Distribution Optimization in Constraint Programming

Guillaume Perez
Huawei Technologies Ltd, CSI Paris, Boulogne-Billancourt, France

Gaël Glorian
Huawei Technologies Ltd, CSI Paris, Boulogne-Billancourt, France

Wijnand Suijlen
Huawei Technologies Ltd, CSI Paris, Boulogne-Billancourt, France

Arnaud Lallouet
Huawei Technologies Ltd, CSI Paris, Boulogne-Billancourt, France

Abstract

Stochastic Constraint Programming introduces stochastic variables following a probability distribution to model uncertainty. In the classical setting, probability distributions are given and constant. We propose a framework in which random variables are given a set of possible distributions and only one should be selected. A solution is obtained when all variable distributions are assigned, and all decision variables are assigned too. In such a setting, a constraint on random variables limits the possible distributions its random variables may take. We generalize the notion of *chance* as the probability of satisfaction of a constraint, called *probabilization*, given variable distributions. Probabilization can be seen as a generalization of reification in a random setting whose result is a random variable. We define minimal arithmetic to work with stochastic variables having a variable distribution. Using the introduced representation, our framework can in theory save an exponential number of decisions, and represents problems that were previously not representable with finite integer domains. Finally, we model and solve two industrial problems that require this extension – virtual network configuration and assignment of chemical delivery – and show improvement in terms of quality of solution and speed.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Mathematics of computing → Solvers; Computing methodologies → Probabilistic reasoning

Keywords and phrases Constraint Programming, Optimization, Stochastic Optimization

Digital Object Identifier 10.4230/LIPIcs.CP.2023.29

1 Introduction

Stochastic optimization and chance-constrained programming [6, 41] are classes of problems in which uncertainty is present. In such a setting, both decision variables and random variables are present. Usually the probability distributions of the random variables are known and constant. The goal is to optimize a given objective function on these variable sets and to satisfy a set of constraints with sufficient probability. Optimal production planning, optimal power flow, textile manufacturing, vehicle sharing, and parcel delivery services are just a few of the many industrial areas where stochastic optimization is required [28, 29, 34, 55, 21].

Learning probabilistic distributions, or distribution learning, is a machine learning framework that consists of learning the probability distribution that could generate a given set of samples [22, 24]. In the usual settings, the input is a set of samples drawn from an unknown distribution and the goal is to uncover this unknown distribution. Since, many methods have been proposed, for example using assumptions on the class of probability distribution to be a mixture of Gaussian or a Poisson law etc. [10, 9], or directly using neural networks [1].

Consider the virtual network functions design problem [12, 13, 16, 48]. The main part of this problem consists of selecting the settings of different nodes of a network function, such that the global computation latency is robust. Robustness here implies that the total latency (i.e. the sum of the latency of each node) is smaller than a given value, with a probability of at least γ. Locally, for each node the latency distribution is a given by a random variable whose distribution is conditioned by the settings of the node. Different settings will lead to different probability distributions for the nodes.

In this paper, we propose to work on *probability distribution optimization.* In this setting, the input is a stochastic constraint optimization problem, and the goal is to find both the assignment of the decision variables and the distribution of the random variables. This can be seen as a generalization of distribution learning in the sense that it is not restricted to the fitting constraints.

Constraint Programming (CP) is an expressive optimization framework often used to solve combinatorial problems such as scheduling. In CP, optimization under chance, confidence, probability, or statistical constraints is a prolific research area [32, 40, 38, 37, 19, 27]. The early and impacting works on stochastic constraint programming defined the basics [14, 53]. Then, optimization methods for chance constraints, Markov, or sampling probability distribution constraints etc. have been proposed [49, 18, 46, 39, 30]. These works have focused either on the multi-stage framework or on one global constraint to extend the optimization process to a stochastic context. But they all consider random variables with fixed distributions.

In this paper, the stochastic CP framework is extended to handle distribution optimization. It is another set of stochastic problems where random variables are given a domain of possible distributions, from which only *one* should be selected. The abstract object *probability distribution variable* is introduced for modeling purposes. It represents the variability of its sample space and its probability. A random variable is *assigned* when its associated probability distribution is known and fixed. *Distribution variables* are now one of the many variables of the problem to solve. A solution is found when all the decision variables are assigned, and when the distribution of all the random variables is fixed. In addition, we propose the definition of *distribution constraints*, which are constraints involving distribution variables. In the *hard* case, a distribution constraint restricts the possible distributions of the random variables it involves. Using the introduced representation, our framework can in theory save an exponential number of decisions, and represents problems that previously could not be represented with a finite integer domain.

Then, we propose to focus on a particular case of constraints namely *relational constraints.* These constraints represent the usual CP constraints, as they are defined by the set of allowed tuples. For these constraints, we propose two new consistency levels, namely P consistency, for probability consistency, and Ω consistency, which is a probability distribution encoding of the usual arc consistency. A direct implication is that most existing stochastic constraints (confidence, sampling, PMF, etc.) that consider the distribution as data of the problem, can be upgraded to deal with variable distributions. That is why in this paper we extend the Confidence/Chance of relational constraints [49, 30, 36]. Furthermore, the notion of *probabilization* of a constraint is proposed. It is a generalization of the chance concept [49], closely related to the reification of constraint, but for probability purpose. The *probabilization* returns a Boolean random variable associated to a distribution variable representing the probability of satisfaction of the constraint.

Distribution variables are abstract objects that can be implemented in diverse ways. We propose an implementation of distribution variables using a direct encoding as a probability mass function. For each possible value of a random variable, we define its probability as a continuous variable. Then, we propose filtering rules defining the minimal arithmetic allowing to model distribution optimization problems.

Finally, we exploit this implementation to solve two distribution optimization problems in the experimental section. The first problem is the stochastic design of virtual network functions, where a simple, yet efficient, model allows to find optimal designs. The second problem we model is a chemical delivery application. It shows how using *variable distributions* allows to have smaller search trees and higher quality solutions.

2 Related Work

Chance constrained optimization [6, 31, 33] is a prolific research area. They are classes of problems in which uncertainty is present. In such setting, both decision variables and random variables are present. Usually the probability distributions of the random variables are known and constant. The first definition was given for discrete distributions and piece-wise linear functions with linear inequalities involving random variables. These inequalities had to be maintained at a given level of probability. In these problems, the random variables can be defined using a joint probability distribution, making them conditional on each other [8], which usually makes the problem harder to solve. This framework has been used for the case where the distribution of random variables was conditioned on storage levels or stream flows [20]. Moreover, joint chance constrained optimization problem are problems that contain multiple uncertain constraints on multivariate random variables. They are jointly required to be satisfied with probability exceeding a threshold [54]. However, it is uncommon to have the probability distribution defined as conditioned by the decision variables, or even constrained. The work proposed in this paper is part of chance constrained optimization and aims to use a constraint programming solver as a modeling and solving framework for problems where finding a distribution that satisfies constraints is also part of the problem.

The probabilistic graphical model community proposed Bayesian networks and influence diagram framework [35, 26] to represent the interactions between decision variables (agents) and the probability distributions of random variables. Such influence diagrams, while they tend to grow exponentially, could be powerful tools to represent the constrained distribution during the optimization process. Several methods have been defined to solve chance constrained optimization. In general, non-linear programming solvers are used [28] and dedicated models are defined. For example, dynamic programming coupled with influence diagram has been used to optimize agent decisions to maximize utility functions [50], in a similar way as reinforcement learning. Furthermore, it is not unusual to use sampling methods to approximate chance optimization [3, 33].

In CP, the *chance* constraint is defined as a policy under uncertainty, guaranteeing robustness of the assignment [53, 49]. More precisely, consider a problem where first a set X_1 of decision variables are assigned. Next, some random variables Y_1 reveal their value, then values are selected for the set of variables X_2 and so on for as many iterations as it is required. The solution of such problem is a tree, and no longer a tuple, for which the specified fraction of the scenarios is satisfied. Since then, this work has been extended multiple times, [17, 18] proposed generic algorithms reusing existing filtering for the global chance constraint. More recently, [30] proposed to restrict the chance constraint to 2 stages optimization, as it is a widely used approach. They proposed a filtering for the conjunction of binary inequalities. Finally, [36] proposed to use MDDs as a new generic filtering for the confidence constraint. All of these are different from the work proposed in this paper. On the one hand, policies are not considered, the proposed framework is restricted to 2-stages similarly to [30, 36]. On the other hand, the proposed framework is the first introducing variable distribution inside of constraint programming. Note that it is one of the possible extensions to stochastic CP as mentioned in the *Extensions* section of [53]. Finally, a recent trend called randomness optimization [23, 2, 15]. is used to optimize the randomness of the solution of a decision problem. For example randomness of solution is important for the design of systems to prevent reverse engineering. Random solution as a constraint is already part of the constraint programming framework [42].

3 Distribution Optimization

3.1 Preliminaries

A sample space Ω is a set representing all the possible outcomes of an experiment. In the general case, the probability of an outcome $v \in \Omega$ is noted $\Pr[v]$. An event is a set of outcomes from the sample space and an event space $\mathcal{F}$ is a set of events. Events are often useful for characterising particular subsets of outcomes. For example, the latency of a process might be a real number, but the probability will be usually defined by segments of time (i.e. intervals). Finally, P is a probability function that maps each element e of $\mathcal{F}$ to a probability $P(e) \in [0, 1]$. These three elements form a *probability space*, denoted by the triplet $\psi = (\Omega, \mathcal{F}, P)$. A random variable y follows a probability space ψ if its distribution is defined by the probability space ψ. This will be noted $y \sim \psi$. When Ω is discrete or countable, a probability mass function (PMF) $pmf \colon \Omega \to \mathbb{R}$ assigns a probability $\Pr[v]$ to each value $v \in \Omega$. Let $Y = (y_1, ..., y_r)$ be a vector of r independent random variables. By definition, the probability of a sample of independent variables (i.e. a tuple) is given by the product of the probabilities of the selected values. More precisely, the probability of a tuple $t = (a_1, ..., a_r)$ is defined by: $\Pr[Y = t] = \prod_{i=1}^{r} \Pr[y_i = a_i]$. When a subset Y is composed of variables that are not independent, it is possible to replace them by a single random variable Y^* representing their aggregation [7]. The domain of outcomes of this variable is the Cartesian product of the outcomes of the variables of Y. So we can assume in this paper that all random variables are independent.

In the rest of this paper, $X = \{x_1, \ldots, x_r\}$ denotes a set of variables (usually integer). $Y = \{y_1, \ldots, y_k\}$ denotes a set of random variables. Given a constraint C, $\mathbb{T}_C(v_1, \ldots, v_n) = 1$ (resp. $= 0$) implies that constraint C is satisfied by the tuple $(v_1, \ldots, v_n)$ (resp. unsatisfied). $D(x)$ denotes the current domain of variable x. We denote by $\overline{D(x)}$ (resp. $\underline{D(x)}$) the upper bound (resp. lower bound) of variable x. we denote by $v \in D(x)$ the fact that a value v belongs to the domain of a variable x. In a similar way, let $\forall t \in D^{\times}(X) = \prod_{i=1}^{n} D(X_i)$ denotes all the tuples in the Cartesian product of the current domain of variables in X.

3.2 Distribution Variables

In this section, we extend the CP framework with the definition of *probability distribution variables.*

▶ **Definition 1.** *A probability distribution variable r is a variable defining a probability space. The domain $D(r)$ of a distribution variable r is a set of probability spaces.*

Recall that a probability space defines its sample space Ω, its event space $\mathcal{F}$ and the probability P of each event in it. *Distribution variables* are now one of the many variables of the problem, similar to any other decision variable. A probability distribution variable is assigned when the probability of each value in its event space is known and fixed. Let r be a random distribution variable. $\psi \in D(r)$ denotes the membership of a probability space ψ to the current domain of the distribution variable r. We denote by $y \sim r$ the fact that the random variable y will be drawn following the distribution variable r, and by $y \sim \psi$ that random variable y follows the distribution defined by the probability space ψ (a constant). Let y_1 and y_2 be two random variables such that $y_1 \sim r$, $y_2 \sim r$. Such notation implies that they are independent and identically distributed (IID). We denote $D(y)$ or $D(r)$ the domain of the probability distribution variable of random variable $y \sim r$. Let $\forall \Psi \in D^{\times}(Y)$ denotes all the tuples of probability spaces in the Cartesian product of the current domain of distribution variables of Y. Let Ω_y be the set of all the outcomes such that there exists a probability space in $D(y)$ where the outcome has a non-zero probability. Let $\forall t_Y \in D^{\times}_{\Omega}(Y)$ denote all the tuples of outcomes (i.e. value) in the Cartesian product of the current sample space Ω_y of random variables in $y \in Y$. We note by $\overline{Pr[Y = t]}$ (resp. $\underline{Pr[Y = t]}$) the upper bound (resp. lower bound) probability of tuple t to be drawn by random variables in Y. We have $\overline{Pr[Y = t]} = \max\{Pr[Y \sim \Psi = t] \mid \Psi \in D^{\times}(Y)\}$ and $\underline{Pr[Y = t]} = \min\{Pr[Y \sim \Psi = t] \mid \Psi \in D^{\times}(Y)\}$. All along the paper, Ψ represents a vector of probability spaces, ψ a probability space, and $\Psi[\{y\}]$ the probability space associated to variable y in Ψ. Finally, $\mathbb{P}$ is the space of all the possible probability spaces (i.e. $\forall \psi \in \mathbb{P}$).

It is interesting to note that integer variables can be represented by the proposed distribution variables.

▶ **Proposition 2.** *A integer variable always has an equivalent distribution variable.*

Proof. Let x be an integer variable with domain $D(x)$. Let r be a distribution variable such that $D(r) = \{(\{v\}, \{\emptyset, \{v\}\}, \text{Pr}[v]{=}1) | \forall v \in D(x)\}$. In other words, for each value v in the domain of x, there is a probability space whose sample space $\Omega = \{v\}$ is restricted to value v. There is a one-to-one correspondence between the values in the domains of r and x, hence x and r are equivalent. Note that the converse is not true as the number of probability spaces in a distribution variable may not be countable. In the rest of this paper, without loss of generality, we assume that all the integer variables have been replaced by equivalent distribution variables for simplicity of notation. ◀

3.3 Distribution Constraints

A random distribution variable being a modeling object of CP, it can be constrained. In this section, we propose a basic definition of distribution constraints. We use this definition to derive a generalization of arc consistency to distribution variables. Finally, we propose to focus on a sub-part of the constraint space, namely relational constraints, that generalizes usual CP constraints.

Distribution constraints are all the constraints involving random variables with variable distribution. A distribution constraint defines the possible probability spaces of the random variables it involves. Integer constraints can always be defined by the set of satisfying tuples. This basic notion is extended to distribution constraints:

▶ **Definition 3.** *Let C be a constraint defined on random variables Y. C is defined by a set of tuples T_R such that $\forall \Psi \in T_R$, Ψ is a tuple of valid probability spaces for the probability distribution of random variables in Y.*

▶ **Example 4.** Table of PMFs The following table represents a constraint on two variables: integer variable x and random variable $y \sim r$.

$\text{Pr}[x = 0]$	$\text{Pr}[x = 1]$	$\text{Pr}[y = 1]$	$\text{Pr}[y = 2]$	$\text{Pr}[y = 3]$	$\text{Pr}[y = 4]$	$\text{Pr}[y = 5]$	$\text{Pr}[y = 6]$
1	0	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
0	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$

Such a table restricts the distribution variables of x and y to two probability spaces, defined by probability mass functions. x is a integer variable encoded as a probability distribution. The possible probabilities are either 0 or 1. y is a true random variable. Depending on the assigned value of x, random variable y will behave differently. Note that PMFs are not the only way to encode probability spaces.

The notion of consistency is of utmost important in constraint programming [45]. We propose to directly map the generalized arc consistency to distribution variables. The principal difference is that the domain of a distribution variable is a set of probability spaces, and not a set of integers as for integer variables for example. Let C be a constraint defined on random variables Y defined by a set of tuples of probability spaces T_R. The consistency properties are defined by:

- A probability space $\psi \in D(y)$ of $y \in Y$ is consistent with C if it exists a support $\Psi \in T_R$ such that $\Psi[\{y\}] = \psi$ and $\Psi \in D^{\times}(Y)$.
- A distribution variable of $y \in Y$ is consistent if each probability space in its domain is consistent.
- A constraint C is generalized arc consistent if all its variables are consistent.

3.4 Relational Constraints

Relational constraints are a subset of distribution constraints. They consider the sampled values of the random variables for the satisfiability check of the constraint. In other words, the constraint can always be defined by the possible assignments for the integer variables and for the random variables. They are composed of all the usual constraints of the CP framework but generalized to variable distributions.

▶ **Definition 5.** *Let C be a relational constraint defined on random variables Y. C is defined by a table of tuples T_C, such that $\forall t \in T_C$ is a valid sample that can be drawn for random variables in Y.*

Note that, for each table T_C, an infinite number of probability spaces are valid. Indeed, consider the binary constraint defined by the tuples $[(0,0),(0,2)]$ on variables x and y. This constraint restricts the sampled value of x to be 0, and restricts the probability distribution of y to any distribution such that $\Pr[y = 0]+\Pr[y = 2] = 1$. This implies that such a definition can be highly compressing, as an infinite number of distributions satisfy this equality. On the other hand, such a definition is not expressive enough to restrict to all subsets of probability spaces. For example, it is impossible to define $E(y) = 3$ (expectation), or simply the constraint defined by the table from the example in section 3.3. That is the reason why relational constraints are a subset of distribution constraints.

▶ **Definition 6** (Relational Constraint). *Let $\mathbb{T}_C(t) = 1$ if $t \in T_C$ and 0 otherwise be a predicate function for constraint C. We propose to use the following definition for the relational constraint:*

$$\int_{t \in D^{\times}_{\Omega}(Y)} Pr[Y = t]\mathbb{T}_C(t) = 1 \quad ; \quad \int_{t \in D^{\times}_{\Omega}(Y)} Pr[Y = t]\neg\mathbb{T}_C(t) = 0 \tag{1}$$

The most intuitive description of such constraint is that it does not imply that all valid tuples must be reached, but that all reachable tuples must be valid.

The notion of consistency of a probability space in the case of a relational constraint can be redefined using its definition. Given a relational constraint C defined on random variables Y by the relational table T_C.

▶ **Definition 7.** *A probability space $\psi \in D(y)$ of $y \in Y$ is consistent with C if and only if:*

$$\exists \Psi \in D^{\times}(Y), \Psi[\{y\}] = \psi \wedge \forall t_Y \in D^{\times}_{\Omega}(Y), Pr[Y \sim \Psi = t_Y] \leq \mathbb{T}_C(t) \tag{2}$$

It is a direct rewriting of the consistency of distribution constraints using the relational definition (equation (1)). Then, the domain of probability variable $y \in Y$ is consistent with constraint C if all the probability spaces in it are consistent with C. Finally, a constraint C is consistent if all the variables in it are consistent.

Let ψ be a probability space and Y_y^{ψ} be a random distribution variable vector where variable y is restricted to probability space ψ. A weaker consistency can be defined for relational constraints and distribution variables. This consistency is close to bound-consistency. It considers the bounds of the probability of events given the current domain of a distribution variable.

▶ **Definition 8.** *A probability space $\psi \in D(y)$ of $y \in Y$ is P-bound consistent with C if:*

$$\forall t \in D^{\times}_{\Omega}(Y_y^{\psi}), \underline{Pr[Y_y^{\psi} = t]} \leq \mathbb{T}_C(t) \tag{3}$$

Then, the domain of probability variable $y \in Y$ is P-bound consistent with constraint C if all the probability spaces in it are P-bound consistent with C. Finally, a constraint C is P-bound consistent if all the variables in it are P-bound consistent: $\forall t \in D^{\times}_{\Omega}(Y), \underline{\Pr[Y = t]} \leq \mathbb{T}_C(t)$. Now that the P-bound consistency is defined, we propose to focus on the case $\mathbb{T}_C(t) = 0$ as it is the only one able to invalidate probability spaces. Let $Y_{\setminus\{y\}}$ be the vector of variables Y without variable y. Let $t_{\setminus\{y\}}$ be the tuple t without the value at the position of variable y. Propagating the following logical implication is enough to enforce P-bound consistency constraint C:

$$\forall y \in Y, \forall t \in D^{\times}_{\Omega}(Y), \underline{\Pr[Y_{\setminus\{y\}} = t_{\setminus\{y\}}]} > \mathbb{T}_C(t) \implies \Pr[y = t_y] = 0 \tag{4}$$

This pruning rule implies that all the probability spaces that could lead to an invalid tuple should be removed. It will later be derived to extract filtering rules for the implementation of this framework.

▶ **Definition 9.** *Given a relational constraint C defined on random variables Y by the relational table T_C. A probability space $\psi \in D(y)$ of $y \in Y$ is Ω-consistent with C if:*

$$\exists t \in D^{\times}_{\Omega}(Y_y^{\psi}), \mathbb{T}_C(t) \tag{5}$$

Ω-consistency ensures that it exists at least one tuple with non-zero probability to satisfy the predicate given the other domains. Ω-consistency, when all the variables are integer variables encoded as distribution variables, is equivalent to global arc consistency as it finds a support for each value in the domain of each variable.

Consider a total order on the outcomes of Ω_y. Let $\overline{\Omega_y}$ (resp. $\underline{\Omega_y}$) be the largest (resp. smallest) element of Ω_y with respect to the total order. Let $D^{\times}_{\underline{\Omega}}(Y)$ denote the Cartesian product of the intervals $[\underline{\Omega_y}, \overline{\Omega_y}], \forall y \in Y$.

▶ **Definition 10.** *A probability space $\psi \in D(y)$ of $y \in Y$ is Ω-bound-consistent with C if:*

$$\exists t \in D^{\times}_{\underline{\Omega}}(Y_y^{\psi}), \mathbb{T}_C(t) \tag{6}$$

When all the variables are integer variables encoded as distribution variables and the total order is defined by the operator $<$, Ω-bound-consistency is equivalent to bound-consistency.

Consider the Confidence constraint that restricts the decision variables to values such that the probability of satisfaction is greater than a given threshold [49, 30, 36]. We propose to generalize it by considering variable distributions for random variables.

▶ **Definition 11.** *Given a relational constraint $C(Y)$, with Y a vector of random variables with variable distributions. Given a confidence threshold γ. The generalized confidence is:*

$$\int_{t \in D_\Omega^\times(Y)} Pr[Y = t]\mathbb{T}_C(t) > \gamma \tag{7}$$

Previously, propagation of chance constraint was only removing values from integer variables that could not lead to a robust enough solution. The new propagation of this constraint will impact both the integer variables (here encoded as distributions) and the distribution variables as probability spaces that are not robust enough will be removed too.

3.5 Views on Random Variables

We propose to make a clear distinction between the three equality operators: "$y_1 = y_2$", "$y_1 \simeq y_2$", and "$y_1 \equiv y_2 + y_3$". First, $y_1 = y_2$ is the classic equality constraint, it implies that the value of the random variables must be equal, but they are considered different random variables. Second, the constraint $y_1 \simeq y_2$ ensures that $y_1 \sim r_1$ and $y_2 \sim r_2$ are *equal in law*. More precisely it ensures that $r_1 = r_2$. This implies that their random assignment might be different, but the constraint is satisfied if they follow the same distribution. From a probability point of view, they are independent. Finally, $y_1 \equiv y_2 + y_3$, where y_1 is the actual summation of the two random variables y_2 and y_3. From a probability point of view, they are dependent.

Views are a useful abstraction in constraint programming [47, 51]. They prevent the creation of intermediate variables representing some function on a variable ($x' = f(x)$). Views can be adapted to random variables directly. Consider for example the affine view $y_1 \equiv ay + b$. Iterating on the values of y_1 is done by iterating on the values of y and applying the affine transformation. In addition, $\Pr[y = i]$=$\Pr[y_1 = ai + b]$. In the rest of this paper, notations such as $-y$ represent a view on random variable y.

3.6 Probability Valuation of Constraints

In constraint programming, the reification of a constraint is very useful while modeling a problem. The reification variable is usually a Boolean variable denoting the satisfiability of the related constraints. Other valuations of constraints have been proposed, such as soft constraints [52, 25] and cost-version of constraints [44]. In this paper, we introduce the probability valuation of a constraint. It is a generalization of the chance from [18] for the case of variable distributions as the result is a Boolean random variable.

▶ **Definition 12.** *Let C be a relational constraint defined on the random variables Y. Let p_C be a Boolean random variable. The probability valuation $p_C \equiv C(Y)$ is defined by:*

$$Pr[p_C = 1] = \int_{t \in D_\Omega^\times(Y)} Pr[Y = t]\mathbb{T}_C(t) \tag{8}$$

Note that in the general case processing the probability of satisfaction of a constraint C is hard because the table T_C of the constraint may be untractable.

▶ **Proposition 13.** *The probabilization of a constraint is a generalization of the reification.*

Proof. Let C be a relational constraint defined on the random variables Y. Let $p_C \in [0, 1]$ be a Boolean random variable such that $Pr(p_C = 1) = \Pr[C(Y)]$. The two possible values for the reification of a constraint are satisfied and unsatisfied. If $Pr(p_C = 1) = 1$ the constraint is always satisfied, if $Pr(p_C = 1) = 0$, the constraint is always unsatisfied. For all the other values, p_C represents the probability of satisfaction of the constraint. ◀

Note that the probabilization can be seen as a random variable representing the projection of the distribution of the random variables on the constraint. This implies that p_C and the random variables in C are dependents. In the rest of this paper, p_C will be used to denote $\Pr[p_C = 1]$ when no ambiguity is present.

3.7 Multiple Stochastic Relational Constraints

In chance-constrained optimization, applications can be made of several independent parts [54], and the joint probability must be satisfied. Those are problems for which constraints of the problem can be split into independent ones. Let $\mathbf{1}$ be the vector of ones. More formally, these problems can be reformulated as:

$$\begin{aligned}
\arg\min_{Y \in \mathbb{D}} \quad & F(Y) && (9)\\
\text{subject to} \quad & p_1 \equiv C_1(Y_1) && (10)\\
& \dots && (11)\\
& p_n \equiv C_n(Y_n) && (12)\\
& \Pr[(p_1, \dots, p_n) = \mathbf{1}] \geq \gamma && (13)
\end{aligned}$$

Where $Y_i \bigcap Y_j = \emptyset, \forall i, j$, and $\bigcap$ being the set intersection operator. Using the probability valuation of constraints (10-12), the global chance or confidence constraint is (13). This implies that the hard part of the chance constraining will be located into the probability valuation of the constraints.

4 Implementation

4.1 Distribution Implementation

In the rest of this paper, a possible implementation for the discrete random variables is proposed. Then, propagation algorithms are proposed using the proposed implementation. Implementing a variable distribution *random variable* inside a solver might be challenging, as probability distributions can be expressed in very diverse ways [26]. Indeed, as for set or sequence variables [11], the choice of implementation will impact the design of propagation algorithms, and the capability of representation. For known probabilistic distributions, a straightforward implementation would be the parameters that describe it. For example, it would be sufficient to have two continuous variables μ and σ for a normal distribution. Indeed, with such a definition, the propagation could use the closed form of the cumulative distribution function and probability density functions. In this paper, for the implementation of the distribution, we assume that random variables are independent and discrete. We propose to represent distribution variables using probability mass function variables PMF.

▶ **Definition 14.** *Let y be a discrete independent random variable with sample space $\Omega_y = \{v_1, \dots, v_d\}$. The* PMF *variable of y is represented as a vector $d^y = (d^y_v)_{v \in \Omega_y}$ of continuous variables where $D(d^y_v) \subseteq [0, 1]$ represents the probability $Pr[y = v]$ and $\sum_i d^y_i = 1$.*

Note that using continuous or floating point variables inside of CP solver is a known topic [5, 43]. They are usually represented by intervals. For evident reasons, integer variables are considered encoded as usual.

▶ **Example 15.** Consider a Constraint Satisfaction Problem (CSP) with one decision variable $x \in \{0, 1\}$ and one random variable $y \in [1, 6]$. The CSP contains two constraints: one distribution constraint and one relational constraint. The first one defines the possible distributions of y and is defined by the table from the example in section 3.3. The second constraint is the constraint $\Pr[y > 2] \geq \gamma$ with $\gamma = 0.6$. A solution of such a CSP is an assignment of both x and the d_i^y variables that satisfies both constraints.

Consider the propagation of the $y > 2$ constraint. For each of the variables d_3^y to d_6^y, the lower bound is set to $\gamma - 3\frac{1}{6} = 0.1$. Indeed, for each of the possible values, the cumulative probability must be at least equal to γ. Once the propagation of constraint $y > 2$ is done, the distribution constraint can filter out value 1 from x as the associated distribution is no longer valid ($\underline{d_4^y} = 0.1 > \frac{1}{12}$). Values of d^y are assigned to the tuple $(\frac{1}{6}, ..., \frac{1}{6})$, and a solution is found.

▶ **Example 16** (Probability valuation)**.** Consider again the problem of the previous example. Once y is assigned, the probability valuation of constraint $y > 2$ is $p_{y>2} = \sum_{i=3}^{6} d_i^y = \frac{4}{6}$.

4.2 Filtering Algorithms for Relational Constraints on Random variables

It is stated in [28] that *the major challenge towards solving chance constrained optimization problems lies in the computation of the probability and its derivatives of satisfying inequality constraints*. In this section, probability filtering is provided for several constraints. Assumptions made by these filtering algorithms are that all random variables are independent and implemented using the distribution variables proposed in this paper. Given $p_C \equiv C(X, Y)$ the probability valuation of a constraint, the filtering algorithm should filter inconsistent values from X, from the distributions of Y and p_C.

Let C be a relational constraint defined on the random variables Y. We propose here a few filtering rules deriving from equations (4) and (8).

▶ **Example 17** (Unary Equal)**.** Let $p_{y=c} \equiv (y = c)$ be the probability valuation of the *equal* constraint. In such settings, $\Pr[p_{y=c} = 1] = d_c^y$. In addition for all the other values, another filtering can be defined by $\forall v \neq c, d_v^y \leq 1 - \Pr[p_{y=c} = 1]$. Note that this filtering should be directly done by the domain definition of the probability distribution.

▶ **Example 18** (Unary Greater Than)**.** Let $p_{y>c} \equiv (y > c)$ be the probability valuation of the unary *greater-than* constraint. We propose to decompose equation (8) into two equations on the bounds of the random variable: $p_{y>c} \geq \sum_{v_i=c+1}^{\overline{\Omega(y)}} d_{v_i}^y$; $\quad p_{y>c} \leq \sum_{v_i=c+1}^{\overline{\Omega(y)}} d_{v_i}^y$. This rewriting is usual in CP, and linear propagation can be applied. Moreover, analogously to the reification, filtering can also be defined for values lower than, or equal to c. Indeed, as $p_{y>c}$ is the probability of satisfaction of the constraint, $p_{y\leq c} = 1 - p_{y>c}$ is the probability of violating the constraint. This implies that the two filtering rules above should be used to propagate d_v^y, $\forall v \leq c$.

▶ **Example 19** (Binary Equal)**.** Let $p_{y_1=y_2} = \Pr[y_1 = y_2]$ be the probability valuation of the *equal* constraint between two random variables. A filtering can be extracted from equation: $p_{y_1=y_2} = \sum_i d_i^{y_1} d_i^{y_2}$.

▶ **Example 20** (Constraints $y \simeq y_1$ **op** y_2)**.** $y \simeq y_1 + y_2$ is the definition of the distribution of variable y from the distributions of variables y_1 and y_2. It is a probability-based constraint. In the general case, the probability distribution of the sum of two independent variables is given by the convolution: $\Pr[y = k] = \sum_{i=-\infty}^{\infty} \Pr[y_1 = k - i]\Pr[y_2 = i]$. This convolution can be used to extract propagation rules for the ternary constraint. For each value v_k of the possible values of y, a constraint of the form of $d_{v_k}^y = \sum_i d_{v_{k-i}}^{y_1} d_{v_i}^{y_2}$ is posted. Note that the same equation can be defined for the multiplication, subtraction, division, and modulo constraints.

▶ **Example 21** (Constraints $y = y_1$ **op** y_2)**.** For example, $y = y_1 + y_2$ is the test of equality between the sampled values of y and the sum of the sampled values of y_1 and y_2. This constraint can be expressed using the two constraints above. First, let $y_{1+2} \equiv y_1 + y_2$ be the distribution of the addition of y_1 and y_2. Second, let $p_{y=y_1+y_2}$ be the probability valuation of $y = y_{1+2}$. Finally, $p_{y=y_1+y_2} = \Pr[y = y_{1+2}]$.

4.3 Filtering Relational Constraints on Integer and Random Variables

Consider a relational constraint C on decision variables X and on random variables Y.

▶ **Proposition 22.** *Set the variable $d^{y_i}_{v_i}$ from $y_i \in Y$ for constraint C to 0 if:*

$$\forall t_X \in D^\times(X), \exists t_Y \in D^\times_\Omega(Y), t_{Y_i} = v_i \wedge \prod_{\forall v_j \in t_{Y_{\setminus\{v_i\}}}} d^{y_j}_{v_j} > \mathbb{T}_C(t_X, t_Y) \tag{14}$$

This proposition is a direct rewriting of equation equations (4) and (8). In addition to the classical filtering of values in the decision variables if they do not have a valid tuple, another filtering should be defined.

▶ **Proposition 23.** *Filter value v_i from the domain of $x_i \in X$ for constraint C if:*

$$\exists t_Y \in D^\times_\Omega(Y), \forall t_X = (\dots, v_i, \dots) \in D^\times(X), \prod_{\forall v_j \in t_Y} d^{y_j}_{v_j} > \mathbb{T}_C(t_X, t_Y) \tag{15}$$

Let $p_{C(X,Y)} \equiv C(X, Y)$ be the probability valuation of the constraint. A direct simple filtering can be extracted from equation (8). In addition, the filtering should be propagated back to the decision variables, for example:

▶ **Proposition 24.** *Filter value v_i from the domain of $x_i \in X$ for constraint C if:*

$$\forall t_x = (\dots, v_i, \dots) \in D^\times(X), \Pr[C(t_x, Y)] \neq p_C \tag{16}$$

Here, equation (8) is applied to each tuple containing the value v_i. If none of them satisfies the probability valuation, then the value can be safely removed. Recall that p_C represents $\Pr[p_C = 1]$. Moreover, in general, the $\neq$ will be checked for $>$ and $<$ to filter the bounds of the variables. The exact rule depends on the expression of the satisfaction of the constraint in terms of distribution variables, as shown in the following examples.

▶ **Example 25** (Lower Than)**.** Let $p_{x<y} \equiv (x < y)$ be the probability valuation of the *lower-than* constraint. This constraint is one of the most used in chance optimization as it can be used to model the deviation of scheduled amounts [8]. Analogously to the equal constraint, the pruning of variable x can be defined by the following rule: $x < \max\{v_j \in x \mid \sum_{v_i=v_j}^{\overline{\Omega(y)}} \overline{D(d^y_{v_i})} > \underline{D(p_{x<y})}\}$ First, the opposite rule can also be defined by using $\overline{D(p_{x<y})}$. Then, note that this pruning rule, when both the d^y_v and $p_{x<y}$ are fixed, is equivalent to the pruning of [30]. In addition, as the authors proposed, a closed formula may be used to directly extract the bounds of x. Now that x is propagated, the probability distribution and valuation should be filtered. This can be done by using the following sum: $p_{x<y} = \sum_{v_i=x+1}^{\overline{\Omega(y)}} d^y_{v_i}$.

▶ **Example 26** (Equal)**.** Let $p_{x=y} \equiv (x = y)$ be the probability valuation of the *equal* constraint. First the pruning of variable x can be defined using the following rule:

$$\forall v \in \Omega(y), \overline{D(d^y_v)} < \underline{D(p_{x=y})} \vee \underline{D(d^y_v)} < \overline{D(p_{x=y})} \implies x \neq v \tag{17}$$

Then, the pruning of the probability valuation $p_{x=y}$ can be defined using:

$$p_{x=y} \geq \min(\{\underline{D(d_v^y)}|\forall v \in x\}); p_{x=y} \leq \max(\{\overline{D(d_v^y)}|\forall v \in x\}) \tag{18}$$

Finally, the pruning of the distribution of y is an adaptation of the usual propagation:

$$x = v \implies d_v^y = p_{x=y} \tag{19}$$

4.4 Rewriting of constraints

In the previous sections, several constraints and propagation algorithms have been proposed. In this section, it is shown how to combine these constraints to model most use cases. Consider the not equal constraint. Even though no dedicated algorithm has been defined, propagation can be based on other ones. Let $p_{x \neq y} \equiv (x \neq y)$ be the probability valuation of the not-equal constraint, $p_{x \neq y} = 1 - p_{x=y}$. This is analogous to the reification, indeed, $(x_1 = x_2) = \neg(x_1 \neq x_2)$. Several more constraints can be reformulated, for example the greater-less-than relationship: $p_{x \leq y} = p_{x-1<y}$; $p_{x>y} = p_{-x<-y}$ (alternatively, $p_{x \geq y} = p_{-x-1<-y}$). The in-not in relationship: $p_{y \notin X} = 1 - p_{y \in X}$. And the many logical relationships such as $p_{c_1 \wedge c_2} = p_{c_1} p_{c_2}$; $p_{c_1 \vee c_2} = 1 - (1 - p_{c_1})(1 - p_{c_2})$; and $p_{c_1 \oplus c_2} = p_{c_1}(1 - p_{c_2}) + (1 - p_{c_1})p_{c_2}$. With $\oplus$ the xor logical operator. Note that some of these reformulations are direct encodings of the probability rules, such as for the *and* and *or* constraints.

5 Application: Virtual Network Design

In virtual network functions design, the main performance criterion is the total latency of the virtual network [12, 13, 16, 48]. In such a graph, each node is either a machine or a virtual machine, and has to process blocks of data. The processing time of a data block by a node is following a random distribution, which is conditioned by its configuration (both hardware and software). The global latency is the sum of the processing times of the nodes of the network. The goal of the virtual network design problem is to set the configurations for all the nodes such that the global latency is below L milliseconds with probability γ, while the cost is minimized and other configuration constraints are satisfied.

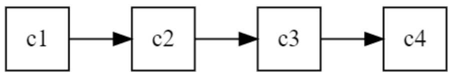

Figure 1 Example of virtual network graph.

Consider the network from Figure 1. This network contains 4 nodes $(c_1, ..., c_4)$. In this graph, when a data block needs to be sent, it will go through c_1, then c_2, then c_3 and finally c_4. Note that in practice, the depth of the graph is often shallow. In our industrial application the number of nodes was 3. Moreover, each node contains several settings (*cpu redundancy, queue maximum read times, queue batch pkt num, dpe*, etc.) The latency of a node is influenced by all these settings. In addition, each node serves a particular role, and some operations must be done in at least one node. All of these requirements are contained in the constraints on the settings variables. The influence of the setting variables on the random variables' distribution is encoded as a table T_s for each node.

5.1 Model

Let $N = (n_1, ..., n_k)$ be the nodes to configure. Let $y_i \sim D^{y_i}$ be the random variable representing the latency of node n_i. Let $S = (S^1, ...S^k)$ with $S^i = s^i_1, ..., s^i_m$ being the settings variables associated to each node. Let L be the maximum latency and γ the minimal probability. Let $acc_i \in (acc_2, ..., acc_k)$ be the accumulated sum of the i first random variables. Let c_i be the cost associated with node n_i.

$$\begin{aligned}
\arg \min_{S\in\mathbb{Z}, Y\in\mathbb{P}} \quad & \textstyle\sum_{i=1}^{k} c_i && (20)\\
\text{such that} \quad & \text{Table}(S^i, c_i, y_i, T_s), \quad \forall i \in [1, k] && (21)\\
& acc_1 \simeq y_1 && (22)\\
& acc_i \simeq acc_{i-1} + y_i, \quad \forall i \in [2, k] && (23)\\
& \Pr[acc_k < L] \geq \gamma && (24)\\
& \text{ValidSettings}(S) && (25)
\end{aligned}$$

5.2 Data

A large number of configurations have been heavily sampled to extract the distribution of probability. Once those were built, they have been used to extrapolate to unknown configurations. The latency is given in micro-seconds, with values in the set $(50, 100, ..., 1450, 1500)$. Each node may be defined by around 100 configurations (valid assignment of the settings). The number of nodes varies between 2 to 5. Two algorithms are compared. First, *Variable* is the work proposed by this paper, where distribution variables are used. The second one is *Fixed*, it is an adaptation of the algorithm from [30, 36]. In *Fixed*, the distributions are considered unknown until all the setting variables used to define them are instantiated. It can be seen as a form of generate and test. The search strategy is the same for both algorithms. It starts by assigning the possible settings. Note that this is the only fair comparison with *Fixed* as the setting variables are the one influencing the possible distributions. In the general case, better strategies could be defined for *Variable*. Time out is set to 30 minutes. The integer CP solver used for all the experiments of this paper is our internal solver. The continuous part is solved by call to IBEX [4], analogously to CHOCO-solver [43].

5.3 Results

First, consider the results from Table 1. As we can see, finding a solution for the *fixed* model is not too hard, but as the size increases, it is not able to find the optimal one, or to prove it. In contrast, for the *variable* model, the optimal solution is always found and proved. The main reason is that invalid distributions are removed by the latency constraint before any decision leading to them is taken.

Table 1 For each instance set, the values are #SAT (**#OPT**). The last row is the mean time in seconds for proving optimality.

# Nodes	**2**	**3**	**4**	**5**
Fixed (baseline)	30 (**30**)	30 (22)	30 (0)	30 (0)
Variable	30 (**30**)	30 (**30**)	30 (**30**)	30 (**30**)
Variable avg time	0.57 s	10.2 s	125.4 s	752.3 s

Moreover, consider the plots from Figure 2. Instances are sorted by size. On the middle, the comparison on the objective value is made. As we can see, the objective value of the *fixed* method is optimal for small instances, even if no proof can be reached. Then, for larger instances, it is often quite far from the optimal (objective $\in [0, 3]$). In addition, on the left and right plots, it is seen that most of the time, the *fixed* method reaches the timeout. Note that the time increases exponentially with the number of nodes for both methods, but not the number of decisions for the *variable* method.

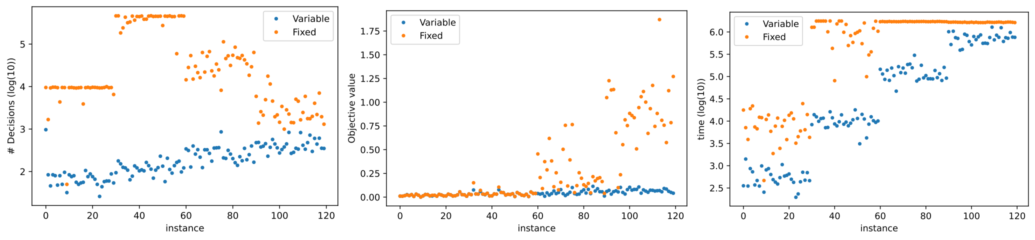

Figure 2 Instances are ordered w.r.t. size. (left) Number of decision (log base 10). (middle) Best objective value comparison. (right) Running time (log base 10).

In conclusion, this experiment shows that the *variable* method proposed in this paper is better both in terms of quality of solution and in terms of time. In practice, solving the industrial instances of size 3 takes 17.75 seconds on average for the optimal solution.

6 Application: Chemical Deliveries

The following problem is part of a pipeline of chemicals processing and is composed of two assignment problems. First, chemicals are received by the factory every day. Once received, they must be stored into containers. These containers are restricted to some type of products. Each container already contains a known amount of products and has a maximum storage capacity. In addition, for some chemicals the total amount delivered is larger than the remaining storage capacity of the containers, given their currently stored quantities. In practice, this is not an issue as the chemicals are also processed, hence emptying the containers. The second problem concerns product assignment. Indeed, the chemicals are used to manufacture bio-sourced bases for perfumes. Those are later used by home-perfume makers to create reeds, candles, etc. A dozen of teams are spread over 4 buildings. Each building/team is specialized in a set of types of product, yet there are overlaps. Every day, in addition to the incoming deliveries, the factory must produce a given amount of several products. Selecting which team should work on which product is part of the problem to solve. The probability of emptying a container depends on the products associated to the teams in the building. A solution is an assignment of products to teams and deliveries to containers such that the stored quantity in each container minus the quantity that will be used does not exceed the maximum capacity with a high confidence.

More formally, this is an assignment problem. Each product $p \in P$ must be produced by a team $t \in T$. Each team $t \in T$ has a work capacity W_t, and work in a building $b_t \in B$. Each product w requires an given amount of work w_p, and is compatible to a subset of the teams $T_p \in T$. Each delivery $j \in D$ stores a quantity q_j in a container $c_i \in C$. Each container c_i has a maximum capacity C_i. Each container c_i will be emptied in parallel of a quantity y_i unknown in advance. The exact quantity y_i is unknown, but its distribution is influenced by the products processed in its building.

6.1 Data

The dataset contains 40 instances, with $|P| \in [15, 25]$, $|C| \in [4, 10]$, $T = 12$,$B = 4$,$|D| \in [20, 50]$. Maximum running time is set to 30 minutes. The dataset is based on past data for the generation of probabilistic distributions. Figure 3 (left) shows an example of a set of distribution for one container. As we can see, the more the building has to make products, the higher the chance to consume.

6.2 Model

For each product p, variables $p_{p,t} \forall t \in T$ indicates if the product is made by the team t. For each delivery d, variable $d_{d,c} \forall c \in C$ indicates if the delivery is stored in container c. For each container c, y_c is the emptying random variable. For each building b, q_b is the quantity of product produced in the building. Let $q = (q_1, \ldots, q_{|B|})$ be a vector of all q_b variables. Let $x_c \forall c \in C$ be the additional capacity required by container c such that it is no longer overflowing.

$$\arg \max_{x \in \mathbb{Z}, Y \in \mathbb{P}} \quad \gamma \tag{26}$$

$$\text{Such that} \quad \textstyle\sum_{t=1}^{|T|} p_{i,t} = 1, \qquad \forall i \in P \tag{27}$$

$$\textstyle\sum_{i=1}^{|P|} p_{i,t} w_i \leq W_t, \qquad \forall t \in T \tag{28}$$

$$q_b = \textstyle\sum_{i=1}^{|P|} \sum_{t \in \{t_i | b_{t_i} = b\}} p_{i,t}, \qquad \forall b \in B \tag{29}$$

$$\textstyle\sum_{i=1}^{|D|} d_{i,c} q_i \leq C_c + x_c, \qquad \forall c \in C \tag{30}$$

$$\textstyle\sum_{c=1}^{|C|} d_{i,c} = 1, \qquad \forall i \in D \tag{31}$$

$$\text{Table}(q, y_c), \qquad \forall c \in C \tag{32}$$

$$p_{x_c \leq y_c} \equiv x_c \leq y_c, \qquad \forall c \in C \tag{33}$$

$$\Pr[(p_{x_1 \leq y_1}, \ldots, p_{x_{|C|} \leq y_{|C|}}) = \mathbf{1}] \geq \gamma \tag{34}$$

6.3 Results

Figure 3 shows the results of the *fixed* and *variable* (this paper) methods. First, it is interesting to see that for some instances a confidence of 1 can be found. For these instances, no consumption was required to find a solution. In contrast, we can also see that for some instances, is is hard to find solutions with more than 2 percent confidence. Those instances are hard instances, where the delivered amount is too large. Then in the middle of the plot, it is shown that the confidence of solutions found by the *variable* method is higher in general than the *fixed* method. In addition, the *variable* method makes one order of magnitude less decisions to find better or equivalent solutions. When profiled, the reason why less decisions are made is the repetitive call to the continuous integrated solver. An actual hybrid CP solver would be drastically more efficient. Nevertheless, even with this drawback, in both experiments, the *variable* showed significant improvements in term both of quality of solution and time.

7 Conclusion and Future works

This paper proposed to extend the CP framework to distribution optimization. First, random variables have been extended to the case of distribution variables, then the CP constraints have been extended to deal with this extension. We proposed a generalization of integer

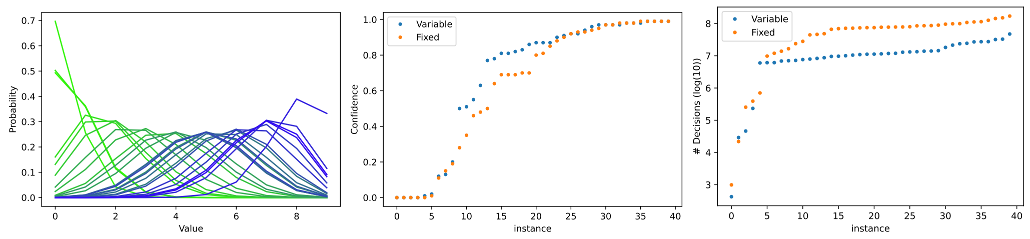

Figure 3 Instances are ordered w.r.t. size. (left) Range of probability distributions for a random variable having 9 possible values. The gradient of color indicate the increase of q. (middle) Best confidence comparison. (right) Number of decisions (log base 10).

variables and new consistencies for probability distributions. In addition, generic filtering algorithms have been proposed, pruning invalid distributions. We defined an implementation based on a probability mass function decomposition and the minimal arithmetic to model most problems, together with the associated filtering algorithms. Finally, as shown in the experimental section, we used the proposed framework to solve two optimization problems where the distributions of probability are not fixed at the beginning of the problem.

The main future direction is to no longer restrict the search to a finite set of distributions using a table as done in our experiments, but to be able to search directly into the distribution space, which is closer to known methods in machine learning. Doing this would bring CP as one of the main frameworks to do constrained distribution learning, and will require hybridization of CP solvers. Other future directions include the design of specialized global constraints filtering, the generalization of the reuse of existing filtering algorithms [18], and the implementation of different types of encoding of the probability distribution variables. Another important direction is the encoding of dependent random variables. In this paper, we purposely introduced the event space of probability space to encode the dependency between random variables. In future works, constraints on the event spaces will lead to efficient dependency implementation. Finally, this paper is restricted to 2-stage policy, it will be important in the future to extend it to multi-stages.

References

1 Eric Baum and Frank Wilczek. Supervised learning of probability distributions by neural networks. In *Neural information processing systems*, 1987.

2 N Bharanidharan and Harikumar Rajaguru. Improved chicken swarm optimization to classify dementia mri images using a novel controlled randomness optimization algorithm. *International Journal of Imaging Systems and Technology*, 30(3):605–620, 2020.

3 Giuseppe Calafiore and Marco C Campi. Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, 102(1):25–46, 2005.

4 Gilles Chabert et al. Ibex, an interval-based explorer, 2007.

5 Gilles Chabert and Luc Jaulin. Contractor programming. *Artificial Intelligence*, 173(11):1079–1100, 2009.

6 Abraham Charnes and William W Cooper. Chance-constrained programming. *Management science*, 6(1):73–79, 1959.

7 Abraham Charnes and William W Cooper. Deterministic equivalents for optimizing and satisficing under chance constraints. *Operations research*, 11(1):18–39, 1963.

8 Abraham Charnes, William Wager Cooper, and MJL Kirby. Chance-constrained programming: an extension of statistical method. In *Optimizing methods in statistics*, pages 391–402. Elsevier, 1971.

9 Constantinos Daskalakis, Ilias Diakonikolas, and Rocco A Servedio. Learning poisson binomial distributions. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 709–728, 2012.

10 Constantinos Daskalakis and Gautam Kamath. Faster and sample near-optimal algorithms for proper learning mixtures of gaussians. In *Conference on Learning Theory*, pages 1183–1213. PMLR, 2014.

11 Augustin Delecluse, Pierre Schaus, and Pascal Van Hentenryck. Sequence variables for routing problems. In *28th International Conference on Principles and Practice of Constraint Programming (CP 2022)*, 2022.

12 Mihai Dobrescu, Katerina Argyraki, and Sylvia Ratnasamy. Toward predictable performance in software {Packet-Processing} platforms. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 141–154, 2012.

13 Paul Emmerich, Daniel Raumer, Florian Wohlfart, and Georg Carle. Assessing soft-and hardware bottlenecks in pc-based packet forwarding systems. *ICN 2015*, 90, 2015.

14 Hélène Fargier and Jérôme Lang. Uncertainty in constraint satisfaction problems: a probabilistic approach. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty: European Conference ECSQARU'93 Granada, Spain, November 8–10, 1993 Proceedings 2*, pages 97–104. Springer, 1993.

15 Jakob Feldtkeller, David Knichel, Pascal Sasdrich, Amir Moradi, and Tim Güneysu. Randomness optimization for gadget compositions in higher-order masking. *Cryptology ePrint Archive*, 2022.

16 Juliver Gil Herrera and Juan Felipe Botero. Resource allocation in nfv: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532, 2016.

17 Brahim Hnich, Roberto Rossi, S Armagan Tarim, and Steven Prestwich. Synthesizing filtering algorithms for global chance-constraints. In *Principles and Practice of Constraint Programming-CP 2009: 15th International Conference, CP 2009 Lisbon, Portugal, September 20-24, 2009 Proceedings 15*, pages 439–453. Springer, 2009.

18 Brahim Hnich, Roberto Rossi, S Armagan Tarim, and Steven Prestwich. Filtering algorithms for global chance constraints. *Artificial Intelligence*, 189:69–94, 2012.

19 JN Hooker. Stochastic decision diagrams. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 138–154. Springer, 2022.

20 Mark H Houck. A chance constrained optimization model for reservoir design and operation. *Water Resources Research*, 15(5):1011–1016, 1979.

21 Bahareh Kargar, Mir Saman Pishvaee, Hamed Jahani, and Jiuh-Biing Sheu. Organ transportation and allocation problem under medical uncertainty: A real case study of liver transplantation. *Transportation Research Part E: Logistics and Transportation Review*, 134:101841, 2020.

22 Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E Schapire, and Linda Sellie. On the learnability of discrete distributions. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 273–282, 1994.

23 Lin Kemeng, Wang Xiaoyan, Xia Weijie, Zhang Jiaming, et al. Optimization of the randomness in einstein which based on monte carlo algorithms. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 6305–6309. IEEE, 2019.

24 Stefan Kern, Sibylle D Müller, Nikolaus Hansen, Dirk Büche, Jiri Ocenasek, and Petros Koumoutsakos. Learning probability distributions in continuous evolutionary algorithms–a comparative review. *Natural Computing*, 3:77–112, 2004.

25 Minh Thanh Khong, Christophe Lecoutre, Pierre Schaus, and Yves Deville. Soft-regular with a prefix-size violation measure. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 333–343. Springer, 2018.

26 Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

27 Anna LD Latour, Behrouz Babaki, Daniël Fokkinga, Marie Anastacio, Holger H Hoos, and Siegfried Nijssen. Exact stochastic constraint optimisation with applications in network analysis. *Artificial Intelligence*, 304:103650, 2022.

28 Pu Li, Harvey Arellano-Garcia, and Günter Wozny. Chance constrained programming approach to process optimization under uncertainty. *Computers & chemical engineering*, 32(1-2):25–45, 2008.

29 Xiaoxia Lin, Stacy L Janak, and Christodoulos A Floudas. A new robust optimization approach for scheduling under uncertainty:: I. bounded uncertainty. *Computers & chemical engineering*, 28(6-7):1069–1085, 2004.

30 Alexandre Mercier-Aubin, Ludwig Dumetz, Jonathan Gaudreault, and Claude-Guy Quimper. The confidence constraint: A step towards stochastic cp solvers. In *International Conference on Principles and Practice of Constraint Programming*, pages 759–773. Springer, 2020.

31 Arkadi Nemirovski and Alexander Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2007.

32 François Pachet, Pierre Roy, Alexandre Papadopoulos, and Jason Sakellariou. Generating 1/f noise sequences as constraint satisfaction: The voss constraint. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

33 Bernardo K Pagnoncelli, Shabbir Ahmed, and Alexander Shapiro. Sample average approximation method for chance constrained programming: theory and applications. *Journal of optimization theory and applications*, 142(2):399–416, 2009.

34 M Arenas Parra, A Bilbao Terol, B Pérez Gladish, and MV Rodrıguez Urıa. Solving a multiobjective possibilistic problem through compromise programming. *European Journal of Operational Research*, 164(3):748–759, 2005.

35 Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan kaufmann, 1988.

36 Guillaume Perez, Steve Malalel, Gael Glorian, Victor Jung, Alexandre Papadopoulos, Marie Pelleau, Wijnand Suijlen, Jean-Charles Régin, and Arnaud Lallouet. Generalized confidence constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

37 Guillaume Perez, Brendan Rappazzo, and Carla Gomes. Extending the capacity of 1/f noise generation. In *International Conference on Principles and Practice of Constraint Programming*, pages 601–610. Springer, 2018.

38 Guillaume Perez and Jean-Charles Régin. MDDs are efficient modeling tools: An application to dispersion constraints. In *Integration of AI and OR Techniques in Constraint Programming*, 2017.

39 Guillaume Perez and Jean-Charles Régin. Mdds: Sampling and probability constraints. In *International Conference on Principles and Practice of Constraint Programming*, pages 226–242. Springer, 2017.

40 Gilles Pesant. Achieving domain consistency and counting solutions for dispersion constraints. *INFORMS Journal on Computing*, 27(4):690–703, 2015. doi:10.1287/ijoc.2015.0654.

41 Warren B Powell. A unified framework for stochastic optimization. *European Journal of Operational Research*, 275(3):795–821, 2019.

42 Steven D Prestwich, Roberto Rossi, and S Armagan Tarim. Randomness as a constraint. In *Principles and Practice of Constraint Programming: 21st International Conference, CP 2015, Cork, Ireland, August 31–September 4, 2015, Proceedings 21*, pages 351–366. Springer, 2015.

43 Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca. Choco solver documentation. *TASC, INRIA Rennes, LINA CNRS UMR*, 6241:13–42, 2016.

44 Jean-Charles Régin. Arc consistency for global cardinality constraints with costs. In *International Conference on Principles and Practice of Constraint Programming*, pages 390–404. Springer, 1999.

45 Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming.* Elsevier, 2006.

46 Roberto Rossi, Brahim Hnich, S Armagan Tarim, and Steven Prestwich. Confidence-based reasoning in stochastic constraint programming. *Artificial Intelligence*, 228:129–152, 2015.

47 Christian Schulte and Guido Tack. View-based propagator derivation. *Constraints*, 18(1):75–107, 2013.

48 Kalika Suksomboon, Nobutaka Matsumoto, Shuichi Okamoto, Michiaki Hayashi, and Yusheng Ji. Configuring a software router by the erlang-k-based packet latency prediction. *IEEE Journal on Selected Areas in Communications*, 36(3):422–437, 2018.

49 S Armagan Tarim, Suresh Manandhar, and Toby Walsh. Stochastic constraint programming: A scenario-based approach. *Constraints*, 11:53–80, 2006.

50 Joseph A Tatman and Ross D Shachter. Dynamic programming and influence diagrams. *IEEE transactions on systems, man, and cybernetics*, 20(2):365–379, 1990.

51 Pascal Van Hentenryck and Laurent Michel. Domain views for constraint programming. In *International Conference on Principles and Practice of Constraint Programming*, pages 705–720. Springer, 2014.

52 Willem-Jan Van Hoeve, Gilles Pesant, and Louis-Martin Rousseau. On global warming: Flow-based soft global constraints. *Journal of Heuristics*, 12(4):347–373, 2006.

53 Toby Walsh. Stochastic constraint programming. In *ECAI*, volume 2, pages 111–115, 2002.

54 Weijun Xie and Shabbir Ahmed. On deterministic reformulations of distributionally robust joint chance constrained optimization problems. *SIAM Journal on Optimization*, 28(2):1151–1182, 2018.

55 Hui Zhang and Pu Li. Chance constrained programming for optimal power flow under uncertainty. *IEEE Transactions on Power Systems*, 26(4):2417–2424, 2011.