Artificial Intelligence 228 (2015) 129–152

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Confidence-based reasoning in stochastic constraint programming ☆

Roberto Rossi [a,*], Brahim Hnich [b], S. Armagan Tarim [c,d,1], Steven Prestwich [d,1]

[a] Business School, University of Edinburgh, United Kingdom
[b] Department of Computer Science, Taif University, Taif, Saudi Arabia
[c] Department of Management, Cankaya University, Ankara, Turkey
[d] Insight Centre for Data Analytics, University College Cork, Ireland

ARTICLE INFO

Article history:
Received 7 November 2014
Received in revised form 5 July 2015
Accepted 8 July 2015
Available online 15 July 2015

Keywords:
Confidence-based reasoning
Stochastic constraint programming
Sampled SCSP
(α, ϑ)-solution
(α, ϑ)-solution set
Confidence interval analysis
Global chance constraint

ABSTRACT

In this work we introduce a novel approach, based on sampling, for finding assignments that are likely to be solutions to stochastic constraint satisfaction problems and constraint optimisation problems. Our approach reduces the size of the original problem being analysed; by solving this reduced problem, with a given confidence probability, we obtain assignments that satisfy the chance constraints in the original model within prescribed error tolerance thresholds. To achieve this, we blend concepts from stochastic constraint programming and statistics. We discuss both exact and approximate variants of our method. The framework we introduce can be immediately employed in concert with existing approaches for solving stochastic constraint programs. A thorough computational study on a number of stochastic combinatorial optimisation problems demonstrates the effectiveness of our approach.

1. Introduction

The stochastic constraint satisfaction/optimisation framework introduced in [2,3] constitutes an expressive declarative formalism for modelling problems of decision making under uncertainty. A stochastic constraint satisfaction problem (SCSP), alongside decision variables, features *random variables*, which follow some probability distribution and can be used to model uncertainty. Relationships over subsets of random and decision variables can be expressed in a declarative manner via *stochastic constraints*. The fact that a given relationship over subsets of random and decision variables should be satisfied according to a prescribed probability can be expressed by means of *chance constraints*. Finally, since problems of decision making under uncertainly are sequential in nature, the modeller can define a *stage structure*, that is a sequence of *decision stages*, in each of which a subset of all possible decisions are taken and a subset of all possible random variables are observed. A solution to an SCSP can be represented in general by means of a *policy tree*, which records feasible or optimal decisions associated with each possible set of random variable realisations.

☆ This work is an extended version of [1].

* Corresponding author at: Business School, University of Edinburgh, 29 Buccleuch place, EH8 9JS, Edinburgh, UK. Tel.: +44 (0)131 6515239; fax: +44 (0)131 650 8077.
E-mail addresses: roberto.rossi@ed.ac.uk (R. Rossi), hnich.brahim@gmail.com (B. Hnich), armtar@yahoo.com (S.A. Tarim), s.prestwich@cs.ucc.ie (S. Prestwich).

[1] This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.

http://dx.doi.org/10.1016/j.artint.2015.07.004
0004-3702/

As shown in [3, Theorem 1], solving SCSPs is a computationally hard task. Even trivial instances with a dozen of decision and random variables require a computational effort out of reach even for the most advanced hardware/software combination. This is due to the fact that the size of the policy tree grows exponentially in the number of random variables in the model and in the size of their support. Furthermore, a major limitation of all existing complete SCSPs solution methods, such as [3] and [4], is the fact that they assume the support of random variables is *finite*, otherwise a solution cannot be expressed as a finite policy tree. In practice, however, it is often the case that random variables either range over continuous supports or have a very large number (possibly infinite) of values in their domain. To date, no general purpose method exists for solving large-scale SCSPs, or SCSPs featuring random variables with continuous or discrete infinite support; for the sake of brevity we shall name this latter class of SCSPs "infinite SCSPs."

The main contribution of this paper is to propose a framework for solving large-scale or infinite SCSPs. More specifically, we argue that in solving large-scale or infinite SCSPs, one should not consider the ultimate feasible/optimal solution, which in some cases may even be impossible to represent; rather, the decision maker should aim for a solution that she "sufficiently trusts," which she may claim to be optimal or feasible with a given confidence level, and for which a certain degree of error may be tolerated. In order to obtain such a solution, the decision maker should only look at a possibly limited number of samples drawn from the random variables in the model. In other words, she should try to "estimate" the quality of this solution.

Our approach has several analogies with established techniques in statistics. When a survey is conducted on a sample population — e.g. an electoral poll — a statistician typically associates a certain confidence level with the results obtained from the chosen sample population. For instance, one may claim that there is a 90% chance that the actual mean being estimated is within a given interval. We argue that the very same approach may be adopted in stochastic decision making. If the infinite or large-scale m-stage SCSP does not admit any closed form solution and is complex enough to rule out any chance of obtaining an exact solution, we suggest that — as is done in statistics — one may introduce a confidence level α and a tolerated estimation error $\pm\vartheta$. The decision maker, instead of looking for an exact solution, may then aim to "estimate" — according to the chosen α and ϑ — whether the actual satisfaction probability guaranteed by an assignment is greater than or equal to the given target value for each of the chance constraints in the model. By choosing given values for α and ϑ the set of solutions may vary. For this reason we will introduce a new notion of solution that is parameterised by these two parameters and that we call an (α, ϑ)-solution. Intuitively, as α tends to 1 and ϑ tends to 0 the set of (α, ϑ)-solutions will converge to the set of actual solutions to the original stochastic constraint satisfaction problem, which we therefore rename $(1, 0)$-solutions. One should note that an approach of this kind has been recently advocated in [5, Chap. 4].

In this work, we make the following contributions to the stochastic constraint programming literature:

- we discuss how to obtain compact instances of infinite or large-scale stochastic constraint programs via sampling: we call these instances "sampled SCSPs;"
- we introduce the concepts of (α, ϑ)-solution and of (α, ϑ)-solution set; and show how to compute a priori the minimum sample size that guarantees the attainment of such classes of solutions;
- we show how the above tools can be employed in order to find approximate solutions to infinite or large-scale stochastic constraint satisfaction/optimisation problems that cannot be solved by existing exact approaches in the stochastic constraint programming literature;
- we conduct a thorough computational study on three well-known stochastic combinatorial problems to validate our theoretical framework and assess its effectiveness, efficiency, and scalability.

This work is structured as follows: in Section 2 we introduce the relevant formal background in constraint programming, stochastic constraint programming, and confidence interval analysis; in Section 3 we introduce sampled SCSPs; in Section 4 we discuss properties of the solutions of sampled SCSPs and formally introduce (α, ϑ)-solutions; in Section 5 we introduce (α, ϑ)-solution sets; in Section 6 we extend our discussion to stochastic constraint optimisation problems; in Section 7 we discuss connections with established techniques in statistics; in Section 8 we present our computational study; in Section 9 we discuss related works; finally, in Section 10 we draw conclusions and discuss future research directions.

2. Formal background

We now introduce the relevant background in constraint programming, stochastic constraint programming, and confidence interval analysis.

2.1. Constraint programming

A Constraint Satisfaction Problem (CSP) [6] consists of a set of decision 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 *generalised arc consistent* (*GAC*) if and only if 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*.

2.2. Stochastic constraint programming

The following definitions are based on [4,7]. An m-stage stochastic constraint satisfaction problem (SCSP) [2] is a 7-tuple $\langle V, S, D, P, C, \beta, L\rangle$, where V is a set of decision variables and S is a set of random variables, D is a function mapping each element of V (respectively, S) 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. To keep the discussion focused, without loss of generality, we assume that this probability distribution is not influenced by the decisions made; extensions to the SCP framework that deal with decision-dependent probabilities are discussed in [8]. C is a set of constraints over a non-empty subset of decision variables and a subset of random variables. If a constraint constrains only decision variables, then we call it a deterministic constraint; if it constrains both decision and random variables, then we call it a stochastic constraint. β is a function mapping each stochastic constraint $h \in C$ to β_h, which is a threshold value in the interval $(0, 1]$. If this threshold is strictly less than 1, then the stochastic constraint is a chance constraint. $L = [\langle V_1, S_1\rangle, \ldots, \langle V_i, S_i\rangle, \ldots, \langle V_m, S_m\rangle]$ is a list of *decision stages* such that each $V_i \subseteq V$, each $S_i \subseteq S$, the V_i form a partition of V, and the S_i form a partition of S.

To solve an m-stage SCSP an assignment to the variables in V_1 must be found such that, given random values for S_1, assignments can be found for V_2 such that, given random values for S_2, ..., assignments can be found for V_m so that, given random values for S_m, the deterministic constraints are satisfied and the stochastic constraints are satisfied in the fraction of all possible scenarios specified by function β. 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* [3]. The arcs in such a policy tree represent values observed for random 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*.

Let $\mathcal{S}$ denote the space of policy trees that are solutions to an SCSP. We may be interested in finding a policy tree $s \in \mathcal{S}$ that maximises the value of a given objective function $f(\cdot)$ over a subset of stochastic variables and a non-empty subset of decision variables. A stochastic constraint optimisation problem (SCOP) is then defined in general as $\max_{s\in\mathcal{S}} f(s)$.

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 $|S_i| > 1$ (see [4]). Intuitively, if S_i comprises more than one random variable, it is always possible to aggregate these variables into a single multivariate random variable [9] by convolving them. If V_i comprises more than one decision variable, in the following discussion the term decision variable should be interpreted as a set of decision variables. Let $S = \{s_1, s_2, \ldots, s_m\}$ be the set of all random variables and $V = \{x_1, x_2, \ldots, x_m\}$ be the set of all decision variables.

Let p be a path from the root node of the policy tree to a leaf. Let Ψ denote the set of all distinct paths of a policy tree. For each $p \in \Psi$, we denote by $\text{arcs}(p)$ the sequence of all the arcs in p whereas $\text{nodes}(p)$ denotes the sequence of all nodes in p. We denote by $\Omega = \{\text{arcs}(p) | p \in \Psi\}$ the set of all scenarios of the policy tree. The probability of $\omega \in \Omega$ is given by $\Pr\{\omega\} = \prod_{i=1}^{m} \Pr\{s_i = \bar{s}_i | s_{i-1} = \bar{s}_{i-1}, \ldots, s_1 = \bar{s}_1\}$, where $\Pr\{s_i = \bar{s}_i | s_{i-1} = \bar{s}_{i-1}, \ldots, s_1 = \bar{s}_1\}$ is the probability that random variable s_i takes value $\bar{s}_i$, given a set of realisations for random variables $s_{i-1}, \ldots, s_1$ already observed.

Now consider a constraint $h \in C$ with a specified threshold level β_h. Consider a policy tree $\mathcal{T}$ for the SCSP and a path $p \in \mathcal{T}$. Let $h_{\downarrow p}$ be the deterministic constraint obtained by substituting the random variables in h with the corresponding values $(\bar{s}_i)$ assigned to these random variables in $\text{arcs}(p)$. Let $\bar{h}_{\downarrow p}$ be the resulting tuple obtained by substituting the decision variables in $h_{\downarrow p}$ by the values $(\bar{x}_i)$ assigned to the corresponding decision variables in $\text{nodes}(p)$. We say that h is *satisfied wrt to a given policy tree* $\mathcal{T}$ iff

$$\sum_{p\in\Psi : \bar{h}_{\downarrow p} \in h_{\downarrow p}} \Pr\{\text{arcs}(p)\} \geq \beta_h.$$

Definition 1. Given an m-stage SCSP $\mathcal{P}$ and a policy tree $\mathcal{T}$, $\mathcal{T}$ is a satisfying policy tree to $\mathcal{P}$ iff every constraint of $\mathcal{P}$ is satisfied wrt $\mathcal{T}$.

Example 1. Let us consider the two-stage SCSP in Fig. 2, whose stage structure is $L = [\langle V_1, S_1\rangle, \langle V_2, S_2\rangle]$; $V_1 = \{x_1\}$ and $S_1 = \{s_1\}$, $V_2 = \{x_2\}$ and $S_2 = \{s_2\}$. Random variable s_1 may take two possible values, 5 and 4, each with probability 0.5; random variable s_2 may also take two possible values, 3 and 4, each with probability 0.5. The domain of x_1 is $\{1, \ldots, 4\}$, the domain of x_2 is $\{3, \ldots, 6\}$. There are two chance constraints[2] in C, $\Pr\{c_1 : s_1x_1 + s_2x_2 \geq 30\} \geq 0.75$ and $\Pr\{c_2 : s_2x_1 = 12\} \geq 0.5$. In this case, the decision variable x_1 must be set to a unique value before random variables are observed, while decision variable x_2 takes a value that depends on the observed value of the random variable s_1. A possible solution to

[2] In what follows, for convenience, we denote a chance constraint by using the notation "$\Pr\{\langle cons\rangle\} \geq \beta$", meaning that constraint $\langle cons\rangle$, constraining decision and random variables, should be satisfied with probability greater than or equal to β.

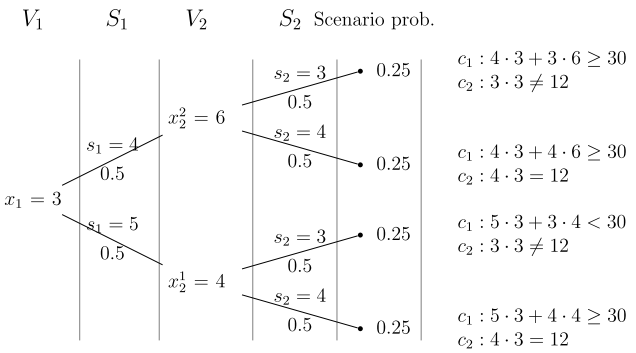

Fig. 1. Policy tree for the SCSP in Example 1.

Constraints:
(1) $\Pr\{s_1x_1 + s_2x_2 \geq 30\} \geq 0.75$
(2) $\Pr\{s_2x_1 = 12\} \geq 0.5$

Decision variables:
$x_1 \in \{1, 2, 3, 4\}$ $x_2 \in \{3, 4, 5, 6\}$

Random variables:
$s_1 \in \{4(0.5), 5(0.5)\}$ $s_1 \in \{3(0.5), 4(0.5)\}$

Stage structure:
$V_1 = \{x_1\}$, $V_2 = \{x_2\}$ $S_1 = \{s_1\}$, $S_2 = \{s_2\}$
$L = [\langle V_1, S_1\rangle, \langle V_2, S_2\rangle]$

Fig. 2. The two-stage SCSP in Example 1.

this SCSP is the satisfying policy tree shown in Fig. 1 in which $x_1 = 3, x_2^1 = 4$ and $x_2^2 = 6$, where x_2^1 is the value assigned to decision variable x_2, if random variable s_1 takes value 5, and x_2^2 is the value assigned to decision variable x_2, if random variable s_1 takes value 4.

As Example 1 shows, a solution to an SCSP is not simply an assignment of the decision variables to values, but it is instead a satisfying policy tree.

It is worth noting that *asking individual constraints to be satisfied according to their probability thresholds* is different from *asking a conjunction of constraints to be satisfied according to a prescribed probability threshold*. Informally speaking, in Example 1 we simply state that $c_1 : s_1x_1 + s_2x_2 \geq 30$ should hold true with probability $\beta_1 = 0.75$, i.e. in at least 75% of the scenarios. If $c_2 : s_2x_1 = 12$ holds true or not in those very same scenarios is not a matter of concern, as long as c_2 holds true in at least 50% of the scenarios — not necessarily the same as those in which c_1 holds true. Essentially, in Example 1 we do not state anything about the conjunction $c_3 : (s_1x_1 + s_2x_2 \geq 30) \wedge (s_2x_1 = 12)$. If we want to state something about this conjunction, we need to post a specific chance constraint c_3 with its own satisfaction threshold β_3. Assuming $\beta_3 = 0.5$, we may for instance require the conjunction c_3 to hold true in at least 50% of the scenarios, i.e. $\Pr\{c_3 : (s_1x_1 + s_2x_2 \geq 30) \wedge (s_2x_1 = 12)\} \geq 0.5$. Incidentally, the policy tree presented in Fig. 1 also satisfies this constraint.

A practical example that further clarifies this discussion is found in inventory control. It is customary in inventory control problems to enforce service level constraints such as

$$\Pr\{I_t \geq 0\} \geq \alpha \qquad t = 1, \ldots, N$$

where N represents the length of the planning horizon and I_t is the inventory level at the end of period t. The above set of constraints means that the probability of stocking out in a given period should be less than $1 - \alpha$; regardless what happens in other periods. A more restrictive service level requirement would be

$$\Pr\{\bigwedge_{t=1}^{N} I_t \geq 0\} \geq \alpha$$

This latter restriction means that the probability of stocking out in at least one of the N periods should be less than $1 - \alpha$.

2.3. Confidence interval analysis

Confidence interval analysis is a well established technique in statistics. Informally, confidence intervals are a useful tool for computing, from a given set of experimental results, a range of values that, with a certain confidence level (or confidence probability), will cover the actual value of a parameter that is being estimated. Consider a discrete random variable that follows a Bernoulli distribution. Accordingly, such a variable may produce only two outcomes, i.e. "yes" and "no", with probability q and $1-q$, respectively. Let us assume that the value q — the "yes" probability — is unknown. Obviously, if we observe the outcome of a Bernoulli trial once, the data collected will not reveal much about the value of q. Nevertheless, in practice, we may be interested in "estimating" q, by repeatedly observing the behaviour of the random variable in a sequence of Bernoulli trials. This problem is well-known in statistics and both exact and approximate techniques are available for performing this estimation [10,11]. The estimation produced by the methods available in the literature typically does not come as a point estimate, rather it consists of an interval of values computed from a set of representative samples for the quantity being estimated. This interval is known as "confidence interval" and consists of a range of values that, with a certain confidence probability α, covers the actual value of the parameter that is being estimated.

A method that is commonly classified as the "exact confidence intervals" for the Binomial distribution has been introduced by Clopper and Pearson in [10]. This method uses the Binomial cumulative distribution function (CDF) in order to build the interval from the data observed. The Clopper–Pearson interval is a symmetric two-sided confidence interval. It can be however also expressed as a single-sided interval. Clopper–Pearson single-sided intervals can be written as $(p_{\text{lb}}, 1)$ and $(0, p_{\text{ub}})$ where

$$p_{\text{lb}} = \min\{q | \Pr\{\text{bin}(N;q) \geq X\} \geq 1-\alpha\},$$
$$p_{\text{ub}} = \max\{q | \Pr\{\text{bin}(N;q) \leq X\} \geq 1-\alpha\},$$

X is the number of successes (or "yes" events) observed in the sample, $\text{bin}(N;q)$ is a binomial random variable with N trials and probability of success q and α is the confidence probability. Note that we assume $p_{\text{lb}} = 0$ when $X = 0$ and that $p_{\text{ub}} = 1$ when $X = N$.

Because of the close relationship between Binomial distribution and the Beta distribution, the Clopper–Pearson interval is sometimes presented in an alternative format that uses percentiles from the beta distribution [12]:

$$p_{\text{lb}} = 1 - \text{beta}^{-1}(\alpha, N - X + 1, X),$$
$$p_{\text{ub}} = 1 - \text{beta}^{-1}(1-\alpha, N - X, X + 1),$$

where beta^{-1} denotes the inverse Beta distribution. This form can be efficiently evaluated by existing algorithms.

An interesting property of confidence intervals related to the estimation of the "success" probability associated with a Bernoulli trial consists in the fact that, given a confidence probability, it is possible to derive mathematically, by performing a worst case analysis, the minimum number of samples that should be observed in order to produce a confidence interval of a given size.

Therefore, for a given confidence probability α, it is possible to determine the minimum number of samples that should be considered in order to achieve a margin of error of $\pm\vartheta$ in the estimation of the "success" probability of a Bernoulli trial. This computation plays a central role in our novel approach. In fact, intuitively estimating the satisfaction probability of a chance constraint is equivalent to estimating the "success" probability of the associated Bernoulli trial.

3. Sampled SCSPs

Consider an SCSP $\mathcal{P}$ over a set S of stochastic variables. Assume that stochastic variables are defined on continuous supports or discrete supports comprising a large or infinite number of values. Solving the original SCSP clearly poses a hard combinatorial challenge, in fact the policy tree comprises a number of scenarios that is exponential in the size of stochastic variable domains. Since the policy tree may comprise an infinite number of scenarios, the computational problem may even be undecidable in general.

In this section we discuss how to *sample* a more compact SCSP, which comprises at most N scenarios, out of the original problem. We shall call this new problem $\widehat{\mathcal{P}}_N$ or "sampled SCSP" over N scenarios. Intuitively, a sampled SCSP is a reduced version of the original problem the solution of which is a policy tree that comprises a bounded number of paths sampled out of the original policy tree. In the following sections we will discuss under which conditions the solution to a sampled SCSP $\widehat{\mathcal{P}}_N$ is, according to a certain confidence probability and within prescribed error tolerance thresholds, likely to be also a solution to the original SCSP $\mathcal{P}$.

We shall here discuss how to employ Simple Random Sampling [13] to obtain a sampled SCSP from the original problem. Of course, more advanced stratified sampling techniques may be used in order to reduce variance and improve the effectiveness of the approach. Nevertheless, due to the large number of topics already covered in this work, we leave this discussion as future work.

Consider a complete realisation, $\bar{s}_1, \ldots, \bar{s}_m$, for the stochastic variables in S obtained by sampling a value from the support $D(s_i)$ of each of the stochastic variables $s_i \in S$ according to its probability distribution $P(s_i)$. From the definition

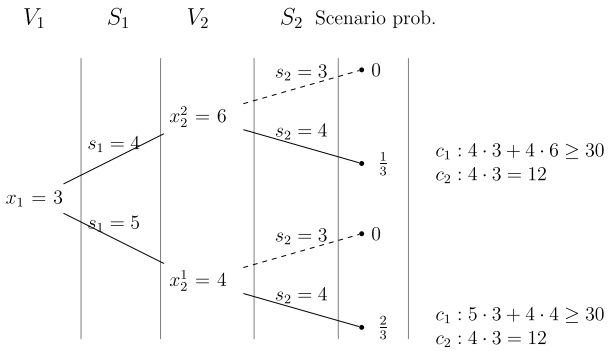

Fig. 3. Reduced policy tree for the sampled SCSP in Example 2; dashed arcs are those that have not been observed in the sample.

of policy tree it is clear that there always exists a path associated with this realisation. In other words, this realisation corresponds to one of the scenarios comprised in the policy tree.

Generate N independent sets of random variable realisations

$$\{\bar{s}_1^1, \ldots, \bar{s}_m^1\}, \{\bar{s}_1^2, \ldots, \bar{s}_m^2\}, \ldots, \{\bar{s}_1^N, \ldots, \bar{s}_m^N\},$$

where $\bar{s}_j^i$ is the realised value for random variable j observed in the i-th set of realisations. Recall that $\mathcal{T}$ denotes the complete, and possibly infinite, policy tree for $\mathcal{P}$. Let a reduced policy tree $\widehat{\mathcal{T}}$ for $\mathcal{P}$ be a policy tree that comprises only arc labellings observed in the former N complete realisations (without repetitions).

Let $\widehat{\Psi}$ denote the reduced set of distinct paths in $\widehat{\mathcal{T}}$. The probability associated with each path $p \in \widehat{\Psi}$, i.e. $\Pr\{\text{arcs}(p)\}$, is simply set equal to the frequency of occurrence of such a path in the above N realisations. Of course, $\widehat{\mathcal{T}}$ represents a policy tree for a different SCSP than the one we started with. We call this new problem the sampled SCSP $\widehat{\mathcal{P}}_N$.

Now consider a chance constraint $h \in C$ with a specified threshold level β_h, a policy tree $\widehat{\mathcal{T}}$ for the sampled SCSP $\widehat{\mathcal{P}}_N$ and a path $p \in \mathcal{T}$. We say that h is *satisfied wrt to a given policy tree* $\widehat{\mathcal{T}}$ iff

$$\sum_{p \in \widehat{\Psi} : \bar{h}_{\downarrow p} \in h_{\downarrow p}} \Pr\{\text{arcs}(p)\} \geq \beta_h.$$

Example 2. Let us consider the two-stage SCSP $\mathcal{P}$ discussed in Example 1. We set $N = 3$ and we derive a sampled SCSP $\widehat{\mathcal{P}}_N$. By using simple random sampling we draw the following three complete realisations for random variables in $\mathcal{P}$:

$$\{\bar{s}_1^1 = 5, \bar{s}_2^1 = 4\}, \{\bar{s}_1^2 = 4, \bar{s}_2^2 = 4\}, \{\bar{s}_1^3 = 5, \bar{s}_2^3 = 4\}.$$

A possible solution to the sampled SCSP $\widehat{\mathcal{P}}_N$ is the satisfying policy tree shown in Fig. 3, in which $x_1 = 3, x_2^1 = 4$ and $x_2^2 = 6$, where x_2^1 is the value assigned to decision variable x_2, if stochastic variable s_1 takes value 5, and x_2^2 is the value assigned to decision variable x_2, if stochastic variable s_1 takes value 4. The above policy tree has two paths sampled out of the original tree: p_1 has an associated probability of $2/3$, since we observed two occurrences of the scenario associated with this path over the 3 complete realisations sampled for the random variables; p_2 has an associated probability of $1/3$, since we observed a single occurrence of the scenario associated with this path over the 3 complete realisations sampled for the random variables. Paths that were not observed in the sampled realisations have an associated probability equal to zero and are not considered.

It should be noted that every policy tree $\widehat{\mathcal{T}}$ for a sampled SCSP $\widehat{\mathcal{P}}$ can be employed as a (partial) policy tree for the original SCSP $\mathcal{P}$. Nevertheless, by sampling we lose completeness. If at stage i in $\mathcal{P}$ we observe, for a given random variable, a realised value that is not comprised in $\widehat{\mathcal{T}}$, it will be of course impossible to determine the correct decisions for subsequent stages. By taking a conservative point of view, this means that all paths in the corresponding subtree will never be satisfied. In multi-stage SCSPs, and especially in those including random variables with continuous support, this prevents the direct use of the approach that will be discussed in this work. In fact, if random variable supports are continuous, there is only an infinitesimal probability of observing a given set of realisations. In this case, it is therefore essential to adopt a "rolling horizon" approach [14] in order to reduce the original multi-stage SCSPs to a sequence of multi-stage sampled SCSPs. Under this strategy, our aim is to fix decisions at stage one, and make sure that compatible values exist for decision variables that appear, for subsequent stages, in $\widehat{\mathcal{T}}$. Future decisions are not fixed because, after observing the realised values for random variables at stage one, the problem is solved again by taking into account new available information; decision variables that were previously associated with stage two become stage one decisions. The original problem is thus reduced to a sequence of multi-stage sampled SCSPs. We will apply this technique to handle the two-stage problem discussed in Section 8.2: the stochastic multiprocessor scheduling problem with release time and deadlines.

4. (α, ϑ)-solutions

We will now characterise the probability that the solution of a sampled SCSPs $\widehat{\mathcal{P}}_N$ over N scenarios, which may be computed by using any of the existing approaches discussed in [3,4], is a solution to the original *single-stage* SCSP $\mathcal{P}$.

These results are also applicable to multi-stage problems, provided that a rolling horizon approach is adopted and that the aim is to characterise the probability that stage one decisions of a sampled SCSPs $\widehat{\mathcal{P}}_N$ over N scenarios are consistent with respect to the original SCSP $\mathcal{P}$.

We will firstly discuss how to compute N such that, if a given policy tree $\mathcal{T}$ satisfies a chance constraint h in the sampled SCSPs $\widehat{\mathcal{P}}_N$, it also satisfies the same chance constraint in the original SCSP $\mathcal{P}$ with probability α. Since a policy tree $\mathcal{T}$ in $\widehat{\mathcal{P}}_N$ by definition only comprises a subset $\widehat{\Psi}$ of all the paths that constitute a policy tree for the original SCSP $\mathcal{P}$, this policy tree, in order to satisfy h in the original SCSP $\mathcal{P}$, must clearly provide a sufficient satisfaction probability regardless of the scenarios that have been ignored by the sampling process.

Consider a confidence probability α and a margin of error of $\pm\vartheta$; The number of scenarios N for the sampled SCSP depends on ϑ, α and also β, which we recall is the target satisfaction probability of chance constraint h.

Definition 2. N is computed as the minimum value for which

$$\max(p^{\beta}_{\text{ub}} - \beta, \beta - p^{\beta}_{\text{lb}}) \leq \vartheta,$$

where p^{β}_{lb} and p^{β}_{ub} are the single-sided Clopper–Pearson confidence interval bounds for a confidence probability α, and $round(\beta N)$ "successes" in N trials; $round()$ approximates the value to the nearest integer.[3]

Definition 3. Any policy tree $\mathcal{T}$, which can be proved to satisfy h in $\mathcal{P}$ with probability α, satisfies h in $\mathcal{P}$ with probability α if it satisfies h in $\widehat{\mathcal{P}}_N$. Conversely, any policy tree $\mathcal{T}$, which can be proved not to satisfy h in $\mathcal{P}$ with probability α, does not satisfy h in $\mathcal{P}$ with probability α, if it does not satisfy h in $\widehat{\mathcal{P}}_N$.

Proposition 1. *A policy tree $\mathcal{T}$ can be proved to satisfy h in $\mathcal{P}$ with probability α if the actual satisfaction probability $\delta > \beta$ provided by $\mathcal{T}$ wrt h is such that $\delta \geq p^{\beta}_{ub}$. Conversely, if the actual satisfaction probability $\delta < \beta$ provided by $\mathcal{T}$ wrt h is such that $\delta \leq p^{\beta}_{lb}$ $\mathcal{T}$ can be proved to not satisfy h in $\mathcal{P}$ with probability α.*

Proof. Let $\delta \geq p^{\beta}_{\text{ub}}$. Since $p^{\beta}_{\text{ub}} = \max\{q | \Pr\{\text{bin}(N; q) \leq \text{round}(\beta N)\} \geq 1-\alpha$, it is clear that $\Pr\{\text{bin}(N; \delta) \leq \text{round}(\beta N)\} < 1-\alpha$. This means that

$$\Pr\left\{ \sum_{p \in \widehat{\Psi} : \bar{h}_{\downarrow p} \in h_{\downarrow p}} \Pr\{\text{arcs}(p)\} \leq \beta \right\} < 1-\alpha,$$

where we recall that $\widehat{\Psi}$ is the set of paths in the sampled SCSP $\widehat{\mathcal{P}}_N$. This implies

$$\Pr\left\{ \sum_{p \in \widehat{\Psi} : \bar{h}_{\downarrow p} \in h_{\downarrow p}} \Pr\{\text{arcs}(p)\} \geq \beta \right\} \geq \alpha.$$

Therefore, by using the test

$$\sum_{p \in \widehat{\Psi} : \bar{h}_{\downarrow p} \in h_{\downarrow p}} \Pr\{\text{arcs}(p)\} \geq \beta,$$

a policy tree $\mathcal{T}$ can be proved to satisfy h in $\mathcal{P}$ with probability α.

Conversely, let $\delta \leq p^{\beta}_{\text{lb}}$. Since $p^{\beta}_{\text{lb}} = \min\{q | \Pr\{\text{bin}(N; q) \geq \text{round}(\beta N)\} \geq 1-\alpha$, it is clear that

$$\Pr\{\text{bin}(N; \delta) \geq \text{round}(\beta N)\} < 1-\alpha.$$

This means that

$$\Pr\left\{ \sum_{p \in \widehat{\Psi} : \bar{h}_{\downarrow p} \in h_{\downarrow p}} \Pr\{\text{arcs}(p)\} \geq \beta \right\} < 1-\alpha,$$

[3] This is justified by the fact that the Clopper–Pearson interval is, in fact, a step function — see [10], p. 405 — since the Binomial is a discrete probability distribution.

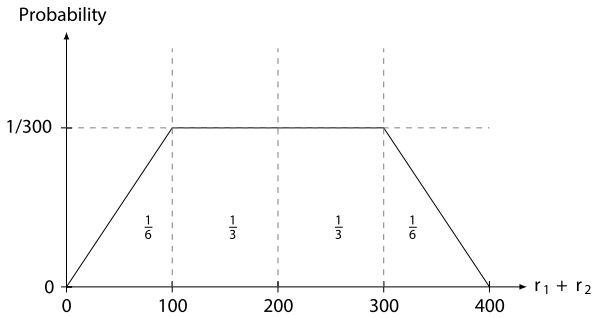

Fig. 4. Probability density function of the convolution of two independently non-identically distributed uniform random variables r_1 and r_2.

which implies

$$\Pr\left\{\sum_{p\in\widehat{\Psi}:\bar{h}_{\downarrow p}\in h_{\downarrow p}} \Pr\{\text{arcs}(p)\} \leq \beta\right\} \geq \alpha.$$

Therefore, by using the test

$$\sum_{p\in\widehat{\Psi}:\bar{h}_{\downarrow p}\in h_{\downarrow p}} \Pr\{\text{arcs}(p)\} \leq \beta,$$

a policy tree $\mathcal{T}$ can be proved to not satisfy h in $\mathcal{P}$ with probability α. □

Proposition 2. *Any policy tree $\mathcal{T}$ which provides a satisfaction probability $\delta \geq \beta + \vartheta$ wrt h in $\mathcal{P}$ can be proved to satisfy h in $\mathcal{P}$ with probability α. Any policy tree $\mathcal{T}$ which provides a satisfaction probability $\delta \leq \beta - \vartheta$ wrt h in $\mathcal{P}$ can be proved to not satisfy h in $\mathcal{P}$ with probability α.*

Proof. This directly follows from Definition 2 and Proposition 1. □

Proposition 3. *Any policy tree $\mathcal{T}$ which cannot be proved to satisfy or not to satisfy h in $\mathcal{P}$ with probability α, can be either proved to satisfy h in $\mathcal{P}$ with probability γ, where γ is a probability ranging in $[0.5, \alpha)$, if it satisfies h in $\widehat{\mathcal{P}}_N$, or not to satisfy h in $\mathcal{P}$ with probability γ, where γ is a probability ranging in $[0.5, \alpha)$, if it does not satisfies h in $\widehat{\mathcal{P}}_N$.*

Proof. Consider the two limiting cases. (i) The actual satisfaction probability δ provided by $\mathcal{T}$ wrt h in $\mathcal{P}$ is exactly equal to β. Since the sample mean, used to estimate the satisfaction probability out of the N samples considered, is an unbiased estimator of δ, it will overestimate β with probability 0.5 and, similarly, it will underestimate β with probability 0.5; this sets the lower bound for γ. (ii) The actual satisfaction probability δ provided by $\mathcal{T}$ wrt h in $\mathcal{P}$ is exactly equal to $\beta + \vartheta$. From the proof of Proposition 1 it immediately follows that, in this case, $\gamma = \alpha$, and also that, if $\delta < \beta + \vartheta$ then $\gamma < \alpha$; this sets the upper bound for γ. □

Definition 4. An (α, ϑ)-solution to an SCSP $\mathcal{P}$ is a policy tree $\widehat{\mathcal{T}}$ that at least with probability α provides for every chance constraint h_i in $\mathcal{P}$ with satisfaction threshold β_i a satisfaction probability greater than or equal to $\beta_i - \vartheta$.

It is apparent that ϑ may be interpreted as a parameter that the user can set in order to define a "region of indifference", i.e. $\beta \pm \vartheta$, for the satisfaction probability. In such a region, we assume that assignments can be safely misclassified with probability greater than α and that satisfaction probabilities remain in an acceptable range.

Example 3. Consider the single-stage SCSP $\mathcal{P} = \langle V, S, D, P, C, \beta, L\rangle$, where $V = \{X_1, X_2\}$, $S = \{r_1, r_2\}$, $D(X_1) = D(X_2) = \{0, 1\}$, $D(r_1) = (0, 100)$, $P(r_1) = \text{uniform}(0, 100)$, $D(r_2) = (0, 300)$, $P(r_2) = \text{uniform}(0, 300)$, $C = \{c : C_1 \geq X_1 r_1 + X_2 r_2\}$, $\beta_c = 0.5$, and $L = [\langle V, S\rangle]$. $C_1 = 185$ is a constant. This problem comprises random variables defined on a continuous support and it cannot be solved by existing complete approaches to SCSPs. If we set $\alpha = 0.95$ and $\vartheta = 0.05$, from Definition 2 we compute the number of samples $N = 290$ required to guarantee that any solution to the sampled SCSP $\widehat{\mathcal{P}}$ over N samples is an (α, ϑ)-solution for $\mathcal{P}$.

Furthermore, the simple structure of the constraint c considered in $\mathcal{P}$ allows us to perform some further analysis. Consider the assignment $X_1 = 1$ and $X_2 = 1$. A simple reasoning on the convolution of two independently non-identically distributed uniform random variables (see [15]) immediately suggests that this assignment is indeed inconsistent. r_1 and r_2

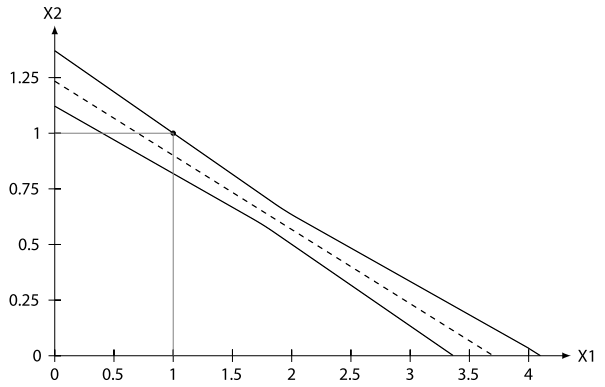

Fig. 5. Feasible region for the SCSP in Example 1; the dashed line denotes the true boundary of constraint c. The upper solid line demarcates the set of solutions providing a satisfaction probability of at least $\beta - \vartheta$; the lower solid line demarcate the set of solutions providing a satisfaction probability of at least $\beta + \vartheta$.

are two independently non-identically distributed uniform random variables. The distribution that results from their convolution is shown in Fig. 4. This distribution is shaped like a trapezoid. Clearly, since the area for the whole figure must be equal to 1, the area of each of the two rectangle triangles at the side of the trapezoid must be equal to 1/6. Consequently, the area of the internal rectangle must be equal to 2/3. It is easy to see that the cumulative distribution function for value 200 returns a probability of 0.5. Then, since $1/3^*(15/100) = 0.05$, the 0.45 quantile of the inverse cumulative distribution function which results from convolving r_1 and r_2 is exactly equal to $C_1 = 185$. Therefore, since the satisfaction probability provided by the assignment $X_1 = 1$ and $X_2 = 1$ is equal to $\beta_c - \vartheta = 0.45$ (Fig. 5), this assignment will be correctly classified as inconsistent with probability α, when the sample size is set to $N = 290$.

Let $h_1, \ldots, h_k$ be k chance constraints in an SCSP $\mathcal{P}$. Let $\widehat{\mathcal{P}}$ be a sampled SCSP over N samples, where N is the number of samples required to guarantee a confidence level α and an error tolerance threshold ϑ for each constraint h_i considered independently, according to Definition 2.

Proposition 4. *Let $\widehat{\mathcal{T}}$ be a policy tree that is a solution to $\widehat{\mathcal{P}}$. Then $\widehat{\mathcal{T}}$ is an (α, ϑ)-solution for $\mathcal{P}$.*

Proof. Consider a chance constraint h_i. Let β_i be the respective satisfaction threshold. By definition, the probability that a solution $\widehat{\mathcal{T}}$ to $\widehat{\mathcal{P}}$ provides a service level less or equal to $\beta_i - \vartheta$ for h_i in $\mathcal{P}$ is less than or equal to $1 - \alpha$. Therefore $\widehat{\mathcal{T}}$ is an (α, ϑ)-solution. Now consider a pair of chance constraints $\langle h_i, h_j \rangle$ with satisfaction thresholds β_i, β_j, respectively. The probability p_{ij} that a solution $\widehat{\mathcal{T}}$ to $\widehat{\mathcal{P}}$ provides a service level less or equal to $\beta_i - \vartheta$ for h_i and to $\beta_j - \vartheta$ for h_j in $\mathcal{P}$ is less than or equal to $(1 - \alpha)^2$, in fact we must misclassify both the constraints in order to accept such a solution. Even a single constraint correctly classified will make $\widehat{\mathcal{T}}$ inconsistent w.r.t. $\widehat{\mathcal{P}}$. The case in which constraints are misclassified independently from each other represents a worst-case reasoning. If constraints are perfectly positively correlated, i.e. if one is misclassified then all other constraints are also misclassified, then p_{ij} is $(1 - \alpha)$; if constraints are perfectly negatively correlated, i.e. if one is misclassified then no other constraint is misclassified, p_{ij} becomes 0. This reasoning can be generalised to k chance constraints, for which the probability becomes $(1 - \alpha)^k$. Noting that $(1 - \alpha)^k < \ldots < (1 - \alpha)^2 < (1 - \alpha)$ and that $1 - (1 - \alpha)^k \geq \alpha$ the probability $1 - \alpha$ that a solution is misclassified in a model comprising a single constraint represents an upper bound for the probability that a solution $\widehat{\mathcal{T}}$ to $\widehat{\mathcal{P}}$ does not provide a satisfaction probability within the required tolerance threshold for one or more constraints in a generic model $\mathcal{P}$. By rephrasing, the probability that a solution $\widehat{\mathcal{T}}$ provides a satisfaction probability greater than or equal to $\beta_i - \vartheta$ for each constraint h_i is greater than or equal to α. Hence, by Definition 4, $\widehat{\mathcal{T}}$ is an (α, ϑ)-solution for $\mathcal{P}$. □

5. (α, ϑ)-solution set

Consider policy tree $\mathcal{T}$, chance constraint h, and the indicator random variable

$$\tau = \begin{cases} 1 & \sum_{p \in \widehat{\Psi}: \bar{h}_{\downarrow p} \in h_{\downarrow p}} \Pr\{\text{arcs}(p)\} \geq \beta \\ 0 & \text{otherwise} \end{cases}$$

representing the test discussed in Definition 3. If the actual satisfaction probability δ provided by a policy tree $\mathcal{T}$ with respect to constraint h in $\mathcal{P}$ is exactly equal to $\beta - \vartheta$, τ takes value 1 with probability $1 - \alpha$.

To motivate the following discussion, we introduce the following example.

Example 4. Consider once more Example 3 and assume that $D(X_1) = D(X_2) = (0, 5)$; i.e. decision variables are defined on continuous domains spanning from 0 to 5. Assignments $(X_1 = 4.1, X_2 = 0)$ and $(X_1 = 0, X_2 = 1.37)$ lie on the upper solid line shown in Fig. 5. Each of these two assignments provides a satisfaction probability of exactly $\beta - \vartheta$ with respect to constraint c in the original problem $\mathcal{P}$. From the discussion in Section 4 it follows that each of these two assignments is recognised as infeasible with probability α if $N = 290$. However, since r_1 and r_1 are independent the probability that these two assignments are **both** recognised as infeasible is only α^2. We next discuss how to address the issue of correctly classifying multiple policy trees according to a prescribed confidence level α.

We introduce the following definition.

Definition 5. An (α, ϑ)-solution set to an SCSP $\mathcal{P}$ is a set of policy trees. All policy trees in this set simultaneously provide, with probability at least α, a satisfaction probability greater than or equal to $\beta_i - \vartheta$ for every chance constraint h_i in $\mathcal{P}$ with satisfaction threshold β_i.

Consider an SCSP and T policy trees $\mathcal{T}_1, \ldots, \mathcal{T}_T$ for which the actual satisfaction probability δ with respect to h in $\mathcal{P}$ is less than or equal to $\beta - \vartheta$. Let $\tau_1 \ldots, \tau_T$ be the associated random variables, each of which according to Proposition 4 takes value 1 with probability less than or equal to $1 - \alpha$. Although we have fully characterised the marginal probability distribution of a test τ_i involving a single policy tree $\mathcal{T}_i$, we have not characterised yet the joint probability among tests carried out on a set of T policy trees.

Proposition 5. *The probability that $\tau_1, \ldots, \tau_T$ are all equal to 0 is at least $1 - T(1 - \alpha)$.*

Proof. A worst-case reasoning can be carried out by considering the case in which events $\tau_i = 1$ and $\tau_j = 1$ are mutually exclusive for all $i, j = 1, \ldots, T$, $i \neq j$; of course it is still true that $\Pr\{\tau_i = 1\} = \Pr\{\tau_j = 1\} \leq 1 - \alpha$. The probability that $\tau_1, \ldots, \tau_T$ are all equal to 0 is then easily seen to be $1 - T(1 - \alpha)$. If events are not mutually exclusive, this probability is greater than or equal to $1 - T(1 - \alpha)$, e.g. in the case of T independent tests it would be $1 - (1 - \alpha)^T \geq 1 - T(1 - \alpha)$. □

Of course, it is not possible to know the value of T a priori, as this would require solving the SCSP. However, for a given chance constraint h, T is clearly less than or equal to the cardinality A_h of the assignment space constrained by h. A_h can be computed as the cartesian product of the domains of the decision variables in the policy tree that are constrained by h. Since the property discussed in Proposition 5 applies to each chance constraint $h \in C$, to compute an (α, ϑ)-solution set we may introduce the following Bonferroni's correction [16], which is *free of correlation and distribution assumptions*, while computing N.

Definition 6. N is computed as the minimum value for which

$$\max(p^{\beta}_{\text{ub}} - \beta, \beta - p^{\beta}_{\text{lb}}) \leq \vartheta,$$

where p^{β}_{lb} and p^{β}_{ub} are the single-sided Clopper–Pearson confidence interval bounds for a confidence probability $\widehat{\alpha}$, where

$$\widehat{\alpha} = 1 - \frac{1 - \alpha}{\sum_{h \in C} A_h},$$

and $round(\beta N)$ "successes" in N trials.

Proposition 6. *A set of policy trees that are solutions to $\widehat{\mathcal{P}}$ for a sample size N computed as discussed in Definition 6 is an (α, ϑ)-solution set for $\mathcal{P}$.*

Proof. Bonferroni's correction, introduced in Definition 6, ensures that, for every chance constraint h in C, the probability $\tau_1, \ldots, \tau_T$ are all equal to 0 simultaneously is at least α. □

Example 5. Consider the following SCSP $\mathcal{P} = \langle V, S, D, P, C, \beta_c, L \rangle$, where $V = \{X_1, X_2\}$, $S = \{r_1, r_2\}$, $D(X_1) = D(X_2) = \{0, 0.01, 0.02, \ldots, 24.99, 25\}$, $D(r_1) = (0, 10)$, $P(r_1) = \text{uniform}(0, 10)$, $D(r_2) = (0, 30)$, $P(r_2) = \text{uniform}(0, 30)$, $D(r_3) = (0, 15)$, $P(r_3) = \text{uniform}(0, 15)$, $D(r_4) = (0, 20)$, $P(r_2) = \text{uniform}(0, 20)$, $C = \{c_1 : C_1 \geq X_1 r_1 + X_2 r_2, c_2 : C_2 \geq X_1 r_3 + X_2 r_4\}$, $\beta_{c_1} = \beta_{c_2} = 0.7$, and $L = [\langle V, S \rangle]$. $C_1 = 245$ and $C_2 = 215$ are constants. We set $\alpha = 0.9$ and $\vartheta = 0.05$. We computed analytically the true boundaries of c_1 and c_2 (see [15,17]), each of which is denoted by a dashed line in Figs. 6 and 7. We also computed confidence bands around these two dashed lines. The upper confidence band is the set of solutions that provide a satisfaction probability of exactly $\beta_i - \vartheta$; the lower confidence band is the set of solutions that provide a satisfaction probability of exactly $\beta_i + \vartheta$. We apply Definition 6 to compute the number of samples $N = 2848$ required to obtain an (α, ϑ)-solution set to $\mathcal{P}$, which is shown in Fig. 6.

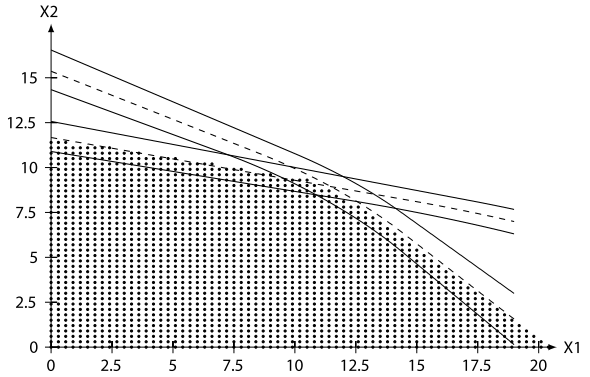

Fig. 6. An (α, ϑ)-solution set for Example 5 computed for $N = 2848$ samples.

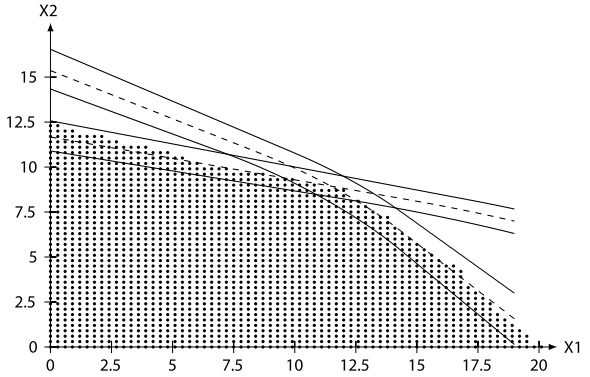

Fig. 7. An approximate (α, ϑ)-solution set for Example 6 computed for $N = 348$ samples.

5.1. Approximating (α, ϑ)-solution sets

Bonferroni's correction is known to be conservative. In particular, as we have seen, this correction assumes events $\tau_i = 1$ and $\tau_j = 1$ are mutually exclusive for all $i, j = 1, \ldots, T$, $i \neq j$. In other words, we are assuming that assignment misclassifications are mutually exclusive. In practice, in an SCSP sets of assignments are often misclassified together depending on random variables realisations. For this reason a correction such as the one introduced in Definition 6 will generally be too conservative and will lead to a sample size much larger than the one strictly needed to obtain an (α, ϑ)-solution set. This fact is well known in statistics and a number of adjusted corrections have been proposed to account for correlated errors [see e.g. 18,19].

In what follows, we shall therefore adopt a less conservative approximate correction strategy. To the best of our knowledge no similar correction has been discussed in the literature. In our computational study (Section 8) we will demonstrate the effectiveness of this technique. Of course, the investigation of other less conservative and possibly exact correction strategies, ideally borrowed from established results in statistics, is an interesting direction for future research.

Consider the general case in which constraint h constrains all m random variables in S.

Lemma 1. *Given realisations* $\{\bar{s}_1^1, \ldots, \bar{s}_m^1\}, \{\bar{s}_1^2, \ldots, \bar{s}_m^2\}, \ldots, \{\bar{s}_1^N, \ldots, \bar{s}_m^N\}$, *where* $\bar{s}_j^k$ *is the realised value for random variable* j *observed in the* k*-th set of realisations,* τ_i *is a deterministic test.*

Proposition 7. τ_i *is a function of random variables* $s_1, \ldots, s_m$ *and of* N.

Proof. Immediately follows from Lemma 1 and from the fact that $s_j^1, \ldots, s_j^N$ are N i.i.d. random variables. □

Proposition 8. *The probability that at least one of* $\tau_1, \ldots, \tau_T$ *takes value 1 is uniquely determined by the probability distributions of* $s_1, \ldots, s_m$ *and the number of samples* N.

Proof. Follows from the definition of τ, Lemma 1 and Proposition 7. □

In practice, this means that assignment misclassifications in a sampled SCSP, e.g. events $\tau_i = 1$ and $\tau_j = 1$, depend on realisations of one or more random variables $s_1, \ldots, s_m$; note that m is generally much smaller than T. Since the multivariate random variable $\{\tau_1, \ldots, \tau_T\}$ is a (deterministic) function of the multivariate random variable $\{s_1, \ldots, s_m\}$ and of N (Proposition 7), and since in the previous section we have fully characterised the marginal probability distribution of a test τ_i, the probability that $\tau_1, \ldots, \tau_T$ are all equal to 0 is approximately bounded from below by $1 - m(1-\alpha)$; i.e. we correct for at most m mutually exclusive misclassifications induced by random variables $s_1, \ldots, s_m$ and we assume that all remaining misclassifications depend on one or more of these. Once more we introduce a correction for each chance constraint $h \in C$. Let m_h be the number of random variables constrained by h, to compute an approximate (α, ϑ)-solution set we introduce the following correction while computing N.

Definition 7. N is computed as the minimum value for which

$$\max(p^{\beta}_{\text{ub}} - \beta, \beta - p^{\beta}_{\text{lb}}) \leq \vartheta,$$

where p^{β}_{lb} and p^{β}_{ub} are the single-sided Clopper–Pearson confidence interval bounds for a confidence probability $\widehat{\alpha}$, where

$$\widehat{\alpha} = 1 - \frac{1-\alpha}{\sum_{h \in C} m_h},$$

and $round(\beta N)$ "successes" in N trials.

A set of policy trees that are solutions to $\widehat{\mathcal{P}}$ for a sample size N computed as discussed in Definition 7 is an approximate (α, ϑ)-solution set for $\mathcal{P}$.

Example 6. Consider once more the SCSP in Example 5. We apply Definition 7 to compute the number of samples $N = 348$ required to obtain an approximate (α, ϑ)-solution set to $\mathcal{P}$, which is shown in Fig. 7; note that there are two constraints each of which constrains two random variables. To assess the quality of this approximation, we generated 1000 different instances and analytically inspected, for each of them, if the (α, ϑ)-solution set generated was fully contained within the upper confidence band in Fig. 7; the result of this simulation study revealed that the (α, ϑ)-solution set was not fully contained within the upper confidence band with probability 0.894, 0.95 confidence interval $(0.873, 0.912)$; this misclassification rate is in line with the prescribed α. Finally, it is worth noting that the random boundary of an (α, ϑ)-solution set remains within the channel identified by the two solid confidence bands with probability at least $1 - 2(1-\alpha)$.

6. Stochastic constraint optimisation problems

The concepts introduced in Sections 4 and 5 can be employed to approximate optimal solutions to sampled SCOPs. In this setting, we must distinguish two possible cases: the case in which the objective function is deterministic and that in which the objective function is stochastic. If the objective function is deterministic, it is possible to exploit the results in Section 5 to obtain a confidence interval for the cost/profit of an optimal plan. Without loss of generality, we discuss the case in which our aim is to maximise a deterministic objective function f of the decision variables in V. Consider an SCOP $\mathcal{P} = \langle V, S, D, P, C, \beta_c, L, f \rangle$. Choose α and ϑ and construct two new SCOPs: $\mathcal{P}_{\text{lb}} = \langle V, S, D, P, C, \beta^1_c, L, f \rangle$, where for all $c \in C$, $\beta^1_c = \beta_c + \vartheta$; and $\mathcal{P}_{\text{ub}} = \langle V, S, D, P, C, \beta^2_c, L, f \rangle$, where for all $c \in C$, $\beta^2_c = \beta_c - \vartheta$.

Proposition 9. *An (α, ϑ)-solution set to $\mathcal{P}_{lb}$ underestimates the true optimal profit with probability greater or equal to α; an (α, ϑ)-solution set to $\mathcal{P}_{ub}$ overestimates the true optimal profit with probability greater or equal to α.*

Proof. The proof follows from Definition 5. □

Proposition 9 can be exploited to generate a confidence interval for the true optimal profit via a binomial reasoning. We solve M independently generated instances of $\mathcal{P}_{\text{lb}}$ and store the optimal profit obtained for each of these instances into an array K_{lb} sorted in ascending order; we solve M independently generated instances of $\mathcal{P}_{\text{ub}}$ and store the optimal profit obtained for each of these instances into an array K_{ub} sorted in ascending order. Let $\text{bin}^{-1}(M, \alpha)$ be the inverse cumulative distribution of a binomial distribution with M trials and a success probability α; let k_{lb} be the $(1-\alpha)/2$-quantile of this distribution; finally, let k_{ub} be the $1 - (1-\alpha)/2$-quantile of $\text{bin}^{-1}(M, 1-\alpha)$. With confidence α element at position k_{lb} of K_{lb} is a lower bound and element at position $k_{\text{ub}} + 1$ of K_{ub} is an upper bound to the true optimal cost.[4]

Example 7. We transform the SCSP in Example 5 into two SCOPs $\mathcal{P}_{\text{lb}}$ and $\mathcal{P}_{\text{ub}}$ that maximise the objective function $f(X_1, X_2) = X_1 + 2X_2$. In other words, we assume the profit per unit of X_1 is 1 and the profit per unit of X_2 is 2. By

[4] Elements of K_i are indexed as follows: $1, \ldots, |K_i|$. Note that in statistics the k^{th}-smallest value of a statistical sample is known as k^{th} order statistic [20].

Table 1
Type I and Type II errors in statistics.

	H_0 is true	H_0 is false
Reject H_0	Type I error (false positive)	Correct outcome (true positive)
Fail to reject H_0	Correct outcome (true negative)	Type II error (false negative)

choosing $M = 20$ we obtain the α confidence interval $(282, 304)$ for the true optimal profit 293; if we reduce ϑ to 0.01 the interval shrinks considerably to $(290, 295)$.

If the objective function is stochastic there is no unique way to proceed. For instance, based on the available samples one may derive standard confidence intervals for the expected value of a stochastic expression based on the Student's t distribution and then compare solutions or partial assignments by comparing upper or lower limits of these intervals. The decision maker must of course choose a suitable confidence level α associated with this estimation. An example of a filtering algorithm that may be employed in such context is discussed in Appendix A. This algorithm is designed to handle the situation in which the objective is to minimise/maximise the expected value of some expression involving decision and random variables. Different algorithms must be designed if the objective involves a different operator, e.g. variance. Our algorithm distinguishes the case in which we are trying to determine an upper or a lower bound for the expected cost of an optimal solution. It then exploits the sampling distribution (i.e. Student's t distribution) of the expected total profit/cost and filters values based on upper/lower confidence limits obtained via this distribution. For instance, if our aim is to determine an upper bound for the optimal profit (problem type $\mathcal{P}_{\text{ub}}$), our algorithm will simply compare the upper confidence limits of the expected profit of two assignments and retain the assignment with the highest upper confidence limit. We will make use of this propagator to solve the models discussed in Section 8.

Finally, one should note that an alternative strategy may instead compare not only the upper confidence limits, but the whole intervals. An assignment would then provide a lower/higher profit than another if and only if their profit confidence intervals do not overlap. However, due to the complexity of the filtering logic that would be required in this case, we prefer to leave this discussion as future work.

7. Connections with statistics

To better understand the concepts just introduced, it is worth discussing the connection between the approach introduced and hypothesis testing in statistical analysis. Let us assume that our *null hypothesis* (H_0), in statistical sense, is that an assignment is feasible. According to classical hypothesis testing we may have four cases, as illustrated in Table 1. We may have a feasible assignment at hand (H_0 true) and we may incorrectly filter it (Type I error); or we may be operating on an infeasible assignment (H_0 false) and we may fail to reject it (Type II error).

In clinical trials or quality control, it is key to control the rate of Type I errors. It is undesirable to put under treatment a healthy a patient or to discard a functioning expensive machine. However, there are cases in which controlling Type II errors is essential. For example, aerospace engineers would prefer to scrap a functioning electronic circuit than to use one that is actually broken on a spacecraft; in such a situation a Type I error raises the budget, but a Type II error would put at risk the entire mission. In general, minimising Type I and Type II errors is not a simple matter. If one tries to reduce the rate of occurrence for Type I errors, the direct consequence is typically an increase in the observed rate for Type II errors and vice-versa. So in practice, one tries to control either Type I or Type II errors and, if the rate of the type that is not controlled is too high, then one increases the sample size.

In our specific case it is clearly essential to control the rate of Type II errors, which are more delicate than Type I errors. Making a Type II error means retaining an infeasible assignment, which is what we want to avoid as much as possible. Making a Type I error means discarding a feasible solution, which may impact optimality for an optimisation problem, or may lead to an empty solution space. Since our approach is essentially a heuristic, it is clear that both these issues — a poor solution quality or an empty solution space — are acceptable and should be dealt with by increasing the number of samples.

8. Computational experience

The aim of this section is to provide numerical insights on the theoretical framework introduced and particularly on the concept of (α, ϑ)-solution set and on its applications to find approximate solution to SCSPs and SCOPs. In our numerical study we will consider three well-known problems: the static stochastic knapsack (Section 8.1), the stochastic multiprocessor scheduling problem with release time and deadlines (Section 8.2), and the static stochastic lot-sizing problem (Section 8.3). The first and the third problems are single-stage, while the second is two-stage. In Section 8.4 we will generate approximate (α, ϑ)-solution sets using Definition 7 for the first two problems and show numerically that, with probability greater than or equal to α, the approach we discussed generates solution sets that satisfy chance constraints in the model with a margin of

Constraints:

(1) $\Pr\{s_1^k x_1 + \ldots + s_N^k x_N \leq C^k\} \geq \beta \quad k = 1, \ldots, L$

(2) $\Pr\{s_1^k x_1 + \ldots + s_N^k x_N \geq C^k\} \geq \beta \quad k = L+1, \ldots, G$

Decision variables:

$x_i \in \{0, \ldots, D\} \quad i = 1, \ldots, N$

Random variables:

$s_i^k \leftarrow \text{Poisson}(\lambda_i^k) \quad i = 1, \ldots, N; k = 1, \ldots, G$

Stage structure:

$V_1 = \{x_1, \ldots, x_N\}$

$S_1 = \{s_1^1, \ldots, s_n^k, \ldots, s_N^G\}$

$L = [\langle V_1, S_1 \rangle]$

Fig. 8. The static stochastic multiple knapsack as an SCSP.

error ϑ. In Section 8.5 we will numerically illustrate that the upper and lower profit/cost bounds obtained with the approach outlined in Section 6 comply with the prescribed confidence level α. We will also show the behaviour of the optimality gap as a function of the chosen error threshold ϑ and number M of independently generated instances of $\mathcal{P}_{\text{lb}}$ and $\mathcal{P}_{\text{ub}}$. Finally, in Section 8.6 we will investigate computational efficiency and scalability. All our experiments were performed by using Choco [21] on an Intel Xeon(r) CPU @ 3.50 Ghz with 16 GB of RAM.

8.1. Static stochastic knapsack

The knapsack problem [22] is a well-known combinatorial optimisation problem. The decision maker is given a set of objects each of which is associated with a weight and a profit. The aim is then to select a subset of these objects that fit into a given capacity and bring the maximum profit. There are several possible stochastic variants of the knapsack problem. Stochastic versions of the knapsack problem can be classified into static or dynamic. In the static stochastic knapsack problem, see e.g. [23], object weights and/or profits are random and the decision maker must choose, before observing any of their weights/profits, a subset of these objects that maximises a given objective, e.g. the expected profit, while meeting a restriction, e.g. a chance constraint, on the given capacity. Conversely, in the dynamic stochastic knapsack, see e.g. [24], the decision maker selects an object and immediately observes its weight and/or profit; based on this information she can then decide whether to select or not other objects.

In our computational study we will consider the SCSP presented in Fig. 8, i.e. a static stochastic multiple knapsack (SSMKP). In this problem we have a set of N types of objects; there are D objects of type i available. Each object of type i is associated random "coefficients" s_i^k that appear in the context of G chance constraints — this set of coefficients is generally denoted as *stochastic technology matrix* [25]; without loss of generality, these coefficients follow a Poisson distribution with mean λ_i^k.[5] The first L of these chance constraints are of type (1), i.e. they can be seen as "capacity restrictions" with respect to a target capacity C_k, and they should be satisfied with probability β. In the context of the first L chance constraints s_i^k represents the "weight" of item i in chance constraint k. The remaining G–L chance constraints are of type (2), i.e. they can be seen as "minimum production requirements" with respect to a target level C_k, and again they should be satisfied with probability β. In the context of the remaining G–L chance constraints s_i^k represents the "production contribution" of item i in chance constraint k.

Our aim is to determine the feasible region of the problem, i.e. the set of assignments that satisfy constraints (1) and (2).

We will also consider an optimisation version of the problem (Fig. 9) in which our aim is to determine what subset of the N objects in the problem maximises the expected total profit while satisfying all chance constraints. For each object i, we therefore introduce a random "profit" p_i, which follows a Poisson distribution with mean π_i; once more the choice of the distribution is made without loss of generality.

8.2. Stochastic multiprocessor scheduling problem with release time and deadlines

We consider a multiprocessor scheduling problem (MPSP, see [27], p. 238). The problem consists in finding a feasible schedule to process a set of K orders (or jobs) using m processors, where $m \leq P$. Processing an order k can only begin after the release date r_k and must be completed at the latest by the due date d_k. Order k requires a certain capacity c_k — expressed in terms of the number of processors — to be processed. The processing time of order k is t_k. The problem just described is well known in scheduling and it is fully deterministic and can easily and compactly be modelled using the `cumulative` constraint [28]. Let the height of a task k be c_k. This constraint considers a set of tasks and enforces that at each point in time the total height of the set of tasks that overlap that point does not exceed a given limit m. A task k

[5] For a discussion on statistic stochastic knapsack problems with Poisson resource requirements see e.g. [26].

Objective:
(1) $\max \mathrm{E}[p_1 x_1 + \ldots + p_N x_N]$

Constraints:
(2) $\Pr\{s_1^k x_1 + \ldots + s_N^k x_N \leq C^k\} \geq \beta \qquad k = 1, \ldots, L$

Decision variables:
$x_i \in \{0, \ldots, D\} \qquad i = 1, \ldots, N$

Random variables:
$s_i^k \leftarrow \text{Poisson}(\lambda_i^k) \qquad i = 1, \ldots, N; k = 1, \ldots, L$
$p_i \leftarrow \text{Poisson}(\pi_i) \qquad i = 1, \ldots, N$

Stage structure:
$V_1 = \{x_1, \ldots, x_N\}$
$S_1 = \{s_1^1, \ldots, s_n^k, \ldots, s_N^L\}$
$L = [\langle V_1, S_1 \rangle]$

Fig. 9. The static stochastic multiple knapsack as an SCOP.

Constraints:
(1) $\Pr\{\text{cumulative}(s, e, t, c, m)\} \geq \beta$

Decision variables:
$s_k \in \{r_k, \ldots, d_k\}, \qquad \forall k \in 1, \ldots, K$
$e_k \in \{r_k, \ldots, d_k\}, \qquad \forall k \in 1, \ldots, K$

Stochastic variables:
$t_k \rightarrow \text{Poisson}(\lambda_k) \qquad \forall k \in 1, \ldots, K$

Stage structure:
$V_1 = \{s_1, s_2, \ldots, s_K\} \quad S_1 = \{t_1, t_2 \ldots, t_K\}$
$V_2 = \{e_1, e_2, \ldots, e_K\} \quad S_2 = \{\}$
$L = [\langle V_1, S_1 \rangle, \langle V_2, S_2 \rangle]$

Fig. 10. An SCSP for the stochastic multiprocessor scheduling problem with release time and deadlines.

overlaps a point i if and only if its origin s_k is less than or equal to i, and its end e_k is strictly greater than i. This constraint also imposes, for each task k, the constraint s_k+t_k=e_k.

However, in reality, some parameters of this problem are uncertain in nature. Jobs may take longer than expected, some processors may break down and become unavailable, the release and due dates may be delayed, etc. In order to better model this problem a number of stochastic generalisations may be considered such as uncertain release date r_k; uncertain due date d_k; uncertain processing capacity c_k; uncertain processing time t_k; and uncertain number m of available processors; and every possible combination stemming from these cases.

We will consider the following stochastic constraint programming formulation of the stochastic multiprocessor scheduling problem (SMPSP), in which only processing time t_k for order k is uncertain; this is shown in Fig. 10. In this model, decision variables s_k and e_k denote the start time and the completion time of each job k, respectively. The processing time t_k of each job k is modelled as a Poisson distributed random variable with mean λ_k. In contrast to the problem presented in Section 8.1, this model is a two-stage SCSP. In the first stage, we decide on the start time of each job then we observe the realisation of the processing time. In the second stage the completion times are decided. Under this stage structure, constraint (1) enforces that the *probability* of not exceeding the given deadline for each job and the number of available processors m stays above the specified threshold β. More specifically, this constraint is a global chance constraint embedding a well-known global constraint: the `cumulative` constraint [28]. This constraint can be filtered using the general purpose method discussed in [4].

In our computational study we will also consider an optimisation version of the above problem in which we aim to minimise the latest start time.

8.3. Static stochastic lot-sizing

The last problem we will consider in our computational study is the single-item stochastic lot-sizing problem introduced in [29]. A SCOP for this problem is shown in Fig. 11. The decision maker faces a finite horizon of T periods and a random demand d_t in each period; which, without loss of generality, we will consider Poisson distributed with mean λ_t. There is a fixed cost a for placing an order of size $0 < Q_t \leq C$ in period t. An order placed in period t is delivered immediately at the beginning of the period, before demand occurs. Binary decision variable δ_t is set to zero if no order is placed (3). There

Objective:
(1) $\min \mathrm{E}[\sum_{t=1}^{N}(a\delta_t + h\sum_{j=1}^{t}(Q_t - d_t))]$

Constraints:
(2) $\Pr\{\sum_{j=1}^{t}(Q_t - d_t) \geq 0\} \geq \beta$ $\quad t = 1, \ldots, T$
(3) $\delta_t = 0 \Longrightarrow Q_t = 0$ $\quad t = 1, \ldots, T$

Decision variables:
$\delta_t \in \{0, 1\}$ $\quad t = 1, \ldots, T$
$Q_t \in \{0, \ldots, C\}$ $\quad t = 1, \ldots, T$

Random variables:
$d_t \leftarrow \mathrm{Poisson}(\lambda_t)$ $\quad t = 1, \ldots, T$

Stage structure:
$V_1 = \{Q_1, \ldots, Q_T, \delta_1, \ldots, \delta_T\}$
$S_1 = \{d_1, \ldots, d_T\}$
$L = [\langle V_1, S_1 \rangle]$

Fig. 11. The stochastic lot-sizing problem in [29] as an SCOP (static uncertainty strategy).

is a holding cost h charged on items that are carried over from one period to the next. Finally, the decision maker must comply with a service level restriction (2) stating that the net inventory at the end of each period should be nonnegative with probability at least β. The aim is to meet these service level restrictions while minimising the expected total cost (1).

The authors in [29] describe a range of control policies that can be used to control such a system. In our study, we will adopt the static uncertainty policy, which fixes all Q_t and δ_t at the beginning of the planning horizon, before demand is observed. Note that other strategies discussed in [29], i.e. dynamic uncertainty and static-dynamic uncertainty, can be easily captured by modifying the stage structure of the SCOP. In what follows, we shall refer to this problem as the static stochastic lot-sizing problem (SSLSP).

8.4. Feasibility

In Section 5 we introduced the notion of approximate (α, ϑ)-solution set. We will now present a computational analysis for the SCSPs presented in Sections 8.1 and 8.2 demonstrating that, with probability α, our approach generates solution sets that satisfy chance constraints in the model with a margin of error ϑ.

We considered thirty randomly generated small instances of the single stage problem in Fig. 8 in which $N = 2$, $L = 2$, $G = 3$, $D = 250$ and $\beta = 0.7$. Means λ_i^k of random variables in the model were integer numbers uniformly distributed between 10 and 20 for constraints (1) and between 20 and 30 for constraints (2). Right hand side constants C^k were integer numbers uniformly distributed between 1500 and 2000 for constraints (1) and between 2500 and 3000 for constraints (2). We fixed $\alpha = 0.9$ and $\vartheta = 0.2$; according to Definition 7 this choice led to a sample size of 31.

We also considered thirty randomly generated small instances of the two stage problem in Fig. 10 in which $K = 2$ and $\beta = 0.6$; r_k and d_k, which represent job k release time and deadline, were set to 0 and 4, respectively. Capacity requirements c_k were generated as integer numbers uniformly distributed between 1 and 2. Finally, expected task durations λ_k were generated as uniformly distributed numbers between 1 and 3; the maximum number of processors P was set to 3. We fixed $\alpha = 0.9$ and $\vartheta = 0.35$, this choice led to a sample size of 6.

Instances were small since in our analysis we generated the complete set of feasible assignments of the respective sampled SCSP, i.e. an (α, ϑ)-solution set, which for the two-stage problem in Fig. 10 was generally extremely large, in the order of tens of thousands of solutions. Feasibility of each of these assignment with respect to the original SCSP was then assessed via Monte Carlo simulation; the number of simulation runs was set in such a way as to guarantee a margin of error of $\vartheta/10$ with a confidence level of 0.9 — so that the Monte Carlo simulation error is an order of magnitude smaller than the approximation error associated with the (α, ϑ)-solution set obtained.

To numerically investigate if those computed are effectively (α, ϑ)-solution sets, for each of the above sixty instances, we repeatedly solved 1000 sampled SCSPs and computed the frequency of event e: "**all feasible assignments** of the sampled SCSP are feasible with respect to the original SCSP within the given tolerance threshold ϑ." In Fig. 12, for both problems and for each instance, we report the frequency of event e and the associated confidence intervals (confidence level of 0.95). These frequencies, are in line with the claim that those computed are (α, ϑ)-solution sets, for the given $\alpha = 0.9$. Note that our aim is to control Type-II errors (an infeasible assignment regarded as feasible), and not Type-I errors (a discarded and yet feasible assignment); for this reason if the sampled SCSP admitted no solution, this was regarded as a degenerate case in which all feasible assignments (i.e. none) of the sampled SCSP were feasible with respect to the original SCSP within the given tolerance threshold ϑ. Finally, it is worth observing that some of the frequencies observed in Fig. 12 are strictly greater than the prescribed value α. This is due to the fact that only assignments providing a satisfaction probability of exactly $\beta - \vartheta$ are correctly classified as infeasible with probability α. However, given the discrete nature of the assignment space, it is likely that instances may not feature any such assignment. Assignments providing a satisfaction probability

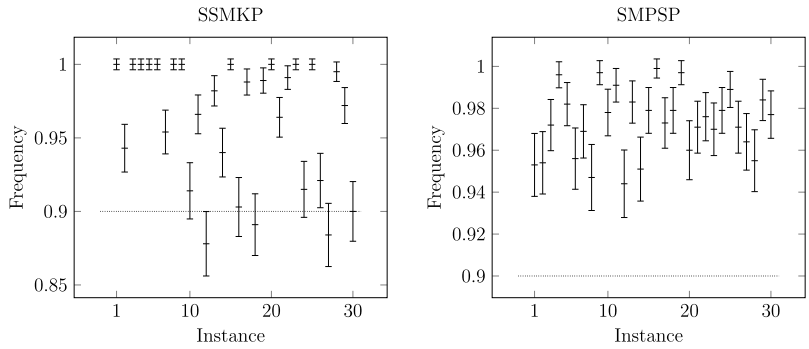

Fig. 12. Frequency of event "all feasible assignments of the sampled SCSP are feasible with respect to the original SCSP within the given tolerance threshold ϑ" over 1000 sampled SCSPs; together with the frequency, we report the associated confidence interval (confidence level of 0.95).

strictly less than $\beta - \vartheta$ are correctly classified as infeasible with probability strictly greater than α. In addition to this, when a model features multiple chance constraints, Bonferroni's correction, which is free of correlation and distribution assumptions, might generate a conservative — i.e. strictly larger than needed — sample size.

8.5. Optimality

We considered fifty randomly generated small instances of the problem in Fig. 9 (SSMKP) in which $N = 10$, $L = 2$, $D = 1$ and $\beta = 0.9$. Means λ_i^k of random variables in the model were integer numbers uniformly distributed between 10 and 20 for constraints (1). Right hand side constants C^k were integer numbers uniformly distributed between 100 and 200 for constraints (1). Means π_i were all set to 10.

We also considered fifty randomly generated small instances of the problem in Fig. 11 (SSLSP) in which $T = 5$, $h = 1$, $a = 10$, $C = 100$, and $\beta = 0.9$. Means λ_i^t of Poisson demand in each period $t = 1, \ldots, T$ were integer numbers uniformly distributed between 5 and 10.

We fixed $\alpha = 0.9$, $\vartheta = 0.05$ and $M = 10$; recall that M is the number of independently generated instances of $\mathcal{P}_{\text{lb}}$ and $\mathcal{P}_{\text{ub}}$ used for computing profit/cost upper and lower bounds as illustrated in Section 6. This led to a sample size of 209 for the SSMKP and of 370 for the SSLSP (Definition 7).

Due to the small size of the SSMKP instances, we managed to obtain optimal solutions by exhaustive enumeration, i.e. we generated all possible assignment and then checked feasibility and expected total profit of each of them via Monte Carlo simulation. The number of Monte Carlo runs was set to guarantee a margin of error of $\vartheta/10$ with a confidence level of 0.9, in such a way as to ensure an approximation error negligible with respect to the chosen ϑ. SSLSP instances can be solved to optimality by using a deterministic equivalent mixed integer linear programming model [30]. In our analysis, we can therefore compare results obtained with our approach against the true optimal solutions.

In Fig. 13, for each instance, we plotted upper and lower bound obtained for its optimal profit (SSMKP) or cost (SSLSP). For clarity, the interval has been normalised by using the profit/cost of the true optimal solution as a normalisation factor, so that value one in the graph denotes the true optimal profit/cost. The confidence level achieved by using our approach is generally higher than the prescribed α. In fact, despite α being set to 0.9, over the hundred instances analysed, the cost confidence interval did not cover the true optimal cost only in one case (SSMKP, instance 21). This is due to the conservative nature of our approach, as already discussed in Section 8.4.

We believe the fluctuations in the size of optimality gaps observed in Fig. 13 for the SSMKP may be related to the fact that this problem features 0–1 integer variables. Depending on the specific instance being solved, different sets of samples may lead to assignments in which "high value" objects belonging to the true optimal solution of the problem are not selected. This may lead to larger optimality gaps than those observed for other instances in which the optimal solution is less sensitive to random fluctuations produced by the sampling process.

Finally, we included in the analysis randomly generated instances of the SMPSP formulated as an SCOP in which the objective is to minimise the latest start time. In these instances $K = 5$ and $\beta = 0.6$; r_k and d_k, which represent job k release time and deadline, were all set to 0 and 20, respectively. Capacity requirements c_k where generated as integer numbers uniformly distributed between 1 and 3. Expected task durations λ_k were generated as uniformly distributed numbers between 1 and 5; the maximum number of processors P was set to 5.

In Fig. 14 we analysed the behaviour of the optimality gap when $\alpha = 0.9$, $M = 10$ and ϑ varies. The sample size ranges as follows: from 209 ($\vartheta = 0.05$) to 5838 ($\vartheta = 0.01$) for the SSMKP; from 370 ($\vartheta = 0.05$) to 6350 ($\vartheta = 0.01$) for the SSLSP; and from 14 ($\vartheta = 0.3$) to 114 ($\vartheta = 0.1$) for the SMPSP. For each value of ϑ considered, we solved 50 different instances of the SSMKP, SSLSP and SMPSP, and we computed the average optimality gap over this pool of instances. The average optimality gap for the SSMKP and the SSLSP is reported in percentage of the true optimal solution. For the case of the SMPSP unfortunately we were not able to compute the true optimal plan, therefore we reported the optimality gap in absolute terms; since we are minimising the latest start time, we expressed the optimality gap in expected number of

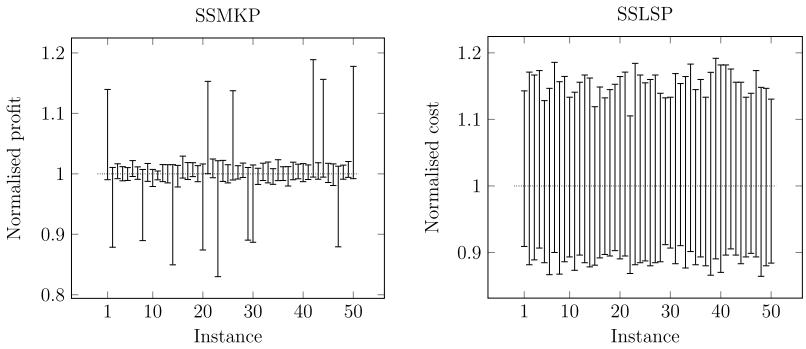

Fig. 13. Normalised profit/cost upper and lower bounds for fifty SSMKP and SSLSP instances; a value of 1 denotes the true optimal profit/cost.

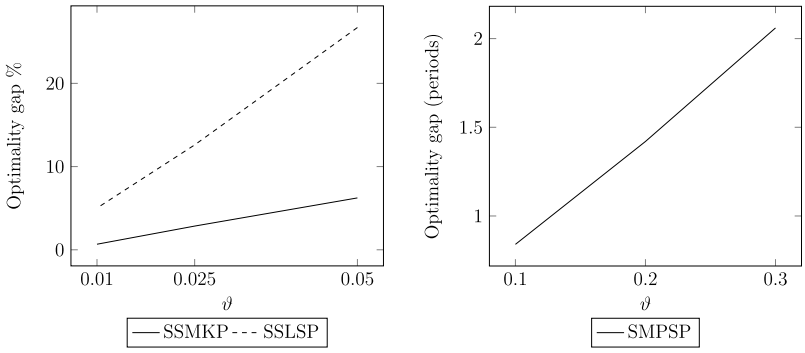

Fig. 14. Average optimality gap for different values of ϑ.

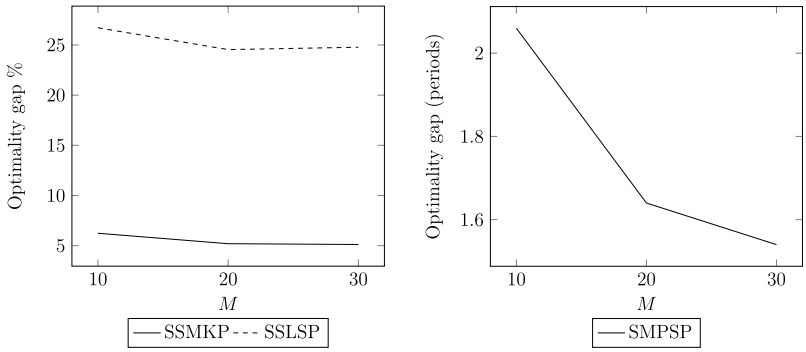

Fig. 15. Average optimality gap for different values of M.

periods. Note that since α and ϑ are linked to the number of samples generated by the relation in Definition 7, similar plots may be obtained by varying α and keeping ϑ fixed.

In Fig. 15 we carried out a similar analysis by keeping ϑ fixed to 0.05 (SSMKP, SSLSP) and to 0.3 (SSMKP) and by varying M.

8.6. Computational efficiency

In this section we reflect on the computational complexity and on the scalability of our approach.

8.6.1. Computational complexity

The computational complexity of SCSPs has been discussed in several works [2–4]; in particular we direct the interested reader to [31,32], which provide comprehensive overviews on the complexity of stochastic programs.

Multi-stage stochastic programming with discretely distributed decision-dependent random variables is PSPACE-hard [32]; the result follows from the PSPACE-hardness of the problem "decision-making under uncertainty" in [33]. SCSPs are PSPACE-complete in general if random variables are defined on discrete supports [3]. However, as pointed out in [32],

Table 2
SSMKP. Larger instances comprising $N = 20$ objects.

Instance	Optimality gap %	Runtime (hours)
1	0.49	1.65
2	0.66	0.98
3	0.57	1.20
4	0.71	0.65
5	0.53	1.77
6	0.51	2.18
7	0.50	2.22
8	0.55	0.77
9	0.44	2.63
10	0.70	0.80
Mean	0.57	1.49

Table 3
SMPSP. Larger instances comprising $K = 10$ jobs.

Instance	Latest start time		Runtime (hours)
	LB	UB	
1	13	15	1.50
2	14	16	0.68
3	11	12	0.72
4	13	14	3.20
5	17	21	0.31
6	17	19	3.27
7	18	20	9.08
8	13	14	0.43
9	13	14	0.16
10	17	20	11.11
Mean			3.05

the complexity of the "standard" multi-stage stochastic programming problem, in which distributions are independent of decisions taken in earlier stages, remains open; the authors in [32] conjecture that this is also PSPACE-hard.

The advantage of sampled SCSPs over generic SCSPs is that sampled SCSPs always comprise a finite number of scenarios whose number is determined by Definition 7, which establishes a relationship among α, ϑ, and N. A decision maker is then free to fix a pair of these values and to derive the remaining one. In principle, one may fix a priori the number of samples N — rather than the confidence level α or the error threshold ϑ — and sacrifice precision for efficiency. This will not make the sampled SCSP fixed-parameter tractable in general — in [4] the authors proved that maintaining GAC on a global chance constraint can be intractable even when maintaining GAC on the corresponding deterministic version of that constraint is tractable — but it may reduce its complexity from PSPACE to NP-hard.

8.6.2. Scalability

To illustrate the scalability of our approach with respect to other state-of-the-art approaches to SCSPs we employ, once more, the SSMKP. Note that this problem is similar to the one discussed in [4, Section 8.3]. In Section 9.4 of the same work, it was discussed that — for this class of problems — even when profits and weights of the objects are defined on a support that comprises only two values, a scenario-based formulation would end up comprising 2^{20} scenarios and a solver such as Choco would run out of memory. It was then shown that the complete approach discussed in that work could solve an instance comprising 10 objects in about an hour on average.

We fixed $\alpha = 0.9$, $\vartheta = 0.01$ and $M = 10$. We solved ten instances of the SSMKP randomly generated as discussed in Section 8.5, but now comprising $N = 20$ rather than ten objects; this led to a sample size of 6916. In Table 2 we report the optimality gap and the runtime for each of these instances as well as the runtime in hours.

Finally, we fixed $\alpha = 0.9$, $\vartheta = 0.1$ and $M = 10$, and we solved larger instances of the SMPSP formulated as an SCOP. These instances comprise ten jobs (i.e. $K = 10$). r_k and d_k were now set to 0 and 30, respectively; this led to a sample size of 142. In Table 3 we report upper and lower bound for the latest start time associated with each of these instances, as well as and the runtime in hours.

It is clear that it would be impossible to directly use the approach in [4] to model these SSMKP instances, as random variables follow a Poisson distribution and therefore have infinite values in their support. Even if one discretises these supports, e.g. by reducing them to only two values, the resulting SCSP would feature millions of scenarios.

However, one may argue that, in the case of the SSMKP, we are analysing a problem that could be analysed by brute force. In other words, one may as well generate all 2^{20} possible assignments for the decision variables and then analytically check the feasibility of each. Unfortunately, it is clear that this is not possible for the two-stage SMPSP just analysed, which features a much larger search space.

These results therefore demonstrate that the discussion in this work provides a viable means for scaling up the approach in [4].

9. Related works

Confidence-based optimisation was originally introduced in [34]. In this work, the authors discuss an application of this methodology in the context of a well-known stochastic inventory control problem. Our work extends the discussion presented there to generic SCSPs and SCOPs by introducing a more general notion of confidence-based reasoning based on two novel concepts: (α, ϑ)-solutions and (α, ϑ)-solution sets. In the context of stochastic modelling and optimisation, as discussed, these tools can be employed to find approximate solutions that possess given statistical properties.

9.1. Related works in stochastic programming

In operations research, and particularly in stochastic programming, the state-of-the-art technique that applies sampling in combinatorial optimisation is the sample average approximation (SAA) method [23]. This is a Monte Carlo simulation-based approach to stochastic discrete optimisation problem. This method replaces the actual distribution of random variables in the combinatorial problem of interest by an empirical distribution obtained via sampling. The obtained "sample average optimisation problem" is then solved and the procedure is repeated multiple times until a given termination criterion is satisfied. The authors in [23] focus on stochastic programs with expected value objectives and discuss convergence rate and stopping rules. In [35] the authors extend their analysis to two-stage stochastic programs with integer recourse; for this latter class of problems [36] carry out a post-hoc computationally intensive analysis of the quality of solutions obtained via SAA. Extensions to problems with expected value constraints, e.g. conditional value-at-risk constraints, were discussed in [37]. However, none of these works investigated the case in which the problem of interest include chance constraints. As [38] remarks, there are formulations of stochastic programming problems that incorporate expectations of penalised constraints in the objective function as a penalty terms. These problems can be solved efficiently since they simply require continuous variables for modelling penalties and they do not require any additional binary variable. However, this modelling approach does not address the issue of finding or approximating feasible or optimal solutions to a chance constrained problem [39, p. 950].

SAA methods for problems comprising a single chance constraint were discussed in [40–42]. In [40] the authors summarise convergence properties (Section 2.1) and post-hoc solution validation strategies (Section 2.2). They remark that "based on this [convergence] analysis, we can compute a priori the sample size required in the SAA problem so that it produces a feasible solution to the true problem with high probability (typically such estimates of a required sample size are quite conservative)." The convergence analysis the authors refer to was originally conducted in [41] and it shows asymptotical convergence properties based on inequalities such as Chernoff's [43] or Hoeffding's [44], which are known to be conservative bounds. Their analysis is conducted under the assumption that the feasible region is finite, since the sample size determined via the aforementioned convergence properties grows linearly in the size of the feasible region [41, p. 683]. Extensions of the analysis in [41] to the case of multiple chance constraints were illustrated in [38]. This latter work is similar to those just discussed, since once more the analysis is based on the above inequalities and the sample size depends on the size of the feasible region. More recently, [45] investigated the relations between chance constrained and penalty function problems under discrete distributions. This analysis extended a number of previous works that analysed this relation under continuous distribution. However, the authors explicitly remark that "our goal is not to show that the penalty problems are able to generate optimal values and solutions of chance constrained problems." Instead they compare the problems with focus on asymptotic equivalence of optimal values and corresponding convergence of optimal solutions.

After surveying the existing literature on SAA, the first important remark is that in none of the above works can we find concepts that resemble those of (α, ϑ)-solution and (α, ϑ)-solution set, which are unique to confidence-based reasoning [34]. This is a subtle conceptual difference that should not be overlooked. The aim of SAA is to find an assignment that, with prescribed confidence probability α, is a solution to the original problem; see e.g. [42, Section 3.1]. In other words, in SAA the decision maker does not fix any a priori tolerated estimation error ϑ. To ensure that the solution of the sampled problem is feasible with respect to the original problem with sufficiently high probability, in SAA the threshold β associated with chance constraints in the sampled problem is increased by a factor ϑ, which however is not explicitly interpreted as an error tolerance threshold in a statistical sense, although in practice it is used as such. In [42, p. 407], the authors point out that, for a fixed α and for a given threshold β, "it is not clear what the best choices for the sample size and ϑ are," since they believe this is a problem-dependent issue that should be addressed numerically. This statement demonstrates the aforementioned fundamental difference. By introducing the two concepts of (α, ϑ)-solution and (α, ϑ)-solution set, we suggest that a decision maker may — in line with established practices in statistics — interpret ϑ as an error tolerance threshold and fix a priori, together with the confidence level α, either the sample size (on the basis of the available observations) or ϑ (on the basis of the estimation error that can be tolerated); finally, the parameter that has not been fixed should be derived via the analysis we presented. In summary, the difference lies in the interpretation. Confidence-based reasoning aims to find a solution that, with confidence α, satisfies the chance constraints in the original problem within the given error tolerance ϑ. In addition to this important semantic difference, we should mention that our analysis is based on

the exact Clopper–Pearson confidence interval, and not on conservative bounds such as Chernoff's or Hoeffding's inequalities. Finally, our approximation strategy for (α, ϑ)-solution sets leads to a sample size (Definition 7) that is independent of the number of assignments in the feasible region; a major difference from all other methods surveyed so far.

To contrast our approach with respect to other existing state-of-the-art approaches, one may consider the stochastic vehicle routing problem with time windows discussed in [38, Section 4]. It is possible to apply our analysis to the instances discussed in [38, Table 1], by converting the parameters used in SAA and setting the confidence level $\alpha = 0.99$, the error threshold $\vartheta = 0.05$ and the chance constraint thresholds $\beta = 0.95$. For the instances with 10 customer orders, the sample size prescribed by our approach is 429 for the model with a single chance constraint and 490 for the model with three chance constraints. For the instances with 50 customer orders, the sample size prescribed by our approach is 608 for the model with a single chance constraint and 669 for the model with three chance constraints. Not only are these sample sizes orders of magnitude smaller than the ones suggested in [38], which range from 200 thousand up to 32 million; but most importantly they do not depend on the number of vessels used or the size of the time windows; in fact, according to Definition 7, they only depend on the number of random variables and chance constraints in the model. Of course, as the authors in [38] remark, finding an exact solution to a scenario-based model with 32 million scenarios is unrealistic. For this reason, they suggest to adopt heuristic solution methods, e.g. tabu search. As demonstrated in our computational study, our approximate (α, ϑ)-solution sets represent a viable alternative to the use of heuristics on instances featuring very large sample sizes.

9.2. Related works in constraint programming

A detailed discussion on hybrid CP/AI/OR approaches for decision making under uncertainty can be found in [46,47]. We direct the reader to these two references for further details on existing works in this research area. We next briefly survey key relevant references. Efforts that try to extend classical CSP framework to incorporate uncertainty have been influenced by works that originated in different fields, namely *chance-constrained programming* [48] and *stochastic programming* [49]. To the best of our knowledge the first work that tries to create a bridge between Stochastic Programming and Constraint Programming is by Benoist et al. [50]. Search and consistency strategies, namely a backtracking algorithm, a forward checking procedure [2] and an arc-consistency [51] algorithm have been proposed for SCSPs. A scenario-based approach for building up constraint programming models of SCSPs was proposed by Tarim et al. [3]. In the same work a fully featured language — Stochastic OPL — for modelling SCSPs was also proposed. In [52] the authors introduce new algorithms for solving multi-objective stochastic problems are proposed. Global chance constraints were introduced first in [53], and bring together the reasoning power of global constraints from CP and the expressive power of chance constraints from SP. A general purpose approach for filtering global chance constraints is proposed in [4,7]. This approach is able to reuse existing propagators available for the respective deterministic global constraint which corresponds to a given global chance constraint when all the random variables are replaced by constant parameters. In [54] the authors discuss some possible strategies to perform cost-based filtering for certain classes of SCOPs. These strategies exploit well-known inequalities borrowed from SP and used to compute valid bounds for any given SCOP that respects some mild assumptions. Unfortunately, the above approaches operate under the assumption that the number of scenarios must be finite, otherwise a solution cannot be expressed as a finite number of possible decisions. Furthermore, these approaches do not scale well. Even problems having a limited number of stochastic variables with large support immediately produce policy trees whose size makes impractical the use of a complete method. In [3] the authors employed sampling in order to reduce the number of scenarios considered for a given stochastic constraint program and produce a solution in reasonable time. Nevertheless, this approach does not provide any optimality/feasibility guarantee for the solution produced. Heuristic approaches such as the one in [55], in which a neural network is employed in order to encode a policy function, suffer from the same limitation and from lack of modularity. Stochastic sampling in the context of Stochastic Boolean Satisfiability was discussed in [56]; forward sampling [50] and sample aggregation [57] are two other techniques that have been employed to solve SCSPs. Nevertheless, none of these approaches introduce a concept that resembles that of (α, ϑ)-solution. Probably, the work discussed in [58] represents the closest attempt to provide some sort of guarantees for a stochastic constraint satisfaction problem. Nevertheless, this work is focused on a specific problem — a two-stage stochastic matching problem — and it does not propose a generic approach for solving SCSPs. Finally, another closely related work is [59], which discusses sample-based approaches to job shop scheduling with probabilistic durations; however, like in the previous case, the approach proposed is focused on a specific problem and not on solving generic SCSPs.

10. Conclusions

We proposed a framework for exploiting sampling in order to solve SCSPs that include random variables over a continuous or very large discrete support. Our framework is based on a number of novel concepts: sampled SCSPs, (α, ϑ)-solutions and (α, ϑ)-solution sets. We employed statistical estimation to determine if a given assignment is consistent with respect to a given set of chance constraints. As in statistical estimation, the quality of our estimate is determined via confidence interval analysis.

In contrast to existing approaches based on sampling, we provide likelihood guarantees for the quality of the solutions found. In fact, we explicitly state a confidence probability α that bounds the probability of exceeding a given error tolerance

threshold ϑ in our estimation. By properly choosing the estimation error ϑ and the confidence probability α it is possible to generate compact sampled SCSPs that can be effectively solved by existing solution methods. We also extended the reasoning to SCOPs and demonstrated how to produce statistical upper and lower bounds for the value of the optimal solution.

We demonstrated our approach on a number of SCSPs and SCOPs: the static stochastic knapsack problem, a stochastic multiprocessor scheduling problem, and a stochastic lot-sizing problem. Our computational study demonstrates the effectiveness of our approach.

We conclude by briefly discussing a number of suggestions for future work.

Online stochastic optimisation A promising direction is that of exploring synergies with online stochastic optimisation [50]. In particular, we suspect that our approach may be used to enhance the results in [57,60–62] by ensuring a better control of the solution quality obtained at each step of the online process.

Sampling strategies A key open issue is related to the fact that simple random sampling [63] is a relatively naive strategy for selecting samples. The use of more refined sampling strategies — for instance a stratified sampling technique such as Latin Hypercube Sampling [64] — may of course reduce the number of samples required to produce an (α, ϑ)-solution. Nevertheless, further research is required in order to clarify how stratified sampling can be effectively employed in this context.

Confidence intervals The Clopper–Pearson interval is an exact interval since it is based directly on the binomial distribution rather than any approximation to the binomial distribution. This interval, however, can be conservative because of the discrete nature of the binomial distribution, as pointed out by Neyman [65]. For example, the true coverage rate of a 95% Clopper–Pearson interval may be well above 95%, depending on n and q. Thus the interval may be wider than it needs to be to achieve 95% confidence. In contrast, it is worth noting that other approximate confidence bounds may be narrower than their nominal confidence width, i.e., the "normal approximation interval," also known as Wald confidence interval, the Wilson Interval, the Agresti–Coull Interval, etc, may in fact achieve a confidence level that is lower than the nominal one [11]. Future research may investigate the application of approximate intervals in the context of sample-based constraint solving. The performance of each of these approximate intervals have been thoroughly analysed in the existing body of literature. The advantage is that approximate intervals may lead to smaller sample sets and therefore to more compact sampled SCSPs.

Computational complexity Finally, an interesting computational complexity questions remains open about the complexity of the standard multi-stage stochastic constraint programs.

Acknowledgements

The authors would like to thank the Associate Editor and the anonymous reviewers for commenting on our drafts and providing insightful remarks.

Appendix A. Filtering strategy for constraint expressions involving expected values

We discuss a filtering strategy for handling constraint expressions involving expected values in sampled SCSPs. This filtering strategy can be employed, in concert with the approach discussed in Section 6, to deal with the case in which the objective function is stochastic. Consider a constraint $x = \mathrm{E}[\langle \exp \rangle]$, where E[] denotes the expectation operator and x is a real valued decision variable, whose domain is stored as an interval with real valued upper and lower bounds. Techniques for handling propagation and search involving real valued decision variables are discussed in [66]. A filtering algorithm that enforces bounds consistency on this constraint is shown in Algorithm 1. It should be noted that the approach discussed in Section 6 distinguishes two cases: the one in which our aim is to underestimates the true optimal profit (SCOP $\mathcal{P}_{\mathrm{lb}}$) and that in which our aim is to overestimates the true optimal profit (SCOP $\mathcal{P}_{\mathrm{ub}}$). The type of problem ($\mathcal{P}_{\mathrm{lb}}$ or $\mathcal{P}_{\mathrm{ub}}$) which the propagator belongs to must be specified as an input parameter "type" that influences propagation. The algorithm constructs two arrays: U and L. U lists, for each scenario, an upper bound for the expected value of $\langle \exp \rangle$, L lists, for each scenario, a lower bound for the expected value of $\langle \exp \rangle$. Then it exploits the Student's t distribution with $|\Psi| - 1$ degrees of freedom (StudentT($|\Psi| - 1$)) to determine upper and lower confidence limits for the expected value of $\langle \exp \rangle$ at the prescribed confidence level α. Note that $\mathrm{CDF}_t^{-1}(\alpha)$ denotes the inverse cumulative distribution function of t; mean(X) and std(X) denote the mean and the standard deviation of the elements in X, respectively. The algorithm operates by exploiting the structure Ψ of the policy tree; therefore it takes implicitly into account the stage structure of the problem while computing the expected value of a given expression and it will correctly evaluate expected values both in a single or multi-stage case. Finally, it is worth remarking that this constraint is closely related to the Student's t test constraint discussed in [67].

Algorithm 1: Filtering expected values in sampled SCSPs.

input : type; $\langle \text{exp} \rangle$; $\mathcal{T}$; x; α.
output: Bound consistent x.

begin
  $U \leftarrow \{\}$; $L \leftarrow \{\}$;
  for *each* $p \in \Psi$ **do**
    $U \leftarrow U \cup \text{Sup}(\langle \text{exp} \rangle_{\downarrow p})$;
    $L \leftarrow L \cup \text{Inf}(\langle \text{exp} \rangle_{\downarrow p})$;
  $t \leftarrow \text{StudentT}(|\Psi| - 1)$;
  if $type = \mathcal{P}_{lb}$ **then**
    $\text{Sup}(x) \leftarrow \text{mean}(U) - \text{CDF}_t^{-1}(1 - (1 - \alpha)/2) \cdot \text{std}(U)/\sqrt{|\Psi|}$;
    $\text{Inf}(x) \leftarrow \text{mean}(L) - \text{CDF}_t^{-1}(1 - (1 - \alpha)/2) \cdot \text{std}(L)/\sqrt{|\Psi|}$;
  else if $type = \mathcal{P}_{ub}$ **then**
    $\text{Sup}(x) \leftarrow \text{mean}(U) + \text{CDF}_t^{-1}(1 - (1 - \alpha)/2) \cdot \text{std}(U)/\sqrt{|\Psi|}$;
    $\text{Inf}(x) \leftarrow \text{mean}(L) + \text{CDF}_t^{-1}(1 - (1 - \alpha)/2) \cdot \text{std}(L)/\sqrt{|\Psi|}$;

References

[1] R. Rossi, B. Hnich, S.A. Tarim, S. Prestwich, Finding (α, ϑ)-solutions via sampled SCSP, in: T. Walsh (Ed.), Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011, July 16–22, Barcelona, Spain, AAAI Press, 2011, pp. 2172–2177.
[2] T. Walsh, Stochastic constraint programming, in: F. van Harmelen (Ed.), European Conference on Artificial Intelligence, Proceedings, ECAI'2002, IOS Press, 2002, pp. 111–115.
[3] S.A. Tarim, S. Manandhar, T. Walsh, Stochastic constraint programming: a scenario-based approach, Constraints 11 (2006) 53–80.
[4] B. Hnich, R. Rossi, S.A. Tarim, S. Prestwich, Filtering algorithms for global chance constraints, Artif. Intell. 189 (2012) 69–94.
[5] D. Costantini, Verso una rappresentazione probabilistica del mondo, Maria Margherita Bulgarini, Firenze, 2014.
[6] F. Rossi, P. van Beek, T. Walsh, Handbook of Constraint Programming (Foundations of Artificial Intelligence), Elsevier Science Inc., New York, NY, USA, 2006.
[7] B. Hnich, R. Rossi, S.A. Tarim, S.D. Prestwich, Synthesizing filtering algorithms for global chance-constraints, in: I.P. Gent (Ed.), 15th International Conference on Principles and Practice of Constraint Programming, Proceedings, CP 2009, Lisbon, Portugal, September 20–24, 2009, in: Lect. Notes Comput. Sci., vol. 5732, Springer, 2009, pp. 439–453.
[8] S.D. Prestwich, S.A. Tarim, R. Rossi, B. Hnich, Hybrid metaheuristics for stochastic constraint programming, Constraints 20 (2015) 57–76.
[9] H. Jeffreys, Theory of Probability, Clarendon Press, Oxford, UK, 1961.
[10] C.J. Clopper, E.S. Pearson, The use of confidence or fiducial limits illustrated in the case of the binomial, Biometrika 26 (1934) 404–413.
[11] A. Agresti, B.A. Coull, Approximate is better than "exact" for interval estimation of binomial proportions, Am. Stat. 52 (1998) 119–126.
[12] M. Evans, N. Hastings, B. Peacock, Statistical Distributions, Wiley, New York, 2000.
[13] I.C.G. Upton, Oxford Dictionary of Statistics, Oxford University Press, Oxford, UK, 2002.
[14] S. Sethi, G. Sorger, A theory of rolling horizon decision making, Ann. Oper. Res. 29 (1991) 387–416.
[15] S. Sadooghi-Alvandi, A. Nematollahi, R. Habibi, On the distribution of the sum of independent uniform random variables, Stat. Pap. 50 (2009) 171–175.
[16] R.G. Miller, Simultaneous Statistical Inference, Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 1981.
[17] F. Killmann, E. von Collani, A note on the convolution of the uniform and related distributions and their use in quality control, Econ. Qual. Control 16 (2001) 17–41.
[18] E.L. Lehmann, J.P. Romano, Generalizations of the familywise error rate, Ann. Stat. 33 (2005) 1138–1154.
[19] C.E. Smith, R.A. Cribbie, Multiplicity control in structural equation modeling: incorporating parameter dependencies, Struct. Equ. Model. 20 (2013) 79–85.
[20] H.A. David, H.N. Nagaraja, Order Statistics, 3rd ed., Wiley-Interscience, 2003.
[21] F. Laburthe, the OCRE project team, Choco: implementing a CP kernel, Technical report, Bouygues e-Lab, France, 1994.
[22] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, Inc., New York, NY, USA, 1990.
[23] A.J. Kleywegt, A. Shapiro, T. Homem-De-Mello, The sample average approximation method for stochastic discrete optimization, SIAM J. Optim. 12 (2001) 479–502.
[24] A.J. Kleywegt, J.D. Papastavrou, The dynamic and stochastic knapsack problem, Oper. Res. 46 (1998) 17–35.
[25] P. Kall, J. Mayer, Stochastic Linear Programming: Models, Theory and Computation, 2nd ed., Internat. Ser. Oper. Res. Management Sci., Springer, 2011.
[26] S. Ağralı, J. Geunes, A single-resource allocation problem with Poisson resource requirements, Optim. Lett. 3 (2009) 559–571.
[27] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.
[28] A. Aggoun, N. Beldiceanu, Extending chip in order to solve complex scheduling and placement problems, Math. Comput. Model. 17 (1993) 57–73.
[29] J.H. Bookbinder, J.Y. Tan, Strategies for the probabilistic lot-sizing problem with service-level constraints, Manag. Sci. 34 (1988) 1096–1108.
[30] V. Vargas, An optimal solution for the stochastic version of the Wagner–Whitin dynamic lot-size model, Eur. J. Oper. Res. 198 (2009) 447–451.
[31] H.B. Hunt, M.V. Marathe, R.E. Stearns, Complexity and approximability of quantified and stochastic constraint satisfaction problems, Electron. Notes Discrete Math. 9 (2001) 217–230.
[32] M. Dyer, L. Stougie, Computational complexity of stochastic programming problems, Math. Program. 106 (2006) 423–432.
[33] C.H. Papadimitriou, Games against nature, J. Comput. Syst. Sci. 31 (1985) 288–301.
[34] R. Rossi, S. Prestwich, S.A. Tarim, B. Hnich, Confidence-based optimisation for the newsvendor problem under binomial, Poisson and exponential demand, Eur. J. Oper. Res. 239 (2014) 674–684.
[35] S. Ahmed, A. Shapiro, E. Shapiro, The sample average approximation method for stochastic programs with integer recourse, SIAM J. Optim. 12 (2002) 479–502.
[36] J. Linderoth, A. Shapiro, S. Wright, The empirical behavior of sampling methods for stochastic programming, Ann. Oper. Res. 142 (2006) 215–241.
[37] W. Wang, S. Ahmed, Sample average approximation of expected value constrained stochastic programs, Oper. Res. Lett. 36 (2008) 515–519.
[38] M. Branda, Sample approximation technique for mixed-integer stochastic programming problems with several chance constraints, Oper. Res. Lett. 40 (2012) 207–211.
[39] M. Branda, Stochastic programming problems with generalized integrated chance constraints, Optimization 61 (2012) 949–968.

[40] S. Ahmed, A. Shapiro, Solving chance-constrained stochastic programs via sampling and integer programming, in: Z.-L. Chen, S. Raghavan (Eds.), Tutorials in Operations Research, INFORMS, 2008, pp. 261–269.
[41] J. Luedtke, S. Ahmed, A sample approximation approach for optimization with probabilistic constraints, SIAM J. Optim. 19 (2008) 674–699.
[42] B.K. Pagnoncelli, S. Ahmed, A. Shapiro, Sample average approximation method for chance constrained programming: theory and applications, J. Optim. Theory Appl. 142 (2009) 399–416.
[43] H. Chernoff, A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations, Ann. Math. Stat. 23 (1952).
[44] W. Hoeffding, Probability inequalities for sums of bounded random variables, J. Am. Stat. Assoc. 58 (1963) 13–30.
[45] M. Branda, On relations between chance constrained and penalty function problems under discrete distributions, Math. Methods Oper. Res. 77 (2013) 265–277.
[46] K.N. Brown, I. Miguel, Uncertainty and change, in: F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, Elsevier, 2006, pp. 729–753 (Chapter 21).
[47] B. Hnich, R. Rossi, S.A. Tarim, S. Prestwich, A survey on CP-AI-OR hybrids for decision making under uncertainty, in: P. van Hentenryck, M. Milano (Eds.), Hybrid Optimization, in: Springer Optim. Appl., vol. 45, Springer New York, New York, NY, 2011, pp. 227–270.
[48] A. Charnes, W.W. Cooper, Deterministic equivalents for optimizing and satisficing under chance constraints, Oper. Res. 11 (1963) 18–39.
[49] J.R. Birge, F. Louveaux, Introduction to Stochastic Programming, Springer Verlag, New York, 1997.
[50] T. Benoist, E. Bourreau, Y. Caseau, B. Rottembourg, Towards stochastic constraint programming: a study of online multi-choice knapsack with deadlines, in: T. Walsh (Ed.), Principles and Practice of Constraint Programming, Proceedings, CP 2001, in: Lect. Notes Comput. Sci., vol. 2239, Springer, 2001, pp. 61–76.
[51] T. Balafoutis, K. Stergiou, Algorithms for stochastic CSPs, in: F. Benhamou (Ed.), Principles and Practice of Constraint Programming, Proceedings, CP 2006, in: Lect. Notes Comput. Sci., vol. 4204, Springer, 2006, pp. 44–58.
[52] L. Bordeaux, H. Samulowitz, On the stochastic constraint satisfaction framework, in: SAC '07: Proceedings of the 2007 ACM Symposium on Applied Computing, ACM, New York, NY, USA, 2007, pp. 316–320.
[53] R. Rossi, S.A. Tarim, B. Hnich, S.D. Prestwich, A global chance-constraint for stochastic inventory systems under service level constraints, Constraints 13 (2008) 490–517.
[54] R. Rossi, S.A. Tarim, B. Hnich, S.D. Prestwich, Cost-based domain filtering for stochastic constraint programming, in: P.J. Stuckey (Ed.), 14th International Conference on Principles and Practice of Constraint Programming, Proceedings, CP 2008, Sydney, Australia, September 14–18, 2008, in: Lect. Notes Comput. Sci., vol. 5202, Springer, 2008, pp. 235–250.
[55] S.D. Prestwich, S.A. Tarim, R. Rossi, B. Hnich, Evolving parameterised policies for stochastic constraint programming, in: I.P. Gent (Ed.), 15th International Conference on Principles and Practice of Constraint Programming, Proceedings, CP 2009, Lisbon, Portugal, September 20–24, 2009, in: Lect. Notes Comput. Sci., vol. 5732, Springer, 2009, pp. 684–691.
[56] M.L. Littman, S.M. Majercik, T. Pitassi, Stochastic boolean satisfiability, J. Autom. Reason. 27 (2001) 251–296.
[57] P. van Hentenryck, R. Bent, Y. Vergados, Online stochastic reservation systems, in: J.C. Beck, B.M. Smith (Eds.), Third International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Proceedings, CPAIOR 2006, Cork, Ireland, May 31–June 2, 2006, in: Lect. Notes Comput. Sci., vol. 3990, Springer, 2006, pp. 212–227.
[58] I. Katriel, C. Kenyon-Mathieu, E. Upfal, Commitment under uncertainty: two-stage stochastic matching problems, in: L. Arge, C. Cachin, T. Jurdzinski, A. Tarlecki (Eds.), 34th International Colloquium on Automata, Languages and Programming, Proceedings, ICALP 2007, Wroclaw, Poland, July 9–13, 2007, in: Lect. Notes Comput. Sci., vol. 4596, Springer, 2007, pp. 171–182.
[59] J.C. Beck, N. Wilson, Proactive algorithms for job shop scheduling with probabilistic durations, J. Artif. Intell. Res. 28 (2007) 183–232.
[60] L. Michel, P.V. Hentenryck, Iterative relaxations for iterative flattening in cumulative scheduling, in: S. Zilberstein, J. Koehler, S. Koenig (Eds.), Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling, ICAPS 2004, Whistler, British Columbia, Canada, June 3–7, 2004, AAAI, 2004, pp. 200–208.
[61] R. Bent, P.V. Hentenryck, Regrets only! online stochastic optimization under time constraints, in: Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25–29, 2004, San Jose, California, USA, 2004, pp. 501–506.
[62] R. Bent, I. Katriel, P.V. Hentenryck, Sub-optimality approximations, in: P. van Beek (Ed.), 11th International Conference on Principles and Practice of Constraint Programming, Proceedings, CP 2005, Sitges, Spain, October 1–5, 2005, in: Lect. Notes Comput. Sci., vol. 3709, Springer, 2005, pp. 122–136.
[63] D.S. Yates, D.S. Starnes, D.S. Moore, The Practice of Statistics, W.H. Freeman & Co, 2002.
[64] M.D. McKay, R.J. Beckman, W.J. Conover, A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics 21 (1979) 239–245.
[65] J. Neyman, On the problem of confidence limits, Ann. Math. Stat. 6 (1935) 111–116.
[66] F. Benhamou, L. Granvilliers, Continuous and interval constraints, in: F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, Elsevier, 2006, p. 569.
[67] R. Rossi, S.D. Prestwich, S.A. Tarim, Statistical constraints, in: T. Schaub, G. Friedrich, B. O'Sullivan (Eds.), ECAI 2014 – 21st European Conference on Artificial Intelligence, Including Prestigious Applications of Intelligent Systems, PAIS 2014, Prague, Czech Republic, 18–22 August 2014, in: Front. Artif. Intell. Appl., vol. 263, IOS Press, 2014, pp. 777–782.