Cost-based domain filtering for stochastic constraint programming

Cost-based Domain Filtering for Stochastic Constraint Programming*

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

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

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

1 Introduction

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

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

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

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

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

2 Formal Background

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

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

In [19], a *stochastic CSP* is defined as a 6-tuple $\langle V, S, D, P, C, \theta\rangle$, where V is a set of decision variables and S is a set of stochastic variables, D is a function mapping each element of V and each element of S to a domain of potential values. A decision variable in V is *assigned* a value from its domain. P is a function mapping each element of S to a probability distribution for its associated domain. C is a set of constraints. A constraint $h \in C$ that constrains at least one

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

3 Value of Stochastic Solutions

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

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

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

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

using the expectation operator $\mathbb{E}$, as

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

with an optimal solution x^*.

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

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

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

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

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

Proof. A proof is given in [2].

It directly follows that $\mathrm{EV(P)} \geq \mathrm{RP(P)}$ ($\mathrm{EV(P)} \leq \mathrm{RP(P)}$). We will base our cost-based filtering strategies on this inequality.[2] Assumption (i) restricts the form of the cost function. As witnessed by much of the Stochastic Programming literature [4, 11], many real life applications exhibit such a behavior in the profit (cost) function. Nevertheless, it is often possible to encounter stochastic constraint programs whose objective exhibits a generalized non-convex dependence on the stochastic variables. Note that, although the classical Jensen (Proposition 1) and Edmundson-Madansky type bounds [4], which we will employ in the following sections, or their extensions are generally not available for such problems, tight bounds may still be constructed under mild regularity conditions as discussed in [13]. Assumption (ii) states that Proposition 1 provides a valid bound only when a feasible solution exists and its existence is not affected by the distribution of the stochastic variables. Intuitively, this means that nothing can be inferred by using Proposition 1 if EV(P) is infeasible or, clearly, if RP(P) is infeasible. Assumption (ii) may be violated in problems where chance-constraints appear. We will not discuss how to handle generic chance-constraints and how to produce deterministic equivalent reformulations for them in EV(P): the reader may refer to [6]. In this work we will consider only examples of stochastic COPs that always satisfy assumptions (i) and (ii). In particular, to comply

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

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

with assumption (ii), we will consider problems for which a feasible solution always exists and for which the chance-constraints are "hard" ($\theta = 1$). Note that "hard" chance-constraints in RP(P) become deterministic in EV(P).

4 Global optimization chance-constraints

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

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

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

In this section and in the following ones we will refer to a running example and we will employ the following problem to better understand the concepts explained. Consider the *Static Stochastic Knapsack Problem* (SSKP) [12]: a subset of k items has to be chosen, given a knapsack of size q into which to fit the items. Each item i has an expected reward of r_i. The size $\mathcal{W}_i$ of each item is not

Objective:

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

Decision variables:

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

Stochastic variables:

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

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

Objective:

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

Constraints:

(1) $Z_j \geq \sum_{i=1}^{k} \mathcal{W}_i^j X_i - q$ $\quad \forall j \in 1,\ldots,n$

Decision variables:

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

(3) $Z_j \in [0, \sum_{i=1}^{k} \mathcal{W}_i^j]$ $\quad \forall j \in 1,\ldots,n$

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

known at the time the decision has to be made, but we assume that the decision maker has an estimate of the probability distribution of $\overline{\mathcal{W}} = (\mathcal{W}_1,\ldots,\mathcal{W}_k)$. A per unit penalty of c has to be paid for exceeding the capacity of the knapsack. By modeling this problem as a one-stage Stochastic COP, the recourse problem RP(SSKP) can be formulated as shown in Fig. 1. The objective function maximizes the trade-off between the reward brought by the objects selected in the knapsack (those for which the binary decision variable X_i is set to 1) and the expected penalty paid for buying additional capacity units in those scenarios in which the low cost capacity q is not sufficient.

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

This solution can be obtained by solving a deterministic equivalent conventional constraint program obtained by employing a scenario-based representation [18]. Let $\mathcal{W}_i^j$ be the realized weight of object i in scenario j. We hand-crafted a deterministic equivalent model DetEquiv(RP(SSKP)) for RP(SSKP) following the guidelines in [18]. This model is shown in Fig. 2. Constraint (1) states that Z_j, total excess weight in scenario j, must be greater than the sum of the weights of the objects selected in this scenario minus the low cost capacity q. Constraint (2) restricts the decision variables X_i to be binary. X_i is equal to 1 iff item i is selected in the knapsack. Constraint (3) fixes an upper bound for Z_j; this

upper bound is the sum of the weights of all the k objects in scenario j. The objective function maximizes the trade-off between the total reward brought by the objects selected and the sum of penalty costs — weighted by the respective scenario probability — paid for those scenarios where the low cost capacity q is not sufficient.

4.1 Expectation-based relaxation for stochastic variables

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

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

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

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

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

4.2 Relaxing the expected value problem

It should be noted that, although the expected value problem is easier than the recourse problem, it may still be difficult to solve (NP-hard). For this reason we

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

can further relax the expected value problem in order to obtain a valid bound by solving an easier problem. Let R(EV(P)) be a generic relaxation of EV(P). Then for a maximization problem EV(P)$\leq$ R(EV(P)) holds, therefore R(EV(P)) provides a valid bound for the recourse problem.

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

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

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

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

4.3 Cost-based filtering

In order to perform cost-based filtering, as in RC-based filtering, we need a *gradient function* **grad(V,v)**, which returns for each variable-value pair (V,v) an optimistic evaluation of the profit obtained if v is assigned to V. This function is obviously problem dependent, but regardless of the strategy adopted in the former section — i.e. whether we are using a relaxation for the expected value problem or solving this problem to optimality — it is possible to specify it and use it to filter provably suboptimal values. In what follows we present a gradient function for SSKP. At each node of the search tree, in order to compute this function, we use a continuous relaxation of the expected value problem similar to the one proposed by Dantzig for the well-known 0-1 Knapsack Problem [15]. We will now define the gradient function for SSKP by reasoning on the expected value problem. Assume that a partial assignment for decision variables is given. Let K be the set of all the items in the problem, $|K| = k$. Let S be the set of items for which a decision has been fixed, with $|S| < k$. Let q^* be the sum of the expected weights of the elements in S that are part of the knapsack. The profit $\overline{r}$ associated with this assignment is equal to the sum of the profits of the items in the knapsack minus the eventual expected penalty cost $c(q^* - q)$, if $q - q^*$ is negative. Now we consider an element $i \in K/S$. There are two possible options: taking it into the knapsack or not. If we take it, we increase the profit by r_i minus

any eventual expected penalty cost we pay if the expected residual capacity is or becomes negative. Finally for every other element in K/S we check if the balance between its profit and the eventual expected penalty gives an overall positive profit and, if so, we add it to the knapsack. This procedure requires at most $O(k)$ steps for each element for which a decision has not yet been taken, therefore it can be applied at each node of the search tree to compute a valid upper bound associated with a certain decision on an item, which therefore may be filtered if suboptimal.

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

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

4.4 Finding good feasible solutions

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

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

5 Experimental results

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

5.1 Static Stochastic Knapsack Problem

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

In our experiments we adopted a randomly generated test bed similar to the one proposed in [12]. There are three sets of instances considered: the first set has $k = 10$, the second set has $k = 15$ and the third has $k = 20$ items. For all the instances, item random weights, W_i, from which scenarios are generated, are independent and normally distributed with probability distribution function $N(\mu_i, \sigma_i)$. The expected weights, μ_i, are generated from the uniform (20,30) distribution, and the weight standard deviations, σ_i, are generated from the uniform (5,10) distribution. Rewards r_i are generated from the uniform (10,20) distribution. The per unit penalty is $c = 4$, while the available low cost capacity is $q = 250$ for 20 items, $q = 187$ for 15 items, and $q = 125$ for 10 items. We randomly generated, using simple random sampling, sets of scenarios having different sizes: $\{100, 300, 500, 1000\}$. Scenarios are equally likely. The variable selection heuristic branches first on items with lower profit over expected weight ratio. The value selection tries first not to insert an item into the knapsack. In Table 1 we report our computational results. In all the instances considered our approach outperforms a pure SCP model in terms of explored nodes: the maximum improvement reaches a factor of 576.5. Run times are also shorter in our approach for almost all the instances. An exception is observed for the smallest instance, where the cost of filtering domains is not compensated by the payoff in terms of reduction of the search space. The maximum speed-up observed for run times reaches a factor of 90.5.

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

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

5.2 Stochastic sequencing with release times and deadlines

We consider a specific sequencing problem similar to the one considered by Hooker et. al [9]. Garey and Johnson [8] also mention this problem in their list of NP-hard problems and they refer to it as "Sequencing with Release Times and Deadlines" (SSEQ). An optimization version of this scheduling problem was also described in [10]. The problem consists in finding a feasible schedule to process a set I of k orders (or jobs) using a set M of n parallel machines. Processing an order $i \in I$ can only begin after the release date r_i and must be completed at the latest by the due date d_i. Order i can be processed on any of the machines. The processing time of order $i \in I$ on machine $m \in M$ is P_{im}. The model just described is fully deterministic, but we will now consider a generalization of this problem to the case where some inputs are uncertain. For convenience we will just consider uncertain processing times $\mathcal{P}_{im}$ for order $i \in I$ on machine $m \in M$. Instead of simply finding a feasible plan we now aim to minimize the expected total tardiness of the plan (the deterministic version of this problem is known as "Sequencing to minimize weighted tardiness" [8] and it is NP-hard). A solution for our SSEQ problem consists in an assignment for the jobs on the machines and in a total order between jobs on the same machine. In such a plan, a job will be processed on its release date if no other previous job is still processing, or as soon as the previous job terminates. The recourse problem RP(SSEQ) can be formulated as a one-stage Stochastic COP. This is shown in Fig. 3.

Decision variable X_{im} takes value 1 iff job i is processed on machine m, decision variable S_{ab} takes value 1 iff job a is processed before job b. Constraints (1) and (2) enforce a total order among jobs on the same machine. Constraint (3) enforces that each job must be processed on one and only one machine. Constraint (4) states that the (stochastic) completion time, $\mathcal{C}_i$, of a job i minus its (stochastic) duration $\mathcal{P}_{im}$ on the machine on which it is processed must be greater than or equal to its release date r_i, where $\mathcal{C}_i$ is an auxiliary variable used for simplifying notation. Let $I_m \equiv \{\mathcal{J}_{1m}, \mathcal{J}_{2m}, \ldots, \mathcal{J}_{qm}\} \subseteq I$ be the ordered set of jobs assigned to machine m. $\mathcal{C}_{\mathcal{J}_{qm}}$ is defined recursively as $\mathcal{C}_{\mathcal{J}_{qm}} = \max\{r_{\mathcal{J}_{qm}}, \mathcal{C}_{\mathcal{J}_{(q-1)m}}\} + \mathcal{P}_{\mathcal{J}_{qm}m}$, and $\mathcal{C}_{\mathcal{J}_{0m}} = 0$. Constraint (5) states that if two jobs a and b are processed on the same machine and if a is processed

Objective:

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

Constraints:

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

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

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

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

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

Decision variables:

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

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

Stochastic variables:

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

Auxiliary variables:

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

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

before b, that is $S_{ab} = 1$, then the (stochastic) completion time of job a plus the (stochastic) duration of job b on the machine on which it is processed must be less than or equal to the (stochastic) completion time of job b. Finally, the objective function minimizes the sum of the expected tardiness of each job. The tardiness is defined as $\max\{0, \mathcal{C}_i - d_i\}$. The cost function to be minimized can easily be proved convex in the random job durations. The expected total tardiness is in fact minimized for n machines. Job completion times on different machines are independent, therefore if we prove convexity for machine $m \in M$, then it directly follows that the cost function of the problem is also convex[4]. The cost function for machine m can be expressed as $\mathbb{E}\left[\sum_{i \in I_m} (\mathcal{C}_i - d_i)^+\right]$.

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

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

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

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

We implemented DetEquiv(RP(SSEQ)) in Choco and we coded an optimization-oriented global chance-constraint which exploits the expected value problem

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

Objective:

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

Constraints:

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

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

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

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

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

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

Decision variables:

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

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

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

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

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

In order to generate instances for our experiments, we adopted release times, deadlines and deterministic processing times from the first two "hard" instances proposed in [9], the one with 3 jobs and 2 machines and the one with 7 jobs and 3 machines. In each scenario, we generated processing times uniformly distributed in $[1, 2 * J_{im}]$, where J_{im} is the deterministic processing time required for job i on machine m for the instance considered. We considered different number of scenarios in $\{10, 30, 50, 100\}$. Scenarios are equally likely in terms of probability. The variable selection heuristic branches first on binary decision variables. The value selection tries increasing values in the domain. In Table 2 we report the results observed with and without the improvement brought by our cost-based filtering approach.

It should be noted that in this case, in contrast to the approach employed for SSKP, we only relax stochastic variables and we do not employ a relaxation for the deterministic equivalent problem, which therefore remains NP-hard. Recall that in SSKP we adopted Dantzig's relaxation to efficiently obtain a bound for the deterministic equivalent problem. A direct consequence of this is that, while in the SSKP example the improvement is significant both in terms of

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

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

explored nodes and run times for all the instances, in this example the run time improvement starts to be significant (a factor of 11.6) only for the largest instance (7 jobs and 3 machines) and for a high number of scenarios (100 scenarios). This is due to the fact that at every node of the search tree we solve a difficult problem (though far easier than the original stochastic constraint program) to obtain bounds and perform cost-based filtering. In terms of explored nodes, however, we obtain a significant improvement for every instance — the maximum improvement factor is of 32.3 — since the bounds generated are tight.

6 Related work

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

7 Conclusions

We proposed a novel strategy to performing cost-based filtering for certain classes of stochastic constraint programs, under the assumptions that (i) the objective function is concave or convex in the stochastic variables, and (ii) the existence of a feasible solution is not affected by the distribution of the stochastic variables. This strategy is based on a known inequality borrowed from Stochastic Programming. We applied this technique to two combinatorial optimization problem involving uncertainty from the literature. Our results confirm that orders-of-magnitude improvements in terms of explored nodes and run times can be achieved. In the future, we aim to apply cost-based filtering to multi-stage Stochastic COPs, define strategies to handle generic chance-constraints, which

are currently ruled out by our assumptions, and to extend the approach to other valid inequalities such as Edmundson-Madansky [4] or to suitable inequalities for non-convex problems [13]. Finally, we plan to exploit the information provided by optimization-oriented global chance-constraints to define search strategies.

References

1. K. Apt. *Principles of Constraint Programming.* Cambridge University Press, Cambridge, UK, 2003.
2. M. Avriel and A. C. Williams. The value of information and stochastic programming. *Operations Research*, 18(5):947–954, 1970.
3. T. Balafoutis and K. Stergiou. Algorithms for stochastic csps. In *Proceedings of the 12th International Conference on the Principles and Practice of Constraint Programming*, pages 44–58. Springer Verlag, 2006. Lecture Notes in Computer Science No. 4204.
4. J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming.* Springer Verlag, New York, 1997.
5. S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, Cambridge, UK, 2004.
6. A. Charnes and W. W. Cooper. Deterministic equivalents for optimizing and satisficing under chance constraints. *Operations Research*, 11(1):18–39, 1963.
7. F. Focacci, A. Lodi, and M. Milano. Optimization-oriented global constraints. *Constraints*, 7:351–365, 2002.
8. M. R. Garey and D. S. Johnson. *Computer and Intractability. A guide to the theory of NP-Completeness.* Bell Laboratories, Murray Hill, New Jersey, 1979.
9. J. N. Hooker, G. Ottosson, E. S. Thorsteinsson, and H. J. Kim. On integrating constraint propagation and linear programming for combinatorial optimization. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 136–141. AAAI, The AAAI Press/MIT Press, Cambridge, Ma., 1999.
10. V. Jain and I. E. Grossmann. Algorithms for hybrid milp/cp models for a class of optimization problems. *INFORMS Journal on computing*, 13:258–276, 2001.
11. P. Kall and S. W. Wallace. *Stochastic Programming.* John Wiley & Sons, 1994.
12. A. J. Kleywegt, A. Shapiro, and T. Homem-De-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal of Optimization*, 12(2):479–502, 2001.
13. D. Kuhn. *Generalized bounds for convex multistage stochastic programs.* Lecture Notes in Economics and Mathematical Systems No. 584.
14. F. Laburthe and the OCRE project team. Choco: Implementing a cp kernel. Technical report, Bouygues e-Lab, France, 1994.
15. S. Martello and P. Toth. *Knapsack Problems.* John Wiley & Sons, NY, 1990.
16. R. Rossi, S. A. Tarim, B. Hnich, and S. Prestwich. A global chance-constraint for stochastic inventory systems under service level constraints. *Constraints*, 13(4):xxx–xxx, 2008.
17. S. A. Tarim, B. Hnich, R. Rossi, and S. Prestwich. Cost-based filtering techniques for stochastic inventory control under service level constraints. *Constraints*, 2008 (forthcoming).
18. S. A. Tarim, S. Manandhar, and T. Walsh. Stochastic constraint programming: A scenario-based approach. *Constraints*, 11(1):53–80, 2006.
19. T. Walsh. Stochastic constraint programming. In *Proceedings of the 15th ECAI. European Conference on Artificial Intelligence.* IOS Press, 2002.