On Constraint Linear Decompositions Using Mathematical Variables

Thierry Petit

On Constraint Linear Decompositions Using Mathematical Variables

Thierry Petit
LS2N (TASC) / IMT Atlantique (DAPI), Nantes, France,
Thierry.Petit@imt-atlantique.fr

***Abstract*—A wide literature exists on constraint programming model linearization, based on integer domain decomposition. This paper considers the systematic study of classical global constraints, but in the context of mathematical variables. We consider constraints originally stated using integer domain variables, for which we investigate new definitions and linear decompositions using bounded rational variables. We introduce a generic scheme for reification and softening. Combined with state-of-the-art decompositions on integer variables, this approach permits solving discrete-continuous high level models using a single modeler, connected to a MILP solver.**

I. Introduction

In constraint programming (CP), the practical need of considering both continuous and discrete variables motivated building a bridge between continuous and discrete CP solvers (see, e.g., [1]), or mathematical programming and CP hybridization. In the latter case, linear decompositions of the most commonly used global constraints on integer domain variables have been introduced [2], [3], [4]. In this paper, we investigate the linear decomposition of global constraints, as well as their reification and softening, but considering continuous variables. Four main motivations justify exploring this topic independently from any specific application.

1) Some global constraints naturally involve continuous quantities although most decision variables take integer values. For instance, given a set $X = \{x_1, \ldots, x_n\}$ of integer variables and an integer variable s, `Deviation` [5] is defined as follows: $nz = (\sum_{i=1}^{n} x_i)$ and $s = (\sum_{i=1}^{n} |x_i - z|)$. In its original definition, z is an integer constant (a fixed argument of the constraint), representing a mean. This is due to the need of using integer domain variables and constants. In practice, however, defining this argument as an integer before solving the problem may be tricky [6]. One may prefer using a rational variable.
2) Combined with state-of-the-art decompositions of constraints on integer variables, linear formulations of global constraints on rational variables permit to solve high-level discrete-continuous CP models using a mathematical solver. Even though some continuous constraints might not (reasonably) be decomposed into linear equations, this approach is more expressive than purely discrete CP modelers. This opens new perspectives for fast prototyping: a model consists in the statement of variables and constraints (without search strategy). As default solving parameters of mathematical solvers are generally robust [7], simplicity of use bears no comparison with hybrid techniques [8] or decomposition [9].
3) Interpreting the definition of classical CP global constraints in the continuous case may lead to modeling blocks that capture new interesting concepts. We can deduce from linear decompositions a new violation measure, easy to aggregate in an objective function.
4) It seems reasonable to impose that, when used with (bounded) integer variables, a linear decomposition exactly matches the semantics of the original CP global constraint. In this case, decompositions consistent with continuous variables become alternatives to state-of-the-art decompositions on integer variables. The new ones can be used without stating a domain decomposition that would likely require many extra binary variables [3], [4].

The contributions of this article are presented as follows. In section II, we provide some useful background. We introduce a new decomposition in the context of integer domain variables. In section III, we consider two families of CP global constraints and we derivate definitions that are suitable to continuous variables. Section IV introduces a generic scheme for reification and softening and new linear decompositions on continuous variables. In section V, we present a prototype based on this work and some experimental results.

II. Constraint Linearization on Domain Variables

Mixed integer linear programming (MILP) models support integer and rational variables, linear constraints, objectives and linearized logical constraints. CP models make no restriction about the constraints. On the other hand, CP variables must range over a finite domain of values. This domain constraint can be encoded by linear constraints, which permits linearizing many CP global constraints [2], [3], [4].

Definition 1 (from Refalo [3]*):* Let X be a set of n integer domain variables and $D = d_1 \cup \ldots \cup d_n = \{v_1, \ldots, v_m\}$ be their domain union. All the domain constraints of the variables in X can be stated using a set B of $O(\sum_{i \in \{1,\ldots,n\}} |d_i|)$ binary variables and $O(n)$ inequalities. $\forall v_j \in D$, $\forall x_i \in X$, if $v_j \in d_i$ state a binary variable $b_{ij} \in B$, and $\forall x_i$ state:

$$(\sum_{v_j \in d_i} v_j b_{ij}) - x_i = 0 \quad \text{and} \quad \sum_{b_{ij} : v_j \in d_i} b_{ij} = 1.$$

Name	New scope	CP scope	Definition
`Deviation` [5]	vars: X: $\mathbb{Q}^n$ var: s: $\mathbb{Q}$ var: z: $\mathbb{Q}$	vars: X: $\mathbb{Z}^n$ var: s: $\mathbb{Z}$ constant: z: $\mathbb{Z}$	$nz = (\sum_{i=1}^{n} x_i)$ $\wedge\ s = (\sum_{i=1}^{n} \lvert x_i - z\rvert)$
`DistanceXYZ` [10]	vars:$\{x, y, z\}$: $\mathbb{Q}^3$	vars: $\{x, y, z\}$: $\mathbb{Z}^3$	$\lvert x - y\rvert = z$
`Diverse_sum` ([11], from [12])	vars: X: $\mathbb{Q}^n$ fixed vectors: V, $V = \{v_1, \ldots, v_k\}$, $v_j \in \mathbb{Q}^n$ constant: d: $\mathbb{Q}^+$	vars: X: $\mathbb{Z}^n$ fixed vectors: V, $V = \{v_1, \ldots, v_k\}$, $v_j \in \mathbb{Z}^n$ constant: d: $\mathbb{Z}^+$	$\sum_{i=1,j=1}^{i=n,j=k} \lvert x_i - v_j[i]\rvert \geq d$
`InterDistance` [13], [14]	vars: X: $\mathbb{Q}^n$ var: p: $\mathbb{Q}^+$	vars: X: $\mathbb{Z}^n$ constant: p: $\mathbb{Z}^+$	$\forall x_i, x_j, i < j,$ $\lvert x_i - x_j\rvert \geq p$
`Smooth`$_{\geq}$ [15]	vars: X: $\mathbb{Q}^n$ var: z: $\mathbb{N}$ constant: cst: $\mathbb{Q}^+$	vars: X: $\mathbb{Z}^n$ var: z: $\mathbb{N}$ constant: cst: $\mathbb{Z}^+$	$z =$ $\lvert\{x_i : \lvert x_i - x_{i+1}\rvert \geq cst\}\rvert$

TABLE I
EXAMPLES OF DISTANCE CONSTRAINTS AND THEIR CONTINUOUS INTERPRETATIONS.

Some inequalities in Definition 1 can be delayed during the solving process [3], and refined domain definitions can be considered [4]. From the domain decomposition, most usual CP global constraints can be encoded. Rather than providing a state-of-the art example of linearized global constraint, we introduce a new convex hull decomposition of the `Gcc` constraint with fixed occurrence bounds. This is a direct extension of an existing `Alldifferent` decomposition [2], [3].

Definition 2 (`Gcc` [16]*):* Let $X = \{x_1, \ldots, x_n\}$ be a set of variables, t an array of values. Each $t[k] \in t$ is associated with two integers $\underline{t[k]}$ and $\overline{t[k]}$, $0 \leq \underline{t[k]} \leq \overline{t[k]}$. `Gcc` holds if and only if for all indexes k: $\underline{t[k]} \leq |\{i : x_i = t[k]\}| \leq \overline{t[k]}$.

Lemma 1 (`Gcc` *linearization):* We use the notations of definitions 1 and 2. For all $t[k] \in t, B_{t[k]} = \{b_{ij} \in B : v_j \in d_i, v_j = t[k]\}$. `Gcc` can be represented by the domain decomposition of X and $O(|t|)$ inequalities. $\forall t[k] \in t$ state:

$$(\sum_{b \in B_{t[k]}} b) \geq \underline{t[k]} \text{ and } (\sum_{b \in B_{t[k]}} b) \leq \overline{t[k]}.$$

Proof: From Definition 1, $\forall b_{ij} \in B, b_{ij}=1 \Leftrightarrow x_i = v_j$. ■

III. GLOBAL CONSTRAINT DEFINITIONS

This section investigates how "classical" constraints of discrete CP solvers can be used in the context of continuous variables. The main difference with the decomposition scheme presented in Section II is that we do not consider domains, but only a minimum and maximum bound for each variable. The values of the two bounds should have, in our context, no significant impact on the solving efficacy; one may consider, for instance, the minimum and maximum integer represented by the computer. We use the following notations for rational intervals: $[v, w)$ ranges from v included to w excluded. $[v]$ is the interval reduced to value v. A variable x ranges over $[\underline{x}, \overline{x} + \epsilon)$, where $\epsilon > 0$ is an arbitrary small value. We focus on constraint families that are not necessarily disjoint.

A. Conjunctions of Inequalities.

Some constraints can be decomposed into a conjunction of linear inequalities, without any extra variable. Although not related to the number of variables, this notion conceptually relies to "network decomposable" constraints [17]. Considering linear constraints makes sense in the context of MILP as such decompositions can be added to a model without any transformation. If one wishes to extend this group by introducing disjunctions of linear inequalities, binary variables should be introduced in order to keep a linear model. We do not detail more this family because the related decomposition techniques and constraints are well-known.

B. Distance constraints.

Many constraints have semantics based on the Euclidian distance $abs = |x - y|$, where x, y, and abs are variables of any type, including constants. Identifying those constraints as a specific family makes sense, because this group is far from being restricted to distance constraints [18]. Moreover, calculations based on distances are often rational. We mentioned in introduction that stating a mean of physical distances can be useful in various contexts, and this mean is rarely an integer. Even though rational numbers can be artificially multiplied in order to use integer variables in some models, depending on the solving technique and other constraints in the problem this modification may not be straightforward. In the context of CP models solved by a MILP engine, limiting domain sizes is a key criterion for obtaining an efficient solving, as domains are decomposed using extra variables. Table I provides examples of global constraints that are commonly used in CP and their definition. All these constraints are based on the primitive constraint $abs = |x - y|$. The second and third columns in the table show the type of variables and parameters of the continuous interpretation of each constraint ("New scope"), in comparison with its original statement ("CP scope").

C. Occurrence constraints.

Occurrence constraints, such as `Gcc`, seem at first glance to be specific to variables with enumerated domains, such as

integer domains. On the other hand, a natural generalization of the notion of the "number of occurrences of value v within the assignment of a variable set X" consists in counting the "number of values in a range $[v, w)$, within the assignment of a variable set X". If the range is reduced to a single value $[v]$ the two definitions are equivalent. This interpretation permits adapting occurrence constraints to the continuous case, so that the new definitions match exactly to the original ones when all the variables in X are stated to be fixed with integer values and the intervals are restricted to single (integer) values.

Definition 3 (`RangeGcc`*):* Let $X = \{x_1, \ldots, x_n\}$ be a set of variables, t an array of disjoint intervals $t = \langle t[1] = [s_1, e_1), \ldots, t[m] = [s_m, e_m)\rangle$, not necessarily consecutive. Each $t[k] \in t$ is associated with two integers $\underline{t[k]}$ and $\overline{t[k]}$, $0 \leq \underline{t[k]} \leq \overline{t[k]}$. `RangeGcc` holds if and only if for all indexes k: $\underline{t[k]} \leq |\{i : x_i \in t[k]\}| \leq \overline{t[k]}$.

Using the same principle, by replacing in their definition values by intervals, one may obtain interpretations on continuous variables of the CP constraints `AtLeast` and `AtMost` [10], `Among` [19], [20], and constraints that correspond to conjunction of `Among` constraints, e.g., `Sequence` [19] or `OrderedDistribute` [21]. The case of `Among` is noticeable. Given an array of values t, a domain variable z and a set of domain variables X, `Among` is satisfied if and only if z is equal to the number of variables taking a value in t. Our generalization of the notion of occurrence corresponds implicitly to `Among`; a range replaces the array t. On the other hand, the generalization of `Among` to rational variables considers a series of ranges, one per value in t.

Practical applications may require to state constraints that count quantities, based on intervals of integer or rational values. They correspond to the modeling blocks obtained through the above generalizations. For instance, cruise lines organize trips where, generally, distances between ports vary; some people like practicing gambling activities while the vessel is underway, while other prefer to have time for visiting new places. In such a routing problem, only a subset of ports is selected. If a set of variables X represents the distances between pairs of successive ports in the tour, the `RangeGcc` on X allows to control the number of steps in predefined intervals of distances (e.g., small, medium and long).

IV. Decompositions, Reificaton and Softening

In this section we provide linear decompositions of the constraints of Section III, mostly based on the "Big-M" principle. Prior to this, we investigate a systematic scheme for softening and reifying constraints. We introduce a new violation measure. We consider strict inequalities of the form $\sum_1^n a_i x_i < b$. In the integer case, such an expression can be represented by adding an offset of 1 to turn it into a non strict inequality. In the rational case, we make the assumption that an arbitrary small offset $\epsilon > 0$ is used.

A. Generic Softening and Reification.

Instead of merely posting a constraint c it may be useful to reflect its truth value into a binary variable r. This process is called *reification*. It is used to express logical constraints. An extension of the $\{0, 1\}$ reification to integer *violation measures* (also called *violation costs*) leaded to the notion of soft global constraint [22]. We propose a generic scheme for softening any constraint represented by a set of linear inequalities.

The principle is to soften each individual linear constraint and aggregate the result for obtaining the decomposition of the softened global constraint. In the case of decompositions based on domain constraints, all linear constraints that represent domains should remain hard, as softening constraints does not mean allowing assignments that are not valid. In terms of modeling, this principle allows to tackle, in a generic way, important issues: over-constrained problems, reification and, by fixing the truth variable, negation of global constraints. Although it is new and generic, the following scheme relies to previous studies on frequency assignment problems [23].

Proposition 1 (Linear constraint reification): Let c be a linear constraint on variables $X = \{x_1, \ldots, x_n\}$ and rational coefficients $A = \{a_1, \ldots, a_n\}$, $\sum_1^n a_i x_i - b \leq 0$, such that $b \in \mathbb{Q}$. As all variables in X are bounded, we know the maximal possible value for $\sum_1^n a_i x_i - b$, denoted by M, and the minimum value, denoted by m. Let r be a binary variable. The following decomposition represents the reification of c. If $M <= 0$: $r = 1$; if $m > 0$: $r = 0$; if $m \leq 0 < M$:

$$\sum_1^n a_i x_i - b - M(1-r) \leq 0 \text{ and } \sum_1^n a_i x_i - b - (m-1)r > 0.$$

Proof: If $r = 0$, $\sum_1^n a_i x_i - b - M \leq 0$ is always true, we must have $\sum_1^n a_i x_i - b > 0$. If $r = 1$, we must have $\sum_1^n a_i x_i - b \leq 0$, given that $\sum_1^n a_i x_i - b - m + 1 > 0$ is always true. ∎

Following an equivalent scheme, for rational and/or integer variables the reification of $\sum_1^n a_i x_i - b < 0$ is the following: $\sum_1^n a_i x_i - b - (M+1)(1-r) < 0$ and $\sum_1^n a_i x_i - b - mr \geq 0$.

From Proposition 1, we define a violation measure, equal to the number of violated inequalities in the decomposition.

Definition 4 (Violation measure and soft constraint): Let C be a global constraint defined by a conjunction of linear constraints, i.e., $C \Leftrightarrow c_1 \wedge \ldots \wedge c_k$. Let c_j^r be the reification of each c_j and r_j the truth variable (Proposition 1). We define a new violation measure expressed by an integer variable μ, equal to the number of truth variables equal to 0, i.e., the soft version $softC$ of C is decomposed as follows:

$$\forall j \in \{1, \ldots, k\}, c_j^r \text{ and } \mu = k - \sum_{j=1}^{k} r_j.$$

Compared with existing violation measures (see, e.g., [24], [25]), this measure has some links with the primal graph based violation measure [22], as it refers to a decomposition of the global constraint using more primitives ones. However, a main difference is that in most cases extra-variables are used in the linear decomposition, making the measure more universal. It can actually be used for all the most commonly used CP global constraints when we consider domain variables and their linear representation (see [4] for a survey of

existing decompositions). Furthermore, it is uniform, as all decompositions consist of a set of linear inequalities. This homogeneity should facilitate aggregation into an objective function. Normalization over a set of constraints can be based on the number of inequalities within each decomposition. This point is important because normalization of violation measure remains a main issue, notably in constraint based local search (CBLS). Contrary to the variable based measure [22], it does not suffer from pathological cases where it reduces to a $\{0, 1\}$ cost for global constraints. A perspective of our work is to investigate the use of violation measures based on Definition 4 in the context of CBLS. We think that determining the criteria for selecting the appropriate linear decomposition of each constraint cannot be done independently from applications.

Reified decompositions of global constraints can be derived from this measure.

Lemma 2 (Global constraint reification): We keep the notations of Definition 4. Let r be a binary variable. The reification of C is obtained by extending Definition 4 decomposition:

$$\mu - k(1 - r) \leq 0 \quad \text{and} \quad \mu + r > 0.$$

Proof: If $r = 0$ then $\mu \leq k$ is always true and we must have $\mu > 0$. If $r = 1$ then we must have $\mu = 0$ (from Definition 4, $\mu \geq 0$) and $\mu > -1$ is always true. ■

It is possible to simplify, for a specific case, the set of linear constraints obtained through this reification scheme, notably by skipping the use of a μ variable or by considering particular properties of the global constraint to be decomposed. For instance, the Gcc (Definition 2) reification can be stated with the following decomposition, that can be viewed as a specialization of the general scheme.

*Lemma 3 (*Gcc *reification):* We use Definition 2 and Lemma 1 notations. Let r_c be the binary variable for the truth value of the *Gcc* to be reified. We create $O(|t|)$ binary variables r_c^{k+}, mapped with values in t and $O(|t|)$ binary variables r_c^{k-}, also mapped. R is the set of all r_c^{k-} and all r_c^{k+} variables, of size $2|t|$. The reified *Gcc* is obtained by the domain decomposition of X. In addition, $\forall t[k] \in t$ state:

$$(\sum_{b \in B_{t[k]}} b) - \underline{t[k]} r_c^{k-} \geq 0. \quad (1)$$

$$(\sum_{b \in B_{t[k]}} b) - r_c^{k-}(n+1) \leq \underline{t[k]} - 1. \quad (2)$$

$$(\sum_{b \in B_{t[k]}} b) + (\overline{t[k]} + 1) r_c^{k+} \geq \overline{t[k]} + 1.$$

$$(\sum_{b \in B_{t[k]}} b) + n r_c^{k+} \leq \overline{t[k]} + n.$$

Moreover, state once:

$$(\sum_{r \in R} r) - r_c \leq 2|t| - 1. \quad (3) \qquad (\sum_{r \in R} r) - 2|t| r_c \geq 0. \quad (4)$$

Proof: We first consider the case of one value $t[k] \in t$. Without loss of generality we restrict to $\underline{t[k]}$ (the case of maximum occurence is symmetrical). If $r_c^{-k} = 0$ then *(1)* is always satisfied and *(2)* is satisfied if and only if $\sum_{b \in B_{t[k]}} \leq \underline{t[k]} - 1$, i.e., the minimum required number of occurrences of $\underline{t[k]}$ is not reached. If $r_c^{k-} = 1$ then *(2)* is always satisfied and *(1)* is satisfied if and only if $(\sum_{b \in B_{t[k]}} b) \geq \underline{t[k]}$. Then, consider all values. If $\sum_{r \in R} r = 2|t|$ then all lower and upper cardinalities are in the request bounds, $r_c = 1$ to satisfy constraint *(3)*. Otherwise, *(3)* is always satisfied. If $\sum_{r \in R} r < 2|t|$ then $r_c = 0$ to satisfy constraint *(4)*. Otherwise, *(4)* is always satisfied. The Lemma holds. ■

B. Decomposing Distance Constraints.

All constraints in this family are based on the Euclidian distance $abs = |x - y|$. We consider the general case where $\{abs, x, y\}$ are variables in $\mathbb{Q}^3$. We first need to decompose this distance constraint into a set of linear equations.

Lemma 4 ($abs = |x - y|$ *linearization):* We use the following notations: Given $\underline{x}, \overline{x}, \underline{y}$, and $\overline{y}$, a is the minimum possible value of $|x - y|$ and A the maximum possible value. d is the minimum possible value of $x - y$ and D the maximum possible value. We then distinguish three cases.

1) $D \leq 0$. Add $abs - y + x = 0$.
2) $d \geq 0$. Add $abs - x + y = 0$.
3) $d < 0 < D$. Add $a \leq abs$ and $abs \leq A$, define three variables dif, $difp$ and $difn$, a binary variable b and state:

$dif - x + y = 0.$ (1) $0 \leq difp$ and $difp \leq D.$ (2)

$0 \leq difn$ and $difn \leq |d|.$ (3) $d \leq dif$ and $dif \leq D.$ (4)

$dif - difp + difn = 0.$ (5) $difp - Db \leq 0.$ (6)

$difn + |d| b \leq |d|.$ (7) $abs - difp - difn = 0.$ (8)

Proof: Cases 1 and 2 are obvious. If $d < 0 < D$, constraints (2) (3) and (4) state the bounds of $difp$, $difn$ and dif. From (1), $dif = x - y$. (5) states $x - y = difp - difn$. From (2) and (3) $difp \geq 0$ and $difn \geq 0$. We have: either $difp > 0$, then from (6) $b = 1$, and from (7) $difn = 0$: from (8), $abs = x - y$; or $difn > 0$, from (7) $b = 0$, from (6) $difp = 0$: from (8), $abs = y - x$; otherwise, $abs = difp = difn = 0$. ■

We denote by DistanceXYZ(x, y, abs) the linearization of constraint $abs = |x - y|$ of Lemma 4. We can decompose the constraints in table I.

*Proposition 2 (*Deviation *linearization):* Let s and z be two rational variables and $A = \{abs_1, \ldots, abs_n\}$ be rational variables. Deviation can be represented by the following set of $O(n)$ linear constraints:

$$\forall x_i \in X, \texttt{DistanceXYZ}(x_i, z, abs_i).$$

In addition state once:

$$nz - (\sum_{i=1}^{n} x_i) = 0 \quad \text{and} \quad s - (\sum_{i=1}^{n} abs_i) = 0.$$

*Proposition 3 (*InterDistance *linearization):* Let p be a rational variable amd $X = \{x_1, \ldots, x_n\}$ be a set of rational variables. The decomposition requires $O(n^2)$ inequalities. $\forall x_i, x_j, i < j$, state:

$$\texttt{DistanceXYZ}(x_i, x_j, abs_{ij}) \quad \text{and} \quad abs_{ij} \geq p.$$

The case of `Diverse_sum` is similar to `InterDistance`. `Smooth` requires reification, as it counts the number of times constraint $|x_i - x_{i+1}| \geq c$ is satisfied.

Proposition 4 (`Smooth` *linearization):* Let cst be a positive integer, $X = \{x_1, \ldots, x_n\}$ be a set of rational variables and z be an integer variable. $\forall i \in \{1, \ldots, n\}$, let M_i be the maximum possible value of $cst - |x_i - x_{i+1}|$ and m_i the minimum possible value. In addition, we add $n-1$ binary variables $\{r_1, \ldots, r_n\}$. The decomposition requires $O(n)$ inequalities. $\forall x_i, i < n$, state:

If $m_i > 0$, $r_i = 0$. If $M_i \leq 0$, $r_i = 1$.

Otherwise:

$$\texttt{DistanceXYZ}(x_i, x_{i+1}, abs_i).$$

$$\text{and} \quad cst - abs_i - M_i(1 - r_i) <= 0.$$

$$\text{and} \quad cst - abs_i - (m_i - 1)r_i > 0.$$

In addition state once: $z = \sum_{i=1}^{n} r_i$.

C. Decomposing Occurrence Constraints.

Occurrence constraints generalized to rational variables (section III-C) are basically based on the count of the number of variables taking their value within an interval $[v, w)$.

Definition 5 (Occurrence): Let $X = \{x_1, \ldots, x_n\}$ be a set of variables of any type, z be an integer variable and $[v, w)$ an interval of rational values. $\texttt{Occ}(X, z, [v, w))$ holds if and only if $z = |\{i : x_i \in [v, w)\}|$.

We first need to decompose this constraint.

Lemma 5 (Occurrence linearization): We use the notations of Definition 5. $\forall x_i \in X$, $X = \{x_1, \ldots, x_n\}$, state three binary variables $r_i, r_i^<$, and $r_i^\geq$. Let m_i be the minimum possible value of x_i and M_i the maximum possible value. Let $m_i^<$ be the minimum possible value of $x_i - v$ and $M_i^<$ the maximum possible value. Let $m_i^\geq$ be the minimum possible value of $w - x_i$ and $M_i^\geq$ the maximum possible value. $\texttt{Occ}(X, z, [v, w))$ can be decomposed by stating $\forall x_i \in X$:

If $M_i < v \vee m_i \geq w$ then $r_i = 0$

If $M_i < w \wedge m_i \geq v$ then $r_i = 1$.

Otherwise:

$$x_i - v - (M_i^< + 1)(1 - r_i^<) < 0 \text{ and } x_i - v - m_i^< r_i^< \geq 0.$$

$$w - x_i - M_i^\geq(1 - r_i^\geq) \leq 0 \text{ and } w - x_i - (m_i^\geq - 1)r_i^\geq > 0.$$

$$r_i \leq (1 - r_i^<) \text{ and } r_i \leq (1 - r_i^\geq) \text{ and } r_i + r_i^< + r_i^\geq \geq 1.$$

In addition, state once: $z = \sum_{i=1}^{n} r_i$.

Proof: The cases $M_i < v \vee m_i \geq w$ and $M_i < w \wedge m_i \geq v$ are straightforward. Otherwise, from proposition 1 $r_i^<$ reifies the constraint $x_i < v$, and $r_i^\geq$ reifies the constraint $x_i \geq w$. If either $r_i^< = 1$ or $r_i^\geq = 1$ then $r_i \leq 0$ and $r_i + r_i^\geq + r_i^< \geq 1$ is satisfied. If none of the two are equal to 1, then we have $x_i \geq v \wedge x_i < w$. $r_i^< = r_i^\geq = 0$. $r_i \leq (1 - r_i^<)$ is $r_i \leq 1$, $r_i \leq (1 - r_i^\geq)$ is $r_i \leq 1$ and to satisfy $r_i + r_i^< + r_i^\geq \geq 1$, $r_i = 1$. Each r_i is the truth value of $x_i \in [v, w)$. $\sum_{i=1}^{n} r_i$ is equal to the number of values in $[v.w)$, equal to z. ∎

We deduce from Lemma 5 a linear formulation of the constraint of Definition 3 of Section III-C.

Proposition 5 (`RangeGcc` *linearization):* We use the notations of Deinifion 3. Let m be the size of array t. The decomposition requires $O(n \times m)$ inequalities. $\forall t[k] \in t$, state the linear decomposition of $\texttt{Occ}(X, z_k, t[k])$, where z_k is the integer variable equal to the number of variables in X taking a value in $\underline{t[k]}$. In addition, $\forall t[k] \in t$, state: $\underline{t[k]} - z_k \leq 0$ and $z_k - \underline{t[k]} \leq 0$.

When used with integer mathematical variables, thanks to Proposition 5 a `Gcc` constraint can be added to a model by stating t as an array of intervals reduced to one integer value $t[k] = v$. `Alldifferent` can be derived from this scheme by considering $\forall k \in \{1, \ldots, m\}, \underline{t[k]} = 0$ and $\underline{t[k]} = 1$. These decompositions are alternatives to Lemma 1, but they do not require a domain decomposition.

V. Implementation and Experiments

We used an Intel-I7 processor and 8GB of RAM and the MILP commercial solver Gurobi 7.0 [26].

A. Modeler implementation.

We implemented a Java modeler, MICE, for solving CP models using any MILP engine, providing that this engine can be called from Java. We argue that using a standard language makes sense, given our objective of simplicity of use. We underline three aspects.

1) Modularity:

Our prototype provides its own API for stating linear constraints. Therefore, all predefined constraints as well as user constraints are stated using MICE objects. To plug a new MILP solver, the main class, called `Solver`, should be augmented by a call to the method creating a new model in the MILP solver. In addition, a unique new class must be created. This class implements an interface that states the methods used to create mathematical variables and linear constraints within the MILP engine (this makes the link with MICE linear expressions), to call specific methods for running the MILP solver, limiting time, and some getters (value of a mathematical variable, solver statistics, etc.).

2) Constraints on integer domain variables:

Our prototype provides an API for integer domain variables and a wide range of state-of-the-art global constraint linear formulations, such as `Alldifferent`, `Gcc`, `Element`, positive tables (i.e., defined by the set of allowed combinations of values), sum and scalar product, arithmetic operators. In addition, it provides modeling facilities such as negative tables, constraints $z = x^k$, $z = |x - y|$, $z = x \times y$, represented by tables, reified `Gcc` and reified tables. We do not detail all these features, except the decompositions of negative tables and table constraint reification based on Refalo's domain representation, which are, as far as we know, new.

Lemma 6 (Negative table linearization): We use the notations of Definiton 1. Let c be a constraint defined on $var(c) \subseteq X$ by a set $T = \{\tau_1, \ldots, \tau_{|T|}\}$ of forbidden tuples. Value of variable x_i in tuple τ_k is denoted by $\tau_k[x_i]$. c can be represented by the domain decomposition of X and $O(|T|)$ inequalities. $\forall \tau_k \in T$ state: $(\sum_{i, v_j \in d_i : v_j = \tau_k[x_i]} b_{ij}) \leq |var(c)| - 1$.

Proof: For c to be satisfied, $\forall \tau_k \in T$ at least one variable $x_i \in var(c)$ should take a value $v_j \neq \tau_k[x_i]$. From Definiton 1, if $x_i = v_j$ then $b_{ij} = 1$, otherwise $b_{ij} = 0$. $\sum_{i,v_j \in d_i : v_j = \tau_k[x_i]} b_{ij}$ must be $< |var(c)|$. ■

Lemma 7 (Table constraint reification): We use the notations of Definiton 1. We introduce a set of binary variables B_T, one-to-one mapped with tuples in $T = \{\tau_1, \ldots, \tau_{|T|}\}$. The set T can either represent allowed or forbidden tuples. Value of variable x_i in tuple τ_k is denoted by $\tau_k[x_i]$. The reification of a table constraint c can be represented by the domain decomposition of X, the binary variable r_c used to express the truth value of c and $O(|T|)$ inequalities.
(1) $\forall \tau_k \in T$ and the corresponding variable $b_{\tau_k} \in B_T$ state:

$$|var(c)|b_{\tau_k} - (\sum_{i,v_j \in d_i : v_j = \tau_k[x_i]} b_{ij}) \leq 0,$$

and:

$$(\sum_{i,v_j \in d_i : v_j = \tau_k[x_i]} b_{ij}) - |var(c)|b_{\tau_k} \leq |var(c)| - 1.$$

(2) In addition, if tuples in T are allowed tuples state:

$$(\sum_{k \in \{1,\ldots,|T|\}} b_{\tau_k}) - r_c = 0.$$

otherwise state:

$$(\sum_{k \in \{1,\ldots,|T|\}} b_{\tau_k}) + r_c = 1.$$

Proof: $\forall \tau_k \in T$, inequality (1) ensures that if $b_{\tau_k} = 1$ then all $x_i \in var(c)$ take the value $\tau_k[x_i]$. If $b_{\tau_k} = 0$ it is always satisfied. The second one ensures that if $b_{\tau_k} = 0$ not all the variables $x_i \in var(c)$ take the value $\tau_k[x_i]$. If $b_{\tau_k} = 1$ it is always satisfied. From Definiton 1, $\sum_{k \in \{1,\ldots,|T|\}} b_{\tau_k} \leq 1$. From *(2)*, $r_c = 1 \Leftrightarrow$ the variables in $var(c)$ are fixed with a tuple of T (positive table) or not in T (negative table). ■

The main advantage of Lemma 7 is that although all forbidden and allowed combinations of values should be considered, they are not both stated explicitly: Only either allowed or forbidden tuples are used. In many cases this obviously makes a huge difference in terms of size.

To design a complete CP modeler, we need an API for stating constraints based on logical operators. Logical constraints are then themselves set through tables on reification variables. Given any two reified constraints c_1 and c_2, `ReifOR` is a table on the truth variables r_1 and r_2 and a new variable r: the allowed tuples on $\{r_1, r_2, r\}$ are $\{(0,0,0),(0,1,1),(1,0,1),(1,1,1)\}$. Stating `OR` is then $r = 1$. The cases of `AND`, $\neg$ and $\rightarrow$ (implication) are similar. There is no limitation about the number of logical operator combinations on reified constraints.

3) Discrete-continuous use:
The decompositions on rational variables presented in this paper can be included to a model using our prototype, and mixed with other constraints on integer variables. Recall that constraints on rational variables have been designed to be suited to integer variables as well.

B. Comparative performance assessment.

Although we implicitly compare CP and MILP on the same models, obviously our goal is not to design a winner. Using the best existing models, some problems are best solved by CP and some other are best solved by MILP solvers. In the context of a "model and run" approach (i.e., models are exclusively stated by variables and high level constraints), where linear models are automatically derived from CP models, our objective is rather to provide examples. For each example, if the price to pay for this simplicity of use[1] is not prohibitive concerning solving efficacy, a proof of concept is made. This shows that such an approach can be an interesting alternative for tackling discrete-continuous problems, using both domain, mathematical, binary, integer and/or rational variables. Observe that nothing prevents the user to further improve her model by replacing part of the generic decompositions by an ad hoc linear formulation. Similarly, most CP solvers offer an API to design user's own global constraints, search and propagators.

1) Comparison with hybrid models:
In a first experiment, we compared:

- MICE coupled with Gurobi.
- Hybridization of discrete and continuous CP solvers.

We experimented the Santa Claus discrete-continuous problem [1], where the goal is to minimize the average deviation of gift prices, in order to fairly satisfy kids. This problem was initially shown to justify the need of a bridge between two solvers: a "classical" CP solver with a continuous CP solver.

nb. kids	max. price	nb. gifts	MICE	Choco-Ibex
3	25	5	0 sec.	0 sec.
6	50	10	0 sec.	0.8 sec.
9	75	15	0.3 sec.	371.6 sec.
12	100	20	32.3 sec.	$>$ 600 sec.
15	125	25	341.5 sec.	$>$ 600 sec.

TABLE II
MICE WITH GUROBI VS CHOCO-IBEX.

We used Choco 3.3.3 [10] and Ibex 2.3.1 [27], like in the original experiments performed on this problem [1]. The CP model also stems from this paper. We use the same model with our prototype (except that no search strategy is stated). Building time is negligible. This model embeds `Alldifferent` and `Element` on integer variables and `Deviation` with a continuous variable for the mean. Table II shows time for proving the optimal solution, with a 10 minutes limit. We generated random instances with all-distinct prices for available gifts, according to a maximum price (minimum is \$1), number of kids and total number of available gifts. Table II shows some benefits, despite the simplicity of modeling such a discrete-continuous model using a single API. While this example shows the possible advantages of

[1]Benefiting from the default parameters of modern MILP solvers that make the solving process quite robust, an advantage of this technique in comparison with classical CP solvers is to skip the search strategy definition in models. A similar comment can be made with respect to implicit constraints set in CP models to improve propagation, or concerning symmetry breaking constraints.

our approach, recall that a limitation is that not all global constraints can be (easily, reasonably) represented, contrary to a system like Ibex.

2) Domain versus mathematical integer variables: Concerning models stated on integer domain variables, recent works have shown competitive results on optimization CP benchmarks [4].

Series (10 instances)	MICE optimal proofs	MICE av. obj. value	MICE av. time (sec.)	CP optimal proofs	CP av. obj. value	CP av. time (sec.)
bqp50	100%	1926.8	60.1	60%	1903.9	447.3
g05_20	100%	64.9	1.9	100 %	64.9	3.6
g05_30	100%	138.8	40.1	0 %	137.1	600
g05_40	40%	244.9 (3.2%)	549.1	0 %	228.2	600

TABLE III
MICE WITH GUROBI VS CP: MAXIMIZATION PROBLEMS.

Table III reports complementary results about the CP-style modeling features of MICE to solve the Max-Cut non-linear (quadratic) optimization problem that occurs in statistical physics applications [28], without any handcrafted model transformation. Given an undirected graph with weighted edges $G_w = (V, E_w)$, Max-Cut is the problem of finding a cut in G of maximum weight. i.e., split the vertex set in two so that to have a maximum weighted set S of edges that have an end point in each set. A variable x_i is stated for each vertex in V. $x_i = 1$ if vertex v_i is in S and $x_i = -1$ otherwise. We maximize $\frac{1}{2}\sum_{i<j} w_{ij}(1 - x_i x_j)$. We encoded this problem in Choco 3.3.3 and MICE coupled with Gurobi. Using our prototype the default product constraint corresponds to a positive table. In CP, the best strategy we found is *DomOverWdeg* [29], and first assign vertices. We used Beasley instances from the OR-Libary and problems from g05_60 Rudy instances[2], with a 10 min. time-limit. Graphs have respectively 50 nodes and weights in $[-100, 100]$, and 30-50 nodes unweighted. Building time is not significant.

MICE can also provide competitive results for satisfaction benchmarks on integer domain variables, providing that it is connected to a modern MILP solver, such as Gurobi. For instance, several benchmarks that use `Alldifferent` are as easily solved (most in less than 1 second) by MICE, as they are using the models in the sample directories of Choco 3.3.3 [10] and OR-Tools [30] (e.g., Sudoku, prob03, prob07, prob19 from CSPLib [3]). Furthermore, if in the CP model we replace ad hoc search strategies by the default search strategy, to make the comparison exactly on the same model, the CP solvers are not always able to solve the instances in a comparable time. The CP best strategy for CSPLib prob19-7, for instance, is impact-based search (IBS) with specific parameters that would not be understandable by non-expert practicers. Using IBS, the problem is solved in 6.4 seconds (MICE+Gurobi solves it in 3.5 sec). If we use the default strategy, no solution is found by the CP solver is one minute (both using Choco and OR-Tools).

[2]See http://biqmac.uni-klu.ac.at/biqmaclib.html
[3]http://www.csplib.org/

n	Model A		Model B		Model C	
	Build	Solve (sec.)	build	Solve (sec.)	Build	Solve (sec.)
14	14.8	0.2	0.3	7.1	0.1	0.2
16	35.2	0.3	0.2	13.2	0.1	0.5
18	72.1	0.6	0.8	0.6	0.1	5
20	141.7	1	0.9	> 600	0.1	5.1

TABLE IV
BUILDING AND SOLVING TIME IN SECONDS.

There are, however, issues inherent to the use of a domain representation. The main one is that new binary variables are created for decomposing the domain constraint of each variable (Definition 1). This observation was made in [4], where authors suggest a refined domain representation. If domains are too large or too many linear constraints are stated on those binary variables, the MILP model can become huge even for small instances. Then, in addition to the consequences concerning solving time, even generating the linear equations can take a while. To illustrate this issue, consider the All-interval series problem. It is worth noticing that recent CP solvers, such as Choco 3.3.3, easily solve this problem (in a solving time close to 0 second).

Given $n \in \mathbb{N}$, the problem is to find a vector $s = (s_1, \ldots, s_n)$ such that s is a permutation of $\mathbb{Z}_n = 0, 1, \ldots, n-1$ and the interval vector $v = (|s_2 - s_1|, |s_3 - s_2|, \ldots, |s_n - s_{n-1}|$ is a permutation of $\mathbb{Z}_n \setminus \{0\} = 1, \ldots, n-1$. The model uses `Alldifferent` and `DistanceXYZ` constraints. With integer domain variables, if all constraints are linearized from the binary variables used to decompose domains, `DistanceXYZ` can be encoded through a positive ternary table (model A). Such a table may quickly become big and slow to be generated. It is possible to state a model exclusively on integer mathematical variables, using Proposition 5 for `Alldifferent` and Lemma 4 for `DistanceXYZ` (model B). A drawback is that we do not use the state-of-the-art concise and convex hull representation of `Alldifferent` [3]. The good compromise, with Gurobi, is to state a model on integer domain variables and to use Lemma 4 for `DistanceXYZ` (model C). Even if this observation is simple, it underlines that in some cases selecting the appropriate representation can be important, which can be considered as an argument against simplicity of use. However, the good news are that selection criteria should be problem independent (mainly the number of variables and linear constraints generated), making feasible the perspective of dealing with this issue in a generic way. Table IV shows the model building and solving times of the three models.

VI. CONCLUSION AND PERSPECTIVES

In this paper, we considered constraints originally stated using integer domain variables, for which we derived new definitions using rational variables and linear decompositions. We introduced a generic scheme for reification and softening. Throughout this study, we implemented a prototype that provides an API comparable to existing CP solvers concerning

integer variables, including global constraints and logical operators, plus global constraints on rational variables. Benefiting from the "model and run" features of Gurobi, plugged to our modeler, we provided examples showing that our approach can be an alternative that is noticeably simple to use. We are arguing that simplicity of use is important for democratizing the CP-style modeling. Stating only variables and constraints is simpler than selecting propagators, designing hybrid models and/or dedicated search strategies. The choice of a standard language for the prototype is also motivated by this context of use. Importing a library is probably simpler than learning a new language; integration into broader projects is easier. Most importantly, some practitioners or students may wish to quickly learn about CP modeling before deciding whether to adopt this technology or not at the final stage.

From a technical point of view, we introduced new decompositions, both for domain and mathematical variables: the `Gcc` with fixed occurrence intervals, reified `Gcc`, `DistanceXYZ`, `Deviation`, `InterDistance`, `Diverse_sum`, `Smooth`, `RangeGcc`, `Among`, negative tables and reified tables. Future work includes proposing alternative models for these constraints as well as extending the approach to other constraint families.

Concerning the new violation measure, we came up with a definition that solves the issue of uniformity while it does not have expressivity drawbacks of the variable based cost. A perspective in CP and CBLS is to determine, from a series of practical problems, which are the criteria for selecting the best decomposition of a constraint, when several alternatives exist. Using a CP solver, this may require to design propagators dedicated to the subproblem stated by each decomposition.

REFERENCES

[1] J.-G. Fages, G. Chabert, and C. Prud'homme, "Combining finite and continuous solvers," *CoRR*, vol. abs/1402.1361, 2014. [Online]. Available: http://arxiv.org/abs/1402.1361

[2] R. Rodosek, M. Wallace, and M. Hajian, "A new approach to integrating mixed integer programming and constraint logic programming," *Annals of Operational Research. Recent Advances in Comb. Optimization*, 1997.

[3] P. Refalo, "Linear formulation of constraint programming models and hybrid solvers," in *Principles and Practice of Constraint Programming - CP 2000, 6th International Conference, Singapore, September 18-21, 2000, Proc.*, 2000, pp. 369–383.

[4] G. Belov, P. J. Stuckey, G. Tack, and M. Wallace, "Improved linearization of constraint programming models," in *Principles and Practice of Constraint Programming - CP 2016, 22nd International Conference, CP 2016, Toulouse, France, September 5 - 9, 2016, Proc.*, vol. 9392, 2016.

[5] P. Schaus, Y. Deville, P. Dupont, and J.-C. Régin, "The deviation constraint," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR 2007, Brussels, Belgium, May 23-26, 2007, Proc.*, 2007, pp. 260–274.

[6] P. Schaus, P. V. Hentenryck, and J.-C. Régin, "Scalable load balancing in nurse to patient assignment problems," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR 2009, Pittsburgh, PA, USA, May 27-31, 2009, Proc.*, ser. Lecture Notes in Computer Science, W.-J. van Hoeve and J. N. Hooker, Eds., vol. 5547. Springer, 2009, pp. 248–262.

[7] J.-F. Puget, "Constraint programming next challenge: Simplicity of use," in *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proc.*, 2004, pp. 5–8.

[8] M. Milano and P. V. Hentenryck, *Hybrid Optimization - The Ten Years of CPAIOR*, ser. Springer Optimization and its Applications. Springer, 2011, vol. 45.

[9] J. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Computational Management Science*, vol. 2, no. 1, pp. 3–19, 2005. [Online]. Available: http://EconPapers.repec.org/RePEc:spr:comgts:v:2:y:2005:i:1:p:3-19

[10] C. Prud'homme, J.-G. Fages, and X. Lorca, *Choco3 Documentation*, TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2014. [Online]. Available: http://www.choco-solver.org

[11] T. Petit and A. C. Trapp, "Finding diverse solutions of high quality to constraint optimization problems," in *Proc. of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 2015, pp. 260–267.

[12] E. Hebrard, B. Hnich, B. O'Sullivan, and T. Walsh, "Finding diverse and similar solutions in constraint programming," in *Proc., The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, 2005, pp. 372–377.

[13] J.-C. Régin, "The global minimum distance constraint," ILOG (IBM), Tech. Rep., 01 1997.

[14] C.-G. Quimper, A. López-Ortiz, and G. Pesant, "A quadratic propagator for the inter-distance constraint," in *Proc., The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, 2006, pp. 123–128.

[15] N. Beldiceanu, M. Carlsson, and J.-X. Rampon, "Global Constraint Catalog, 2nd Ed." SICS, Tech. Rep. T2010-07, 2010.

[16] J.-C. Régin, "Generalized arc consistency for global cardinality constraint," in *Proc. of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, August 4-8, 1996, Volume 1.*, 1996, pp. 209–215.

[17] I. P. Gent, K. Stergiou, and T. Walsh, "Decomposable constraints," *Artif. Intell.*, vol. 123, no. 1-2, pp. 133–156, 2000.

[18] E. Hebrard, B. O'Sullivan, and T. Walsh, "Distance constraints in constraint satisfaction," in *IJCAI 2007, Proc. of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, 2007, pp. 106–111.

[19] N. Beldiceanu and E. Contejean, "Introducing global constraints in CHIP," *Journal of Mathematical and Computer Modelling*, vol. 20(12), pp. 97–123, 1994.

[20] C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan, and T. Walsh, "Among, common and disjoint constraints," in *Recent Advances in Constraints, Joint ERCIM/CoLogNET Int. Workshop on Constraint Solving and Constraint Logic Programming, CSCLP 2005, Uppsala, Sweden, June 20-22, 2005, Revised Selected and Invited Papers*, 2005, pp. 29–43.

[21] T. Petit and J.-C. Régin, "The ordered distribute constraint," *International Journal on Art. Intelligence Tools*, vol. 20, no. 4, pp. 617–637, 2011.

[22] T. Petit, J.-C. Régin, and C. Bessière, "Specific filtering algorithms for over-constrained problems," in *Principles and Practice of Constraint Programming - CP 2001, 7th International Conference, CP 2001, Paphos, Cyprus, 2001, Proc.*, 2001, pp. 451–463.

[23] A. Koster, "Frequency assignment: models and algorithms," Ph.D. dissertation, Maastricht University, 1999.

[24] N. Beldiceanu and T. Petit, "Cost evaluation of soft global constraints," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR 2004, Nice, France, April 20-22, 2004, Proc.*, 2004, pp. 80–95.

[25] W.-J. van Hoeve, G. Pesant, and L.-M. Rousseau, "On global warming: Flow-based soft global constraints," *J. Heuristics*, vol. 12, no. 4-5, pp. 347–373, 2006.

[26] Gurobi, *Gurobi 6.5 Manual (`www.gurobi.com`)*, 2016.

[27] G. Chabert, B. Neveu, J. Ninin, L. Jaulin, and G. Trombettoni, *Ibex*, Mines Nantes, ENPC Paris, ENSTA Brest, Univ. Montpellier II, 2016. [Online]. Available: http://www.ibex-lib.org

[28] F. Liers, "Contributions to determining exact ground-states of ising spin-glasses and to their physics," Ph.D. dissertation, Univ. zu Koln, 2004.

[29] F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais, "Boosting systematic search by weighting constraints," in *Proc. of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, Valencia, Spain, August 22-27, 2004*, 2004, pp. 146–150.

[30] N. van Omme, L. Perron, and V. Furnon, "*OR-Tools users manual*," Google, Tech. Rep., 2014.