Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Finding Diverse Solutions of High Quality to Constraint Optimization Problems

Thierry Petit
Foisie School of Business,
Worcester Polytechnic Institute, USA
Mines de Nantes / LINA, France
tpetit@{wpi.edu,mines-nantes.fr}

Andrew C. Trapp
Foisie School of Business,
Worcester Polytechnic Institute, USA
atrapp@wpi.edu

Abstract

A number of effective techniques for constraint-based optimization can be used to generate either diverse or high-quality solutions independently, but no framework is devoted to accomplish both simultaneously. In this paper, we tackle this issue with a generic paradigm that can be implemented in most existing solvers. We show that our technique can be specialized to produce diverse solutions of high quality in the context of over-constrained problems. Furthermore, our paradigm allows us to consider diversity from a different point of view, based on generic concepts expressed by global constraints.

1 Introduction

Most existing techniques for solving optimization problems in Artificial Intelligence naturally give the priority to the value of an objective function. While this is essential from the optimality point of view, there are scenarios where it may be advantageous to consider multiple solutions. In Constraint Programming (CP), a number of effective methods can be used to generate multiple solutions that are significantly diverse, e.g., [Hebrard *et al.*, 2005; 2007]. However, these techniques deal with satisfaction problems and are not concerned by solution quality, i.e., the value of the objective function of each diverse solution. On the other hand, constraint solvers routinely generate a single solution to optimization problems, ideally optimal. In this paper, we suggest a generic constraint-based paradigm for assessing both solution diversity and solution quality, which can take as argument any constraint optimization problem. Motivated by practical applications, we propose to consider several diversity norms and to use them in conjunction with a quality constraint. In this way, we provide a set of solutions that optimize a compromise between solution diversity and quality. Such a tradeoff is especially useful, for instance, in personnel scheduling (when not all user preferences are known in advance), for routing problems (alternative paths are useful in case of unexpected events), and in configuration (a good quality product can be made using diverse sets of components). As there is no requirement for the initial solution to be optimal, there is therefore no theoretical restriction on solving large problems, using for instance Large Neighborhood Search. Our approach can easily be implemented in most existing solvers. Moreover, our paradigm can be specialized to produce diverse solutions of high quality in the context of over-constrained problems. A second contribution of our paper is to consider diversity from a different point of view, based on global constraints. The intuitive idea is to propose to the end-user solutions that correspond to some generic concepts, such as concentration or fair distribution of high values, deviation, etc. We show that our paradigm is naturally well-suited to use these new notions, extending the state-of-the art in CP also for satisfaction problems.

2 Background

Constraint Programming. A *constraint network* $\mathcal{N}$ consists of a sequence of problem variables $\mathcal{X} = (x_1, \dots, x_n)$, a sequence of domains $D = \{D(x_1), D(x_2), \dots, D(x_n)\}$, where the domain $D(x_i)$ is the finite set of at most d values that variable x_i can take (we also use the notation D_i), and a set $\mathcal{C}$ of constraints that specify the allowed combinations of values for given subsets of variables. A *relation* R of arity k is any set of tuples $\langle a_1, a_2, \dots, a_k \rangle$ where all a_j's are values from the given universe U. A *constraint* of arity k is a pair $C = (\mathcal{X}, R)$, where $\mathcal{X}$ is a sequence of k variables and R is a relation of arity k. An *assignment* t is a function from $\mathcal{X}$ to the universe of values U. $t[x]$ is the value of x in t. An assignment t to a sequence $\mathcal{X} = (x_1, \dots, x_k)$ of variables, or to a superset of $\mathcal{X}$, *satisfies* $C = (\mathcal{X}, R)$ iff $\langle t[x_1], \dots, t[x_k] \rangle \in R$. Otherwise, t *violates* C. A *solution* of $\mathcal{N} = (\mathcal{X}, D, \mathcal{C})$ is an assignment t to $\mathcal{X}$ such that $t[x_i] \in D_i$ for all $x_i \in \mathcal{X}$, and t satisfies all the constraints in $\mathcal{C}$. In a *Constraint Optimization Problem* (COP), a variable $obj \in \mathcal{X}$ represents the objective value. Without loss of generality, consider the minimization COP. An *optimal solution* is then a solution t^* of the constraint network such that no other solution t exists with $t[obj] < t^*[obj]$. An alternative way to represent optimization problems is to extend CP, using Semi-ring CSPs/Cost Functional Networks [Bistarelli *et al.*, 1999]. In such frameworks, conversely to classical COP, costs involved in the objective function are not encoded by problem variables. The two alternatives were successfully used for solving real-world problems (see for example [Zampelli *et al.*, 2013; Allouche *et al.*, 2014]). As our contributions in sections 5 and 6 require to set *hard* constraints on cost variables, in this paper we deal with classical COP.

Solution Quality. Solution *quality* [Trapp and Konrad, 2015], also called *performance* [Billaut *et al.*, 2010], aims to measure the distance between the value $t[obj]$ in a given solution t of a COP and the objective value $t^*[obj]$ of a "good" solution t^*, ideally an optimal solution. Given the objective variable obj, the relative deterioration of quality of solution t can be quantified using the following function: $Q(\mathcal{X}, t, t^*) = \max(0, t[obj] - t^*[obj])$.

Solution Diversity. Although in CP solution diversity received less exposure than solution quality, some significant contributions have been made in the last decade. Hebrard et al. [Hebrard *et al.*, 2005] distinguish *offline diversity*, computed given the whole set of solutions, from *online diversity* where solutions are computed incrementally. In this paper we will consider online diversity, which is intuitive for COP.

1) Hamming distance. The most usual diversity measure is the sum of Hamming distances [Hamming, 1950].

Definition 1 (Hamming Distance $\delta_{\mathcal{H}}$) *Given two assignments t_i and t_j of n variables $\mathcal{X}$ and $x_l \in \mathcal{X}$,*

- $\delta_{\mathcal{H}}(t_i[x_l], t_j[x_l]) = 0$ *iff* $t_i[x_l] = t_j[x_l]$, 1 *otherwise.*
- $\delta_{\mathcal{H}}(\mathcal{X}, t_i, t_j) = \sum_{l=1}^{n} \delta_{\mathcal{H}}(t_i[x_l], t_j[x_l])$.

The MAXDIVERSE*kSet* problem [Hebrard *et al.*, 2005] consists of finding a set of solutions $\{t_1, t_2, \ldots, t_k\}$ maximizing $\sum_{1 \le i < j \le k} \delta_{\mathcal{H}}(\mathcal{X}, t_i, t_j)$. Heuristic approaches [Hebrard *et al.*, 2005; Hentenryck *et al.*, 2009] and knowledge compilation techniques [Hadzic *et al.*, 2009] have been used to solve this problem.

2) Alternative Distance Functions. Depending on the problem and the solving technique, Hamming distances may not be appropriate. For instance, consider a scheduling problem with a horizon of one day and a time unit of one minute. Given a solution, delaying all activities by one minute leads to a diverse solution according to Hamming distances, yet in practice the two solutions are quite similar. Conversely, if we swap the k first and k last activities, even with small values of k the end-user may consider that the new solution significantly differs from the initial one. Following the work of Hebard et al., this observation was made in the context of knowledge compilation [Hadzic *et al.*, 2009] and Answer-set programming [Eiter *et al.*, 2013]. To address this issue in CP, we consider measures based on the L_1 and L_2 norms.

Definition 2 (Manhattan (δ_{L_1}), Euclidian (δ_{L_2}) distances) *Given two assignments t_i and t_j of n variables $\mathcal{X}$ and $x_l \in \mathcal{X}$,*

- $\delta_{L_1}(t_i[x_l], t_j[x_l]) = |t_i[x_l] - t_j[x_l]|$.
  $\delta_{L_1}(\mathcal{X}, t_i, t_j) = \sum_{l=1}^{n} \delta_{L_1}(t_i[x_l], t_j[x_l])$.
- $\delta_{L_2}(t_i[x_l], t_j[x_l]) = (t_i[x_l] - t_j[x_l])^2$.
  $\delta_{L_2}(\mathcal{X}, t_i, t_j) = \sqrt{\sum_{l=1}^{n} \delta_{L_2}(t_i[x_l], t_j[x_l])}$.

3 Assessing Both Diversity and Quality

Simultaneously addressing both diversity and quality requires two objective functions, Q and δ, $\delta \in \{\delta_{\mathcal{H}}, \delta_{L_1}, \delta_{L_2}\}$. As the MAXDIVERSE*kSet* problem is $F^{NP[log(n)]}$-complete [Hebrard *et al.*, 2005] without regard to quality, we introduce a pragmatic technique for dealing with the two notions at the same time. Just as importantly, we propose a framework that can be easily implemented using reference constraint programming solvers. Following a principle recently suggested for MIP [Trapp and Konrad, 2015], a ratio can be defined to balance between solution quality and diversity. Starting from a "good" solution t^*, i.e., with an objective value that satisfies the end-user, we consider the following ratio.

$$robj = \frac{\delta}{Q} = \frac{\text{Relative Solution Diversity}}{\text{Relative Loss in Quality}}.$$

We assume in this section that the problem has at least one feasible solution, and subsequently investigate the specific case of over-constrained problems. The goal is to find the set of optimal, or at least high-quality, solutions of a new problem derived from the original one. The new problem maximizes the ratio starting from a solution t^*. Unlike the MIP paradigm proposed by Trapp and Konrad, however, the solution t^* may be suboptimal. The first step is to express the relative solution diversity and relative loss of quality according to solution t^* within the CP framework, in order to obtain a second (diverse) solution t_2. Then, iteratively, we will compute the next solution according to the solution set $\mathcal{S} = \{t_1 = t^*, t_2\}$ to obtain a third solution t_3, and so on.

Relative Solution Diversity. This part of our methodology does not fundamentally differ from the constraint satisfaction case. We use a new problem variable x_δ for storing the minimum allowed diversity threshold. In the case of Hamming Distances, we use the DIVERSESUM$_{\mathcal{H}}$ constraint [Hebrard *et al.*, 2005]. This constraint takes as argument a set of problem variables, a set of previously computed solutions and the variable x_δ. In its initial formulation [Hebrard *et al.*, 2005], x_δ is an integer, but we aim to be as generic as possible in the modeling description.

Definition 3 DIVERSESUM$_{\mathcal{H}}(\mathcal{X}, \mathcal{S}, x_\delta) \Leftrightarrow$
$(\sum_{i=1}^{n} \sum_{j=1}^{|\mathcal{S}|} x_i \neq t_j[x_i]) \ge x_\delta$.

We now define similar constraints for L_1 and L_2 norms.

Definition 4 DIVERSESUM$_{L_1}(\mathcal{X}, \mathcal{S}, x_\delta) \Leftrightarrow$
$(\sum_{i=1}^{n} \sum_{j=1}^{|\mathcal{S}|} |x_i - t_j[x_i]|) \ge x_\delta$.

Definition 5 DIVERSESUM$_{L_2}(\mathcal{X}, \mathcal{S}, x_\delta) \Leftrightarrow$
$(\sum_{j=1}^{|\mathcal{S}|} \sqrt{(\sum_{i=1}^{i=n} |x_i - t_j[x_i]|^2)} \ge x_\delta$.

Some alternative definitions of distance constraints exist for Hamming distances and metrics that are component-wise decomposable like Hamming distances [Hebrard *et al.*, 2005; 2007; 2011]. These feature idempotent operators (minimum, maximum) instead of a sum, or are dedicated to finding sets of similar solutions to satisfaction problems rather than diverse solutions.

Relative Loss of Quality. The relative deterioration in objective function quality can be expressed using a constraint QUALITYSUM(obj, t^*, x_Q), where variable x_Q expresses the loss of quality with respect to the first solution t^*.

Definition 6 QUALITYSUM$(obj, t^*, x_Q) \Leftrightarrow$
$\max(0, obj - t^*[obj]) = x_Q$.

A New Maximization Problem. We introduce a new maximization problem for simultaneously assessing both diversity and quality. We aim to maximize the ratio $\frac{x_\delta}{x_Q+1}$ in each new produced solution. Our paradigm is based on an iterative scheme, summarized by Algorithm 1. Given a variable x, we use the notation $\underline{x}$ (respectively $\overline{x}$) for the minimum value (respectively the maximum value) in its domain.

Algorithm 1 takes as argument a COP encoded with the constraint network $\mathcal{N}$ and an integer k, as well as the two variables x_δ and x_Q for defining the allowed range of values for the ratio numerator and denominator. The algorithm returns a solution set $\mathcal{S}$ of size at most k, maximizing the balance between relative diversity and quality with respect to solutions in $\mathcal{S}$. The initial values $\underline{x_\delta}$ and $\overline{x_Q}$ provide the user preferences in terms of minimum acceptable diversity value between two solutions and maximum loss of quality between two solutions. The initial maximum diversity $\overline{x_\delta}$ can be easily estimated for each metric, e.g., for the L_1 norm an upper-bound is the sum of domain sizes, which is convenient in our context. As several optimal solutions may exist, we initially set $\underline{x_Q} = 0$.

There are several ways to implement the objective constraint in the solving scheme described by Algorithm 1. Therefore, we do not directly express the ratio as stated. We use an objective constraint OPTRATIO$(x_\delta, x_Q, robj \ldots)$, where $robj$ is the integer objective variable explicitly maximized. The dots in the signature mean that, depending on the selected implementation, OPTRATIO may require some additional parameters. We will detail this constraint later.

The original objective constraint used to obtain a value for variable obj is never removed from the model. However, except for the first solution, we maximize $robj$ instead of minimizing obj. After having solved the initial problem and identified a solution, the constraints DIVERSESUM and QUALITYSUM control diversity and quality of subsequent solutions through the two variables x_δ and x_Q.

While a variety of metrics could be used in Algorithm 1, we consider the L_1 norm in our pseudo-code. The domain of the diversity variable x_δ must be updated each time a new solution is added to the set $\mathcal{S}$, so as to maintain the same global diversity criterion. To illustrate this need, consider a simple example. Assume after the first run producing a solution t_1 to the initial problem we start searching for a second solution t_2 with $\underline{x_\delta} = 4$. Assume $t_2[x_\delta] = 5$. Then for the third solution t_3 we have to consider three of pairs of solutions instead of one (t_1 vs t_2, t_2 vs t_3 and t_1 vs t_3), thus we add $2 * 4 = 8$ to the previously counted distance $t_2[x_\delta] = 5$, which leads to $\underline{x_\delta} = 13$. At the next step, we have six pairs of solutions instead of three, we add $3 * 4 = 12$ to $t_3[x_\delta]$, and so on.

```
1  $t_1$ := Solution to $\mathcal{N}$ minimizing $obj$;
2  $\mathcal{S} := \{t_1\}$;
3  $j := 1$; $lb_\delta := \underline{x_\delta}$; $ub_\delta := \overline{x_\delta}$;
4  $\mathcal{C} := \mathcal{C} \cup$ DIVERSESUM$_{L_1}(\mathcal{X}, \mathcal{S}, x_\delta)$;
5  $\mathcal{C} := \mathcal{C} \cup$ QUALITYSUM$(obj, t_1, x_Q)$;
6  $\mathcal{C} := \mathcal{C} \cup$ OPTRATIO$(x_\delta, x_Q, robj \ldots)$;
7  repeat
8      $j := j + 1$;
9      $t_j$ := Solution to $\mathcal{N}$ maximizing $robj$;
10     $\underline{x_\delta} := t_j[x_\delta] + j \times lb_\delta$; $\overline{x_\delta} := t_j[x_\delta] + j \times ub_\delta$;
11     $\mathcal{S} := \mathcal{S} \cup \{t_j\}$;
12 until Fail exception $\vee$ $|\mathcal{S}| \geq k$;
13 return $\mathcal{S}$;
```

Algorithm 1: kBESTOPT($\mathcal{N}$, k, x_δ, x_Q): Solutions set $\mathcal{S}$

Performing a similar update of x_Q variable is not necessary, as the quality constraint QUALITYSUM does not deal with the entire solution set $\mathcal{S}$, but rather with the initial objective variable of the problem and the first solution t_1.

Concerning time complexity, we define a new but classical COP. The generic framework takes as an argument any COP. The distance measures should be computable in polynomial time for any fixed assignment (this is the case for Hamming, L_1 and L_2). As the new objective function is linked to the initial objective variable by the quality constraint and no constraints are removed, starting from a NP-Hard COP the resulting problem will generally remain in the same complexity class. However, the domain size of the new objective variable may differ from the size of the original one.

4 Filtering Algorithms

Filtering Diversity Constraints To obtain a satisfactory solving process we need to propagate all the constraints involved in Algorithm 1. We first focus on diversity.

1) Hamming distance. A filtering algorithm for DIVERSESUM$_\mathcal{H}(\mathcal{X}, \mathcal{S}, x_\delta)$ has already been introduced [Hebrard *et al.*, 2005]. This algorithm maintains Generalized Arc Consistency (GAC, see [Bessière, 2006]) in the case where x_δ is an integer, in $O(n(d + k))$ time, with $n = |\mathcal{X}|$, $k = |\mathcal{S}|$ and d is the maximum domain size. In our context, the most useful filtering of the constraint is performed according to the minimum current value $\underline{x_\delta}$ for variable x_δ, thus the algorithm of Hebrard et al. can be adapted in straightforward manner, even handling $\overline{x_\delta}$, and so we omit it for the sake of brevity.

2) L_1 norm. We propose an algorithm for the L_1 norm that runs in $O(k \cdot \sum_i D_i)$ time, where $\sum_i D_i$ is the sum of domain sizes of variables in $\mathcal{X}$. We have $\sum_i D_i \leq n \cdot d$ but depending on the current domains $n \cdot d$ may be a gross over-estimation. Algorithm 2 takes as arguments the variables, the set $\mathcal{S}$ of previous solutions, as well as the value of x_δ in the last previous solution, $prev_{x_\delta}$. The principle is to remove values that are not consistent with the current bounds of $D(x_\delta)$ using a regret mechanism, according to the minimum and maximum possible distances. The algorithm eliminates all the values that are not consistent, enforcing GAC.

3) L_2 norm. This case is more complex as the distance between two solutions is not computed with a simple sum.

However, we sill can design a propagator (Algorithm 3) in $O(k \cdot \sum_i D_i)$ time. The principle is first to fill a three dimensional matrix with all squares and then to compute maximum L_2 norms between $\mathcal{X}$ and each $t \in \mathcal{S}$. With such data we can again use a regret mechanism to remove all the values that are not consistent with $\underline{x_\delta}$ and $\overline{x_\delta}$, as Algorithm 2 does for L_1 norm.

```
foreach $i \in \{1, 2, \ldots, n\}$ do
    foreach $v_j \in D(x_i)$ do
        $dist[i][j] := \sum_{k=1}^{|\mathcal{S}|} |v_j - t_k[x_i]|$;
$maxSum := \sum_{i=1}^{|\mathcal{X}|} \max(dist[i]) + prev_{x_\delta}$;
$minSum := \sum_{i=1}^{|\mathcal{X}|} \min(dist[i]) + prev_{x_\delta}$;
$\overline{x_\delta} := \min(maxSum, \overline{x_\delta})$;  $\underline{x_\delta} := \max(minSum, \underline{x_\delta})$;
foreach $i \in \{1, 2, \ldots, |\mathcal{X}|\}$ do
    foreach $v_j \in D(x_i)$ do
        if $maxSum - \max(dist[i]) + dist[i][j] < \underline{x_\delta}$ then
            $D(x_i) := D(x_i) \setminus \{v_j\}$ ;
        if $minSum - \min(dist[i]) + dist[i][j] > \overline{x_\delta}$ then
            $D(x_i) := D(x_i) \setminus \{v_j\}$ ;
```

Algorithm 2: FILTERDIVERSESUM$_{L_1}(\mathcal{X}, \mathcal{S}, x_\delta, prev_{x_\delta})$

```
// Fill a three dimensional matrix with all squares
foreach $i \in \{1, 2, \ldots, |\mathcal{X}|\}$ do
    foreach $k \in \{1, 2, \ldots, |\mathcal{S}|\}$ do
        foreach $v_j \in D(x_i)$ do
            $dist[i][k][j] := (v_j - t_k[x_i])^2$;
        $maxd[i][k] := \max_j(dist[i][k])$;
        $mind[i][k] := \min_j(dist[i][k])$;
// Min. and max. $L_2$ norms between $\mathcal{X}$ and each $t \in \mathcal{S}$
$maxSum := 0$;
foreach $k \in \{1, 2, \ldots, |\mathcal{S}|\}$ do
    $maxSum[k] := 0$;
    foreach $i \in \{1, 2, \ldots, |\mathcal{X}|\}$ do
        $maxSum[k] := maxSum[k] + maxd[i][k]$;
    $maxSum = maxSum + \sqrt{maxSum[k]}$;
$minSum := 0$; ... Symmetric to $maxSum$
// Filtering
$maxSum := maxSum + prev_{x_\delta}$;
$minSum := minSum + prev_{x_\delta}$;
$\overline{x_\delta} := \lceil \min(maxSum, \overline{x_\delta}) \rceil$;  $\underline{x_\delta} := \lfloor \max(minSum, \underline{x_\delta}) \rfloor$;
foreach $i \in \{1, 2, \ldots, |\mathcal{X}|\}$ do
    foreach $v_j \in D(x_i)$ do
        $nsummax := 0$;
        foreach $k \in \{1, 2, \ldots, |\mathcal{S}|\}$ do
            $nsummax[k] :=$
            $maxSum[k] - maxd[i][k] + dist[i][k][j]$;
            $nsummax := nsummax + \sqrt{nsummax[k]}$;
        $nsummin := 0$; ... Symmetric to $maxSum$
        if $\lceil nsummax \rceil < \underline{x_\delta}$ then $D(x_i) := D(x_i) \setminus \{v_j\}$ ;
        ;
        if $\lfloor nsummin \rfloor > \overline{x_\delta}$ then $D(x_i) := D(x_i) \setminus \{v_j\}$ ;
        ;
```

Algorithm 3: FILTERDIVERSESUM$_{L_2}(\mathcal{X}, \mathcal{S}, x_\delta)$

Filtering the Quality Constraint. Bounds can be sharply adjusted in constant time complexity. As this filtering procedure is obvious and standard in CP, for sake of space we do not present it in this paper. The lower bound of obj is not affected by this constraint.

Propagating the New Objective. This section investigates the filtering algorithm of OPTRATIO$(x_\delta, x_Q, robj \ldots)$, the constraint used to balance between diversity and quality. We suggest two ways for expressing this constraint: Directly defining a ratio, or using a weighted sum.

1) Using a Ratio. We can directly define OPTRATIO$(x_\delta, x_Q, robj)$ as $\frac{x_\delta}{x_Q+1}$. However, diversity expressed by variable x_δ and quality expressed by x_Q variable may be of widely varying scale. Fortunately, we designed our paradigm so that the domain size of x_δ does not change. Its lower bound increases after each new solution proportionally to its upper bound. This observation enables straightforward normalization of the computation.

Let $|x_\delta|$ be the size of $D(x_\delta)$. Before starting the search of a new solution, we define at each new step j in the loop, that is, after line 8 of Algorithm 1, the two following rounded integer quantities:

$$f = \max\left(1, \frac{\overline{x_Q}}{|x_\delta| \times j}\right); \quad g = \max\left(1, \frac{|x_\delta| \times j}{\overline{x_Q}}\right).$$

Then, we add two variables x'_δ and x'_Q such that

$$D(x'_\delta) = [f \times \underline{x_\delta}, f \times \overline{x_\delta}]; \quad D(x'_Q) = [g \times \underline{x_Q}, g \times \overline{x_Q}].$$

We add constraints to the model stating that $x'_\delta = f \times x_\delta$ and $x'_Q = g \times x_Q$ and we use x'_δ and x'_Q instead of x_δ and x_Q in the ratio to obtain a normalized result.

2) Using a Weighted Sum. We define the constraint as follows. Given two positive integers α and β, an assignment of values to variables $\{x_\delta, x_Q, robj\}$ to constraint OPTRATIO$(x_\delta, x_Q, robj, \alpha, \beta)$ is satisfied iff: $robj = \alpha \times x_\delta - \beta \times x_Q$. This constraint can be filtered using standard features available in most solvers. To ensure a proper balance between the two factors of the sum, a straightforward adjustment of α or β is all that is required, somewhat simpler than the normalization of a ratio; we just need to augment either α or β. Typically, the diversity values should grow quickly, proportional to the number of solutions in $\mathcal{S}$, notably when we initially have set $\alpha = \beta = 1$. Given the last previous solution t_j, we suggest in this case to normalize the sum by updating β so as $\beta = \max(\beta, \frac{t_j[x_\delta]}{x_Q})$. At last, one may consider multi-objective optimization instead of a weighted sum. There are no theoretical issues specific to our approach concerning this alternative methodology.

5 Application to Over-Constrained Problems

A problem is over-constrained when no assignment of values to variables satisfies all constraints. In this situation, constraint violations are allowed in solutions, providing that such solutions retain practical interest. Diversity in this problem class is important, as it is not clear which combinations of violations are acceptable, thus making the anticipation of such rules challenging. Typically, we would like to propose a solution set where the violated constraints are not the same from one solution to another. On the other hand, in these problem violations should still be minimized. The quantity of violations that can be expressed in different ways (number of violations, weights, etc.), measures the quality of solutions.

Petit et al. [Petit *et al.*, 2000] suggested to express an over-constrained problem as a classical optimization problem, using new variables to express violations. These variables can have integer domains if a distance to satisfaction is measured or boolean domains if we solve the Max-CSP, where the objective is to minimize the number of constraint violations. For the sake of simplicity, and without loss of generality, we consider Max-CSP. Let $\mathcal{X}_v = \{xv_1, xv_2, \dots xv_{|\mathcal{C}|}\}$ be the set of boolean violation variables, given the constraint set $\mathcal{C}$. The objective constraint of Max-CSP is $obj = \sum_{i=1}^{|\mathcal{C}|} xv_i$.

Using such a formulation, applying Algorithm 1 with Hamming distances restricted to the set $\mathcal{X}_v$ (instead of considering all variables) generates solutions maximizing the balance between the number of constraint violations and the diversity of these violations. The diversity occurs when different constraints are violated in pairs of solutions. No new implementation issues are introduced. We are considering a subset of variables instead of all problem variables $\mathcal{X}$. In this way, our framework adapts to over-constrained problems.

6 An Alternative View of Solution Diversity

Semantics-Based Distance Definitions. Quite often, real-world optimization problems contain constraints that involve the objective function, or at least portions of it. Such constraints express some generic concepts which distinguish useful solutions from solutions that may be optimal with respect to the objective function, but not applicable in practice. In CP, this need was introduced, for instance, in the context of over-constrained problems [Petit *et al.*, 2000] and subsequently by the introduction of several new global constraints, e.g., [Pesant and Régin, 2005; Schaus *et al.*, 2007; Petit, 2012; Narodytska *et al.*, 2013]. However, end-users are not always able to formulate their respective preferences, even less using global constraints. In this context, we would like a system that is able to propose different classes of solutions, each one corresponding to one particular concept. Following this idea, we propose a new approach for defining diversity. Consider the following example.

Example 1 *Let $\mathcal{X} = \langle x_1, x_2, \dots, x_{n=10}\rangle$ be a sequence of variables ordered by their indexes, all involved in an objective function $\sum_{i=1}^{n} x_i \leq s$ to minimize. Consider two solutions of equal quality $t_1 = \langle 1,0,0,1,0,0,1,0,0,1\rangle$ and $t_2 = \langle 0,0,0,1,1,1,1,0,0,0\rangle$. $\delta_{\mathcal{H}}(\mathcal{X}, t_1, t_2) = \delta_{L_1}(\mathcal{X}, t_1, t_2) = 4$ and $\delta_{L_2}(\mathcal{X}, t_1, t_2) = 2$.*

In this example, the two solutions have equivalent quality and they are quite similar, as 60% of variables share the same value. Nevertheless, one may observe that in t_1 zero costs are homogeneously distributed in the sequence, whereas in t_2 costs equal to 1 are concentrated in a unique area. From the point of view of the end-user, this characteristic may make the two solutions appear very distinct. In CP, the generic concept of concentrating costs is a sequence is captured by FOCUS [Petit, 2012], a counting constraint. By counting, we mean that a particular variable is used to measure a given property, which is here the concentration of high values.

Definition 7 *Let $s_{i,j}$ be any sequence of indices of consecutive variables in $\mathcal{X}$, such that $s_{i,j} = [i, i+1, \dots, j]$, $1 \leq i \leq j \leq n$.*
Let y_c be a variable. Let k and len be two integers, $1 \leq len \leq n$. An instantiation of $\mathcal{X} \cup \{y_c\}$ satisfies FOCUS*$(\mathcal{X}, y_c, len, k)$ iff there exists a set $S_{\mathcal{X}}$ of* disjoint *sequences of indices $s_{i,j}$ such that three conditions are all satisfied: (1) $|S_{\mathcal{X}}| \leq y_c$. (2) $\forall x_l \in \mathcal{X}$, $x_l > k \Leftrightarrow \exists s_{i,j} \in S_{\mathcal{X}}$ such that $l \in s_{i,j}$. (3) $\forall s_{i,j} \in S_{\mathcal{X}}$, $j - i + 1 \leq len$.*

Assume we do not constrain sequence length ($len = n$) and k (highest "non costly" value) is equal to 0. Regarding Example 1, the minimum value for y_c in t_1 is 4, while the minimum value for y_c in t_2 is 1. This means that solution t_2 is more concentrated than solution t_1, and we can use this result as a metric for estimating solution diversity. While Example 1 is restricted to constraint networks where the ordering (or more generally the topology) of variables is important, many other concepts are expressed by counting constraints, such as the CHANGE constraint for timetabling [Cosytec, 1997], DEVIATION [Schaus *et al.*, 2007], AMONG [Bessière *et al.*, 2005], BALANCE [Bessière *et al.*, 2014].

Integration in our Framework The simplest way to incorporate this new approach would be to replace the diversity constraint by a given counting constraint. However, it is likely more appropriate and desirable to prioritize classical diverse solutions, e.g., obtained using Hamming-based, L_1 or L_2 norms, which *additionally* vary according to the concept captured by a given counting constraint. As generic concepts may be of heterogeneous nature, we may provide a diverse set of solutions for each concept class. Therefore, we suggest to integrate the counting variable y_c in the quality constraint QUALITYSUM instead of modifying the diversity approach. This is possible because quality is taken into account in the objective of the new maximization problem. We replace the quality variable x_Q in Algorithm 1 by a new variable x'_Q, such that $x'_Q = \gamma x_Q + \theta y_c$, where γ and θ are positive integers. In the case of satisfaction problems, we may consider only $x'_Q = y_c$. Our framework can thus be considered as an extension of previous CP approaches, able to tackle new definitions of diversity for satisfaction and optimization problems.

7 Experiments

We used an IntelXeon 2.27GHz machine and the Choco 3.2.1 solver. [1] We exclusively provided a constraint network as argument without any specific tuning to the problem. We systematically re-use the search strategy of the original model (e.g., DOM/WDEG [Boussemart *et al.*, 2004]), and then assign the new variables in static order. We present the results using the L_1 norm for diversity. Other cases are similar.

1) Sorting chords. We implemented the *chords* musical benchmark [Truchet and Codognet, 2004; Petit, 2012]. Chords have to be sorted in order to minimize a sum of costs. All the chords should be different. Each cost corresponds to two consecutive chords and is equal to the number of notes changing from one chord to the next one less one, plus additional penalties in some cases. A pair of consecutive chords with a non zero cost is *a violation.*

[1] http://www.emn.fr/z-info/choco-solver/

Instances		Average $t_i[x_Q]$ / $(t_2[x_\delta], t_{20}[x_\delta])$ / # backtracks / % of optimal proofs, for 20 generations of 20 solutions sets			
$\|\mathcal{X}\|/maxc$	Av. $t_1[obj]$ / Av. $t_1^\delta[x_\delta]$	R	NR	S	NS
$15/3$	$1/42$	$0/(31, 4278)/2K/100\%$	$0/(31, 4675)/1.4K/100\%$	$4/(35, 5550)/15K/100\%$	$0.1/(35, 4779)/4K/100\%$
$15/6$	$4/54$	$0/(32, 4927)/0.9K/100\%$	$0/(32, 4891)/0.9K/100\%$	$11/(46, 7376)/15K/100\%$	$0.1/(46, 5133)/3.7K/100\%$
$19/3$	$0/65$	$0/(45, 6307)/1.3K/100\%$	$0/(45, 6297)/1.3K/100\%$	$7.3/(53, 8343)/700K/40\%$	$0.1/(53, 6655)/70K/95\%$
$19/6$	$3/78$	$0.1/(47, 6657)/2.8K/100\%$	$0.1/(47, 6640)/2.7K/100\%$	$0.6/(65, 10860)/563K/25\%$	$0.1/(65, 7163)/99K/90\%$
$23/3$	$0/91$	$0.1/(69, 9278)/6K/100\%$	$0.1/(69, 9279)/6K/100\%$	$5.2/(75, 10978)/623K/5\%$	$0.1/(75, 9510)/242K/65\%$
$27/3$	$0/110$	$0.1/(93, 12739)/113K/90\%$	$0.1/(93, 12741)/112K/90\%$	$2.1/(100, 13496)/414K/5\%$	$0.1/(100, 12902)/341K/20\%$

Table 1: Comparison of ratios for assessing simultaneously solution diversity and quality in optimization.

(A)	(B)	NS	NS with $x'_Q = x_Q + y_c$
$15/8$	$6/7$	$1.2/(5, 696)/1.8K/100\%$	$2.8/(5, 696)/2K/100\%$
$19/10$	$12/9$	$1.5/(6, 896)/43K/100\%$	$3.4/(6, 897)/45K/100\%$
$23/12$	$11/10$	$1.5/(8, 1095)/200K/100\%$	$4.7/(8, 1096)/333K/100\%$
$27/14$	$13/13$	$2.1/(9, 1276)/2648K/80\%$	$5/(9, 1284)/2980K/80\%$

Table 2: Results for Over-Constrained instances of the chords problem. (A) $|\mathcal{X}|/maxc$. (B) Av. $t_1[obj]$ / Av. $t_1^\delta[x_\delta]$.

An instance with optimal sum value that is strictly positive is *over-constrained*. We generated a random cost value for each possible pair of chords. In a first experiment, we compare methods for computing the new objective variable $robj$, namely a ratio (R), a normalized ratio (NR), a sum with $\alpha = \beta = 1$ (S) and a normalized sum (NS). Let nc be the number of chord variables and $maxc$ be the maximum possible value of a cost. For each instance, we give average results obtained with 20 randomly generated cost matrices, and we generate 20 solutions per matrix and ratio, with a time-limit of 15 seconds for each new solution. Moreover, after having generated the first solution $t^* = t_1$ with objective value $t_1[obj]$ and prior to searching for additional solutions, we run our paradigm once with a weighted sum with $\alpha = 1$ and $\beta = 0$, in order to obtain a solution t_1^δ with the maximum possible diversity $t_1^\delta[x_\delta]$, without regarding solution quality. This provides a relevant upper bound of the maximum diversity we may obtain. Thus, for each triplet ($|\mathcal{X}|$, $maxc$, $r \in \{$ R, NR, S, NS $\}$), we solve 420 problems. Then, Table 1 shows the average quality loss / $(t_2[x_\delta], t_{20}[x_\delta])$ / number of backtracks / percentage of optimal proofs for over 20 solution sets generated. All first solutions were proven to be optimal. The results show that using a ratio is the most robust approach, both in terms of optimality proof and number of backtracks. With respect to S and NS, t_2 and t_3 solutions are almost always obtained with a loss in quality, because this allows high increases in diversity and a better objective than with $x_Q = 0$. Using a sum without normalization leads to a progressive removal of the quality criterion as the number of previous solutions increases. In a second experiment, we consider over-constrained instances. We augment the model with a set of boolean reified variables $\mathcal{Y} = \{y_1, y_2, \ldots, y_{nc-1}\}$, one-to-one mapped with cost variables that represent violations, i.e., $y_i = (cost_i > 0)$. Instead of considering diversity among the $\mathcal{X}$ variables, we consider $\mathcal{Y}$. Diversity is applied on the over-constrained aspect of the instances, not on the values taken by variables. Table 2 shows the results for 20 solutions per generated set, with a 5 minute time-limit, using NS. We consider instances with and without the addition of an alternative notion of diversity, here represented by the FOCUS constraint. In the fourth column, FOCUS($costs, y_c, |costs|, 0$) is used. Quality loss is then represented by a variable $x'_Q = x_Q + y_c$ instead of x_Q. As in Table 1, for each ratio we show: The average quality loss / $(t_2[x_\delta], t_{20}[x_\delta])$ / number of backtracks / percentage of optimal proofs for over 20 solution sets generated. Table 2 highlights that solution quality and diversity as well as the solving process are not strongly affected when our paradigm is used in the context of over-constrained problems or with alternative diversity notions.

2) TSPLIB. We used a graph-variable model [Fages and Lorca, 2012] for solving TSPLIB symmetric instances [Reinelt, 1991], which are state-of-the-art routing benchmarks. We used channeling constraints [Cheng *et al.*, 1999] to make the link between the edges in the graph variable and boolean variables expressing whether a route takes the road between two cities or not, used as diversity variables.

	Att48	Bayg29	Berlin52	Brazil58	Burma14	Dantzig42	Eil51	Eil76	Gr17
$t_1[obj]$	10628	1610	7542	25395	3323	699	426	538	2085
% of optimal proof with R	100	100	100	100	100	47	0	0	100
Av. % Quality loss	0.3%	0.8%	0.4%	0.1%	1.6%	2.9%	5.4%	11.9%	0.8%

	Gr21	Gr24	Gr48	Hk48	Rat99	Rd100	St70	Ulysse16	Ulysse22
$t_1[obj]$	2707	1272	5046	11461	1211	7910	675	6859	7013
% of optimal proof with R	100	100	100	100	0	5	0	100	100
Av. % Quality loss	2.4%	0.7%	0.2%	0.5%	7%	0.5%	9.2%	0.6%	0.5%

Table 3: Results for Symmetric TSPLIB instances.

Table 3 shows the results for instances such that the first optimal solution can be found and proved in less than one minute (without giving the optimum value as an upper-bound of the objective variable). We generated a total of 20 solutions per instance, defining a ratio (R) with L_1 norm. For most instances, proof of optimality can be done in less than one minute for all solutions. If it is not the case for one solution, generally the other diverse solutions are not proved in under a one minute. The results show low average quality losses, which is interesting in practice for routing problems.

8 Conclusions and Future Work

We have proposed a generic paradigm for producing diverse solutions of high quality in constraint-based optimization, which can be specialized to the context of over-constrained problems. We suggest considering diversity from additional points of view, based on concepts expressed by counting constraints. Future work includes using other counting constraints and implementation in an industrial context.

References

[Allouche *et al.*, 2014] D. Allouche, I. André, S. Barbe, J. Davies, S. de Givry, G. Katsirelos, B. O'Sullivan, S. David Prestwich, T. Schiex, and S. Traoré. Computational protein design as an optimization problem. *Artif. Intell.*, 212:59–79, 2014.

[Bessière *et al.*, 2005] C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan, and T. Walsh. Filtering algorithms for the nvalue constraint. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Second International Conference, CPAIOR*, pages 79–93, 2005.

[Bessière *et al.*, 2014] C. Bessière, E. Hebrard, G. Katsirelos, Z. Kiziltan, É. Picard-Cantin, C.-G. Quimper, and T. Walsh. The balance constraint family. In *Principles and Practice of Constraint Programming - CP 2014, 20th International Conference, CP*, pages 174–189, 2014.

[Bessière, 2006] C. Bessière. Constraint propagation. Research report 06020 (Chapter 3 of the Handbook of Constraint Programming, F. Rossi, P. van Beek and T. Walsh eds. Elsevier 2006.), LIRMM, 2006.

[Billaut *et al.*, 2010] J.-C. Billaut, A. Moukrim, and E. Sanlaville, editors. *Flexibility and Robustness in Scheduling*. Wiley, 2010.

[Bistarelli *et al.*, 1999] S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier. Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. *Constraints*, 4(3):199–240, 1999.

[Boussemart *et al.*, 2004] F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais. Boosting systematic search by weighting constraints. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including PAIS*, pages 146–150, 2004.

[Cheng *et al.*, 1999] B. M. W. Cheng, K. M. F. Choi, J. H. Lee, and J. C. K. Wu. Increasing constraint propagation by redundant modeling: an experience report. *Constraints*, 4(2):167–192, 1999.

[Cosytec, 1997] Cosytec. *CHIP 5.1 Ref. Manual*, 1997.

[Eiter *et al.*, 2013] T. Eiter, E. Erdem, H. Erdogan, and M. Fink. Finding similar/diverse solutions in answer set programming. *TPLP*, 13(3):303–359, 2013.

[Fages and Lorca, 2012] J.-G. Fages and X. Lorca. Improving the asymmetric TSP by considering graph structure. *CoRR*, abs/1206.3437, 2012.

[Hadzic *et al.*, 2009] T. Hadzic, A. Holland, and B. O'Sullivan. Reasoning about optimal collections of solutions. In *Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP*, pages 409–423, 2009.

[Hamming, 1950] R. W. Hamming. Error detecting and error correcting codes. *Bell system technical journal*, 29:147–160, 1950.

[Hebrard *et al.*, 2005] E. Hebrard, B. Hnich, B. O'Sullivan, and T. Walsh. Finding diverse and similar solutions in constraint programming. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, pages 372–377, 2005.

[Hebrard *et al.*, 2007] E. Hebrard, B. O'Sullivan, and T. Walsh. Distance constraints in constraint satisfaction. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 106–111, 2007.

[Hebrard *et al.*, 2011] E. Hebrard, D. Marx, B. O'Sullivan, and I. Razgon. Soft constraints of difference and equality. *J. Artif. Intell. Res. (JAIR)*, 41:97–130, 2011.

[Hentenryck *et al.*, 2009] P. Van Hentenryck, C. Coffrin, and B. Gutkovich. Constraint-based local search for the automatic generation of architectural tests. In *Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP*, pages 787–801, 2009.

[Narodytska *et al.*, 2013] N. Narodytska, T. Petit, M. Siala, and T. Walsh. Three generalizations of the FOCUS constraint. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 2013.

[Pesant and Régin, 2005] G. Pesant and J.-C. Régin. SPREAD: A balancing constraint based on statistics. In *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP*, pages 460–474, 2005.

[Petit *et al.*, 2000] T. Petit, J.-C. Régin, and C. Bessière. Meta-constraints on violations for over constrained problems. In *12th IEEE International Conference on Tools with Artificial Intelligence*, pages 358–365, 2000.

[Petit, 2012] T. Petit. Focus : A constraint for concentrating high costs. In *Principles and Practice of Constraint Programming - 18th International Conference, CP*, pages 577–592, 2012.

[Reinelt, 1991] G. Reinelt. TSPLIB, a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

[Schaus *et al.*, 2007] 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, 4th International Conference, CPAIOR*, pages 260–274, 2007.

[Trapp and Konrad, 2015] Andrew C. Trapp and Renata A. Konrad. Finding diverse solutions of high quality to binary integer programs. *to appear, IIE Transactions*, 2015.

[Truchet and Codognet, 2004] C. Truchet and P. Codognet. Musical constraint satisfaction problems solved with adaptive search. *Soft Comput.*, 8(9):633–640, 2004.

[Zampelli *et al.*, 2013] S. Zampelli, Y. Vergados, R. Van Schaeren, W. Dullaert, and B. Raa. The berth allocation and quay crane assignment problem using a CP approach. In *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013*, pages 880–896, 2013.