2(\frac{1}{3}), 3(\frac{1}{3})\}$.

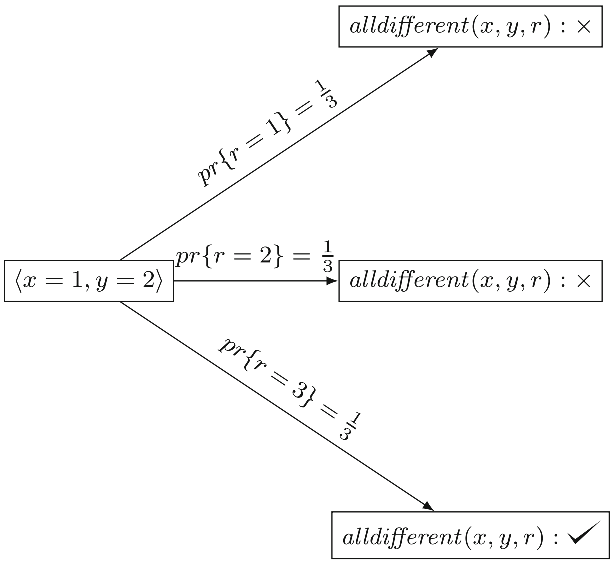
$alldifferent(x, y, r) : \times$
$pr\{r = 1\} = \frac{1}{3}$
$\langle x = 1, y = 2\rangle$
$pr\{r = 2\} = \frac{1}{3}$
$alldifferent(x, y, r) : \times$
$pr\{r = 3\} = \frac{1}{3}$
$alldifferent(x, y, r) : \checkmark$

Fig. 2 Policy tree for the assignment $\langle x = 1, y = 2\rangle$ wrt to chance constraint $pr\{x \neq r \wedge y \neq r\}$ which is equal to $\frac{1}{3}$

As illustrated in Fig. 2, the global chance constraint is unsatisfiable since:

$$pr\{alldifferent(x, y, r)\} = \frac{1}{3} < \frac{1}{2}$$

whereas, as shown in Fig. 3, the decomposition is satisfiable since $pr\{x \neq r\} = pr\{y \neq r\} = \frac{2}{3}$ which is greater than $\frac{1}{2}$ and $x \neq y$ is true. Note that this anomaly cannot be resolved by sampling from the joint distribution of the random variables in general since in this example we only have one random variable.

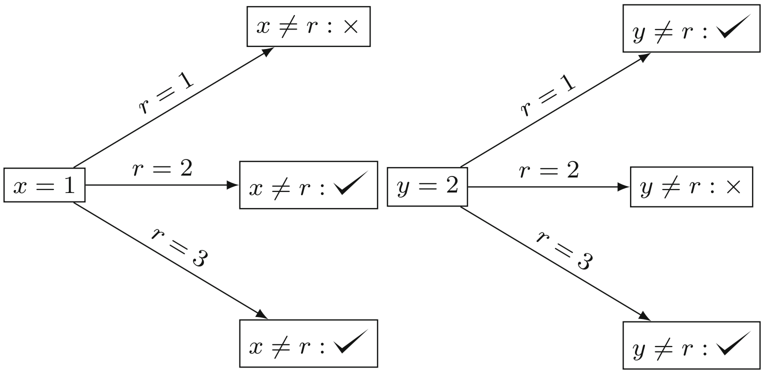

Fig. 3 (Left) Policy tree for the assignment $x = 1$ wrt to chance constraint $pr\{x \neq r\}$ which is equal to $\frac{2}{3}$. (Right) Policy tree for the assignment $y = 2$ wrt to chance constraint $pr\{y \neq r\}$ which is equal to $\frac{2}{3}$

6 Conclusion

Chance constraints are a major modeling tool for combinatorial problems under uncertainty. Within the constraint programming literature, there exits no survey paper that deals specifically with chance constraints. In this paper, we summarize the basic modeling ingredients of uncertain combinatorial problems and shown how the SCSP formalism is able to support high-level declarative constructs that allow for ease of modeling of uncertain problems in general. Finally, we described the various types of chance constraints (stochastic, single, and joint) and their applicability in various modeling situations. Furthermore, we outlined the different propagation methods of such constraints and discussed some modeling subtleties that arise when modeling with chance constraints.

References

1. Balafoutis, T., & Stergiou, K. (2006). Algorithms for stochastic csps. In *Principles and Practice of Constraint Programming, CP 2006* (pp. 44–58): Proceedings.
2. Bessiere, C., Hebrard, E., Hnich, B., Walsh, T. (2007). The complexity of reasoning with global constraints. *Constraints*, *12*(2), 239–259.
3. Bessière, C., & Van Hentenryck, P. (2003). To be or not to be ... a global constraint. In *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003* (pp. 789–794). Kinsale: Proceedings.
4. Birge, J.R., & Louveaux, F. (1997). *Introduction to Stochastic Programming*. New York: Springer Verlag.
5. Brown, K.N., & Miguel, I. (2006). Uncertainty and change. In Rossi, F., van Beek, P., Walsh, T. (Eds.) *Handbook of Constraint Programming, chapter 21*: Elsevier.
6. Charnes, A., & Cooper, W.W. (1959). Chance-constrainted programming. *Management Science*, *6*(1), 73–79.
7. Cire, A.A., Coban, E., van Hoeve, W.-J. (2012). *Flow-Based Combinatorial Chance Constraints*, (pp. 129–145). Berlin: Springer Berlin Heidelberg.
8. Fargier, H., Lang, J., Martin-Clouaire, R., Schiex, T. (1995). A constraint satisfaction framework for decision under uncertainty. In Besnard, P., & Hanks, S. (Eds.) *UAI '95: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence* (pp. 167–174). Montreal: Morgan Kaufmann.
9. Hnich, B., Rossi, R., Tarim, S.A., Prestwich, S. (2011). A survey on CP-AI-OR hybrids for decision making under uncertainty. In van Hentenryck, P., & Milano, M. (Eds.) *Hybrid Optimization, volume 45 of Springer Optimization and Its Applications, chapter 7* (pp. 227–270). New York: Springer.
10. Hnich, B., Rossi, R., Armagan Tarim, S., Prestwich, S.D. (2012). Filtering algorithms for global chance constraints. *Artificial Intelligence*, *189*, 69–94.
11. Kall, P., & Wallace, S.W. (1994). *Stochastic Programming*. Hoboken: Wiley.
12. Regin, J.-C. (1994). A filtering algorithm for constraints of difference in csps. In *Proceedings of the 12th National Conference on Artifcial Intelligence*, (Vol. 1 pp. 362–367). Seattle: AAAI Press.
13. Rossi, R., Tarim, S.A., Bollapragada, R. (2012). Constraint-based local search for computing non-stationary replenishment cycle policy under stochastic lead-times. *INFORMS Journal on Computing*, *24*(1), 66–80.
14. Rossi, R., Tarim, S.A., Hnich, B., Prestwich, S. (2008). A global chance-constraint for stochastic inventory systems under service level constraints. *Constraints*, *13*(4), 490–517.
15. Rossi, R., Tarim, S.A., Hnich, B., Prestwich, S.D. (2008). Cost-based domain filtering for stochastic constraint programming. In Stuckey, P.J. (Ed.) *Principles and Practice of Constraint Programming, CP 2008, Proceedings, volume 5202 of LNCS* (pp. 235–250): Springer.
16. Rossi, R., Tarim, S.A., Hnich, B., Prestwich, S. (2010). Computing replenishment cycle policy under non-stationary stochastic lead time. *International Journal of Production Economics*, *127*(1), 180–189.
17. Rossi, R., Hnich, B., Armagan Tarim, S., Prestwich, S. (2015). Confidence-based reasoning in stochastic constraint programming. *Artificial Intelligence*, *228*, 129–152.
18. Sahinidis, N.V. (2004). Optimization under uncertainty: State-of-the-art and opportunities. *Computers and Chemical Engineering*, *28*, 971–983.
19. Tarim, S.A., Hnich, B., Prestwich, S.D., Rossi, R. (2008). Finding reliable solution: Event-driven probabilistic constraint programming. Annals of Operations Research.

20. Tarim, S.A., Hnich, B., Rossi, R., Prestwich, S. (2009). Cost-based filtering techniques for stochastic inventory control under service level constraints. *Constraints*, *14*(2), 137–176.
21. Tarim, S.A., Manandhar, S., Walsh, T. (2006). Stochastic constraint programming: A scenario-based approach. *Constraints*, *11*(1), 53–80.
22. Walsh, T. (2002). Stochastic constraint programming. In *European Conference on Artificial Intelligence, ECAI'2002* (pp. 111–115): Proceedings.
23. Zghidi, I. (2011). *Computing optimal (s,s) policy parameters under service level constraints: A stochastic constraint programming approach*. Tunisia: Master's thesis, Sfax University.
24. Zghidi, I. (2016). *Towards Statistical Consistency for Stochastic Constraint Programming*. Tunisia: PhD thesis, University of Sfax.