A Survey on CP-AI-OR Hybrids for Decision Making under Uncertainty

Brahim Hnich, Roberto Rossi, S. Armagan Tarim, and Steven Prestwich

Abstract In this survey we focus on problems of decision making under uncertainty. Firstly, we clarify the meaning of the word "uncertainty" and we describe the general structure of problems that fall into this class. Secondly, we provide a list of problems from the Constraint Programming, Artificial Intelligence and Operations Research literatures in which uncertainty plays a role. Thirdly, we survey existing modeling frameworks that provide facilities for handling uncertainty. A number of general purpose and specialized hybrid solution methods are surveyed, which deal with the problems in the list provided. These approaches are categorized into three main classes: stochastic reasoning-based, reformulation-based and sample-based. Finally, we provide a classification for other related approaches and frameworks in the literature.

1 Introduction

In this work we survey problems in which we are required to make decisions under uncertainty, and we categorize existing hybrid techniques in

Brahim Hnich
Department of Computer Engineering, Izmir University of Economics, Turkey,
brahim.hnich@ieu.edu.tr

Roberto Rossi
Logistics, Decision and Information Sciences, Wageningen UR, the Netherlands,
roberto.rossi@wur.nl

S. Armagan Tarim
Department of Management, Hacettepe University, Ankara, Turkey,
armtar@yahoo.com

Steven Prestwich
Cork Constraint Computation Centre, University College Cork, Ireland,
s.prestwich@4c.ucc.ie

Constraint Programming (CP), Artificial Intelligence (AI) and Operations Research (OR) for dealing with them. The word *uncertainty* is used to characterize the existence, in these problems, of uncontrollable or “random” variables[1], which cannot be influenced by the decision maker. In addition to these random variables, problems also comprise controllable or “decision” variables, to which a value from given domains has to be assigned. More specifically, a problem classified as *deterministic* with respect to the degree of uncertainty does not include random variables, while a *stochastic* problem does.

Random variables are typically employed to model factors such as the customer demand for a certain product, the crop yield of a given piece of land during a year, the arrival rate of orders at a reservation center and so forth. A continuous or discrete domain of possible values that can be observed is associated with each random variable. A probabilistic measure — typically a probability distribution — over such a domain is assumed to be available in order to fully quantify the likelihood of each value (respectively, range of values in the continuous case) that appears in the domain.

The decision making process comprises one or more subsequent *decision stages*. In a decision stage, a decision is taken by the decision maker who assigns a value to each controllable variable related to this decision stage of the problem and, subsequently, the uncontrollable variables related to this stage are observed and their realized values become known to the decision maker.

It should be noted that, in this work, we do not consider situations in which the decision maker has the power to modify the probability distribution of a given random variable by using his decisions. Random variables are therefore fully uncontrollable. To clarify, this means that a situation in which the decision maker has the option of launching a marketing campaign to affect the distribution of customer demands will not be considered.

This work is structured as follows: in Section 2 we employ a motivating example and a well established OR modeling framework — Stochastic Programming — in order to illustrate key aspects associated with the process of modeling problems of decision making under uncertainty; in Section 3 we provide a list of relevant problems from the literature on hybrid approaches for decision making under uncertainty and, for each problem, we also provide a short description and a reference to the work in which such a problem has been proposed and tackled; in Section 4 we introduce frameworks, respectively from AI and from CP, that aim to model problems of decision making under uncertainty; in Section 5, we classify existing hybrid approaches for tackling problems of decision making under uncertainty into three classes: in the first class (Section 6) we identify general and special purpose approaches that perform “stochastic reasoning”, in the second class (Section 7) we list approaches, general and special purpose, that use reformulation, and in the third class (Section 8) we categorize approximate techniques based on a va-

[1] Alternatively, in the literature, these variables are also denoted as “stochastic”.

riety of strategies employing sampling; finally, in Section 9 we point out connections with other related works, and in Section 10 we draw conclusions.

2 Decision Making Under Uncertainty

Several interesting real world problems can be classified as "stochastic". In this section we use a variant of the Stochastic Knapsack Problem (SKP) discussed in [34] as a running example to demonstrate ideas and concepts related to stochastic problems.

Single-stage Stochastic Knapsack. A subset of k items must be chosen, given a knapsack of size c into which to fit the items. Each item i, if included in the knapsack, brings a deterministic profit r_i. The size ω_i of each item is stochastic and it is not known at the time the decision has to be made. Nevertheless, we assume that the decision maker knows the probability mass function PMF(ω_i) [31], for each $i = 1, \ldots, k$. A per unit penalty cost p has to be paid for exceeding the capacity of the knapsack. Furthermore, the probability of the plan not exceeding the capacity of the knapsack should be greater than or equal to a given threshold θ. The objective is to find the knapsack that maximizes the expected profit.

We now discuss Stochastic Programming, which is one of the most well known modeling approaches in OR for problems of decision making under uncertainty, such as the SKP. We arbitrarily chose to employ such a framework to introduce the key concepts of decision making under uncertainty. In the next sections, the following frameworks will be also introduced: Stochastic Boolean Satisfiability, Probabilistic Constraint Satisfaction Problems, Event-Driven Probabilistic Constraint Programming and Stochastic Constraint Programming.

Stochastic Programming (SP) [11, 32] is a well established technique often used for modeling problems of decision making under uncertainty. A *Stochastic Program* typically comprises a set of decision variables defined over continuous or discrete domains, a set of random variables also defined over continuous or discrete domains and, for each random variable, the respective probability density function (PDF) if continuous or probability mass function (PMF) if discrete. Decision and random variables are partitioned into decision stages. Within a decision stage, firstly, all the associated decision variables are assigned values; and secondly, all the associated random variables are observed. A set of constraints is usually enforced over decision and random variables in the model. These constraints may be *hard*, that is they should always be met regardless of the values that are observed for the random variables, or they may be *chance-constraints* [15]. Chance-constraints are constraints that should be satisfied with a probability exceeding a given

Objective:

$$\max\left\{\sum_{i=1}^{k} r_i X_i - p\mathbb{E}\left[\sum_{i=1}^{k} \omega_i X_i - c\right]^+\right\}$$

Subject to:

$$\Pr\left\{\sum_{i=1}^{k} \omega_i X_i \leq c\right\} \geq \theta$$

Decision variables:

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

Random variables:

$\omega_i \rightarrow$ item i weight $\forall i \in 1,\ldots,k$

Stage structure:

$V_1 = \{X_1,\ldots,X_k\}$

$S_1 = \{\omega_1,\ldots,\omega_k\}$

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

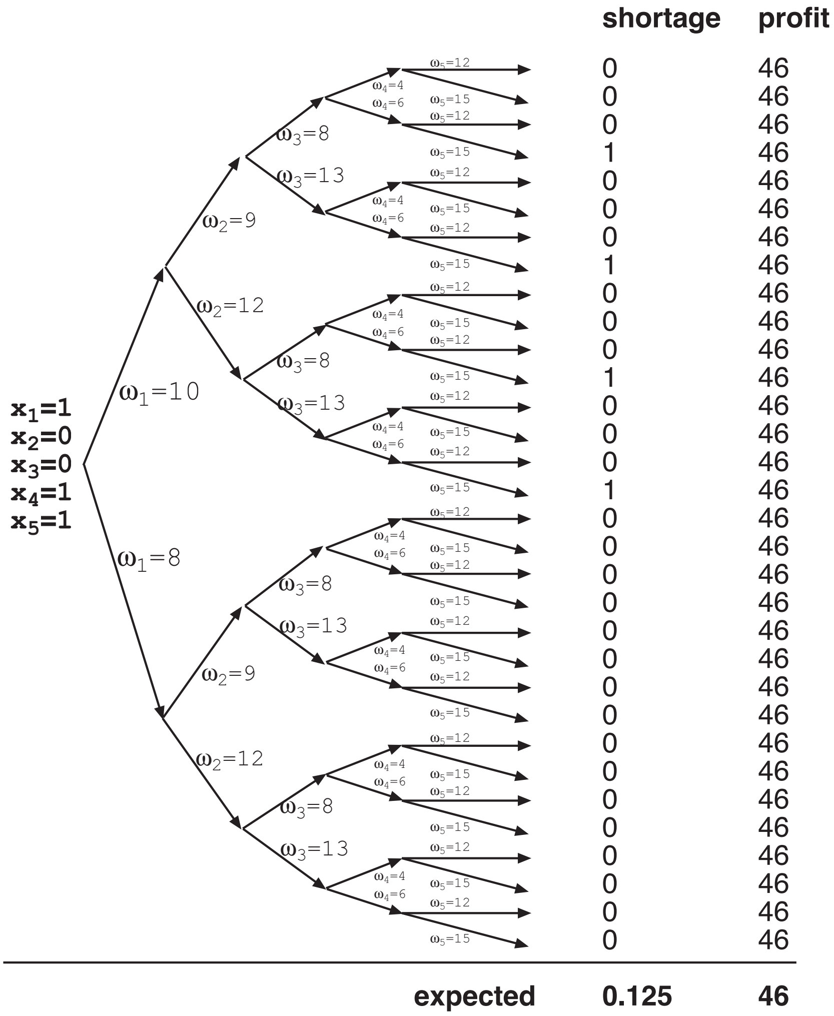
Fig. 1 A Stochastic Programming formulation for the single-stage SKP. Note that $[y]^+ = \max\{y, 0\}$ and $\mathbb{E}$ denotes the expected value operator

threshold. If the problem is an optimization one, it may minimize/maximize an objective function defined over some expressions on possible realisations (for example, maximize the worst case performance of the stochastic system under control, or minimize the difference between the maximum and minimum values a performance measure may take to increase the robustness of a system) or some probabilistic measure — such as expectation or variance — of decision and random variables in the model.

To clarify these concepts we now introduce a Stochastic Programming model for the single-stage SKP (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 available capacity c is not sufficient. Control actions that are performed after the uncertainty is resolved — such as buying additional capacity at a high cost — are called, in SP, "recourse actions". The only chance-constraint in the model ensures that the capacity c is not exceeded with a probability of at least θ. There is only a single decision stage in the model. Decision stages define how uncertainty unfolds in the decision making process. In other words, what the alternation should be between decisions and random variable observations. In a decision stage $\langle V_i, S_i\rangle$, first we assign values to all the decision variables in the set V_i, then we observe the realized values for all the random variables in the set S_i. More specifically, in the single decision stage $\langle V_1, S_1\rangle$ of the SKP, first we select all the objects that should be inserted into the knapsack, that is we assign a value to every decision variable $X_i \in V_1$, $\forall i \in 1,\ldots,k$; second, we observe the realized weight $\omega_i \in S_1$ for every object $i \in 1,\ldots,k$.

We now introduce a numerical example for the single-stage SKP.

Example 1. Consider $k = 5$ items whose item rewards r_i are $\{16, 16, 16, 5, 25\}$. The discrete probability mass functions for the weight ω_i of item $i = 1, \ldots, 5$ are respectively:
$\mathrm{PMF}(\omega_1) = \{10(0.5), 8(0.5)\}$, $\mathrm{PMF}(\omega_2) = \{9(0.5), 12(0.5)\}$, $\mathrm{PMF}(\omega_3) = \{8(0.5),$
$13(0.5)\}$, $\mathrm{PMF}(\omega_4) = \{4(0.5), 6(0.5)\}$, $\mathrm{PMF}(\omega_5) = \{12(0.5), 15(0.5)\}$.
The figures in parenthesis represent the probability that an item takes a certain weight. The other problem parameters are $c = 30$, $p = 2$ and $\theta = 0.6$.



	shortage	profit
	0	46
	0	46
	0	46
	1	46
	0	46
	0	46
	0	46
	1	46
	0	46
	0	46
	0	46
	1	46
	0	46
	0	46
	0	46
	1	46
	0	46
	0	46
	0	46
	0	46
	0	46
	0	46
	0	46
	0	46
	0	46
	0	46
	0	46
	0	46
	0	46
	0	46
	0	46
	0	46
expected	**0.125**	**46**

Fig. 2 Scenario tree representing the solution of the single-stage SKP in Example 1

As discussed, the problem has a single decision stage. This means that every decision has to be taken in a proactive way, before any of the random variables is observed. Therefore the optimal solution can be expressed as a simple assignment for the decision variables X_i, $\forall i \in 1, \ldots, k$. More specifically, the optimal solution for Example 1 proactively selects items $\{1, 4, 5\}$ and achieves an expected profit of 45.75. Such a solution can be validated using a scenario tree, as shown in Fig. 2. This tree considers every possible future realisation for the random variables ω_i, $\forall i \in 1, \ldots, k$. Since every random variable in the problem takes each of the possible values in its domain with uniform probability, all the paths in the scenario tree are equally likely. Therefore, it is easy to compute the expected profit of such an assignment and the expected additional capacity required. By plugging these values into the objective function, the profit associated with this solution can be easily obtained (i.e. $46 - 2 \cdot 0.125 = 45.75$). Finally, it can be easily verified that the chance-constraint in the model is also satisfied by this solution. In fact, a shortage is observed only in 4 out of 32 scenarios, therefore the chance constraint is satisfied, in this solution, with probability $0.875 \geq \theta = 0.6$.

The problem discussed in the former paragraphs only comprises a single decision stage. However, in general, stochastic programs may comprise multiple decision stages, that is a sequence of decisions and observations. In order to clarify this, we slightly modify the SKP presented above in such a way as to allow for multiple decision stages. Therefore, we introduce the multi-stage SKP.

Multi-stage Stochastic Knapsack. The single-stage problem description and assumptions are valid here with the exception that the items are considered sequentially, starting from item 1 up to item k. In other words, first we take the decision of inserting or not a given object into the knapsack, then we immediately observe its weight, which is a random variable, before any further item is taken into account.

A stochastic programming model for the multi-stage SKP is shown in Fig. 3. The model is similar to the one presented in Fig. 1, but the structure of the objective function is different. In this new model, expectation ($\mathbb{E}_{\omega_i}$) and $\max_{X_i}$ operators are nested and parameterized each by, respectively, the random variable ω_i over which the expectation is computed and the decision variable X_i that should be assigned in order to maximize the objective function value. This means, in practice, that an object may be selected or not, depending on the realized weights for previous objects. The stage structure is also different, because now the problem comprises multiple decision stages that alternate decisions and observations according to the arrival sequence of the objects.

We refer, once more, to the Example 1 presented above. The numerical data introduced there can be used to obtain an instance of the multi-stage SKP. As discussed, the problem now has multiple decision stages. This means

Objective:

$$\max_{X_1}\{r_1X_1+\mathbb{E}_{\omega_1}\{\max_{X_2} r_2X_2+\mathbb{E}_{\omega_2}\{\ldots\{\max_{X_{k-1}} r_{k-1}X_{k-1}+$$
$$\mathbb{E}_{\omega_k}\{\max_{X_k} r_kX_k+p[\textstyle\sum_{i=1}^{k}\omega_iX_i-c]^+\}\}\ldots\}\}\}$$

Subject to:

$$\Pr\left\{\textstyle\sum_{i=1}^{k}\omega_iX_i\leq c\right\}\geq\theta$$

Decision variables:

$X_i\in\{0,1\}$ $\qquad\forall i\in 1,\ldots,k$

Random variables:

$\omega_i\rightarrow$ item i weight $\;\forall i\in 1,\ldots,k$

Stage structure:

$V_i=\{X_i\}\;\forall i\in 1,\ldots,k$

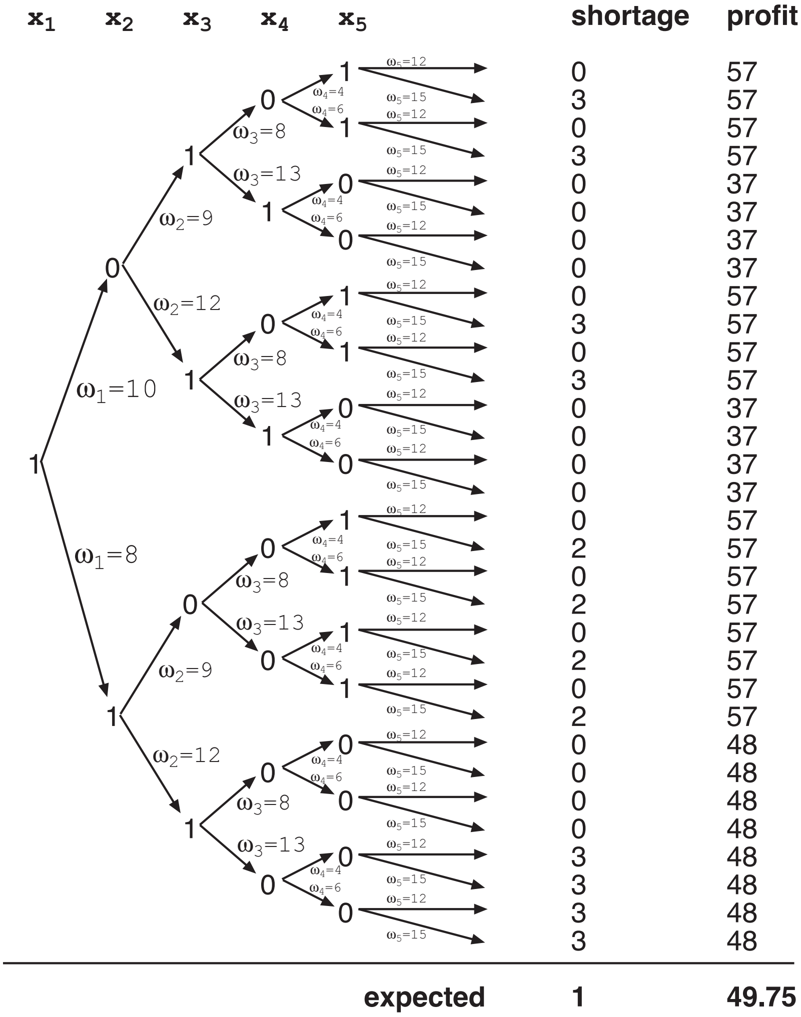
$S_i=\{\omega_i\}\;\forall i\in 1,\ldots,k$

$L=[\langle V_1,S_1\rangle,\langle V_2,S_2\rangle,\ldots,\langle V_k,S_k\rangle]$

Fig. 3 Stochastic programming formulation for the multi-stage SKP

that decisions are taken in a dynamic way, and they are alternated with observations for random variables. Therefore, the optimal solution is now expressed by using a *solution tree.* A solution tree encodes full information on how to act at a certain decision stage, when some random variables have been already observed. More specifically, the optimal solution tree for the instance of the multi-stage SKP defined by the data in Example 1 achieves an expected profit of 47.75 and it is shown in Fig. 4. To clarify: at the root node no uncertainty has been unfolded. The optimal solution tree in Fig. 4 shows that it is always optimal to take item 1 in the knapsack. Nevertheless, depending on the observed value for the weight of item 1, two alternative decisions may be optimal: not taking item 2 if the observed weight for item 1 is 10; or taking item 2 if the observed weight for item 1 is 8. To reiterate, since every random variable in the problem takes each of the possible values in its domain with uniform probability, all the paths in the solution tree are equally likely. Therefore, it is easy to compute the expected profit of such an assignment and the expected additional capacity required. By plugging these values into the objective function, the profit associated with this solution can be easily obtained. Finally, it can be also easily verified that the chance-constraint in the model is also satisfied by this solution. In fact, a shortage is observed only in 12 out of 32 scenarios, therefore the chance constraint is satisfied, in this solution, with probability $0.625\geq\theta=0.6$.

In this section we discussed the SKP; in Section 3 we provide a further list of problems from the literature discussing hybrid approaches to decision making under uncertainty.



shortage	profit
0	57
3	57
0	57
3	57
0	37
0	37
0	37
0	37
0	57
3	57
0	57
3	57
0	37
0	37
0	37
0	37
0	57
2	57
0	57
2	57
0	57
2	57
0	57
2	57
0	48
0	48
0	48
0	48
3	48
3	48
3	48
3	48
expected 1	**49.75**

Fig. 4 Solution tree for the multi-stage SKP in Example 1

3 A Collection of Stochastic Problems

In this section we provide a list of 9 other problems of decision making under uncertainty for which hybrid approaches have been proposed in the literature. This list is comprehensive, in the sense that it contains representative problems for each hybrid CP-AI-OR approach for decision making under uncertainty surveyed in this work.

The problems are:

- Stochastic queueing control problem [8, 75, 74]
- Scheduling Conditional Task Graphs [40]
- Stochastic reservation [5]
- Job shop scheduling with probabilistic durations [3]
- Two-stage stochastic matching problem [33]
- Production/inventory management [78]
- Stochastic template design [52, 71]
- Scheduling internal audit activities [60]
- Stochastic sequencing with release times and deadlines [57].

For each of these problems we provide a textual description; the reader may refer to the respective works where these problems were first introduced to obtain a more detailed description. In Section 5 we discuss and classify the hybrid solution methods proposed for modeling and solving these problems.

3.1 Stochastic Queueing Control Problem

In a facility with front room and back room operations, the aim is to switch workers between the rooms in order to cope with changing customer demand. Customer arrival and service time are stochastic and the decision maker seeks a policy for switching workers such that the expected customer waiting time is minimized, while the staff in the back room remains sufficient to perform all work. The problem was originally proposed and analyzed in [8]. Terekhov and Beck investigated it in [75, 74].

3.2 Scheduling Conditional Task Graphs

This is the problem, discussed in [40], of scheduling conditional task graphs in presence of unary and cumulative resources, minimizing the expected makespan. Conditional task graphs are directed acyclic graphs containing activities linked by precedence relations. Some of the activities represent branches. At run time only one of the successors of a branch is chosen for execution, depending on the occurrence of a condition labeling the corresponding arc. Since the truth or the falsity of those conditions is not known a priori, the problem is stochastic. Therefore all the possible future scenarios must be taken into account while constructing the schedule.

3.3 Stochastic Reservation

This problem, introduced in [5], is a particular application of the stochastic multi-knapsack problem. A travel agency may aim at optimizing the reservation of holiday centers during a specific week with various groups in the presence of stochastic demands and cancellations. The requests are coming according a given probability distribution and they are characterized by the size of the group and the price the group is willing to pay. The requests cannot specify the holiday center. However, the travel agency, if it accepts a request, must inform the group of its destination and must commit to it. Groups can also cancel the requests at no cost. Finally, the agency may overbook the centers, in which case the additional load is accommodated in hotels at a fixed cost.

3.4 Job Shop Scheduling with Probabilistic Duration

This problem was originally proposed in [3]. The problem is a classic Job Shop Scheduling (JSP) (see [23], p. 242) in which the objective is to find the minimum makespan. In contrast to the classic formulation for the JSP presented in [23] the authors assume, in this case, that the job durations are probabilistic. The objective is therefore accordingly modified to account for uncertainty: the authors search for a proactive plan, consisting of a partial order among activities and of resource-activity allocations, which attains the lowest possible makespan with probability greater or equal to a given threshold.

3.5 Two-stage Stochastic Matching Problem

We consider the minimum cost maximum bipartite matching problem discussed in [33]. The task is to buy edges of a bipartite graph which together contain a maximum-cardinality matching in the graph. The problem is formulated as a two-stage stochastic program with recourse, therefore edges can be bought either during the first stage, or with a recourse action after uncertainty has been resolved. There are two possible variants of this problem. In the first, the uncertainty is in the second stage edge-costs, that is the cost of an edge can either increase or decrease in the second stage. In the second variant all edges become more expensive in the second stage, but the set of nodes that must be matched is unknown. This problem can model real-life stochastic integral planning problems such as commodity trading, reservation systems and scheduling under uncertainty.

3.6 Production/Inventory Management

Uncertainty plays a major role in production and inventory management. In this simplified production/inventory planning example there are a single product, a single stocking point, production capacity constraints, service level constraints and a stochastic demand. The objective is to find a replenishment plan associated with the minimum expected total cost. The cost components taken into account are inventory holding costs and fixed replenishment (or setup) costs. The optimal plan gives the timing of the replenishments as well as the order quantities, which depend upon the previously realized demand. This production/inventory management problem has been investigated in [71, 78]. In [69] the authors investigate the same problems under the assumption that the production capacity constraints are relaxed.

3.7 Stochastic Template Design

The deterministic template design problem (prob002 in CSPLib[2]) is described as follows. We are given a set of variations of a design, with a common shape and size and such that the number of required pressings of each variation is known. The problem is to design a set of templates, with a common capacity to which each must be filled, by assigning one or more instances of a variation to each template. A design should be chosen that minimises the total number of runs of the templates required to satisfy the number of pressings required for each variation. As an example, the variations might be for cartons for different flavours of cat food, such as fish or chicken, where ten thousand fish cartons and twenty thousand chicken cartons must be printed. The problem would then be to design a set of templates by assigning a number of fish and/or chicken designs to each template such that a minimal number of runs of the templates is required to print all thirty thousand cartons. Proll and Smith [55] address this problem by fixing the number of templates and minimising the total number of pressings. In the stochastic version of the problem [52] the demand for each variation is uncertain. In compliance with production/inventory theory, the authors incorporate two conventional cost components: scrap cost, incurred for each template that is produced in excess of the realized demand, and shortage cost, incurred for each unit of demand not fulfilled. The objective is then to minimize the expected total cost.

[2] http://www.csplib.org

3.8 Scheduling Internal Audit Activities

Based on costs and benefits that change over time, the focus of the internal audit scheduling problem is how often to conduct an internal audit on an auditable unit. Auditable units are the units upon which internal control procedures are applied, in order to safeguard assets and assure the reliability of information flows. The problem, originally introduced in [60], can be stated as follows. We consider a planning horizon comprising of N time periods. We are given a set of M audit units over which random losses may accrue over time. Losses in each period are assumed to have a known probability mass function that could easily be estimated from available historical data. The distribution of losses may vary from period to period, i.e., it is non-stationary. Losses at different periods are assumed to be independent. Auditing is a time-consuming task, and the auditing team is given a strict deadline for performing an audit. Specifically, an audit must be completed in T time periods. Therefore after T periods the accrued losses drop to zero. If a team has already started auditing a unit at a given time period, then no other audit can be initiated during this period for the given audit team. The timing of audits are fixed once and for all at the beginning of the planning horizon and cannot be changed thereafter, even if it is suspected that certain auditable units have accrued unexpected losses. The objective is to find the optimal audit schedule while respecting the maximum loss criteria. That is, the invariant audit cost (i.e., fixed audit costs incurred each time an audit is conducted) and expected total discounted audit losses (i.e., cumulative losses accrued at the end of each period) are minimized by satisfying a minimum probability α that the losses will not exceed a predetermined level (allowed maximum loss) in any given audit period for any auditable unit.

3.9 Stochastic Sequencing with Release Times and Deadlines

The problem, introduced in [57], consists in finding an optimal schedule to process a set of orders using a set of parallel machines. The objective is to minimize the expected total tardiness of the plan. Processing an order can only begin after its release date and should be completed at the latest by a given due date for such an order. An order can be processed on any of the machines. The processing time of a given order, when processed on a certain machine, is a random variable. A solution for this problem consists in an assignment for the jobs on the machines and in a total order between jobs on the same machine. 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.

4 Frameworks for Decision Making Under Uncertainty in CP and AI

In Section 2 we introduced SP, a well established OR framework for decision making under uncertainty. In this section, we introduce other existing frameworks for decision making under uncertainty from, respectively, AI and CP. Stochastic Boolean Satisfiability extends a well established AI modeling framework, Propositional Satisfiability, by considering uncertainty. Probabilistic CSP, Event-Driven Probabilistic Constraint Programming and Stochastic Constraint Programming set the scene for dealing with uncertainty in CP. Where appropriate, we describe connections and similarities among these different frameworks.

4.1 Stochastic Boolean Satisfiability

The Boolean Satisfiability (SAT) community have investigated problems involving uncertainty, with the *Stochastic Satisfiability* (SSAT) framework. SSAT aims to combine features of logic and probability theory, and has been applied to probabilistic planning, belief networks and trust management. We base our discussion on a recent survey [43].

4.1.1 Definitions

The SAT problem is to determine whether a Boolean expression has a satisfying labelling (set of truth assignments). The problems are usually expressed in conjunctive normal form: a conjunction of clauses $c_1 \wedge \ldots \wedge c_m$ where each clause c is a disjunction of literals $l_1 \vee \ldots \vee l_n$ and each literal l is either a Boolean variable v or its negation $\bar{v}$. A Boolean variable can be labelled true (T) or false (F). Many constraint problems can be SAT-encoded (modelled as a SAT problem) and vice-versa. In fact any SAT problem can be viewed as a Constraint Satisfaction Problem (CSP) with binary domains and non-binary constraints via the *non-binary encoding* [77]: for example a clause $a \vee b \vee \bar{c}$ corresponds to the constraint (or conflict) preventing the assignments $\{a \leftarrow F,\ b \leftarrow F,\ c \leftarrow T\}$. The SSAT terminology is somewhat different than that of SP but there are many correspondences.

An SSAT problem $\Phi = Q_1 v_1 \ldots Q_n v_n \phi$ is specified by:

- a *prefix* $\Phi = Q_1 v_1 \ldots Q_n v_n$ that orders the Boolean variables $v_1 \ldots v_n$ of the problem and *quantifies* them. Each variable v_i is quantified by its quantifier Q_i either as *existential* ($\exists$) or *randomised* (Ɐ);
- a *matrix* ϕ: a Boolean formula containing the variables, usually in conjunctive normal form (CNF).

An existential variable is a standard SAT variable (corresponding to a decision variable in SP), while a randomised variable v_i is a Boolean variable that is true with associated probability π_i (corresponding to a random variable in SP). Sequences of similarly quantified variables may be grouped together into (existential or randomised) *blocks*, and an SP stage corresponds to an existential block followed by a randomised block. The values of existential variables may be contingent on the values of (existential or randomised) variables earlier in the prefix, so an SSAT solution takes the form of an *assignment tree* (corresponding to the solution tree in SP) specifying an assignment to each existential variable for each possible instantiation of the randomised variables preceding it in the prefix. An *optimal assignment tree* is one that yields the maximum probability of satisfaction; alternatively, the decision version of SSAT asks whether the probability of satisfaction exceeds a threshold θ.

SSAT is simpler than a Stochastic Program in three ways: the variable domains are Boolean only (as in SAT), the constraints (clauses) are of a fixed type (as in SAT), and no distinction is made between scenarios in which different clauses are violated. The latter means that SSAT is akin to a stochastic program with a single chance-constraint.

4.1.2 Restrictions and Generalizations

Some special cases have been identified in the literature: if all variables are randomised then we have a MAJSAT problem; if the prefix has only an existential block followed by a randomised block then we have an E-MAJSAT problem; and if each block contains a single variable then we have an Alternating SSAT (ASSAT) problem. SSAT has also been extended by the addition of *universal* quantifiers ($\forall$) to give Extended SSAT (XSSAT). A formula $\forall v \phi$ must be true for both $v = T$ and $v = F$. XSSAT subsumes Quantified Boolean Formulae (QBF), which is the archetypal PSPACE-complete problem: QBF is XSSAT without randomised quantifiers.

4.2 Probabilistic Constraint Satisfaction Problems

The Probabilistic CSP framework, proposed in [19], is an extension of the CSP framework [1] that deals with some decisions problems under uncertainty. This extension relies on a differentiation between the agent-controllable decision variables and the uncontrollable parameters whose values depend on the occurrence of uncertain events. The uncertainty on the values of the parameters is assumed to be given under the form of a probability distribution.

4.2.1 Definitions

A probabilistic CSP is a CSP equipped with a partition between (controllable) decision variables and (uncontrollable) parameters, and a probability distribution over the possible values of the parameters. More specifically, the authors define a Probabilistic CSP as a 6-tuple $\mathcal{P} = \langle \Lambda, W, X, D, \mathcal{C}, pr \rangle$, where $\Lambda = \{\lambda_1, \ldots, \lambda_p\}$ is a set of parameters; $W = W_1 \times \cdots \times W_p$, where W_i is the domain of λ_i; $X = \{x_1, \ldots, x_n\}$ is a set of decision variables; $D = D_1 \times \ldots \times D_n$, where D_i is the domain of x_i; $\mathcal{C}$ is a set of constraints, each of them involving at least one decision variable; and $pr : W \to [0, 1]$ is a probability distribution over the parameter assignments. Constraints are defined as in classical CSP. A complete assignment of the parameters (resp. of the decision variables) is called a "world" (resp. a "decision").

The authors consider successively two assumptions concerning the agents awareness of the parameter values at the time the decision must imperatively be made.

- "No more knowledge": the agent will never learn anything new before the deadline for making a decision; all it will ever know is already encoded by the probability distribution.
- "Complete knowledge": the actual parameters will be completely revealed before the deadline is reached (possibly, just before), so that it it useful to the agent to compute off-line a ready-to-use conditional decision, that the agent will be able to instantiate on-line, as soon as it knows what the actual parameters are.

For the first case, a solution is an unconditional decision that is most likely to be feasible according to world probabilities. For the second case, a solution provides a set of decisions with their conditions of applicability — i.e. under which world(s) a given decision should be used — together with the likelihood of occurrence of these conditions, which also follows from world probabilities.

4.3 Event-driven Probabilistic Constraint Programming

In Event-driven Probabilistic Constraint Programming (EDP-CP), which is an extension of the Probabilistic CSP framework, some of the constraints can be designated by the user as *event constraints*. The user's objective is to maximize his/her chances of realizing these "events". In each world — as defined in the Probabilistic CSP framework — events are subject to certain pre-requisite constraints and to certain conditions. If a pre-requisite is unsatisfied in a given world then the event is also classed as unsatisfied in that world; and if a condition is unsatisfied in a world then the event is classed as satisfied in that scenario. Intuitively, this means that in EDP-CP it is possible to express the fact that the feasibility of certain event constraints may depend on the satisfaction of other constraints (denoted as “pre-requisite constraints”) under certain “conditions”. In order to model such situations, a new meta-constraint — the *dependency meta-constraint* — is introduced.

4.3.1 Definitions

An EDP-CP is a 9-tuple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \Lambda, \mathcal{W}, \mathcal{E}, \mathcal{C}, \mathcal{H}, \Psi, \mathrm{Pr} \rangle$ where:

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of decision variables;
- $\mathcal{D} = D_1 \times \ldots \times D_n$, where D_i is the domain of X_i;
- $\Lambda = \{\lambda_1, \ldots, \lambda_l\}$ is a set of uncertain parameters;
- $\mathcal{W}= W_1 \times \ldots \times W_l$, where W_i the domain of λ_i;
- $\mathcal{E} = \{e_1, \ldots, e_m\}$ is a set of event constraints. Each e_i may either be probabilistic (involving a subset of $\mathcal{X}$ and a subset of Λ) or deterministic (involving only a subset of $\mathcal{X}$);
- $\mathcal{C} = \{c_1, \ldots, c_o\}$ is a set of dependency meta-constraints. For each dependency meta-constraint c_i : $\textsc{Dependency}(e, p, f)$ we have $e \in \mathcal{E}$, where p may be either a probabilistic or a deterministic pre-requisite constraint, and f is a deterministic condition constraint;
- $\mathcal{H}= \{h_1, \ldots, h_p\}$ is a set of hard constraints. Each h_i may either be probabilistic (involving a subset of $\mathcal{X}$ and a subset of Λ) or deterministic (involving only a subset of $\mathcal{X}$);
- Ψ is any expression involving the event realization measures on the event constraints in $\mathcal{E}$;
- $\mathrm{Pr} : \mathcal{W} \rightarrow [0, 1]$ is a probability distribution over uncertain parameters.

An optimal solution to an EDP-CP $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \Lambda, \mathcal{W}, \mathcal{E}, \mathcal{C}, \mathcal{H}, \Psi, Pr \rangle$ is any assignment S to the decision variables such that:

1. the hard constraints are satisfied in each possible world; and
2. there exists no other assignment satisfying all the hard constraints with a strictly better value for Ψ, according to the Dependency constraints introduced in the model.

4.3.2 Relations to Other Frameworks

The Event-driven Probabilistic Constraint Programming (EDP-CP) framework, proposed in [67], extends both the Probabilistic CSP framework [19] and the Dependent-chance Programming framework [38]. In contrast to probabilistic CSP, which treats all probabilistic constraints uniformly, EDP-CP distinguishes between event, pre-requisite, condition, and hard constraints. Furthermore, in Dependent-chance Programming a feasible solution satisfies all event constraints, whilst in EDP-CP such a requirement is relaxed. This gives the decision-maker more flexibility in modeling. Finally, the notion of constraint dependency introduced in [67] comprises condition constraints, in addition to the event and pre-requisite constraints. As the authors remark, constraint dependency without condition constraints does not guarantee optimal plans since in certain instances common variables may take values which break the link between two dependent constraints.

4.4 Stochastic Constraint Programming

Stochastic Constraint Programming (SCP) was first introduced in [78] in order to model combinatorial decision problems involving uncertainty and probability. According to Walsh, SCP combines together the best features of CP (i.e. global constraints, search heuristics, filtering strategies, etc.), of SP (expressiveness in representing problems involving random variables), and of Stochastic Satisfiability.

4.4.1 Definitions

An m-stage Stochastic Constraint Satisfaction Problem (SCSP) is defined, according to [78], as a 7-tuple $\langle V, S, D, P, C, \theta, L\rangle$[3], 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 and each element of S to a domain of potential values. In what follows we assume that both decision and random variable domains are finite. P is a function mapping each element of S to a probability distribution for its associated domain. C is a set of chance-constraints over a non-empty subset of decision variables and a subset of random variables. θ is a function mapping each chance-constraint $h \in C$ to θ_h which 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. $L = [\langle V_1, S_1\rangle, \ldots, \langle V_i, S_i\rangle, \ldots, \langle V_m, S_m\rangle]$ 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 hard constraints are satisfied and the chance constraints are satisfied in the specified fraction of all possible scenarios. The solution of an m-stage SCSP is represented by means of a

[3] The original formulation, proposed in [78], does not directly encode the stage structure in the tuple and actually defines a SCSP as a 6-tuple; consequently the stage structure is given separately. We believe that a more adequate formulation is the one proposed in [30], that explicitly encodes the stage structure as a part of the tuple, giving a 7-tuple.

Objective:

$$\max \left\{ \sum_{i=1}^{k} r_i X_i - p\mathbb{E}\left[\sum_{i=1}^{k} \omega_i X_i - c \right]^{+} \right\}$$

$\langle V, S, D, P, C, \theta, L \rangle$**:**

$V = \{X_1, \ldots, X_k\}$

$S = \{\omega_1, \ldots, \omega_k\}$

$D = \{X_1, \ldots, X_k \in \{0, 1\}, D(\omega_1), \ldots, D(\omega_k)\}$

$P = \{PDF(\omega_1), \ldots, PDF(\omega_k)\}$

$C = \left\{ \Pr\left\{ \sum_{i=1}^{k} \omega_i X_i \le c \right\} \ge \theta \right\}$

$L = [\langle \{X_1, \ldots, X_k\}, \{\omega_1, \ldots, \omega_k\} \rangle]$

Fig. 5 Stochastic Constraint Programming formulation for the single-stage SKP

policy tree. 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. The policy tree, in fact, corresponds to the solution tree adopted in SP.

Let $\mathcal{S}$ denote the space of policy trees representing all the solutions of a SCSP. 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 a set $\widehat{S} \subseteq S$ of random variables (edges of the policy tree) and over a set $\widehat{V} \subseteq V$ of the decision variables (nodes in the policy tree). A *stochastic constraint optimization problem* (SCOP) is then defined in general as $\max_{s \in \mathcal{S}} f(s)$.

Unlike SP, SCP offers a richer modeling language which supports chance-constraints over global, nonlinear, and logical constraints in addition to linear ones.

It is easy to reformulate the running example discussed in Section 2 (SKP) as a single-stage SCOP, the respective model is given in Fig. 5. As in the SP model, in the SCP model we have sets of decision and random variables with their respective domains. For the random variables the respective probability mass function is specified. There is a chance-constraint with an associated threshold θ. In fact, the SCOP in Fig. 5 fully captures the structure of the stochastic program in Fig. 1.

5 A Classification of Existing Approaches

In previous sections we stressed the fact that this survey is centered on "uncertainty", and we also clarified the precise meaning we associate with the term uncertainty. Other literature surveys tend to merge uncertainty with other concepts; in Section 9 we will briefly discuss related works in these different areas, and the reader may refer to these surveys for more details. Furthermore, there exist surveys that are more explicitly focused on pure AI

[10] or OR [62] techniques, but little attention has been dedicated so far to hybrid techniques.

In this section, we propose a classification for existing hybrid approaches and frameworks that blend CP, AI and OR for decision making under uncertainty. The integration of CP, AI and OR techniques for decision making under uncertainty is a relatively young research area. We propose to classify existing approaches in the literature within three main classes (Fig. 10).

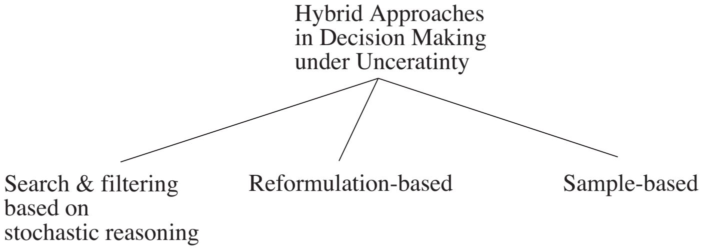


Fig. 6 A classification of hybrid approaches in CP-AI-OR for decision making under uncertainty

- The first class comprises those approaches that perform some form of "stochastic reasoning" by using dedicated — general or special purpose — search procedures, filtering algorithms, neural networks, genetic algorithms etc.
- The second class, in contrast, includes approaches that exploit reformulation — once again employing either a specialized analytical derivation for a given problem, or general purpose techniques — in order to produce a deterministic model that can be solved using existing solvers.
- Finally, the third class comprises incomplete approaches that exploit sampling in order to attain a near-optimal solution for problems of optimization under uncertainty. We believe that approaches based on sampling are particularly attractive and deserve a dedicated class. In fact, a high level of complexity is a typical trait of decision problems involving uncertainty, therefore it seems that the only feasible way of tackling many of these problems consists in developing effective approximation strategies.

Before discussing further this classification, it is worth mentioning that we believe it would be impractical to list all existing applications of hybrid methods from CP, AI, and OR in decision making under uncertainty. For this reason we aim rather to classify the different strategies — and not the specific applications — adopted in the literature for solving this class of problems using hybrid approaches. Nevertheless, for each strategy mentioned in this section, we will report some of the respective applications.

In Section 6, we will discuss approaches performing "stochastic reasoning"; in Section 7 we will discuss approaches that exploit reformulation; and finally in Section 8 we will discuss incomplete approaches that exploit sampling.

6 Approaches Based on Stochastic Reasoning

In this section we will analyze existing approaches that perform some sort of "stochastic reasoning" by using dedicated — general or special purpose — techniques. These techniques take several different forms: search procedures, filtering algorithms, neural networks, genetic algorithms etc.

Firstly, we shall distinguish between *general purpose* and *problem specific* strategies (Fig. 7).

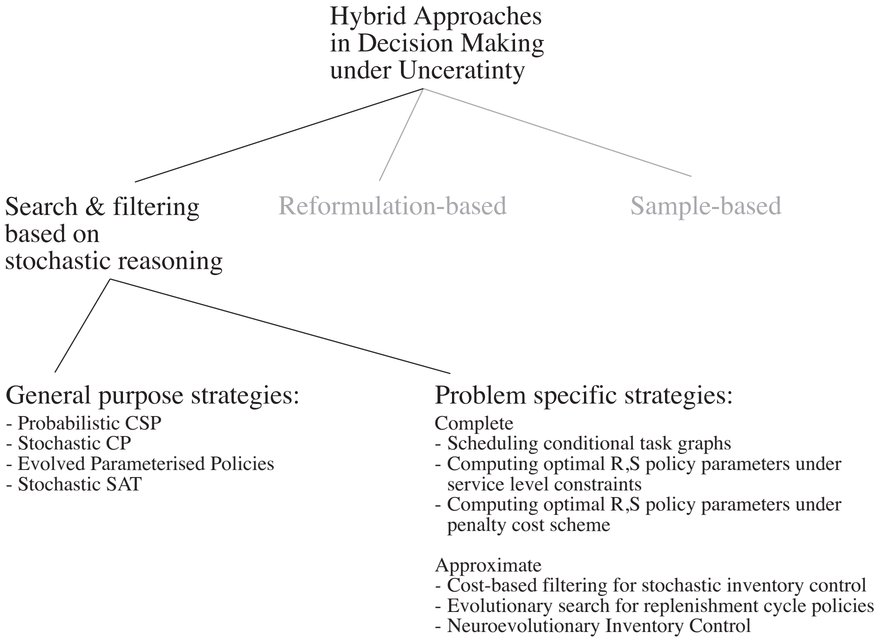

Hybrid Approaches
in Decision Making
under Unceratinty
Search & filtering
based on
stochastic reasoning
Reformulation-based
Sample-based
General purpose strategies:
- Probabilistic CSP
- Stochastic CP
- Evolved Parameterised Policies
- Stochastic SAT
Problem specific strategies:
Complete
- Scheduling conditional task graphs
- Computing optimal R,S policy parameters under service level constraints
- Computing optimal R,S policy parameters under penalty cost scheme
Approximate
- Cost-based filtering for stochastic inventory control
- Evolutionary search for replenishment cycle policies
- Neuroevolutionary Inventory Control

Fig. 7 A classification of hybrid approaches in CP-AI-OR for decision making under uncertainty: approaches based on stochastic reasoning

General purpose strategies aim to develop frameworks that provide modeling and solving facilities to handle generic problems of decision making under uncertainty. The modeling frameworks proposed in the literature typically aggregate concepts from different domains, for instance *global constraints* from CP, *chance-constraints* and *random variables* from SP (OR). These frameworks exploit well established AI strategies, such as forward checking procedures and genetic algorithms, in the solution process.

Problem specific strategies typically develop specialized reasoning algorithms that, during the search, are able to perform inference by exploiting the specific structure of the problem. For instance a typical approach is to encapsulate the reasoning within a dedicated global constraint that prunes decision variable domains according to the underlying stochastic reasoning.

In addition, both general purpose and problem specific strategies may be complete or heuristic. We shall now discuss in more detail these two different classes of approaches based on stochastic reasoning, by providing pointers to works in the literature.

6.1 General Purpose Strategies

We survey four different general purpose strategies for modeling and solving different classes of problems of decision making under uncertainty. These are: Probabilistic CSP, Stochastic CP, Evolving Parameterised Policies, and Stochastic SAT.

6.1.1 Probabilistic CSP.

One of the first general purpose frameworks for modeling uncertainty in CP is the Probabilistic CSP [19]. In the Probabilistic CSP a distinction is made between *controllable* and *uncontrollable* variables which correspond, respectively, to decision and random variables in SP. As in SP, a probability density function is associated with each uncontrollable variable. The authors discuss two different settings. Under the first of these settings, for each of the possible realizations that may be observed for the uncontrollable variables, the best decision is determined. This strategy corresponds to the wait-and-see policy in SP ([32], pp. 8) and it presents a posterior analysis. The second setting simply corresponds to a conventional single stage stochastic program where an optimal decision has to be taken before observing the realized values for the uncontrollable variables. The optimal decision, in this second case, is the one that guarantees the maximum likelihood to result feasible with respect to the given probability density functions for the uncontrollable variables.

The authors propose two algorithms for solving Probabilistic CSPs. The first algorithm, used for solving problems formulated under the first setting discussed, borrows ideas from solution methods developed in for solving Dynamic CSPs [18] and, in particular, reuses a procedure proposed in [21]. The second proposed algorithm consists of a depth first branch and bound algorithm and of a forward checking procedure. These are employed to solve problems formulated under the second setting discussed.

6.1.2 Stochastic Constraint Programming.

The Probabilistic CSP represents the first attempt to include random variables, and thus uncertainty, within the CP framework. Nevertheless, only in [78] is a clear link established between CP and SP with the introduction of SCP. We have already discussed in detail SCP as a modeling framework in Section 4.4. In [78] Walsh discusses the complexity of Stochastic CSPs, and proposes a number of complete algorithms and of approximation procedures for solving them. Namely, a backtracking algorithm and a forward checking procedure are proposed, which resemble those proposed in [19] for Probabilistic CSPs. Nevertheless, we want to underscore the fact that the key difference between a Probabilistic CSP and a Stochastic CSP is the fact that the former does not handle multiple decision stages.

In [2] Balafoutis et al. build on the SCP framework introduced in [78], they correct a flaw in the original forward checking procedure for Stochastic CSPs and they also extend this procedure in order to better take advantage of probabilities and thus to achieve stronger pruning. In addition, *arc-consistency* is defined for Stochastic CSPs and an arc-consistency algorithm able to handle constraint of any arity is introduced. Tests are carried on random binary Stochastic CSPs formulated as single and multi-stage problems.

In [13] Bordeaux and Samulowitz investigate two extensions to the original SCP framework. Firstly, they investigate situations in which variables are not ordered sequentially, corresponding to situations in which the future can follow different branches; they show that minor modifications allow the framework to deal with non-sequential forms. Secondly, they investigate how to extend the framework in such a way as to incorporate multi-objective decision making. An algorithm is proposed, which solves multi-objective stochastic constraint programs in polynomial space.

Global chance-constraints — which we discussed in Section 4.4 — were introduced first in [58], and they 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 [30]. 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 addition, in [57] Rossi et al. discuss some possible strategies to perform cost-based filtering for certain classes of Stochastic COPs. These strategies exploit well-known inequalities borrowed from SP and used to compute valid bounds for any given Stochastic COP that respects some mild assumptions. Examples are given for a simplified version of the stochastic knapsack problem previously discussed and for the stochastic sequencing problem discussed in Section 3.9.

6.1.3 Evolved Parameterised Policies.

Inspired by the success of machine learning methods for stochastic and adversarial problems, a recent approach to Stochastic CSPs/COPs called *Evolved Parameterised Policies* (EPP) is described in [53]. Instead of representing a policy explicitly in a Stochastic Constraint Program, an attempt is made to find a rule that decides, at each decision stage, which domain value to assign to the decision variable(s) at that stage. The quality of a rule can be determined by constructing the corresponding policy tree and observing the satisfaction probability of each chance constraint (and the value of the objective function if there is one). Evolutionary or other non-systematic search algorithms can be used to explore the space of rules.

EPP treats a Stochastic CSP/COP problem as an unconstrained noisy optimisation problem with at worst the same number of (real-valued) variables. This allows a drastic compression of the policy tree into a small set of numbers, and this compression together with the use of evolutionary search makes EPP scalable to large multi-stage Stochastic CSPs/COPs. It has the drawback that only policies of a relatively simple form can be discovered, but it results much more robust than a scenario-based approach on a set of random multi-stage problems [53]. Moreover, arbitrarily complex rules could be discovered by using artificial neural networks instead of these simple functions, a *neuroevolutionary* approach that has been successfully applied to many problems in control [24, 29, 64].

6.1.4 Stochastic SAT.

Another general purpose framework for modeling and solving a well established class of problems under uncertainty in AI — and especially in planning under uncertainty — is Stochastic SAT. We introduced the modeling framework in Section 4.1. Current SSAT algorithms fall into three classes: *systematic*, *approximation*, and *non-systematic*.

The systematic algorithms are based on the standard SAT backtracking algorithm — the Davis-Putnam-Logemann-Loveland (DPLL) algorithm [17, 16] — and correspond roughly to some current SCSP algorithms. The first such algorithms were described in [36], in particular the `evalssat` algorithm for XSSAT which formed the basis for future systematic SSAT algorithms. `evalssat` did not use branching heuristics as in current SAT and CSP solvers, though [36] also used some restricted branching heuristics, but assigned variables in the order specified by the prefix. However, it did use SAT-based techniques (unit propagation and pure variable elimination) and reasoning on the probability threshold θ to prune the search tree. The policy-based SCSP algorithm of [78] is essentially `evalssat` with forward checking. Systematic algorithms have also been devised for special cases of XSSAT.

MAXPLAN [45], ZANDER [46] and DC-SSAT [44] all use special techniques for planning problems modelled as XSSAT problems.

The `sampleevalssat` approximation algorithm uses random sampling to select paths, then uses SAT techniques to search the restricted tree to maximise θ. The `APPSSAT` algorithm [42] considers scenarios in decreasing order of probability to construct a partial tree for the special case of planning problems modelled as SSAT problems.

The `randevalssat` algorithm [36] is based on the `sampleevalssat` algorithm mentioned above, but applies stochastic local search to the existential variables in a random set of scenarios, thus it is non-systematic. Other ways of applying local search were described in [41], including periodically restarting `randevalssat` with different sampled scenarios, an approach used by the WALKSSAT algorithm [79].

6.2 Problem Specific Strategies

In the previous section we discussed general purpose solution methods that bring together CP, AI and OR techniques for decision making under uncertainty. We will now discuss some special purpose approaches proposed in the literature that perform stochastic reasoning on specific problems.

6.2.1 Scheduling Conditional Task Graphs.

The work of [40] describes a complete, special purpose approach that concerns the problem — discussed in Section 3.2 — of scheduling conditional task graphs. Similarly to the approach in [56], the authors propose an analytical formulation of the stochastic objective function, in this case based on the task graph analysis, and a conditional constraint able to handle such a formulation efficiently. The authors show the benefit of such an approach by comparing the results with a deterministic model, which disregards uncertainty, and with a scenario-based formulation [71] that requires an exponential number of scenarios to fully represent the stochastic objective function.

6.2.2 Computing Optimal R,S Policy Parameters Under Service Level Constraints.

Another special purpose strategy is presented in [58], and proposes a dedicated global chance-constraint for computing replenishment cycle inventory policy parameters under service level constraints. More specifically, the problem considered in this work is the production/inventory problem described in Section 3.6. Computing optimal replenishment cycle policy parameters for such a problem is a complex task [69]. By using a dedicated global chance-constraint the authors were able to perform the complex stochastic reasoning required to compute optimal replenishment cycle policy parameters. Such a complete algorithm performs a numerical integration step in order to compute the real service level provided in each period by a given set of policy parameters and the associated expected total cost.

6.2.3 Computing Optimal R,S Policy Parameters Under a Penalty Cost Scheme.

Similarly, a dedicated global constraint has been proposed in [56] in order to solve to optimality the problem of computing optimal replenishment cycle policy parameters under a penalty cost scheme. Such a problem has been investigated in [70], but in this work the authors could only solve the problem in a heuristic way, by employing a piecewise linear approximation of the convex cost function in the problem in order to build up a deterministic equivalent MIP model. In [56] the authors were able to embed a closed-form non-linear analytical expression for such a convex cost function within a global constraint, thus obtaining a complete model able to compute optimal replenishment cycle policy parameters.

6.2.4 Cost-based Filtering for Stochastic Inventory Control.

The work in [68] has a different flavor. In this case, the underling model is the deterministic equivalent CP formulation proposed in [73] for computing near-optimal replenishment cycle policy parameters under service level constraints. The CP formulation was originally proposed as a reformulation of the MIP model in [69]. Such a reformulation showed significant benefits in terms of efficiency. The authors, in [68], propose three independent cost-based filtering strategies that perform stochastic reasoning and that are able to significantly speed up the search when applied to the original CP model in [73].

6.2.5 Evolutionary Search for Replenishment Cycle Policies.

A recent application of a genetic algorithm to a multi-stage optimisation problem in inventory control is described in [51]. Each chromosome represents a replenishment cycle policy plan as a list of order-up-to levels, with a level of 0 representing no order, and the fitness of a chromosome is averaged over a large number of scenarios. This approach is enhanced in [50] by hybridising the genetic algorithm with the SARSA temporal difference learning algorithm [61]. This is shown to greatly improve the performance of genetic search for replenishment cycle policies, both with and without order capacity constraints.

6.2.6 Neuroevolutionary Inventory Control.

One may evolve an artificial neural network to optimally control an agent in an uncertain environment. The network inputs represent the environment and its outputs the actions to be taken. This combination of evolutionary search and neural networks is called *neuroevolution*. A recent paper [54] applies neuroevolution to find optimal or near-optimal plans in inventory control, following no special policy. The problems are multi-stage and involve multi-echelon systems (they have more than one stocking point). Such problems have no known optimal policy and rapidly become too large for exact solution. The inputs to the network are the current stock levels and the outputs are the order quantities.

7 Reformulation-based Approaches

In this section, we will analyze existing approaches that are based on a reformulation that produces a deterministic model, which can be solved using an existing solver.

Once more, we shall distinguish between *general purpose* and *problem specific* strategies (Fig. 8).

Hybrid general purpose reformulation strategies have recently appeared especially at the borderline between CP and OR. These typically take the form of a high level language — such as Stochastic OPL — used to formulate the problem under uncertainty, and of a general purpose compiler that can handle the high level stochastic model and produce a compiled deterministic equivalent one. Often, the compilation relies on a well known technique in SP: *scenario-based modeling.* In addition, due to the complexity of stochastic programs in general, approximation strategies are often proposed in concert with these general purpose frameworks in order to make the size of the compiled model manageable.

In contrast, problem specific strategies aim to fully exploit the structure of the problem in order to produce a deterministic — and possibly equivalent — model that can be handled efficiently by existing solver. In many cases, in order to obtain a model that is manageable by existing solvers, it is necessary to introduce some assumptions that affect the completeness and, thus, the quality of the solution found the the deterministic model. We will provide examples of applications in which a special purpose deterministic equivalent model is built, which is equivalent to the original model and also examples in

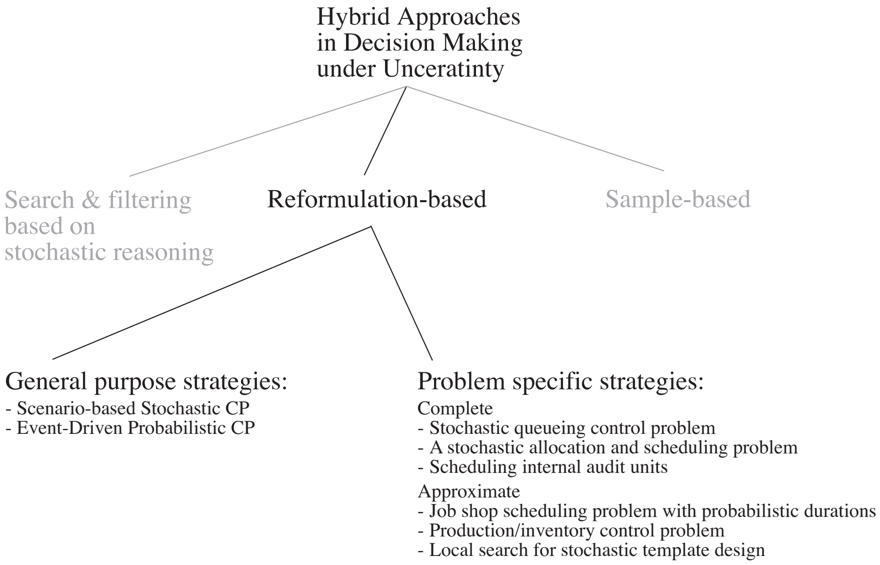

Hybrid Approaches
in Decision Making
under Unceratinty
Search & filtering
based on
stochastic reasoning
Reformulation-based
Sample-based
General purpose strategies:
- Scenario-based Stochastic CP
- Event-Driven Probabilistic CP
Problem specific strategies:
Complete
- Stochastic queueing control problem
- A stochastic allocation and scheduling problem
- Scheduling internal audit units
Approximate
- Job shop scheduling problem with probabilistic durations
- Production/inventory control problem
- Local search for stochastic template design

Fig. 8 A classification of hybrid approaches in CP-AI-OR for decision making under uncertainty: approaches based on a deterministic reformulation

which the deterministic model can only approximate the original stochastic model.

7.1 General Purpose Strategies

We survey two different general purpose strategies based on reformulation for modeling and solving classes of problems of decision making under uncertainty. These are Scenario-based Stochastic CP and Event-Driven Probabilistic Constraint Programming.

7.1.1 Scenario-based Stochastic Constraint Programming.

The first general purpose framework based on reformulation that we present is Scenario-based Stochastic Constraint Programming, which was proposed by Tarim et al. in [71]. The novelty in this work is the fact that the authors adopt a semantics for stochastic constraint programs based on scenario trees. By using this semantics the authors can compile stochastic constraint programs into conventional (non-stochastic) constraint programs and they can therefore use existing constraint solvers to effectively solve this class of problems.

In a scenario based approach — frequently used in SP [11] — a scenario tree is generated which incorporates all possible realizations of discrete random variables into the model explicitly. A path from the root to an extremity of the event tree represents a scenario. With each scenario a given probability is associated. Within each scenario we have a conventional (non-stochastic) constraint program to solve. All we need to do is replace the random variables by the values taken in the scenario, and ensure that the values found for the decision variables are consistent across scenarios, as certain decision variables are shared across scenarios. Constraints are defined (as in traditional constraint satisfaction) by relations of allowed tuples of values, and can be implemented with specialized and efficient algorithms for consistency checking. Furthermore, the scenario-based view of stochastic constraint programs also allows later-stage random variables to take values which are conditioned by the earlier-stage random variables. This is a direct consequence of employing the scenario representation, in which random variables are replaced with their scenario dependent values.

Scenario-based SCP has been outlined in Section 4.4. Tarim et al. [71] not only defined a general way to compile stochastic constraint programs into conventional constraint programs, but they also proposed a language, Stochastic OPL, which is based on the OPL constraint modeling language [28]. Using this language the authors modeled optimization problems under uncertainty from a variety of fields, such as portfolio selection, agricultural planning, and production/inventory management (Section 3.6). We will not discuss the language in detail, but in the Appendix we show how to model the single and multi-stage SKP problems of Section 2 by using the Stochastic OPL.

Among the benefits of the scenario based approach in [71] is the fact that it allows multiple chance-constraints and a range of different objectives to be modeled. The authors point out that each of these changes would require substantial modifications in the backtracking and forward checking algorithms proposed in [78]. The scenario based view allows each of these extensions to be modeled easily using stochastic OPL, compiled down into standard OPL, and solved by means of existing solvers. It should be noted that the approach is general and the compilation need not necessarily be performed using OPL, but it can be implemented using any available CP language and/or software package. The main drawback of this approach is the fact that the scenario tree required to model a given problem grows exponentially in size when random variable domains are large, thus leading to large models that are difficult to solve.

In addition to this general purpose modeling/solving framework the authors also proposed some techniques to improve the efficiency of the solution process. In order to do so, they proposed scenario reduction techniques, such as Monte Carlo Sampling or Latin Hypercube Sampling [65], to reduce the number of scenarios considered in the model. Their experimental results show the effectiveness of this approach, which in practice is able to find high quality solutions using a small number of scenarios. Finally, inspired by robust optimization techniques used in OR [35], the authors also proposed some techniques to generate robust solutions, that is solutions that adopt similar (or the same) decisions under different scenarios.

7.1.2 Event-Driven Probabilistic Constraint Programming.

We now briefly discuss a second general purpose framework based on reformulation: Event-Driven Probabilistic Constraint Programming [67]. This framework was introduced to address different problems than those for which SCP is a suitable modeling tool. Event-Driven Probabilistic Constraint Programming, as the name suggest, is connected to Probabilistic CSPs and, mainly, to Dependent-chance Programming [37, 38].

Sometimes a complex probabilistic decision system undertakes multiple tasks, called *events* here, and the decision-maker wishes to maximize chance functions which are defined as the probabilities of satisfying these events. This is especially useful in situations where a particular measure of the "reliability" or "robustness" of a given plan has to be maximized. The Event-Driven Probabilistic Constraint Programming modeling framework allows users to designate certain probabilistic constraints, involving both decision and random variables, as *events* whose chance of satisfaction must be maximized, subject to hard constraints which should be always satisfied, and also logical dependencies among constraints. Event-Driven Probabilistic Constraint Programming builds on Dependent-chance Programming and provides more expressiveness to the user, in order to capture a more realistic and accurate measure of plan reliability [59]. It also provides an exact solution method, employing scenario-based reformulation, in contrast to the approximate genetic algorithm in [38].

7.2 Problem Specific Strategies

We now discuss some problem specific strategies based on deterministic equivalent reformulations.

7.2.1 Stochastic Queueing Control Problem.

In [75, 74] the authors propose a set of deterministic equivalent CP models for solving the stochastic queueing control problem discussed in Section 3.1. [75] not only provides the first application of CP to solving a stochastic queueing control problem, but it also provides a complete approach for a problem for which only a heuristic algorithm [8] existed. Three deterministic equivalent constraint programming models and a shaving procedure are proposed. The complete models provide satisfactory performances when compared with the heuristic procedure, which nevertheless remains superior in terms of solution quality over time. A hybrid method is therefore proposed, which combines the heuristic in [8] with the best constraint programming method. Such a hybrid approach performs better than either of these approaches separately.

The interesting aspect of this work is that, as in [60], all the stochastic information is encoded as constraints and expected values, and there is no need of random variables or scenarios. The three models proposed explore different sets of variables and different configurations for the constraint set, for instance using duality. Nevertheless, all the three models use predefined constraints available in standard CP solvers.

7.2.2 A Stochastic Allocation and Scheduling Problem.

The problem, discussed in [39], is the scheduling problem described in Section 3.2 applied to multiprocessor systems on chip: given a conditional task graph characterizing a target application and a target architecture, with alternative memory and computation resources, the authors compute an allocation and schedule that minimize the expected value of communication costs, since — as they point out — communication resources are one of the major bottlenecks in modern multiprocessor systems on chips. The approach they propose is complete and efficient. As in the previous cases, it is based on a deterministic equivalent reformulation of the original stochastic integer linear programming model. More specifically, the authors employ logic based Benders decomposition. The stochastic allocation problem is solved through an Integer Programming solver, while the scheduling problem with conditional activities is handled with CP. The two solvers interact through no-goods. Once more, one of the main contributions is the derivation of an analytical deterministic expression employed in order to compute the expected value of communication costs in the objective function. This expression makes it possible for the authors to transform the original stochastic allocation problem into a deterministic equivalent one that can be solved using any available Integer Programming solver.

7.2.3 Scheduling Internal Audit Units.

In [60] the authors analyze the problem of scheduling internal audit units discussed in Section 3.8. A stochastic programming formulation is proposed with Mixed Integer Linear Programming and CP certainty-equivalent models. Both the models transform analytically the chance-constraints in the model into deterministic equivalent ones. In experiments neither approach dominates the other. However, the CP approach is orders of magnitude faster for large audit times, and almost as fast as the MILP approach for small audit times.

Finally, we discuss works in which the deterministic model obtained through reformulation for a given stochastic program is not "equivalent"; rather, it is based on some simplifying assumption that makes it possible to obtain a compact deterministic formulation able to provide a near-optimal solution and an approximate value for the cost of such a solution, or a bound for such a cost.

7.2.4 Job Shop Scheduling with Probabilistic Durations.

In [3] an approximate deterministic reformulation is employed to compute valid bounds to perform cost-based filtering. In this work the authors analyze the Job Shop Scheduling problem discussed in Section 3.4, in which the objective is to find the minimum makespan. In contrast to the classic formulation presented in [23], in [3] the authors assume that the job durations are probabilistic. The objective is therefore accordingly modified to account for uncertainty. More specifically, the authors search for a proactive plan, consisting of a partial order among activities and of resource-activity allocations, which attains the lowest possible makespan with probability greater or equal to a given threshold. For this problem the authors propose a deterministic formulation, which depends on a given non-negative parameter q. A correct choice of such a parameter guarantees that the minimum makespan for the deterministic model is a lower bound for the minimum makespan that can be attained with a certain threshold probability in the original model. This deterministic model can be efficiently solved with classic constraint programming techniques and can provide tight bounds at each node of the search tree that are employed to perform cost-based filtering. A number of heuristic techniques are proposed for correctly choosing a "good" value for the parameter q.

7.2.5 Production/Inventory Control Problem.

Consider the production/inventory problem discussed in Section 3.6. The deterministic reformulation proposed in Tarim et al. [73] relies on some mild assumptions — discussed in [69] — concerning order-quantities. Under these assumptions, it was possible for the authors to obtain analytical deterministic expressions for enforcing the required service level in each period of the planning horizon, and to compute the expected total cost associated with a given set of policy parameters. By using these expressions, it was possible for the authors to formulate a deterministic model by employing standard constraints available in any CP solver. In [58], the authors compare the solutions obtained through a complete formulation with those obtained with the model in [73]. This comparison shows that the assumptions do not significantly compromise optimality, whereas they allow the construction of a model that can significantly outperform the complete one, and solve real-world instances comprising long planning horizons and high demand values.

7.2.6 Local Search for Stochastic Template Design.

In [52] the stochastic template design problem discussed in Section 3.7 is reformulated as a deterministic equivalent constrained optimisation problem, using all possible scenarios and a novel modeling technique to eliminate non-linear constraints. The result is a standard integer linear program that proved to be hard to solve by branch-and-bound. However, a local search algorithm design for linear integer programs performed very well, and was more scalable than the Bender's decomposition algorithm in [72].

8 Approaches Based on Sampling

In this section we will discuss sample-based approximation strategies for solving problems of decision making under uncertainty. Due to the complexity of these problems in general, several works in the literature have been devoted to analyzing the effectiveness of heuristic approaches based on sampling. In Fig. 9 it is possible to observe how three main trends have been identified in the CP and AI literature, which apply sampling in a hybrid setting for solving problems of decision making under uncertainty: the Sample Average Approximation approach (SAA), Forward Sampling and Sample Aggregation.

- In OR, and particularly in SP, the state-of-the-art technique that applies sampling in combinatorial optimization is the Sample Average Approximation approach [34]. In this approach a given number of samples is drawn from the random variable distributions, and the combinatorial problem of interest is repeatedly solved by considering different samples as input in each run. The real expected cost/profit of a solution produced for a given sample is then computed by simulating a sufficient number of samples. Among all the solutions computed, the one that provides the minimum expected cost (or the maximum expected profit) is retained. Two criteria are given by the authors: one for deciding when a given sample size is no more likely to produce better solutions, and one to decide if it increasing the sample size may lead to better solutions.
- Forward sampling, as the name suggests, is a sort of forward checking that employs samples in order to make inference about which values are not

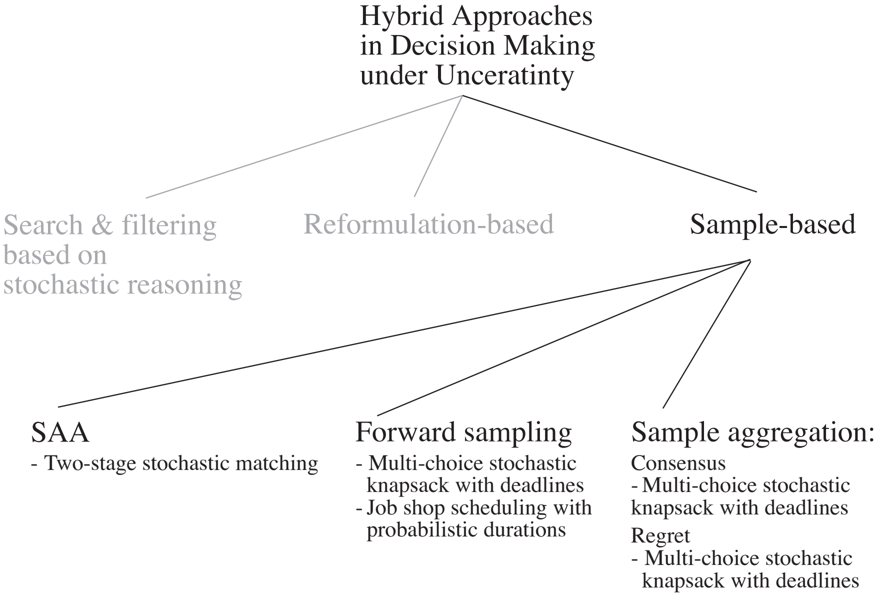

Hybrid Approaches
in Decision Making
under Unceratinty
Search & filtering
based on
stochastic reasoning
Reformulation-based
Sample-based
SAA
- Two-stage stochastic matching
Forward sampling
- Multi-choice stochastic knapsack with deadlines
- Job shop scheduling with probabilistic durations
Sample aggregation:
Consensus
- Multi-choice stochastic knapsack with deadlines
Regret
- Multi-choice stochastic knapsack with deadlines

Fig. 9 A classification of hybrid approaches in CP-AI-OR for decision making under uncertainty: approaches based on sampling
text under figure

consistent in decision variable domains or about the expected cost/profit of associated with a given (partial) assignment for decision variables, which is assessed against the generated samples by computing, for instance, the expected profit/cost of such an assignment with respect to these samples.

- Sample aggregation is a strategy in which a number of samples is generated, for each of these samples a deterministic problem is solved, then the results obtained for all these samples are aggregated and analyzed according to some rule. The "best" among these decisions is implemented in practice. For instance, a possible rule may always choose the decision that is optimal for the highest number of samples.

In the CP and AI literature, sampling is often applied in concert with a so called "online" optimization strategy. Online refers to the fact that decisions and observations are interleaved in the problem, and each time an observation occurs an optimization step takes place to compute the next decision, by taking into account the probability density function of future random variables and the observed values for the past ones. It is easy to notice that a multi-stage stochastic program subsumes an online strategy if the decision maker has a complete knowledge of the probability density function of the random variables in the problem. In this case we may compute the entire solution tree at the beginning, and use it in order to find the best following decision each time a random variable is observed. Nevertheless, several reasons justify the use of an online strategy (also called a “rolling horizon” approach in the OR literature and especially in Inventory Control). The most compelling reason for using an online approach is that it does not require the decision maker to have a complete knowledge of the probability density functions of the random variables. Consider, for instance, the Stochastic Knapsack Problem introduced in the previous sections. If the problem is formulated as a multi-stage stochastic program and we have a full knowledge about the possible weights that can be observed for all the objects, the policy tree will prescribe exactly what to do in each possible future course of action. Nevertheless, if at some stage one of the objects takes a weight that is not part of the probability density function we considered for such an object, the policy tree will not be able to prescribe an appropriate action. In contrast, an online approach would simply take into consideration this weight in the following optimization step and it would however provide a valid decision to be implemented next.

Stochastic problems solved using online strategies, and to which either forward sampling or sample aggregation strategies are applied, appear in a number of works within the CP and AI literatures. In what follow we shall classify some of these works on the basis of which sampling technique is applied.

8.1 Sample Average Approximation

In this section we provide a pointer to a work that proposes to apply SAA to a modified version of a classic matching problem: the two-stage stochastic matching problem.

8.1.1 Two-stage Stochastic Matching Problem.

In [33] Katriel et al. consider the two-stage stochastic matching problem discussed in Section 3.5. The authors prove lower bounds and analyze efficient strategies. We do not provide here a general survey for this work, as the reader may refer to the cited article for more details. Instead, we focus on one of the authors’ contributions in which they firstly observe that, in this problem, with independently activated vertices the number of scenarios is extremely large. However, in such a situation there is often a black box sampling procedure that provides, in polynomial time, an unbiased sample of scenarios; then they observe that one can use the SAA method to simulate the explicit scenarios case and, under some mild assumptions, obtain a tight approximation guarantee. The main observation is that the value of the solution defined by taking a polynomial number of samples of scenarios tightly approximates the value of the solution defined by taking all possible scenarios.

8.2 Forward Sampling

In this section we survey two relevant works in which forward sampling is applied: the multi-choice stochastic knapsack with deadlines and the job shop scheduling with probabilistic durations.

8.2.1 Multi-Choice Stochastic Knapsack with Deadlines.

In [5] the authors analyze different techniques for performing online stochastic optimization. A benchmark problem is proposed in order to assess all these different techniques. The benchmark stemmed from the authors' industrial experience and it consist of a Multi-Choice Stochastic Knapsack with Deadlines. This problem corresponds, in practice, to the stochastic reservation problem discussed in Section 3.3 and it is used to test four different online strategies exploiting combinations of the stochastic and combinatorial aspects of the problem. These strategies are, respectively, forward sampling, average values, most likely scenario analysis and yield management techniques.

Initially, the authors propose two naive order handling policies: a first-come/first-serve policy and a best-fit policy. Furthermore, in order to assess the quality of a given policy, the authors also discuss "far seeing" strategies, which assume advanced knowledge of the realized demand and can therefore solve the associated deterministic multi-choice knapsack problem.[4]

One of the strategies used in this work to estimate the quality of a given policy — for instance first-come/first-serve or best-fit — employs forward sampling in order to generate samples from the current date to the end of the planning horizon. The evaluation of a sample can be done, for instance, by simulating the behavior of a best-fit strategy for the specific sample. The policy evaluation then will be a measure (for instance the average) over the evaluations of many generated samples.

8.2.2 Job Shop Scheduling with Probabilistic Durations.

Forward sampling is also employed in [3]. We recall that in this work the authors analyze the Job Shop Scheduling problem discussed in Section 3.4, in which the authors assume that the job durations are probabilistic. A number of algorithms are proposed for solving this problem through sampling. Firstly, a branch-and-bound procedure is introduced, which exploits at each node of the search tree a Monte Carlo simulation approach to compute — with respect to the partial assignment associated with such a node — a valid lower bound for the minimum possible makespan that may be attained with

[4] We recall that in SP this corresponds to using a wait-and-see policy and performing a posterior analysis.

a probability greater or equal to the given threshold. Since sampling is employed for computing the bound, confidence interval analysis is employed to estimate if the attainment probability associated with the given makespan is a sufficiently reliable estimate. Secondly, the authors propose a number of heuristic techniques that aim to limit the amount of time spent on Monte Carlo simulation during the search, by using the deterministic makespan as an oracle for selecting and simulating only the most promising plans in order to save CPU time and to dedicate more time to the exploration of the search space rather than on simulating non-promising plans. Finally, dedicated tabu search strategies are proposed in order to propose a valid alternative to the constructive search techniques above, which are mainly based on tree-search.

8.3 Sample Aggregation

In this section, we discuss works in which two alternative sample aggregation strategies are employed: the "Consensus" strategy and the "Regret" strategy. The problem to which these strategies are applied is, once more, the multi-choice stochastic knapsack with deadlines.

8.3.1 Multi-Choice Stochastic Knapsack with Deadlines.

In [27] the authors consider the same Online Multi-Choice Knapsack with Deadlines problem considered in [5]. In order to solve this problem the authors employ the following online algorithm. The algorithm receives a sequence of online requests and starts with an empty allocation. At each decision point the algorithm considers the current allocation and the current request, and chooses a bin in which to allocate the request, which is then included in the current assignment. Eventually, the algorithm returns the final allocation and the respective value. In order to decide in which bin to allocate a given request, the algorithm employs a function "chooseAllocation" which is based on two black boxes: a function "getSample" that returns a sample of the arrival distribution; and a function "optSol" that, given the current assignment and a request, returns an optimal allocation of the request by taking into account the past decisions. The authors then consider four possible options for implementing "chooseAllocation":

- The best-fit strategy discussed in [5].
- A strategy called "Expectation" — in practice performing a forward sampling —- that generates future requests by sampling and that evaluates each possible allocation for a given request (i.e. in which bin to fit such a request) against the samples.
- A strategy called "Consensus", which was introduced in [47], and whose key idea is to solve each sample only once. More specifically, instead of

evaluating each possible bin at a given time point with respect to each sample, "consensus" executes the optimization algorithm only once per sample. The bin to which the request is eventually allocated by this optimization step is then credited with the respective profit, while the other bins receive no credit. The algorithm eventually returns the bin with which the highest profit is associated.

- A strategy called "Regret" [6, 7] based on a sub-optimality approximation, which is a fast estimation of the loss caused by sub-optimal allocations. The key steps in the process of choosing a bin resemble the "consensus" algorithm. But in "regret", instead of assigning some credit only to the bin selected by the optimal solution, the sub-optimality approximation is used to compute, for each possible request allocation, an approximation of the best solution that makes such a choice. Therefore every available bin is given an evaluation for every sample at a given time, at the cost of a single optimization.

Consensus and regret are two examples of what we previously defined as "sample aggregation" strategies.

9 Related Works

In this section we will first briefly discuss Stochastic Dynamic Programming, a related and well established technique in OR that deals with decision making under uncertainty. We will also clarify why this technique has not been covered in the former sections. Secondly, we will cast our work within a broader picture, and contrast our survey with existing similar works that address the topics of uncertainty and change.

9.1 Stochastic Dynamic Programming

An alternative and effective technique for modeling problems of decision making under uncertainty is Dynamic Programming. In [4] Bellman explicitly states that Dynamic Programming was initially conceived for modeling multi-stage decision processes. He also argues that these processes arise in practice in a multitude of diverse field and in many real life problems, for instance in stock control, scheduling of patients through a medical clinic, servicing of aircraft at an airfield, etc. Dynamic Programming has been applied to a multitude of deterministic multi-stage decision problems, but in [4] Bellman also discussed its application to stochastic multi-stage decision processes. As in the deterministic case, in the stochastic case the modeling also relies mainly on the development of adequate functional equations capturing the dynamics of the system, and the expected cost (or profit) function associated with the possible decisions and affected by the random variables in the problem. The multi-stage decision process, in Dynamic Programming, is typically defined recursively, starting from a bounding condition that describes a degenerate state of the system that can be easily characterized. Depending on the specific nature of the process being analyzed (Markovian, Semi-Markovian, etc. — see [25], Chapter 8) it is possible to exploit its structure to devise efficient solution methods or closed form solutions for the optimal control policy, which corresponds to the policy tree that constitutes a solution of a given Stochastic Program.

In this work we mainly focused on the connections between and integration of SP, CP and AI. So far Dynamic Programming has not played a role as significant as SP in the development of hybrids approaches for decision making under uncertainty. For this reason, Stochastic Dynamic Programming and its extension to infinite horizon case Markov Decision Processes are not thoroughly covered here. For more details on Stochastic Dynamic Programming the reader may refer to the seminal work of Bellman [4], and to the works of Bertsekas [9], Warren [49], Sutton and Barto [66] and Gosavi [25].

9.2 Related Modeling Frameworks

Recently, the topic of decision making under *uncertain* and *dynamic* environment has been discussed in two literature surveys [76, 14]. Nevertheless, these two works discuss a variety of different problems that can hardly be classified within a unique group. For instance, consider a problem whose structure changes dynamically over time. As an example we may refer to the Dynamic Constraint Network discussed in [18], in which, from time to time, new facts that become known about the model induce a change in the constraint network. We find that such a problem has almost nothing in common with a problem where some parameters are random — thus may assume a certain value with a given probability — and a decision has to be taken proactively, before the realized values for these parameters are known. As an example for this second class, we may consider the proactive Job Shop Scheduling problem discussed in [3], in which an optimal plan — that achieves a minimum makespan with a certain probability — has to be determined before the actual job durations are known. Also consider, as in [22], a constraint satisfaction problem in which we allow some of the constraints to be violated by a solution, and in which we search for a solution that tries to satisfy the original constraint problem as much as possible; or, alternatively, consider a constraint satisfaction problem, as in [26], in which some of the values in the decision variable domains may suddenly become unavailable after a solution has been computed and for which we are looking for robust solutions that can be "repaired" with little effort. These two latter examples, again, significantly differ from the previous ones and among each others.

A clear and comprehensive classification of all these different problems and frameworks is still missing. For this reason, in this section we propose a classification in three distinct classes and we try to position in each of these classes some of the frameworks proposed in the literature.

In our classification (Fig. 10) there are three criteria based on which a particular framework is classified: Degree of Change, Degree of Satisfiability and Degree of Uncertainty.

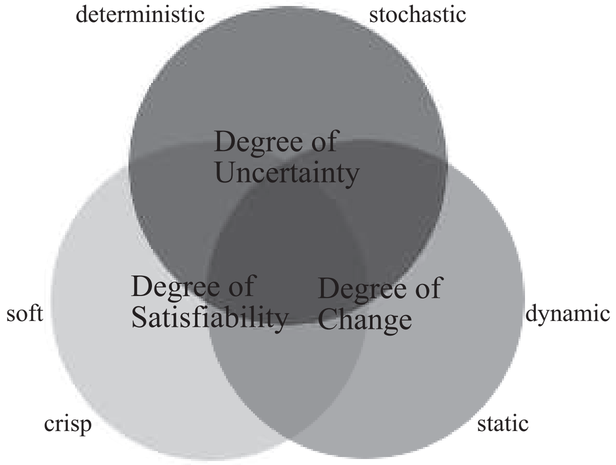


Fig. 10 A classification for existing frameworks based on problem structure

- With respect to the **Degree of Change**, "static" refers to a classic, static CSP, while "dynamic" refers to the fact that the model is assumed to change dynamically, since constraints are added/removed. The solution has to be flexible enough to be adapted to these changes without too many modifications and with limited computational effort. Existing frameworks that, with respect to the Degree of Change, are classified as "dynamic" are: Dynamic Constraint Satisfaction (Dechter [18]); Conditional CSP (Minton et al. [48]); and Super-solutions in CP (Hebrard et al. [26]).
- With respect to the **Degree of Satisfiability** "crisp" refers to a classic CSP in which all the constraints have to be satisfied by a given solution, while "soft" refers to the fact that some of the constraints in the model may be violated by a solution. The aim is to find a solution that typically violates the minimum number of constraints or that, in general, minimizes some violation measure. Existing frameworks that, with respect to the Degree of Satisfiability, are classified as "soft" are: Partial Constraint Satisfaction (Freuder [20]); Constraint solving over semi-rings (Bistarelli et al. [12]); and Valued Constraint Satisfaction (Schiex et al. [63]).

- With respect to the **Degree of Uncertainty**, "deterministic" refers to classic CSPs, while "stochastic" refers to the existence of uncontrollable (random) variables in the model for which a probability distribution is given. Stochastic problems present an alternation of decisions and observations. Constraints are assigned a satisfaction threshold that must be met by any given solution.

Some of the frameworks presented in the literature do, in fact, cover more than one of the classes presented, and for this reason the circles are intersecting each others. Clearly, this classification does not cover several other frameworks that in the years have been proposed to deal with other problem classes.

We have introduced pointers to relevant frameworks that can be either classified under Degree of Change ("dynamic") or Degree of Satisfiability ("soft"). Problems that are classified as "stochastic" with respect to their Degree of Uncertainty have been widely surveyed in the former part of this work. We argue that such a classification better positions existing works with respect to aspects that are, in fact, orthogonal among each others.

10 Conclusions

In this survey we focused on hybrid CP-AI-OR methods for decision making under uncertainty. Firstly, we explicitly defined what "uncertainty" is and how it is possible to model it by using SP, a well established existing modeling framework in OR. We surveyed additional existing frameworks — one from AI and one from CP — for modeling problems of decision making under uncertainty and we also identified the relevant connections among these frameworks. Secondly, we introduced a list of problems from the literature in which uncertainty plays a role and we categorized existing hybrid techniques that have been proposed for tackling these problems into three classes. In the first class we identified general and special purpose approaches that perform "stochastic reasoning". In the second class we listed approaches, once more general and special purpose, that use reformulation. In the third class we categorized approximate techniques based on a variety of strategies employing sampling. Finally, we pointed out connections with other related works.

Acknowledgements S. Armagan Tarim and Brahim Hnich are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027. S. Armagan Tarim is supported also by Hacettepe University BAB.

```
int k = ...;
int p = ...;
int c = ...;
float θ = ...;
range Items 1..k;
range onestage 1..1;
stoch myrand[onestage]=...;
float W[Items,onestage]^myrand = ...;
float r[Items] = ...;
dvar float+ z;
dvar int x[Items] in 0..1;

maximize sum(i in Items) x[i]*r[i] - expected(p*z)
subject to{
   z >= sum(i in Items) W[i]*x[i] - c;
   prob(sum(i in Items) W[i]*x[i] <= c) >= θ;
};
```

Fig. 11 Stochastic OPL formulation for the single-stage SKP

Appendix

In [71] Stochastic OPL, a language for modeling stochastic constraint programs, is proposed. We will now show how the single and multi-stage SKP problems introduced in Section 2 can be easily modeled using such a language.

In Fig. 11 the Stochastic OPL model for the single stage SKP is presented. As in the model presented in Fig. 1, the objective function maximizes the revenue brought by the objects in the knapsack minus the expected penalty for exceeding capacity. Chance-constraint `prob(sum(i in Items) W[i]*x[i] <= c) >=` θ ensures that the capacity is not exceeded with a probability higher than θ.

We now refer to the numerical Example 1 for SKP. In Fig. 12 the Stochastic OPL data file corresponding the numerical instance in Example 1 is presented. We recall that the strategy proposed in [71] employs a scenario-based formulation. In fact it is easy to see that, given the random variables in the example and the values in their domains, there are a total of 32 scenarios that should be considered. Each row for variable `W` in Fig. 12 has, in fact, 32 entries (i.e. `[<10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10, 8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8>]`). There are in total 5 rows, each having 32 entries, so a column — containing all the entries at the same position in each row — therefore fully encodes one of the possible 32 scenarios. The probability of each of the 32 scenarios is provided using the array `myrand`. By using the compilation strategy proposed in [71], any model and data file written using Stochastic OPL can be easily compiled into a classic (deterministic) constraint program and solved by using classic solvers. The optimal solution for Example 1 — computed using the compiled OPL code obtained from the

```
k = 5;
p = 2;
c = 30;
θ = 0.6;
W = [
          [<10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,8,8,8,
            8,8,8,8,8,8,8,8,8,8,8,8,8>],
          [<9,9,9,9,9,9,9,9,12,12,12,12,12,12,12,12,9,9,9,9,9,9,
            9,9,12,12,12,12,12,12,12,12>],
          [<8,8,8,8,13,13,13,13,8,8,8,8,13,13,
            13,13,8,8,8,8,13,13,13,13,8,8,8,8,13,13,13,13>],
          [<4,4,6,6,4,4,6,6,4,4,6,6,4,4,6,6,4,4,
            6,6,4,4,6,6,4,4,6,6,4,4,6,6>],
          [<12,15,12,15,12,15,12,15,12,15,12,15,12,15,12,15,
            12,15,12,15,12,15,12,15,12,15,12,15,12,15,12,15>]
];
myrand = [
          <0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),
          0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),
          0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),
          0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),
          0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),
          0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),
          0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),
          0.0 (0.03125),0.0 (0.03125),0.0 (0.03125),0.0 (0.03125)>
];
r = [16,16,16,5,25];
```

Fig. 12 Stochastic OPL Data File for the single-stage SKP

Stochastic OPL model and data file presented — selects items $\{1, 4, 5\}$ and achieves an expected profit of 45.75, as shown in Fig 2.

The SKP can be also formulated as a multi-stage stochastic constraint program as shown in Fig 3. In Fig. 13 the Stochastic OPL model for the multi-stage SKP is presented. The model is similar to the one presented in Fig. 11. Nevertheless, now the weight of each object is observed at a different decision stage. Therefore we have an array of k random variables (`stoch W[Items]`) in contrast to the previous model that only had one random variable (`myrand`) to model the probability distribution of the possible scenarios. In Fig. 14 the data file corresponding to the numerical instance in Example 1 is presented. The optimal solution for Example 1, when the problem is formulated as a multi-stage Stochastic COP, can be computed using the compiled OPL code obtained from the Stochastic OPL model in Fig. 13 and from the data file presented in Fig. 14. This solution takes the form of a policy tree — graphically rendered in Fig. 4 — and achieves an expected profit of 47.75.

```
int k = ...;
int p = ...;
int c = ...;
float θ = ...;
range Items 1..k;
stoch W[Items]=...;
float r[Items] = ...;
dvar float+ z;
dvar int x[Items] in 0..1;

maximize expected(sum(i in Items) x[i]*r[i]) - p*expected(z)
subject to{
   z >= sum(i in Items) W[i]*x[i] - c;
   prob(sum(i in Items) W[i]*x[i] <= c) >= θ;
};
```

Fig. 13 Stochastic OPL formulation for the multi-stage SKP

```
k = 5;
p = 2;
c = 30;
θ = 0.6;
W = [
           <10(0.5),8(0.5)>,
           <9(0.5),12(0.5)>,
           <8(0.5),13(0.5)>,
           <4(0.5),6(0.5)>,
           <12(0.5),15(0.5)>
];
r = [16,16,16,5,25];
```

Fig. 14 Stochastic OPL Data File for the multi-stage SKP

References

1. K. Apt. *Principles of Constraint Programming.* Cambridge University Press, Cambridge, UK, 2003.
2. T. Balafoutis and K. Stergiou. Algorithms for stochastic csps. In Frédéric Benhamou, editor, *Principles and Practice of Constraint Programming, CP 2006, Proceedings*, volume 4204 of *Lecture Notes in Computer Science*, pages 44–58. Springer, 2006.
3. J. C. Beck and N. Wilson. Proactive algorithms for job shop scheduling with probabilistic durations. *J. Artif. Intell. Res. (JAIR)*, 28:183–232, 2007.
4. R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.
5. T. Benoist, E. Bourreau, Y. Caseau, and B. Rottembourg. Towards stochastic constraint programming: A study of online multi-choice knapsack with deadlines. In Toby Walsh, editor, *Principles and Practice of Constraint Programming, CP 2001, Proceedings*, volume 2239 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2001.

6. R. Bent and P. Van 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*, pages 501–506, 2004.
7. R. Bent, I. Katriel, and P. Van Hentenryck. Sub-optimality approximations. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, pages 122–136. Springer, 2005.
8. O. Berman, J. Wang, and K. P. Sapna. Optimal management of cross-trained workers in services with negligible switching costs. *European Journal of Operational Research*, 167(2):349–369, 2005.
9. D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.
10. L. Bianchi, M .Dorigo, L. Gambardella, and W. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2):239–287, 2009.
11. J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming.* Springer Verlag, New York, 1997.
12. S. Bistarelli, U. Montanari, and F. Rossi. Constraint solving over semirings. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI '95*, pages 624–630, 1995.
13. L. Bordeaux and H. Samulowitz. On the stochastic constraint satisfaction framework. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 316–320, New York, NY, USA, 2007. ACM.
14. K. N. Brown and I. Miguel. Uncertainty and change. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 21. Elsevier, 2006.
15. A. Charnes and W. W. Cooper. Deterministic equivalents for optimizing and satisficing under chance constraints. *Operations Research*, 11(1):18–39, 1963.
16. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.
17. M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.
18. R. Dechter and A. Dechter. Belief maintenance in dynamic constraint networks. In *Proceedings of the 7th National Conference on Artificial Intelligence, AAAI '88*, pages 37–42, 1988.
19. H. Fargier, J. Lang, R. Martin-Clouaire, and T. Schiex. A constraint satisfaction framework for decision under uncertainty. In *UAI '95: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, August 18-20, 1995, Montreal, Quebec, Canada*, pages 167–174, 1995.
20. E. C. Freuder. Partial constraint satisfaction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, IJCAI '89*, pages 278–283. Morgan Kaufmann, 1989.
21. E. C. Freuder and P. D. Hubbe. Extracting constraint satisfaction subproblems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI '95, Montral, Qubec, Canada, August 20-25*, pages 548–557. Morgan Kaufmann, 1995.
22. E. C. Freuder and R. J. Wallace. Partial constraint satisfaction. *Artif. Intell.*, 58(1-3):21–70, 1992.
23. 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.
24. F. J. Gomez, J. Schmidhuber, and R. Miikkulainen. Efficient non-linear control through neuroevolution. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*, volume 4212 of *Lecture Notes in Computer Science*, pages 654–662. Springer, 2006.
25. A. Gosavi. *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning.* Kluwer Academic Publishers, Norwell, MA, USA, 2003.

26. E. Hebrard, B. Hnich, and T. Walsh. Super solutions in constraint programming. In Jean-Charles Régin and Michel Rueher, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, First International Conference, CPAIOR 2004, Nice, France, April 20-22, 2004, Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 157–172. Springer, 2004.
27. P. Van Hentenryck, R. Bent, and Y. Vergados. Online stochastic reservation systems. In J. Christopher Beck and Barbara M. Smith, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Third International Conference, CPAIOR 2006, Cork, Ireland, May 31 - June 2, 2006, Proceedings*, volume 3990 of *Lecture Notes in Computer Science*, pages 212–227. Springer, 2006.
28. P. Van Hentenryck, L. Michel, L. Perron, and J.-C. Régin. Constraint programming in opl. In Gopalan Nadathur, editor, *Proceedings of the International Conference on Principles and Practice of Declarative Programming (PPDP'99)*, volume 1702 of *Lecture Notes in Computer Science*, pages 98–116, September 29 - October 1 1999.
29. N. M. Hewahi. Engineering industry controllers using neuroevolution. *AI EDAM*, 19(1):49–57, 2005.
30. B. Hnich, R. Rossi, S. A. Tarim, and S. D. Prestwich. Synthesizing filtering algorithms for global chance-constraints. In *Principles and Practice of Constraint Programming, CP 2009, Proceedings*, volume 5732 of *Lecture Notes in Computer Science*, pages 439–453. Springer, 2009.
31. H. Jeffreys. *Theory of Probability.* Clarendon Press, Oxford, UK, 1961.
32. P. Kall and S. W. Wallace. *Stochastic Programming.* John Wiley & Sons, 1994.
33. I. Katriel, C. Kenyon-Mathieu, and E. Upfal. Commitment under uncertainty: Two-stage stochastic matching problems. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 171–182. Springer, 2007.
34. 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.
35. M. L. Littman, J. Goldsmith, and M. Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1–36, 1998.
36. M. L. Littman, S. M. Majercik, and T. Pitassi. Stochastic boolean satisfiability. *J. Autom. Reasoning*, 27(3):251–296, 2001.
37. B. Liu. Dependent-chance programming: A class of stochastic optimization. *Computers & Mathematics with Applications*, 34:89–104, 1997.
38. B. Liu and K. Iwamura. Modelling stochastic decision systems using dependent-chance programming. *European Journal of Operational Research*, 101:193–203, 1997.
39. M. Lombardi and M. Milano. Stochastic allocation and scheduling for conditional task graphs in mpsocs. In Frédéric Benhamou, editor, *Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings*, volume 4204 of *Lecture Notes in Computer Science*, pages 299–313. Springer, 2006.
40. M. Lombardi and M. Milano. Scheduling conditional task graphs. In Christian Bessiere, editor, *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, volume 4741 of *Lecture Notes in Computer Science*, pages 468–482. Springer, 2007.
41. S. M. Majercik. *Planning under uncertainty via stochastic satisfiability.* PhD thesis, Durham, NC, USA, 2000. Supervisor-Littman, Michael L.
42. S. M. Majercik. Appssat: Approximate probabilistic planning using stochastic satisfiability. *Int. J. Approx. Reasoning*, 45(2):402–419, 2007.

43. S. M. Majercik. *Stochastic Boolean Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 27, pages 887–925. IOS Press, February 2009.
44. S. M. Majercik and B. Boots. Dc-ssat: A divide-and-conquer approach to solving stochastic satisfiability problems efficiently. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 416–422. AAAI Press / The MIT Press, 2005.
45. S. M. Majercik and M. L. Littman. Maxplan: A new approach to probabilistic planning. In *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems, Pittsburgh Pennsylvania, USA*, pages 86–93. AAAI Press, 1998.
46. S. M. Majercik and M. L. Littman. Contingent planning under uncertainty via stochastic satisfiability. *Artif. Intell.*, 147(1-2):119–162, 2003.
47. L. Michel and P. Van Hentenryck. Iterative relaxations for iterative flattening in cumulative scheduling. In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004), June 3-7 2004, Whistler, British Columbia, Canada*, pages 200–208. AAAI, 2004.
48. S. Minton, M. D. Johnston, A. B. Philips, and P. Laird. Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artif. Intell.*, 58(1-3):161–205, 1992.
49. W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2007.
50. S. D. Prestwich, A. Tarim, R. Rossi, and B. Hnich. A cultural algorithm for pomdps from stochastic inventory control. In Maria J. Blesa, Christian Blum, Carlos Cotta, Antonio J. Fernández, José E. Gallardo, Andrea Roli, and Michael Sampels, editors, *Hybrid Metaheuristics, 5th International Workshop, HM 2008, Málaga, Spain, October 8-9, 2008. Proceedings*, volume 5296 of *Lecture Notes in Computer Science*, pages 16–28. Springer, 2008.
51. S. D. Prestwich, A. Tarim, R. Rossi, and B. Hnich. A steady-state genetic algorithm with resampling for noisy inventory control. In Günter Rudolph, Thomas Jansen, Simon M. Lucas, Carlo Poloni, and Nicola Beume, editors, *Parallel Problem Solving from Nature - PPSN X, 10th International Conference Dortmund, Germany, September 13-17, 2008, Proceedings*, volume 5199 of *Lecture Notes in Computer Science*, pages 559–568. Springer, 2008.
52. S. D. Prestwich, S. A. Tarim, and B. Hnich. Template design under demand uncertainty by integer linear local search. *International Journal of Production Research*, 44(22):4915–4928, 2006.
53. S. D. Prestwich, S. A. Tarim, R. Rossi, and B. Hnich. Evolving parameterised policies for stochastic constraint programming. In *Principles and Practice of Constraint Programming, CP 2009, Proceedings*, volume 5732 of *Lecture Notes in Computer Science*, pages 684–691. Springer, 2009.
54. S. D. Prestwich, S. A. Tarim, R. Rossi, and B. Hnich. Neuroevolutionary inventory control in multi-echelon systems. In *1st International Conference on Algorithmic Decision Theory*, volume 5783 of *Lecture Notes in Computer Science*, pages 402–413. Springer, 2009.
55. L. Proll and B. Smith. Integer linear programming and constraint programming approaches to a template design problem. *INFORMS J. on Computing*, 10(3):265–275, 1998.
56. R. Rossi, S. A. Tarim, B. Hnich, and S. D. Prestwich. Replenishment planning for stochastic inventory systems with shortage cost. In Pascal Van Hentenryck and Laurence A. Wolsey, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 4th International Conference, CPAIOR 2007, Brussels, Belgium, May 23-26, 2007, Proceedings*, volume 4510 of *Lecture Notes in Computer Science*, pages 229–243. Springer Verlag, 2007.

57. R. Rossi, S. A. Tarim, B. Hnich, and S. D. Prestwich. Cost-based domain filtering for stochastic constraint programming. In P. J. Stuckey, editor, *Principles and Practice of Constraint Programming, 14th International Conference, CP 2008, Sydney, Australia, September 14-18, 2008. Proceedings*, volume 5202 of *Lecture Notes in Computer Science*, pages 235–250. Springer, 2008.
58. R. Rossi, S. A. Tarim, B. Hnich, and S. D. Prestwich. A global chance-constraint for stochastic inventory systems under service level constraints. *Constraints*, 13(4):490–517, 2008.
59. R. Rossi, S. A. Tarim, B. Hnich, S. D. Prestwich, and Cahit Guran. A note on liu-iwamura's dependent-chance programming. *European Journal of Operational Research*, 198(3):983–986, 2009.
60. R. Rossi, S. A. Tarim, B. Hnich, S. D. Prestwich, and S. Karacaer. Scheduling internal audit activities: a stochastic combinatorial optimization problem. *Journal of Combinatorial Optimization*, 2008.
61. G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. Technical report, CUED/F-INFENG/TR 166, Cambridge University, 1994.
62. N. V. Sahinidis. Optimization under uncertainty: State-of-the-art and opportunities. *Computers and Chemical Engineering*, 28:971–983, 2004.
63. T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI '95*, pages 631–639. Morgan Kaufmann, 1995.
64. K. O. Stanley and R. Miikkulainen. Evolving neural network through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.
65. M. L. Stein. Large sample properties of simulation using latin hypercube sampling. *Technometrics*, 29:143–151, 1987.
66. R. S. Sutton and A/ G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, March 1998.
67. S. A. Tarim, B. Hnich, S. D. Prestwich, and R. Rossi. Finding reliable solution: Event-driven probabilistic constraint programming. *Annals of Operations Research*, 171(1):77–99, 2008.
68. S. A. Tarim, B. Hnich, R. Rossi, and S. D. Prestwich. Cost-based filtering techniques for stochastic inventory control under service level constraints. *Constraints*, 14(2):137–176, 2009.
69. S. A. Tarim and B. G. Kingsman. The stochastic dynamic production/inventory lot-sizing problem with service-level constraints. *International Journal of Production Economics*, 88:105–119, 2004.
70. S. A. Tarim and B. G. Kingsman. Modelling and Computing (R^n,S^n) Policies for Inventory Systems with Non-Stationary Stochastic Demand. *European Journal of Operational Research*, 174:581–599, 2006.
71. S. A. Tarim, S. Manandhar, and T. Walsh. Stochastic constraint programming: A scenario-based approach. *Constraints*, 11(1):53–80, 2006.
72. S. A. Tarim and I. Miguel. A hybrid benders' decomposition method for solving stochastic constraint programs with linear recourse. In Brahim Hnich, Mats Carlsson, François Fages, and Francesca Rossi, editors, *Recent Advances in Constraints, Joint ERCIM/CoLogNET International Workshop on Constraint Solving and Constraint Logic Programming, CSCLP 2005, Uppsala, Sweden, June 20-22, 2005, Revised Selected and Invited Papers*, volume 3978 of *Lecture Notes in Computer Science*, pages 133–148. Springer, 2005.
73. S. A. Tarim and B. Smith. Constraint Programming for Computing Non-Stationary (R,S) Inventory Policies. *European Journal of Operational Research*, 189:1004–1021, 2008.
74. D. Terekhov and J. C. Beck. A constraint programming approach for solving a queueing control problem. *J. Artif. Intell. Res. (JAIR)*, 32:123–167, 2008.

75. D. Terekhov and J. Christopher Beck. Solving a stochastic queueing control problem with constraint programming. In Pascal Van Hentenryck and Laurence A. Wolsey, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 4th International Conference, CPAIOR 2007, Brussels, Belgium, May 23-26, 2007, Proceedings*, volume 4510 of *Lecture Notes in Computer Science*, pages 303–317. Springer, 2007.
76. G. Verfaillie and N. Jussien. Constraint solving in uncertain and dynamic environments: A survey. *Constraints*, 10(3):253–281, 2005.
77. T. Walsh. Sat v csp. In Rina Dechter, editor, *Principles and Practice of Constraint Programming, CP 2000, Proceedings*, volume 1894 of *Lecture Notes in Computer Science*, pages 441–456. Springer, 2000.
78. T. Walsh. Stochastic constraint programming. In Frank van Harmelen, editor, *European Conference on Artificial Intelligence, ECAI'2002, Proceedings*, pages 111–115. IOS Press, 2002.
79. Y. Zhuang and S. M. Majercik. Walkssat: An approach to solving large stochastic satisfiability problems with limited time. Technical report.