Generalized Confidence Constraints

Guillaume Perez[1], Steve Malalel[2], Gael Glorian[1], Victor Jung[2], Alexandre Papadopoulos[1], Marie Pelleau[2], Wijnand Suijlen[1], Jean-Charles Régin[2], Arnaud Lallouet[1]

[1] Huawei Technologie, Boulogne-Billancourt, France
[2] Université Côte d'Azur, CNRS, Sophia Antipolis, France
guillaume.perez06@gmail.com,

Abstract

In robust optimization, finding a solution that solely respects the constraints is not enough. Usually, the uncertainty and unknown parameters of the model are represented by random variables. In such conditions, a good solution is a solution robust to most-likely assignments of these random variables. Recently, the *Confidence* constraint has been introduced by Mercier-Aubin et al. in order to enforce this type of robustness in constraint programming. Unfortunately, it is restricted to a conjunction of binary inequalities

In this paper, we generalize the *Confidence* constraint to any constraint and propose an implementation based on Multi-valued Decision Diagrams (MDDs). The *Confidence* constraint is defined over a vector of random variables. For a given constraint C, and given a threshold, the *Confidence* constraint ensures that the probability for C to be satisfied by a sample of the random variables is greater than the threshold. We propose to use MDDs to represent the constraints on the random variables. MDDs are an efficient tool for representing combinatorial constraints, thanks to their exponential compression power. Here, both random and decision variables are stored in the MDD, and propagation rules are proposed for removing values of decision variables that cannot lead to robust solutions. Furthermore, for several constraints, we show that decision variables can be omitted from the MDD because lighter filtering algorithms are sufficient. This leads to gain an exponential factor in the MDD size. The experimental results obtained on a chemical deliveries problem in factories – where the chemicals consumption are uncertain – shows the efficiency of the proposed approach.

Introduction

Stochastic optimization is a class of problem where uncertainty is present (Powell 2019; Charnes and Cooper 1959). Such problems are present in different domains, and are already part of constraint programming modeling languages (Rendl, Tack, and Stuckey 2014). Optimization and satisfaction problems involving random variables and searching for robust solutions are ubiquitous in modern continuous flux applications. Vehicle sharing, organ transplant, textile manufactures, and deliveries are just a few of the many industrial areas where robust optimization is necessary (Lin, Janak, and Floudas 2004; Parra et al. 2005; Kargar et al. 2020; Mercier-Aubin et al. 2020). In such setting, the chance constrained problem is usually defined as follow:

$$\underset{X\in\mathbb{N}^r}{\text{minimize}} \quad F(X) \quad \text{subject to} \quad \Pr[f(X,Y)] \geq \gamma \tag{1}$$

Where X are decision variables, Y are random variables, F is an objective function, and f represents the constraints' satisfiability. The main difference with problem definitions that do not involve random variables is that Pr returns the probability that the constraints will be satisfied, given the assignment of the decision variables.

In the context of constraint programming, optimization under probability, confidence or statistical constraints is becoming standard (Pachet et al. 2015; Pesant 2015; Perez and Régin 2017; Perez, Rappazzo, and Gomes 2018; Hooker 2022; Latour et al. 2022). At the beginning of this century stochastic constraint programming has been defined (Walsh 2002). Some problems involving *Confidence* constraints, coined "chance constraints" at that time (Rossi et al. 2008; Hnich et al. 2012), Markov or probability distribution constraints (Perez and Régin 2017a) have already been solved. Until recently the *Confidence* constraint lacked a generic filtering algorithm. A first attempt had been made in 2015 (Rossi et al. 2015). In 2020, some filtering algorithms (Mercier-Aubin et al. 2020) were finally presented but they can handle only conjunction of binary inequality constraints.

We propose to generalize the definition of the *Confidence* constraint to any constraint as in problem definition (1). Given X, Y, γ, and a constraint C, the *Confidence*(X,Y,C,γ) constraint ensures that the probability of $C(X,Y)$ to be satisfied by Y given X is greater than γ.

Consider any assignment p of the decision variables. Let T_p be the set of all valid assignments of random variables of $C(p,Y)$. The constraint ensures that p is a valid assignment of the decision variables if and only if $\sum_{t\in T_p} p(t) \geq \gamma$. In the general case, summing all the possible tuples is intractable, hence approximate sampling methods are often used to represent confidence (Calafiore and Campi 2005). We propose to use a multi-valued decision diagram (MDD) to represent the underlined constraints, and propose exact filtering algorithms for the *Confidence* constraint.

Multi-valued decision diagrams (MDDs) are a generic representation tool used to represent any constraint (Cheng and Yap 2010; Perez and Régin 2015; Bergman et al. 2016).

MDDs are graph data-structures that store all the satisfying tuples as paths. They have been used to represent table constraints (Lecoutre 2011; Demeulenaere et al. 2016), regular and automaton constraints (Pesant 2004), and even more complex constraints such as the Allen constraint (Roy et al. 2016). While an MDD can exponentially compress some constraints, its own size may grow exponentially. As an example of this duality, consider the MDD representing the AllDifferent constraint (Régin 1994). The MDD defined over n variables having n values each, will have 2^n nodes, but will contain $n!$ solutions ($2^{10} \approx 1k$, $10! \approx 3628k$). In the general case, many constraints are represented using polynomial-size MDDs. Such compression power and the many different algorithms defined on top of MDDs make them a useful data-structure for optimization (Bergman et al. 2016; Castro, Cire, and Beck 2022).

The representation of the constraint $C(X, Y)$ using an MDD allows a recursive confidence computation. Indeed, instead of extracting all the tuples stored in the MDD, local confidence values are computed per node, so that the confidence of the constraint can be computed in linear time, with respect to the MDD size. A linear propagation algorithm enforcing global arc consistency (GAC) is proposed for the case where the MDD contains both random and decision variables. This algorithm supposes that the ordering of the variables in the MDD is first the decision variables, and then the random variables. It can be used to solve any problem involving a *Confidence* constraint exactly.

In addition, for several constraints, decision variables can be omitted from the MDD because lighter filtering algorithms are sufficient. We propose three of such algorithms for the confidence of the conjunction of binary constraints. This new representation may gain an exponential factor in the MDD size, as shown in our examples.

Probability distributions used in constraint programming are often independent and directly represented by their probability mass functions (Mercier-Aubin et al. 2020). For many problems, this is not true, for example, in music generation, the succession of musical notes is often described by Markov chains (Pachet and Roy 2011). In this paper, working with MDDs allows to use the polynomial-size transformation of an MDD and a Markov chain to an MDD with layer-independent probabilities (Perez and Régin 2017a). This gives the *Confidence* constraint the capability to handle a broad new range of problems.

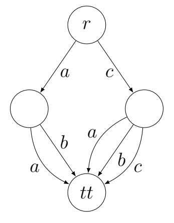
In the end, the experimental section shows the efficiency of the algorithm on the chemical deliveries assignment problem – an assignment problem where deliveries must be assigned to containers and where these containers are emptied in parallel. This experimental evaluation shows the necessity to have a global *Confidence* constraint, containing all the constraints that random variables should satisfy.

Preliminaries

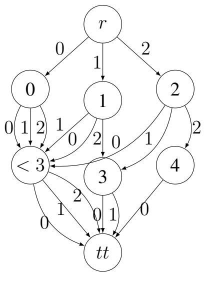
A random variable is a variable whose value depends on a random event A probability mass function (PMF) *pmf*: $V \to \mathbb{R}$ assigns a probability $Pr[v]$ to each discrete event $v \in V$. Let y be a discrete random variable with domain $D_y = \{v_1, \ldots, v_d\}$. Its probability mass function *pmf*$(y) = (p(y, v_1), \ldots, p(y, v_d))$ is defined per value as $p(y, v_i) = Pr[y = v_i]$ and satisfies the following condition: $p(y, v_i) \geq 0$ and $\sum_{i=1}^{d} p(y, v_i) = 1$.

Let $Y = (y_1, \ldots, y_r)$ be a vector of r discrete independent random variables. Let $F = (p_{f_1}, \ldots, p_{f_r})$ be the vector of their associated probability mass functions. The probability of a sample of all the variables (i.e. a tuple) is given by the product of the probabilities of the selected values. More precisely, the probability of a tuple $t = (a_1, \ldots, a_r)$ is equal to:

$$p(t) = \prod_{i=1}^{r} p_{f_i}(y_i, a_i) \tag{2}$$



(a)



(b)

Figure 1: (a) MDD containing the tuple set {(a,a), (a,b), (c,a), (c,b), (c,c)}. (b) MDD for the constraint $\sum v_i < 5$. Values in the nodes are the sum of incoming paths.

Multi-Valued Decision Diagrams

A Multi-valued Decision Diagram (MDD) is a graph data-structure representing discrete functions. It is a multi-valued extension of Binary Decision Diagrams (BDDs) (Bryant 1986). An MDD, as used in CP (Cheng and Yap 2010; Andersen et al. 2007; Hadzic et al. 2008; Hoda, van Hoeve, and Hooker 2010; Bergman, van Hoeve, and Hooker 2011; Gange, Stuckey, and Szymanek 2011; Perez and Régin 2014; Gillard et al. 2021), is a rooted directed acyclic graph (DAG) used to represent some function $f : \{0, \ldots, d-1\}^r \to \{true, false\}$, based on a given integer d. Given the r input variables, the DAG representation is designed to contain $r+1$ layers of nodes, such that each variable is represented as a specific layer of the graph. Each node on a given layer has at most d outgoing arcs to nodes in the next layer of the graph. Each arc is labeled by its corresponding integer. The arc (u, v, a) points from node u to node v and has a as label. All outgoing arcs of the layer r reach the true terminal node tt (the false terminal node is typically omitted). There is an equivalence between $f(a_1, \ldots, a_r) = true$ and the existence of a path from the root node to the true terminal node whose arcs are labeled $a_1, \ldots, a_r$. Let $L(M, i)$ denote the set of arcs of MDD M at layer i. Let $L^{\downarrow}(M, i)$ denotes the set of nodes starting layer i. When there is no ambiguity, the M is omitted. Figure 1 (a) gives an example of MDD containing the tuples (a,a), (a,b), (c,a), (c,b) and (c,c).

MDD of a constraint Let C be a constraint defined on a vector of variables $X(C) = (x_1, \ldots, x_r)$. The MDD associated with C, denoted by MDD(C), is an MDD containing only the set of tuples satisfying C. More precisely, MDD(C) is defined on $X(C)$, such that layer i corresponds to the variable x_i. For each arc emanating from layer i, the label of the arc is one of the possible values for variable x_i. The labels of a path $p = ((u_1, u_2, a_1), (u_2, u_3, a_2), ..., (u_r, tt, a_r))$ represent one tuple satisfying the constraint C.

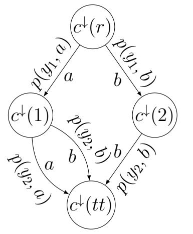
Consistency with MDD(C) Let $D_c(x)$ be the current domain (i.e. remaining values) of variable x. A value a of variable x_i is valid if and only if $a \in D_c(x_i)$. A path p is valid if and only if, for each of its arcs, its label is valid. A node n is valid if and only if at least one valid path goes through it. An arc (u, v, a) at layer i is valid if at least one valid path goes through it. The value a of variable x_i is consistent with a constraint C if and only if there exists at least one valid arc labeled as a emanating from layer i.

MDD propagator for constraint C An MDD propagator associated with MDD(C) is an algorithm which removes some inconsistent values of $X(C)$. The MDD propagator establishes arc consistency of C if and only if it removes all arc inconsistent values with C. This means that it ensures that, for each value of each variable, there is a valid path in MDD(C).

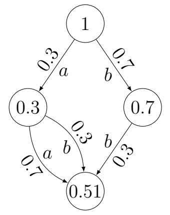
Constraining MDDs and MDD consistency Usually, in optimization, some variables may occur in multiple constraints. To enforce multiple MDDs on overlapping variable sets, their intersection can be taken instead. The issue with consecutive intersections is that the size of the MDD may grow exponentially with the number of constraints. Hence, many works propose to constrain MDDs, instead of directly applying intersections (Hoda, van Hoeve, and Hooker 2010; Perez and Régin 2017b). Constraints such as the AllDifferent, Among, Cost, Probabilities etc. can be directly applied on top of MDDs. Usually, applying a constraint on an MDD implies modifications of the MDD, by removing arcs or other transformations. The goal is to reduce the number of incorrect solutions stored in the MDD, with respect to the applied constraint. Applying a constraint on top of an MDD implies that not all the paths of this MDD are valid. An arc is said *MDD consistent* if it belongs to at least one valid path that also satisfies the constraint applied to the MDD (Hoda, van Hoeve, and Hooker 2010). MDD consistency is achieved when all the inconsistent arcs of the MDD have been removed. In this paper, the *Confidence* constraint is applied on top of a given MDD, the MDD that the random variables should satisfy. The goal of the propagation algorithm is to ensure that given the current state of the MDD, the confidence is still valid.

Generalized Confidence

We consider problems involving random variables for which the probability distribution is known. Recently, the *Confidence* constraint has been defined (Mercier-Aubin et al. 2020). Let $\gamma \in [0, 1]$ be a value. Let $Y = (y_1, ..., y_r)$ be a vector of independent random bounded integer variables. Let $X = (x_1, ..., x_r)$ be a vector of integer variables. This



(a) In each arc, labels are a or b. The probability associated to the arc are $p(y_1, a)$ etc.



(b) The confidence that node tt is reached is $c^{\downarrow}(tt)$ = 0.3 $c^{\downarrow}(2)$ + (0.7 + 0.3) $c^{\downarrow}(1)$ = 0.51

Figure 2: Confidence propagation in MDDs

constraint ensures that with probability at least γ, the random variables Y will be element-wise bounded by X. More precisely it ensures:

$$Pr[\bigwedge_{i=1}^{r} (y_i \leq x_i)] \geq \gamma \quad (3)$$

In this paper, the definition of the confidence is extended to any constraint, and no longer to only $\bigwedge_{i=1}^{r} y_i \leq x_i$. The Generalized confidence constraint ensures that with probability at least γ, the random variables Y will satisfy a given constraint C. Let $C(X, Y)$ denote the satisfaction of the constraint C by the random variables Y given the assignment of the decision variables X.

$$\text{Confidence}(X, Y, C, \gamma) \iff Pr[C(X, Y)] \geq \gamma \quad (4)$$

Note that if $\gamma = 0$, then the constraint is always satisfied. If $\gamma = 1$, then valid assignments of X must allow any non-zero probability assignments of Y to satisfy $C(X, Y)$.

Given a constraint C for which the confidence of satisfaction must be ensured, Let $M_C = MDD(C)$. Define $M_C(t)$ as the validity of the path through M_C denoted by tuple t. The *Confidence* constraint can be rewritten using its MDD implementation:

$$\text{Confidence}(X, Y, M_C, \gamma) \iff Pr[M_C(X, Y)] \geq \gamma \quad (5)$$

Probability, Confidence and MDDs

We consider in the rest of this section that the MDD contains only random variables. In an MDD, the probability of independent random variables can be directly encoded in the arcs. Let $Y = (y_1, ..., y_r)$ be a vector of independent random bounded integer variables. For each random variable y_i, let $pmf(y_i) = (p(y_i, v_1)., ..., p(y_i, v_r))$ be the probability distribution with $p(y_i, v_i) = Pr[y_i = v_i]$. Using MDDs, these probabilities can be associated to arcs as shown in Figure 2 (a). In such settings, the probability of a path (i.e. a tuple satisfying MDD(C)) is the product of the probabilities of its arcs.

In an MDD, each path from the root node to the true terminal node represents a valid tuple. The sum of the probabilities of all the tuples represents the probability that a random sample of the variables Y will satisfy constraint C. Extracting all the tuples of the MDD is usually not practical as it is a compressed data-structure, containing an exponential number of tuples. Luckily, this sum can easily be processed locally (i.e. by node) and then interpreted globally. Let $c^{\downarrow}(u)$ (resp. $c^{\uparrow}(u)$) be the cumulative probability of the incoming paths (resp. outgoing paths) of node u. Let $c^{\downarrow}(r) = 1$ be the value for the root node. Let $c^{\uparrow}(tt) = 1$ be the value for the true terminal node. We have for any node v at layer i the recursive formulas:

$$c^{\downarrow}(v) = \sum_{(u,v,a) \in L(i)} p((u,v,a))c^{\downarrow}(u) \tag{6}$$

$$c^{\uparrow}(u) = \sum_{(u,v,a) \in L(i)} p((u,v,a))c^{\uparrow}(v) \tag{7}$$

and

$$Confidence(Y, M, \gamma) \iff c^{\downarrow}(tt) \geq \gamma \iff c^{\uparrow}(r) \geq \gamma \tag{8}$$

This can be directly cast into the systematic approach to MDD-based constraint (Hoda, van Hoeve, and Hooker 2010), analogously to the probability constraint for MDD. Hence a linear time algorithm can propagate the *Confidence* constraint.

Example Figure 2 (b) shows an example of the application of this algorithm. First for node 1, the incoming confidence is given by $c^{\downarrow}(1) = 0.3c^{\downarrow}(r) = 0.3$. Then for node 2, the incoming confidence is given by $c^{\downarrow}(2) = 0.7c^{\downarrow}(r) = 0.7$. Finally the confidence of node tt is given by $c^{\downarrow}(tt) = 0.3c^{\downarrow}(2) + (0.7 + 0.3)c^{\downarrow}(1) = 0.51$. It is important to note that $p((a,a)) + p((a,b)) + p((b,b)) = 0.3 * 0.7 + 0.3 * 0.3 + 0.7 * 0.3 = 0.51$.

Propagation

We consider the constraint Confidence(X, Y, M, γ). When $X = \emptyset$, no decision variable is present in the constraint. Let $Y = (y_1, ..., y_r)$ be a vector of independent random bounded integer variables. Let MDD M be defined over r layers of arcs. Let $\gamma \in [0, 1]$ be a value. Either $c^{\downarrow}(tt) \geq \gamma$ and the constraint is satisfied, or $c^{\downarrow}(tt) < \gamma$ and the problem is not satisfiable.

In the general case, constraints on random variables also involve decision variables. The *Confidence* constraint is consistent if the decision variables only have values that lead to solutions with enough confidence for the random variables. This section considers first the general case where both random variables and decision variables are contained in the MDD. Then, as this may not always be tractable, three additional dedicated constraint propagation algorithms are provided for the case of MDD containing only random variables.

Filtering of the Decision Variables ($Confidence$) Let MDD M be defined over $k + r$ layers of arcs. Let $\gamma \in [0, 1]$ be a value. Let $X = (x_1, ..., x_k)$ be a vector of integer variables. Let $Y = (y_1, ..., y_r)$ be a vector of independent random bounded integer variables. Let MDD M have the first k layers representing the decision variables and the last r layers representing the random variables. Note that the following algorithms can be adapted to any ordering of the variables. This ordering has been selected for the sake of clarity. Nevertheless, without loss of generality any MDD can be transformed into an MDD with such an ordering. In such settings, we do not consider $c^{\downarrow}(root) = 1$, but $c^{\downarrow}(u) = 1$ for any node u in the layer $k + 1$, the first layer of the random variables. Indeed, as MDDs are deterministic, and the decision variables are all stacked at the top of the MDD, once assigned, only one node at the first layer of the random variables will remain. Equation (9) is a generalization of Equation (8) integrating both decision and random variables. When $X = \emptyset$, $root$ is the only node of the first layer.

$$Confidence(X, Y, M, \gamma) \iff c^{\downarrow}(tt) \geq \gamma \tag{9}$$

The propagation algorithm remove values of decision variables that cannot lead to a solution with a confidence greater or equal than γ. Consider the value $c^{\uparrow}(u)$ of each node of layer $k + 1$. This value represent the accumulated probability of the paths emanating from node u. As stated before, once the decision variables present in the MDD are assigned, only one node of layer $k + 1$ will remain. This implies that the $c^{\uparrow}(u)$ of layer $k + 1$ are similar to the $c^{\uparrow}(r)$ for the case without decision variables. A simple propagation rule can be defined.

Proposition 1 *Global arc consistency on $Confidence(X, Y, M, \gamma)$ is enforced by the following rule. For all the nodes u at layer $k + 1$ of the MDD, if $c^{\uparrow}(u) < \gamma$ then node u becomes invalid and is removed from the MDD.*

The proof can be done by contradiction. Consider a value a from a decision variable x_i that cannot lead to a solution with confidence greater than γ. In order to be supported, a valid arc (u, v, a) at layer i must exists. An arc is valid if it belong to a valid path. A path is valid is all the value of the path are still in the domain of their associated variable and, as stated by the proposition, at layer $k + 1$, $c^{\uparrow}(w) \geq \gamma$. $c^{\uparrow}(w) \geq \gamma$ implies that the confidence is greater than γ, hence the contradiction.

The complexity of enforcing the propagation can be seen in two parts. First the cost of processing $c^{\uparrow}(u)$ of each node of layer $k + 1$. This can be done in linear time over the size of the MDD by using the recursive formula from equation (7). Second, the cost of removing the invalid arcs and values from the MDD. This can be done in linear time by two passes in the MDD, one bottom up and one top bottom. Note that these can be done incrementally during the search as done by known MDD filtering algorithms.

Example Consider the MDD from Figure 1 (b). This MDD represents the constraint where the sum of the variables is bounded by 5. Let $X = (x_1)$ be a vector of one decision variable with domain $\{0, 1, 2\}$. Let $Y = (y_1, y_2)$ be a vector of 2 random discrete uniform variables with possible values $\{0, 1, 2\}$. Let $\gamma = \frac{7}{9}$. First, we have $c^{\uparrow}(< 5) = 1$. Then, $c^{\uparrow}(< 3) = 1, c^{\uparrow}(3) = \frac{2}{3}, c^{\uparrow}(4) = \frac{1}{3}$. Then, $c^{\uparrow}(0) = 1$, $c^{\uparrow}(1) = \frac{8}{9}$, $c^{\uparrow}(2) = \frac{6}{9}$. This implies that $c^{\uparrow}(2) < \gamma$, hence node 2 is removed. The removal of node 2 implies the removal of value 2 from x_1 as it was the only path containing this value for this variable.

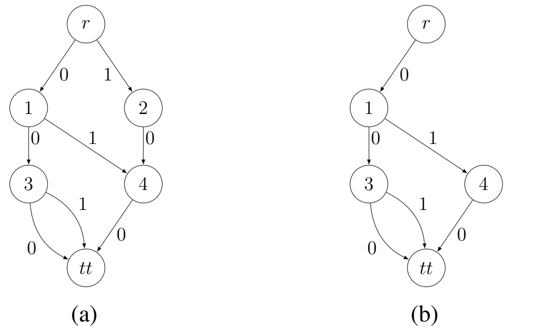


Figure 3: (a) An MDD representing the at most 1 constraint. (b) Propagation of the removal of arc $(r, 2, 1)$.

Particular Cases

It is often interesting to reduce the number of variables contained in an MDD. That is the reason why this section proposes three cases where a complete propagation is possible by removing the decision variables from the MDD, and applying a dedicated algorithm.

1. MDD + *per-arc* management. $Confidence_{\rightarrow}$
2. MDD + *in constraint*. $Confidence_{\in}$
3. MDD + *Lower-than constraint*. $Confidence_{<}$

1) $Confidence_{\rightarrow}$ Let MDD M be defined over r layers of arcs. Let $\gamma \in [0, 1]$ be a value. Let $Y = (y_1, ..., y_r)$ be a vector of independent random bounded integer variables. Let $B = (b_1, ..., b_{|E|})$ be a vector of Boolean variables with one variable per arc of the MDD. Arc $a_i = (u, v, a)$ belongs to the MDD if $b_i = True$, otherwise it is removed. For each variable b_i associated to arc (u, v, a) define vector c as follows:

$$c_i = c^{\downarrow}(u)p((u, v, a))c^{\uparrow}(v) \qquad (10)$$

If omitting arc i from the MDD decreases the satisfaction probability below the threshold, then b_i must be true. This filtering for each value can be defined using:

$$c^{\downarrow}(tt) - c_i < \gamma \quad \rightarrow \quad b_i \qquad (11)$$

Example Consider the MDD from Figure 3 (a). This MDD is defined over three variables, each of them taking values in the set $\{0, 1\}$. Let $Y = (y_1, y_2, y_3)$ be a vector of three independent uniform random bounded integer variables. The four valid tuples of this MDD, representing the at-most-one constraint for value 1 are ((0,0,0),(1,0,0),(0,1,0),(0,0,1)). The probability of each tuple is $0.5^3 = 0.125$. The total confidence is $4 * 0.125 = 0.5$. Hence, randomly sampling assignments of three independent uniform Boolean variables has a probability of 0.5 to satisfy the at most one 1 constraint. Let $\gamma = 0.35$.

Consider the impact of the removal of arc from node r to node 2 labeled by 1. First the propagation of this deletion results in MDD Figure 3 (b). Arc from node 2 to node 4 labeled by 0 is removed as it is no longer part of a valid path. The values of the c of each node are: $c^{\uparrow}(r) = 0.375, c^{\downarrow}(r) = 1$, $c^{\uparrow}(tt) = 1, c^{\downarrow}(tt) = 0.375$, $c^{\uparrow}(1) = 0.75, c^{\downarrow}(1) = 0.5$, $c^{\uparrow}(3) = 1, c^{\downarrow}(3) = 0.25$, $c^{\uparrow}(4) = 0.5, c^{\downarrow}(4) = 0.25$.

Let B be a vector of Boolean variables with one variable per arc of the MDD. Let the variable $b_{(r,2,1)}$ be assigned to *false*, be the cause of the initial arc removal. $c_{(r,1,0)} = c^{\downarrow}(r)*0.5*c^{\uparrow}(1) = 1*0.5*0.75 = 0.375$. $c^{\downarrow}(tt) - c_{(r,1,0)} = 0 < \gamma$, hence $b_{(r,1,0)}$ =*true* (rule (11)). The same goes for all the arcs.

2) $Confidence_{\in}$ Let MDD M be defined over r layers of arcs. Let $\gamma \in [0, 1]$ be a value. Let $Y = (y_1, ..., y_r)$ be a vector of independent random bounded integer variables. Let $X = (X_1, ..., X_r)$ be a vector of set variables. Each set variable X_i contains as many values as there are different labels at layer i of the MDD. $a \notin X_i$ implies that no arc labeled by a at layer i exists. This filtering intersects the constraint contained in the MDD and $\bigwedge_{i=1}^{r}(y_i \in X_i)$. For each variable X_i define vector $c_i = (c_{i,1}, ..., c_{i,d})$ where for each value $a \in X_i$, $c_{i,a}$ is defined as follows:

$$c_{i,a} = \sum_{(u,v,a) \in L(i)} c^{\downarrow}(u)p((u, v, a))c^{\uparrow}(v) \qquad (12)$$

The first filtering for each value can be defined using:

$$c^{\downarrow}(tt) - c_{i,a} < \gamma \quad \rightarrow \quad a \in X_i \qquad (13)$$

The second filtering for the *cardinality* component of the set variable is defined as follows. Let $c_{i\downarrow()}$ be a non-increasing ordering of c_i. Let $c_{i\downarrow(j)}$ be the jth largest value of c_i such that $c_{i\downarrow(1)} \geq c_{i\downarrow(2)} \geq \ldots \geq c_{i\downarrow(d)}$.

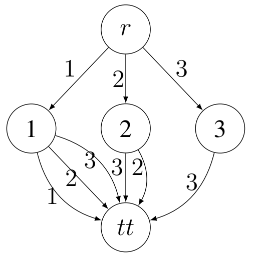
$$card_i \geq \min\{k \in \{1, \ldots, d\} \mid \sum_{j \in \{1,\ldots,k\}} c_{i\downarrow(j)} > \gamma\}\}. \qquad (14)$$

In this settings, values that becomes impossible for a set-variable lead to the removal of the associated arcs.

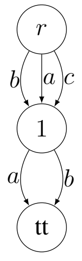
Example Consider again the example of the removal of arc $(r, 2, 1)$ of the MDD from Figure 3 (a). Let $X = (X_1, X_2, X_3)$ be a vector of three set variables. Let $1 \notin x_1$ be the cause of the initial arc removal. $c_{2,0} = c^{\downarrow}(1) * 0.5 * c^{\uparrow}(3) = 0.5*0.5*1 = 0.25$. $c_{2,1} = c^{\downarrow}(1)*0.5*c^{\uparrow}(4) = 0.5* 0.5 * 0.5 = 0.125$. Note that $c_{2,0} + c_{2,1} = c^{\uparrow}(r) = c^{\downarrow}(tt)$. $c_{3,0} = c^{\downarrow}(3) * 0.5 * c^{\uparrow}(tt) + c^{\downarrow}(4) * 0.5 * c^{\uparrow}(tt) = 0.25 * 0.5 * 1 + 0.25 * 0.5 * 1 = 0.25$. $c_{3,1} = c^{\downarrow}(3) * 0.5 * c^{\uparrow}(tt) = 0.25 * 0.5 * 1 = 0.125$. Application of rule (13) leads to $0 \in X_1, 0 \in X_2, 1 \in X_2, 0 \in X_3, 1 \in X_3$. Application of rule (14) leads to $card_2 \geq 2$, $card_2 \geq 2$.

3) $Confidence_{<}$ Let MDD M be defined over r layers of arcs. Let $\gamma \in [0, 1]$ be a value. Let $Y = (y_1, ..., y_r)$ be a vector of independent random bounded integer variables. Let $X = (x_1, ..., x_r)$ be a vector of integer variables defining the representing the upper-bound $y_i \leq x_i$. More precisely, if $x_i < k$, then no arc labeled by values greater than or equal to k are present at layer i. This filtering intersects the constraint contained in the MDD and $\bigwedge_{i=1}^{r} y_i \leq x_i$. For each variable x_i and each value $a \in x_i$ define $c_{i,a}$ using equation (12). Let a be the smallest required value such that:

$$\min_{a \in D(x_i)} \quad a \quad \text{subject to} \quad \sum_{b=\min(D_y)}^{a} c_{i,b} \geq \gamma \qquad (15)$$



(a)



(b)

Figure 4: (a) MDD for the constraint $y \leq x$. y is the first variable of the MDD. (b) A stick MDD for the constraint $\bigwedge_{i=1}^{r} y_i \leq x_i$.

Then, variable x_i can be pruned using:

$$x_i \geq a \tag{16}$$

In this setting, values that are larger than $\max(D(x_i))$ are removed from the MDD. Arcs labeled by values lower or equal to $\min(D(x_i))$ are kept. Note that the same processing can be done for a lower bound of y_i instead of an upper bound. In practice, it works for any permutation of the values of D_y, not only increasing and decreasing.

Example Consider again the example of the removal of arc $(r, 2, 1)$ of the MDD from Figure 3 (a). Let $X = (x_1, x_2, x_3)$ be a vector of three decision variables with domain $\{0, 1\}$. Let the variable x_1 be assigned to 0, be the cause of the initial arc removal. $c_{i,a}$ are processed in the same ways as for the $\in$ version above. Application of rule (16) leads to $x_2 \geq 1$ and $x_3 \geq 1$.

Generalization The original confidence constraint considered the constraint $\bigwedge_{i=1}^{r} y_i \leq x_i$. Consider the MDD from Figure 4 (a). This MDD represents the constraint $y < x$ for one random variable. In the case where the MDD contains first decision variables and then random variables, the size of the MDD grows exponentially with the number of variables. This is not an issue as the decision variables can be safely removed from the MDD. Indeed, a stick MDD as defined in Figure 4(b) defined on random variables is enough. A stick MDD is an MDD such that each layer contains 1 node.

Proposition 2 *Algorithm $Confidence_{\leq}$ and a stick MDD on the random variables is equivalent to the Confidence filtering algorithm defined in (Mercier-Aubin et al. 2020).*

First, the stick MDD contains the Cartesian product of the domain, hence no constraint. Then, by definition, the constraint enforced by $Confidence_{\leq}$ is $P(\bigwedge_{i=1}^{r}(y_i \leq x_i)) \geq \gamma$. The filtering algorithms are equivalent.

The maximum size of such an MDD is linear over the domain of the variables. The complexity of applying the confidence constraint is linear on the size of the MDD. Hence, applying arc consistency is linear with respect to the sizes of the domain. Note that in the general case, the MDD will not be a stick MDD, but an MDD containing additional constraints.

Application: Chemicals Delivery Assignment

The following problem is part of a larger pipeline of chemicals processing. Chemical substances are received by the factory every day. Once received, they are stored into containers. Containers are constrained by both the type of product and the type of truck delivering the product. Each container already contains a known amount of product and has a maximum capacity. Every night, the factory must assign each delivery to a suitable container.

In addition, for products in high demand the total amount delivered is larger than the remaining capacities 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. Some products are in higher demand than others, and more importantly, some containers are easier to access and tend to have a higher rate of emptying. The problem to solve is to assign chemicals to containers, with a potential overflow of the capacities. These overflows, using prior data, should match, with high confidence, the usual emptying of the containers. In practice, these problems involve several plants for the same factory and other details that are ignored in this experimental evaluation.

More formally, this is an assignment problem. Each delivery $j \in D$ stores a quantity q_j in a container $b_i \in B$. Each container b_i has a maximum capacity C_i. Each container b_i will be emptied in parallel of a quantity y_i unknown in advance. The exact quantity y_i is unknown, but its probability mass function is known. The total amount of used chemical is also upper bounded by K, that is $\sum_i^{|B|} y_i \leq K$.

A solution is an assignment of deliveries to containers such that the stored quantity in each bucket minus the quantity that will be used does not exceed the maximum capacity with a high confidence. Let MDD Σ_K represent the sum of $|B|$ variables with domain $[0, d]$ bounded by K. Let $a_{d,b}$ be the Boolean assignment variable indicating that delivery d is assigned to container b. Let x_i be the lower bound variable used to constrain the random variables and the MDD. Constraints of the models are:

$$\sum_{i=1}^{|D|} a_{i,b} q_i - x_b \leq C_b \quad \forall b \in B \tag{17}$$

$$\sum_{b=1}^{|B|} a_{i,b} = 1 \quad \forall i \in D \tag{18}$$

$$\text{Confidence}_{\geq}(x, Y, \Sigma_K, \gamma) \tag{19}$$

With (17) being the capacity constraint, (18) being the assignment constraint. Finally constraint (19) is the confidence constraint. We compare this model to the model that contains only the $\bigwedge_{i=1}^{r} y_i \leq x_i$ in the confidence constraint as proposed in (Mercier-Aubin et al. 2020). A full definition of this other model is given in appendix.

Data Two data sets have been created from existing data: one called *small*, and one called *large*. Both of them contain samples from the distributions of the random variables. In the *small* data set, each delivery has a quantity between 1 and 5. Each container contains some previous content and a max capacity ranging from 25 to 40. For each container, the

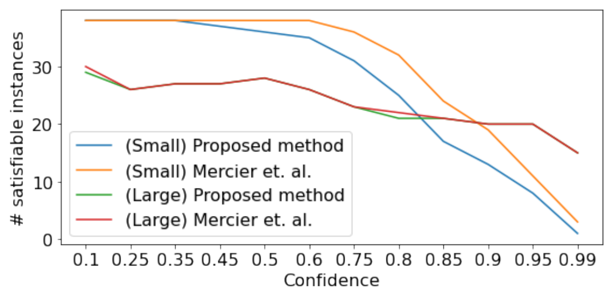


Figure 5: Impact of the bounded sum constraint. x-axis is confidence. y-axis is number of instances satisfiable

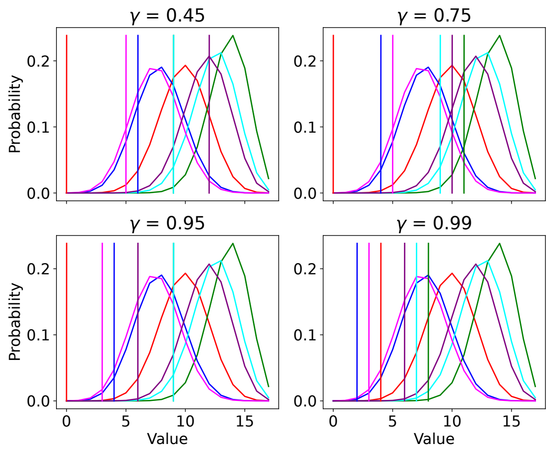


Figure 6: Impact of the confidence.

random variable ranges from 0 to 9. Each random variable has its own distribution. In the *large* data set, each delivery has a quantity between 2 and 8. Each container contains some previous content and a max capacity ranging from 50 to 70. For each container, the random variable ranges from 0 to 17. For both of data sets, the number of deliveries span from 40 to 100. The chemical and truck compatibility is enforced at the initial propagation. The timeout is set to 10 minutes. The implementation uses our internal constraint-programming solver.

Results First, we analyse the impact of constraining the conjunction of constraints. Figure 5 shows the impact of having both the maximum sum Σ_K and the lower-bound into the confidence constraint. As we can see, for the same level of confidence, more instances are unsatisfiable. The reason is that even if independently the $\forall i, y_i \leq x_i$ constraint and the bounded sum constraint might reach the confidence level, constraint (19) contains both of them and processes the confidence of their conjunction. This first benchmark shows why there is a strong need to include all the constraints on the random variables in the same Confidence constraint, as proposed in this paper. Note that in the *large* data set, several instances reached the timeout limit, hence the smaller number of differences. Also, MDDs construction takes few ms, sizes are around 1,000 nodes and 6,000 arcs.

Figure 6 shows both the probability distributions of Y and the selected values of x. The same color is used for the distribution of y_i and the value of x_i. When γ is small, the x variables have a large range of values. Then, as γ is increasing, the range of x decreases. In this example, note that when $\gamma = 0.99$, all the containers are overloaded. Table 1 shows the number of solved instances. Remark that a higher confidence helps to prove unstatisfiablity of instances. The appendix gives more insight on these experiments.

$\lvert D \rvert$	**1%**	**35%**	**50%**	**75%**	**85%**	**95%**	**99%**
40	16,1,3	17,1,2	17,1,2	15,2,3	15,3,2	11,7,2	5,14,1
60	17,1,2	16,1,3	14,3,3	11,4,5	10,6,4	7,8,5	5,13,2
80	18,0,2	17,0,3	17,0,3	14,1,5	6,7,7	5,9,6	2,16,2
100	16,0,4	15,0,5	16,0,4	14,1,5	7,8,5	5,10,5	4,12,4

Table 1: For each cell, values represent respectively, the number of instances solved and satisfied, the number of instances solved and unsatisfied, the number of instances that reached the timeout

Notes on the Complexity

Computing the probability of the sum of random variables requires computing consecutive convolutions. Indeed, the probability of a value j to be reached by summing two random independent variables x and y is given by

$$p(x + y = j) = \sum_k p(x = k)p(y = (j - k)) \qquad (20)$$

Let the constraint $\sum_{i=1}^{r} y_i \leq K$ be represented by an MDD denoted Σ_K. The MDD for the sum of discrete variables is well-defined and has pseudo-polynomial size (Trick 2001). Each node of this MDD, for each layer, is associated to an intermediate sum. Figure 2 (b) is an example of Σ_K.

In this experimental section possible values for the random variables are a range [0,1,2,3,...d]. For each layer j, the maximum number of nodes is bounded by $\sum_{i=1}^{j} y_i < jd$. For this particular problem, the size is in the worst-case $O(rd)$. Moreover, the constraint considered in this problem is upper-bounded by K, hence no more than K nodes can be present in each layer. Finally, as shown in Figure 2 (b), the reduction of the MDD compresses the final MDD even more. Let $C = \min(rd, K)$. Enforcing consistency for the constraint *Confidence*$_{\geq}(x, Y, \Sigma_K, \gamma)$ has a worst-case complexity of $O(rC)$.

Conclusion

In this paper, another step has been made toward efficient solving of stochastic optimization problem using constraint programming and MDDs. A generalization of the confidence constraint, a constraint ensuring that the probability of the constraint to be satisfied by the random assignment of the variables is greater than the threshold, has been proposed. Moreover, an MDD implementation has been proposed. Several different filtering algorithms are proposed to constrain the MDD. These algorithms, in combination with a good model, solved hard instances of the chemical assignment problem.

References

Andersen, H. R.; Hadzic, T.; Hooker, J. N.; and Tiedemann, P. 2007. A Constraint Store Based on Multivalued Decision Diagrams. In *CP*, 118–132.

Bergman, D.; Ciré, A. A.; van Hoeve, W.; and Hooker, J. N. 2016. *Decision Diagrams for Optimization*. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer. ISBN 978-3-319-42847-5.

Bergman, D.; van Hoeve, W. J.; and Hooker, J. N. 2011. Manipulating MDD Relaxations for Combinatorial Optimization. In *CPAIOR*, 20–35.

Bryant, R. E. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Computers*, 35(8): 677–691.

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

Castro, M. P.; Cire, A. A.; and Beck, J. C. 2022. Decision Diagrams for Discrete Optimization: A Survey of Recent Advances. *arXiv preprint arXiv:2201.11536*.

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

Cheng, K. C. K.; and Yap, R. H. C. 2010. An MDD-based generalized arc consistency algorithm for positive and negative table constraints and some global constraints. *Constraints*, 15(2): 265–304.

Demeulenaere, J.; Hartert, R.; Lecoutre, C.; Perez, G.; Perron, L.; Régin, J.-C.; and Schaus, P. 2016. Compact-table: Efficiently filtering table constraints with reversible sparse bit-sets. In *International Conference on Principles and Practice of Constraint Programming*, 207–223. Springer International Publishing.

Gange, G.; Stuckey, P.; and Szymanek, R. 2011. MDD propagators with explanation. *Constraints*, 16: 407–429.

Gillard, X.; Coppé, V.; Schaus, P.; and Cire, A. A. 2021. Improving the filtering of branch-and-bound MDD solver. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 231–247. Springer.

Hadzic, T.; Hooker, J. N.; O'Sullivan, B.; and Tiedemann, P. 2008. Approximate Compilation of Constraints into Multivalued Decision Diagrams. In *CP*, 448–462.

Hnich, B.; Rossi, R.; Tarim, S. A.; and Prestwich, S. 2012. Filtering algorithms for global chance constraints. *Artificial Intelligence*, 189: 69–94.

Hoda, S.; van Hoeve, W. J.; and Hooker, J. N. 2010. A Systematic Approach to MDD-Based Constraint Programming. In *CP*, 266–280.

Hooker, J. 2022. Stochastic Decision Diagrams. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 138–154. Springer.

Kargar, B.; Pishvaee, M. S.; Jahani, H.; and Sheu, J.-B. 2020. 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.

Latour, A. L.; Babaki, B.; Fokkinga, D.; Anastacio, M.; Hoos, H. H.; and Nijssen, S. 2022. Exact stochastic constraint optimisation with applications in network analysis. *Artificial Intelligence*, 304: 103650.

Lecoutre, C. 2011. STR2: optimized simple tabular reduction for table constraints. *Constraints*, 16(4): 341–371.

Lin, X.; Janak, S. L.; and Floudas, C. A. 2004. A new robust optimization approach for scheduling under uncertainty:: I. Bounded uncertainty. *Computers & chemical engineering*, 28(6-7): 1069–1085.

Mercier-Aubin, A.; Dumetz, L.; Gaudreault, J.; and Quimper, C.-G. 2020. The Confidence Constraint: A Step Towards Stochastic CP Solvers. In *International Conference on Principles and Practice of Constraint Programming*, 759–773. Springer.

Pachet, F.; and Roy, P. 2011. Markov constraints: steerable generation of Markov sequences. *Constraints*, 16(2): 148–172.

Pachet, F.; Roy, P.; Papadopoulos, A.; and Sakellariou, J. 2015. Generating 1/f noise sequences as constraint satisfaction: The voss constraint. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Parra, M. A.; Terol, A. B.; Gladish, B. P.; and Urıa, M. R. 2005. Solving a multiobjective possibilistic problem through compromise programming. *European Journal of Operational Research*, 164(3): 748–759.

Perez, G.; Rappazzo, B.; and Gomes, C. 2018. Extending the capacity of 1/f noise generation. In *International Conference on Principles and Practice of Constraint Programming*, 601–610. Springer.

Perez, G.; and Régin, J. 2014. Improving GAC-4 for Table and MDD Constraints. In *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, 606–621.

Perez, G.; and Régin, J. 2017. MDDs are Efficient Modeling Tools: An Application to Dispersion Constraints. In *Integration of AI and OR Techniques in Constraint Programming*.

Perez, G.; and Régin, J.-C. 2015. Efficient operations on MDDs for building constraint programming models. *International Joint Conference on Artificial Intelligence, IJCAI-15, Argentina*.

Perez, G.; and Régin, J.-C. 2017a. MDDs: Sampling and probability constraints. In *International Conference on Principles and Practice of Constraint Programming*, 226–242. Springer.

Perez, G.; and Régin, J.-C. 2017b. Soft and Cost MDD Propagators. In *Proc. AAAI'17*.

Pesant, G. 2004. A Regular Language Membership Constraint for Finite Sequences of Variables. In *Proc. CP'04*, 482–495.

Pesant, G. 2015. Achieving Domain Consistency and Counting Solutions for Dispersion Constraints. *INFORMS Journal on Computing*, 27(4): 690–703.

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

Régin, J.-C. 1994. A Filtering Algorithm for Constraints of Difference in CSPs. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, AAAI '94, 362–367. Menlo Park, CA, USA: American Association for Artificial Intelligence. ISBN 0-262-61102-3.

Rendl, A.; Tack, G.; and Stuckey, P. J. 2014. Stochastic minizinc. In *International Conference on Principles and Practice of Constraint Programming*, 636–645. Springer.

Rossi, R.; Hnich, B.; Tarim, S. A.; and Prestwich, S. 2015. Confidence-based reasoning in stochastic constraint programming. *Artificial Intelligence*, 228: 129–152.

Rossi, R.; Tarim, S. A.; Hnich, B.; and Prestwich, S. 2008. A global chance-constraint for stochastic inventory systems under service level constraints. *Constraints*, 13(4): 490–517.

Roy, P.; Perez, G.; Régin, J.-C.; Papadopoulos, A.; Pachet, F.; and Marchini, M. 2016. Enforcing structure on temporal sequences: The Allen constraint. In *International Conference on Principles and Practice of Constraint Programming*, 786–801. Springer International Publishing.

Trick, M. 2001. A dynamic programming approach for consistency and propagation for knapsack constraints. In *CPAIOR'01*.

Walsh, T. 2002. Stochastic constraint programming. In *ECAI*, volume 2, 111–115.