Modelling Diversity of Solutions

Linnea Ingmar,[1] **Maria Garcia de la Banda,**[2] **Peter J. Stuckey,**[2] **Guido Tack**[2]
[1]KTH Royal Institute of Technology, Sweden
[2]Monash University, Melbourne, Australia
lingmar@kth.se, {maria.garciadelabanda, peter.stuckey, guido.tack}@monash.edu

Abstract

For many combinatorial problems, finding a single solution is not enough. This is clearly the case for multi-objective optimization problems, as they have no single "best solution" and, thus, it is useful to find a representation of the non-dominated solutions (the Pareto frontier). However, it also applies to single objective optimization problems, where one may be interested in finding several (close to) optimal solutions that illustrate some form of diversity. The same applies to satisfaction problems. This is because models usually idealize the problem in some way, and a diverse pool of solutions may provide a better choice with respect to considerations that are omitted or simplified in the model. This paper describes a general framework for finding k diverse solutions to a combinatorial problem (be it satisfaction, single-objective or multi-objective), various approaches to solve problems in the framework, their implementations, and an experimental evaluation of their practicality.

Introduction

The vast majority of the literature on optimization focuses on finding a single optimal solution to a given optimization problem. However, many optimization models only approximate the complex real-word problem they are modelling, and they cannot capture all of the problem details. Hence, the single "best" solution presented to the problem owner may be far from being the best alternative in practice. The same applies to satisfaction problems, which can be seen as optimization problems with a single, constant optimization function. Ideally, the problem owner would be presented with a set of optimal solutions, to either choose one based on preferences that were not modelled, or to at least get an overview of the range of possible optimal solutions.

To overcome this issue one cannot simply provide the problem owner with *all* optimal solutions to the problem, since that would often be overwhelming and computationally impractical. Similarly, one cannot simply provide a small number of *arbitrary* optimal solutions, since there is nothing preventing them from being almost identical (e.g., identical except for a single decision that has no impact on the objective value). Instead, one should provide a reasonably small number of *diverse* solutions that are optimal or close to optimal. This immediately raises the question: what does it mean for solutions to be diverse?

A common case for diversity appears in multi-objective optimization problems, where we are presented with several objectives (rather than a single one) to maximize. This naturally gives rise to diversity issues since, while the ideal answer to a multi-objective problem is the entire *Pareto frontier* or *efficient set* of solutions, in practice this set is often extremely large. Instead, we aim to find k solutions that are diverse with respect to the different objectives.

Example 1. Consider the problem of designing the physical layout of a complex chemical plant. The aim is to allocate the position of all equipment and to route the connecting pipes, while ensuring the solutions satisfy different safety and operational constraints, and minimize three cost objectives: footprint, pipes and supports. While the problem relies on uniformly modelling these costs using present values, they are estimations that depend on future unknowns, and might vary independently of each other. If we present a single "optimal design" to the plant engineers, they are locked into estimations that might be substantially different to what is finally used. Hence, a better approach is to present them with k diverse designs that are optimal (or near-optimal) in terms of estimated cost, but are substantially different in how those costs are spread among the three objectives. This gives decisions makers the ability to choose a robust solution with respect to their understanding of the relative costs of each component. □

While multi-objective problems give rise to a natural notion of diversity, the diversity of solutions may also depend on factors not captured by the objective. Such a notion of diversity is important to all combinatorial problems, including satisfaction and single-objective problems.

Example 2. Consider the plant layout problem introduced in Example 1. It is common for plants to have a special piece of equipment (the pipe *rack*) that traverses the entire plant. The relative position of equipment to this rack is a decision that is often not subject to operational constraints and is thus not captured by any of the objectives. However, some relative positions may be more suitable than others, as they

might affect maintenance access, footprint shape and construction capabilities. Hence, for any given combination of objective values, it makes sense for the plant engineer to explore solutions that differ on the relative position of the equipment. This notion of "diversity" is not captured by the model's objective function. □

The above example shows that measures of diversity are often intimately tied to the model of the problem. Solution diversity in optimization has been studied extensively (see the discussion of related work at the end of this paper). However, current constraint modelling languages do not let modellers directly specify diversity in the model. This paper shows how to extend a solver-independent modelling language to express a very general form of diversity problem. It then defines a number of approaches to solving this diversity problem, and provides an experimental evaluation of an implementation of the framework that demonstrates the versatility of the diversity problem we define, and compares the different solving approaches. In particular, the contributions of this paper are:

- A generic framework to specify diverse solution problems.
- Generic solving approaches to solve the diverse solution problem.
- A practical implementation of the diverse solution problem specification and their solving approaches in the MiniZinc (Nethercote et al. 2007) system.
- Experimental results showing
  - The practicality of the framework.
  - The relative strengths and weaknesses of the different solving approaches.
  - Its use on a large real-world optimization problem where diversity has been requested by the industrial clients.

Preliminaries

Given a universe of variables $\mathcal{V}$ and a set of values $\mathcal{D}$, a valuation θ is a mapping from $v \in \mathcal{V}$ to $\theta(v) \in \mathcal{D}$. Given a set Σ of functions f mapping $\mathcal{D} \times \cdots \times \mathcal{D} \to \mathcal{D}$, we extend θ to map terms over $\Sigma \cup \mathcal{V}$ in the obvious manner $\theta(f(t_1, \ldots, t_n)) = f(\theta(t_1), \ldots \theta(t_n))$. Similarly, given a set Π of predicates in $\mathcal{D} \times \cdots \times \mathcal{D}$, we extend θ to map atoms of Π over terms in $\Sigma \cup \mathcal{V}$ as $\theta(p(t_1, \ldots, t_n)) = \mathit{true}$ iff $(\theta(t_1), \ldots, \theta(t_n)) \in p$.

A *constraint satisfaction problem (CSP)* $P = (V, D, C)$ consists of a set of variables V, an initial (finite) domain D which maps each variable $v \in V$ to its set of possible values $D(v)$, and a set of constraints C over variables V. A valuation θ is a *solution* of P iff $\theta(v) \in D(v), v \in V$ and $\theta(c)$ is *true* for all $c \in C$.

A *constraint optimization problem (COP)* $P = (V, D, C, O)$ consists of a CSP (V, D, C) and an objective function O defined on variables V to be maximized (without loss of generality). An optimal solution of P is a solution θ of (V, D, C) such that for all other solutions θ' of (V, D, C), $\theta(O) \geq \theta'(O)$.

A *multi-objective constraint optimization problem (MCOP)* $P = (V, D, C, \mathbf{O})$ is a COP, where the objective O is replaced by a set of objectives $\mathbf{O}$. Solution θ_1 of P *dominates* solution θ_2 if for each $O \in \mathbf{O}$, $\theta_1(O) \geq \theta_2(O)$, and for at least one $O \in \mathbf{O}$, $\theta_1(O) > \theta_2(O)$. A *non-dominated solution* θ of P is a solution of P that is not dominated by any other solution of P. The *Pareto frontier* PF(P) of an MCOP P is the set of all non-dominated solutions of P, and provides a picture of the whole space of best solutions. This is useful for problem owners, as it allows them to choose the best tradeoff between the objectives. Since for many MCOPs the Pareto frontier may be too large to compute, we are interested in approximating it by a (small) subset $R \subseteq$ PF(P).

There are different ways to measure how well a Pareto frontier is approximated by subset R. One of the simplest measures is given by the hypervolume of the dominated objective space (Zitzler et al. 2003). Given bounds $l_O..u_O$ for each objective $O \in \mathbf{O}$, the *hypervolume* (HV) measures how many valuations in the hypercube defined over the objectives, are dominated by the solutions in R.

$$\mathrm{HV} = |\{(x_1, \ldots, x_n) \mid x_i \in l_{O_i}..u_{O_i}, \\ \exists \theta \in R : \forall i \in 1..n : \theta(O_i) \geq x_i\}|$$

This definition of HV assumes discrete objectives; a similar definition using integration exists for continuous domains.

Modelling Diversity

This section first introduces the various components that make up our diversity framework, and then formally defines the *general diversity problem*.

Distance Measures

Central to the modelling of solution diversity is a model-specific notion of *distance* between solutions. A *distance function* $\delta(\theta_1, \theta_2)$ returns the distance from valuation θ_1 to valuation θ_2. Note that such a distance function is *directional* and, thus, may not be symmetric ($\delta(\theta_1, \theta_2) \neq \delta(\theta_2, \theta_1)$), although symmetry often holds.

Example 3. Consider a model M of the plant layout problem introduced in Example 1 expressed in the MiniZinc modelling language. We can add to M the notion of diversity introduced in Example 2 as follows:

```
enum POS =  OVERLAP, NORTH, SOUTH ;
array[EQUIPMENT] of var POS: rel_pos_rack;
```

where `POS` is an enumerated type representing whether equipment overlaps with the rack or is located north or south of it, `EQUIPMENT` was defined in M as the set of equipment we are trying to position, and `rel_pos_rack` is an array of auxiliary decision variables that map every piece of equipment to its relative position with the rack. Note that the above definitions are added to the model M *only* to measure solution diversity; they play no role in solving the problem.

We can now add to M a distance function (`swaps`), which given the array of relative positions for two solutions (`rpr1` and `rpr2`), returns the total number (`sum`) of non-rack equipment (`e != rack_idx`) that swap their relative position from north to south, or from south to north:

```
var int: swaps(array[int] of var POS: rpr1,
               array[int] of var POS: rpr2)
= sum (e in EQUIPMENT where e != rack_idx)
      (rpr1[e]!=OVERLAP / rpr2[e]!=OVERLAP
        / rpr1[e]!=rpr2[e]);
```

Note that this particular distance function is symmetric. □

Combining Distances into a Single Diversity Measure

There is often more than one single distance function to measure the diversity of solutions. This is common, for example, in multi-objective problems, as each objective provides a diversity measure.

Example 4. Consider a radiation scheduling problem with two objectives: minimizing the beamtime of each radiation beam shot, and minimizing the number `K` of beam shots. These objectives can be expressed in MiniZinc (using maximization to comply with the framework) as:

```
solve maximize -Beamtime;
solve maximize -K;          % number of shots
```

The following distance measures can be automatically derived from these objectives: $\delta_{\text{beamtime}}(\theta_1, \theta_2) = \theta_1(-\texttt{Beamtime}) - \theta_2(-\texttt{Beamtime})$ and $\delta_{\text{shots}}(\theta_1, \theta_2) = \theta_1(-\texttt{K}) - \theta_2(-\texttt{K})$. Note that these are asymmetrical distance functions: a positive distance $\delta_O(\theta_1, \theta_2)$ reflects that θ_1 is better than θ_2 in objective O. □

When multiple distance measures $\Delta = [\delta_1, \ldots, \delta_n]$ exist, modellers must combine them to create a single *diversity measure*. Obvious choices for this include using:

- $\max$: the maximum of the distance measures;
- $\sum$: summing up the distance measures;
- $\min$: the least distance measure, which is useful for ensuring a minimum level of diversity across all measures;
- lex: a lexicographic order on some ordering of the distance measures, which is useful to make each measure more important than those later in the order.

In addition, there might also be problem-specific ways to combine the measures. In general, given a *combinator function* $\mathcal{C} : \mathit{range}(\delta_1) \times \cdots \times \mathit{range}(\delta_n) \to X$ defined by the modeller to map distances $\Delta = [\delta_1, \ldots, \delta_n]$ to some ordered set X, our framework defines the *diversity measure* as $H(\theta_1, \theta_2) = \mathcal{C}(\delta_1(\theta_1, \theta_2), \ldots, \delta_n(\theta_1, \theta_2))$.

Example 5. For multi-objective problems, a useful way of combining (maximization) distance measures is $\mathcal{C} = \max$. This has the property that $H(\theta_1, \theta_2) > 0$ implies there is some distance measure where θ_1 is better than θ_2 and, hence, θ_1 is not dominated by θ_2. Similarly, $H(\theta_1, \theta_2) < 0$ implies that θ_1 is dominated by θ_2. □

Intra-diversity Constraints

For optimization problems, it is often useful to find a diverse set of solutions that are *near* optimal, since this will eliminate diverse solutions that are not interesting to the problem owner (too far from optimal). The definition of how near a solution must be to the optimal value to be acceptable is usually problem-specific and, hence, should be specified in the model. Given an optimal or best known solution to a COP with objective value o, our framework requires the definition in the model of an *intra-diversity constraint* $q(o, O)$, which constrains the objective O to be near the optimal value o.

Example 6. A common intra-diversity constraint for a COP is to allow solutions within $x\%$ of optimal, e.g. $q(o, O) = 100O \geq (100 - x)o$. □

A similar intra-diversity constraint $q(\mathbf{o}, \mathbf{O})$ can be defined for multi-objective problems, where $\mathbf{o}$ contains the optimal value o_i for each objective O_i in the multi-objective $\mathbf{O}$.

Example 7. A typical intra-diversity constraint for a multi-objective problem would be to allow solutions within $x\%$ of optimal in at least one measure, e.g. $q(\mathbf{o}, \mathbf{O}) = \exists_{O_i \in \mathbf{O}} 100O_i \geq (100 - x)o_i$. □

Inter-diversity Constraints

When searching for a set of k diverse solutions, there may be *a priori* restrictions on how different the solutions need to be. Typically, these constraints make use of some distance measure in order to separate the solutions. Hence, the framework includes an *inter-diversity constraint* $p(\delta_1(\theta_1, \theta_2), \ldots, \delta_n(\theta_1, \theta_2))$, which can impose constraints on the various distance measures between solutions.

Example 8. Consider the radiation scheduling problem of Example 4, with the distance measures δ_{beamtime} and δ_{shots}. In order to make the solutions noticeably different, modellers may require δ_{beamtime} to be at least 5 minutes, since otherwise the two solutions may not really be different from a patient's perspective. Thus, we define $p(db, ds)$ as $db \geq 300$, forcing the beamtimes to differ by at least 300 seconds. □

Aggregation of Diversity

In order to select the most diverse k solutions for a problem, our framework uses an aggregator function $\mathcal{A}$ to combine the pairwise diversity measures resulting from each pair of solutions, into an overall measure of diversity for the k solutions. Two common definitions for $\mathcal{A}$ are: $\sum$, summing up the pairwise diversity measures; and $\min$, taking the minimum over all pairs of the diversity measure. For multi-objective problems, the automatically derived aggregator $\mathcal{A}$ is $\min$.

The General Diversity Problem

Let us now specify a general framework for defining diversity problems.

Definition 1 (General Diversity Problem)**.** Given COP $P = (V, D, C, O)$ or MCOP $P = (V, D, C, \mathbf{O})$, a set of pairwise distance functions $\Delta = \{\delta_1, \ldots, \delta_m\}$, a combinator function $\mathcal{C}$, an aggregator function $\mathcal{A}$, an inter-diversity constraint p and an intra-diversity constraint q, the *general diversity problem* $Q = (P, \Delta, \mathcal{C}, \mathcal{A}, p, q)$ for non-negative integer k, is to find a set Θ such that:

- $|\Theta| = k$;
- $\forall \theta \in \Theta$: θ is a solution to the corresponding CSP (V, D, C);

- $\forall\theta \in \Theta$: either $q(o, \theta(O))$ or $q(\mathbf{o}, \theta(\mathbf{O}))$ holds, where o is the value of the best known solution for COP P, and $\mathbf{o}$ contains the values o_i of the best known solution for each objective $O_i \in \mathbf{O}$ of MCOP P;
- $\forall\theta_1, \theta_2 \in \Theta, \theta_1 \neq \theta_2$: $p(\delta_1(\theta_1, \theta_2), \ldots, \delta_m(\theta_1, \theta_2))$ holds; and
- $\mathcal{A}(\{\mathcal{C}(\delta_1(\theta_1, \theta_2), \ldots, \delta_m(\theta_1, \theta_2)) | \theta_1, \theta_2 \in \Theta, \theta_1 \neq \theta_2\})$ is maximized.

Note that not all these features need to be used. For example, if the original problem is a satisfaction problem, then modellers can simply set $O = 0$ to make it a COP. If they do not want to restrict solutions to be near optimal, they can define q to be *true*. Similarly, if they have no a priori distance requirements between solutions, they can define p to be *true*.

Solving Diversity Problems

This section discusses four approaches to solving the general diversity problem $Q = (P, \Delta, \mathcal{C}, \mathcal{A}, p, q)$ for size k, where $P = (V, D, C, O)$, as per Definition 1.

Exact Solution Method

The most straightforward solving approach is to treat the general diversity problem as a COP and obtain an exact solution. To do this we first create k copies of its variables and constraints, k copies of its intra-diversity constraint q, and $k(k-1)/2$ copies of its inter-diversity constraint p. We then construct the objective function, which is of size $|\Delta|k(k-1)/2$, and solve it using any available constraint optimization solver.

The exact solution method, which we denote as $\textsc{Ex}(k)$, is the only method we will explore that has the capability of finding an optimal solution to Q for k. While the problem size of the such COP explodes by at least a factor k, and possibly k^2, it can be useful as a component in hybrid and post hoc approaches (see below).

Greedy Solution Method

To overcome the size explosion of $\textsc{Ex}(k)$, we use a greedy approach that finds one solution at a time, making use of the components in Q to drive towards diverse solutions. While this greedy method may not find optimal solutions to the general diversity problem, it can be much faster in practice, while still producing good diversity.

The greedy approximation method $\textsc{Gr}(k)$ is shown in Algorithm 1, which iteratively builds a set of solutions Θ as follows. Initially, Θ consists of an arbitrary first optimal solution θ_1. Then, in each iteration of the loop, a modified version of the base problem is solved: the set V' of variables for the pairwise distances between solutions with their respective domains is joined with V; the domain D' mapping each variable $v' \in V'$ is added to D; the set C' of constraints on the pairwise distances are added; and the objective is to *lexicographically* maximize first the aggregated distance, and then the original objective O. In other words, in each iteration the *most diverse* solution is found so that its objective value is at least as good as any other solution with the same diversity. If the original problem P is multi-objective, we create a single objective $O = \sum_{O_i \in \mathbf{O}} w_i O_i$ by weighting the components with some positive weights w_i. In this case, the second component will eliminate dominated solutions.

Algorithm 1 A greedy algorithm for the general diversity problem

1: **procedure** $\textsc{Gr}(k)$
2: **Input**
- General diversity problem $Q = ((V, D, C, O), \Delta = \{\delta_1, \ldots, \delta_m\}, \mathcal{A}, \mathcal{C}, p, q)$
- Non-negative integer k

3: $\theta_1 \leftarrow$ solution to (V, D, C) that maximizes O
4: $o \leftarrow \theta_1(O)$
5: $\Theta \leftarrow \{\theta_1\}$
6: **while** $|\Theta| \leq k$ **do**
7: **let** $\forall\theta \in \Theta, \forall\delta \in \Delta : d^\theta_\delta \leftarrow \delta(V, \theta(V))$
8: $V' \leftarrow \{d^\theta_\delta | \theta \in \Theta, \delta \in \Delta\}$
9: $C' \leftarrow \bigcup_{\theta\in\Theta}\{p(d^\theta_{\delta_1}, \ldots, d^\theta_{\delta_m})\} \cup \{q(o, O)\}$
10: $\theta \leftarrow$ solution to $(V \cup V', D \cup D', C \cup C')$ lexicographically maximizing $[\mathcal{A}(\{\mathcal{C}(d^\theta_{\delta_1}, \ldots, d^\theta_{\delta_m}) \mid \theta \in \Theta\}), O]$
11: $\Theta \leftarrow \Theta \cup \{\theta\}$
return Θ

The role of the lexicographic optimization is to ensure that the solutions we return are the best of those with the same diversity measure. Consider Example 6, which defines an intra-diversity constraint requiring solutions to be no more than $x\%$ away from the optimal. Without lexicographic optimization, the greedy algorithm could always return solutions that are exactly $x\%$ non-optimal, even if optimal solutions with the same measure of diversity exist.

Example 9. When solving the radiation scheduling problem of Example 4 using $\textsc{Gr}(k)$, the automatically derived aggregated diversity measure in line 10 of Algorithm 1 is: $\min(\{\max(\delta_{\text{beamtime}}(V, \theta), \delta_{\text{shots}}(V, \theta)) \mid \theta \in \Theta\})$. By maximizing this measure, the algorithm is guaranteed to find the *worst represented point* (Masin and Bukchin 2008) on the Pareto frontier in each iteration.

Hybrid Diversity Solving

The $\textsc{Gr}(k)$ method is blind to the overall space of solutions. Hence, it can have difficulty finding good solutions. In contrast, the $\textsc{Ex}(k)$ method finds optimal solutions but is inefficient as k grows large (as we will see in the experiments, it is not too bad for a small k). This gives us the impetus to develop a hybrid solving method that combines these two. We can straightforwardly achieve this by defining a $\textsc{Hy}(l, k), l \leq k$ method that first solves $\textsc{Ex}(l)$ to generate the most diverse l solutions, and then uses the greedy method to find the next $k - l$ solutions, by simply replacing Θ at line 5 of $\textsc{Gr}(k - l)$ by the result of $\textsc{Ex}(l)$.

Post hoc Diversity Solving

Previous methods compute exactly k solutions. A different approach to tackling the diversity problem is to compute many more solutions (hopefully, very quickly) and select k diverse solutions from them. More formally, the post

```
1 % Define a pairwise distance function on x
2 annotation diverse(array[int] of var $T: x,
3                    ann: dist_fun);
4 % Define the combinator function
5 annotation div_combinator(ann: comb_fun);
6 % Define the aggregator function
7 annotation div_aggregator(ann: agg_fun);
8 % Post the intra diversity constraint p
9 annotation intra_div_constraint(ann: pred);
10 % Post the inter diversity constraint q
11 annotation inter_div_constraint(ann: pred);
```

Figure 1: MiniZinc annotations for diversity.

hoc solving method $\text{PH}(U, K, k)$ first finds $u > k$ solutions S, using method U, to CSP $(V, D, C \cup \{q(o, O)\})$, and then solves the general diversity problem with the restriction that $\Theta \subseteq S$, using method K. Note that we can precompute $\delta(\theta_1, \theta_2), \delta \in \Delta, \theta_1, \theta_2 \in S, \theta_1 \neq \theta_2$ and $p(\delta_1(\theta_1, \theta_2), \ldots, \delta_m(\theta_1, \theta_2)), \theta_1, \theta_2 \in S, \theta_1 \neq \theta_2$, where $\Delta = \{\delta_1, \ldots, \delta_m\}$. Hence, the problem becomes one of selecting k out of u elements of a set, subject to some pairwise constraints and an objective. We can use any approach U to generate the u initial solutions, and any solving method K for selecting k solutions. Obvious choices are random or greedy methods for U, and exact or greedy ones for K.

Implementation in MiniZinc

MiniZinc (Nethercote et al. 2007) is a very expressive solver independent modelling language. Therefore, it is not difficult to extend MiniZinc to define our general diversity problem. We do so by adding annotations to the model in such a way that, if the annotations are ignored, the model specifies the original COP or MCOP. Figure 1 presents the MiniZinc annotations added to the MiniZinc annotation library for modelling the general diversity problem of Definition 1. Note that we do not need to explicitly model the number of solutions k, as MiniZinc already supports the setting of a desired number of solutions.

Given a model M of a COP or MCOP problem, modellers use MiniZinc functions (Stuckey and Tack 2013) to define different distance measures in M. Then, they add to M a `diverse` annotation (lines 2–3) for each distance measure, where the second argument is the name of the function, and the first is the argument to the function (assumed here to be a single array `x` of variables, all of type `T`). Modellers also use MiniZinc functions to define combinator and aggregator functions in M, and then add `div_combinator` and `div_aggregator` annotations (lines 5 and 7) for these functions. Modellers can then use MiniZinc predicates to define the intra- and inter-diversity constraints of the problem, and add `intra_div_constraint` and `inter_div_constraint` annotations (lines 9 and 11) for them.

The implementation builds an appropriate MiniZinc model for the chosen solution approach. For the exact approach it builds one large model. For the greedy and hybrid approaches it builds a new model for each loop iteration. For the post hoc approach it builds a set selection model, precomputing as much as possible.

Experimental Evaluation

Single Distance Measure

Consider the *Bulk Water Management Problem*, which models water reservoirs and bulk transfers over a certain time horizon of multiple years, deployed for a major metropolitan area. Part of the problem is deciding how much water is stored in each reservoir `r` on each month `m`: this value is modelled by the two dimensional array of decision variables `vol_stored`. While the total volume stored in all of the reservoirs is close to constant at any given point in time, the stored volume for a given reservoir might vary over time. To model for diversity in volume difference, we introduce an array `vol_diff` of auxiliary decision variables modelling the maximal difference in stored volume for each reservoir:

```
array[SOURCES] of var 0.0..max_vol: vol_diff
   = [ max(vol_stored[r,..]) -
       min(vol_stored[r,..])
     | r in SOURCES ];
```

We use the following diversity annotations:

```
:: diverse(vol_diff, manhattan);
:: div_aggregator(min);
:: intra_div_constraint(gap);
predicate gap(float: optimal_value) =
  Obj <= optimal_value + 0.1;
```

where `manhattan` is a distance function for the Manhattan distance of two arrays of variables, and `gap` constrains the objective to be equal to the optimal value, using a floating point precision of 0.1.

Table 1 shows runtime and minimum distance between any two solutions of the *Bulk Water Management Problem* for five different solving approaches: greedy; exact; hybrid $\text{HY}(2, k)$; and two post hoc approaches. Post hoc approach $\text{PH}(\text{GR}, \text{EX}, k)$ (column 4) spends 29 minutes generating a pool S_{greedy} of solutions using the greedy approach, and then uses the exact approach to select the k most diverse solutions out of S_{greedy}. The post hoc approach $\text{PH}(\text{RD}, \text{GR}, k)$ (column 5) spends 29 minutes generating a set S_{random} of *random* solutions to the problem, and then uses the *greedy* approach to select k solutions from S_{random}. We use the greedy approach in the selection phase, as the exact approach runs out of memory due to the size of S_{random}. As the results for the greedy and post hoc approaches depend on the first solution, the table shows the mean and standard deviation of diversity over 10 runs using different random start solutions for them.

The *Bulk Water Management Problem* is linear and, thus, it is most efficiently solved by a MIP solver; we choose Gurobi. However, for the selection phase of $\text{PH}(\text{GR}, \text{EX}, k)$ we use a CP solver (Gecode), as the selection problem does not linearize well. We produce random solutions in Gurobi by passing a different random seed in each invocation of the solver. Note that while this does not guarantee that solutions are uniformly distributed, it is often the only method of randomisation available in black-box solvers such as Gurobi.

As expected, the fastest approach is the greedy one. The exact approach cannot prove optimality within the given

Table 1: Runtime in seconds and solution diversity for the *Bulk Water Management Problem* using five solving approaches for finding k diverse solutions. Symbol — indicates the timeout (30 minutes) is reached and the best found diversity before timing out is reported.

	GR(k)		EX(k)		HY($2,k$)	
k	time	div	time	div	time	div
2	**3.2**	202K±3K	106.7	**260K**	106.0	**260K**
3	**5.0**	157K±4K	—	**189K**	113.7	140K
4	**9.7**	122K±2K	—	**156K**	118.7	139K
5	**12.8**	116K±4K	—	**137K**	118.3	130K
6	**15.6**	112K±4K	—	115K	121.4	**116K**
7	**20.5**	102K±3K	—	93K	132.5	97K
8	**27.4**	96K±4K	—	**99K**	138.5	90K
9	**31.9**	90K±2K	—	74K	146.4	90K
10	**40.6**	84K±2K	—	60K	155.3	**90K**
11	**45.3**	82K±2K	—	64K	159.2	**82K**
12	**60.4**	79K±2K	—	57K	173.0	**80K**

	PH(GR, EX, k)		PH(RD, GR, k)	
k	time	div	time	div
2	1740.3	236K±7K	1740.0	86K±5K
3	1740.3	160K±6K	1740.0	60K±6K
4	1740.3	129K±3K	1740.0	55K±5K
5	1740.3	118K±3K	1740.1	43K±3K
6	1740.3	114K±3K	1740.1	40K±3K
7	1740.3	**105K**±3K	1740.1	37K±2K
8	1740.4	97K±2K	1740.1	35K±2K
9	1740.4	**91K**±2K	1740.2	34K±2K
10	1740.4	85K±2K	1740.2	31K±3K
11	1740.4	81K±3K	1740.2	30K±3K
12	1740.4	78K±3K	1740.3	27K±2K

timeout for $k \geq 2$, but finds the highest solution diversity for $k \leq 5$ and $k = 8$. HY$(2,k)$ and PH(GR, EX, k) sometimes find solutions of higher diversity than the greedy approach, but have worse runtime. The diversity found by PH(RD, GR, k) is consistently less than half the diversity of the other approaches.

Multiple Distance Measures

Consider the *Resource-Constrained Project Scheduling Problem with Weighted Earliness/Tardiness objective* (RCPSP/WET) (Vanhoucke, Demeulemeester, and Herroelen 2001) as a MCOP problem, where the two objectives are minimizing the earliness and the tardiness of the tasks. The diversity measure is automatically derived (see Example 4) from the two objectives: maximize $-earliness$; and maximize $-tardiness$. Furthermore, the combinator $\mathcal{C}$ is $\min$ and aggregator $\mathcal{A}$ is $\max$ (automatically derived).

Table 2 shows runtime and hypervolume for the RCPSP/WET problem using three solving approaches for the general diversity problem: the greedy approach and two post hoc approaches. The post hoc approaches use all Pareto optimal solutions S found within 29 minutes by (a) the greedy approach, and (b) random search, and then select the k most diverse solutions from S. The "random" search

Table 2: Runtime in seconds and hypervolume for the RCPSP/WET problem using four solving approaches for finding k diverse solutions. Symbol — indicates the timeout (30 minutes) is reached and the best found hypervolume before timing out is reported.

	GR(k)		PH(GR, EX, k)	
k	time	HV	time	HV
3	**10.6**	13.69K	1740.9	13.90K
4	**16.9**	17.23K	1741.1	17.39K
5	**20.5**	18.78K	1741.1	19.02K
6	**32.2**	19.77K	1741.1	**19.98K**
7	**37.2**	20.56K	1741.2	**20.59K**
8	**41.6**	20.96K	1741.3	21.00K
9	**52.2**	21.26K	1741.6	**21.29K**
10	**65.9**	21.49K	1741.5	21.61K

	PH(RD, EX, k)		Max HV	
k	time	HV	time	HV
3	1740.3	13.90K	273.4	**13.94K**
4	1740.3	**17.40K**	—	17.33K
5	1740.4	**19.14K**	—	18.61K
6	1740.4	19.81K	—	16.61K
7	1740.4	20.58K	—	13.83K
8	1740.4	**21.08K**	—	16.17K
9	1740.4	21.22K	—	9.15K
10	1740.4	**21.71K**	—	0.83K

is done by repeatedly minimizing the weighted sum of the objectives using random positive weights.

Note that while the greedy approach guarantees all solutions found so far are Pareto optimal (Masin and Bukchin 2008), the same does not hold for the exact solving approach EX(k). The inter-diversity constraint q would need to be "the solution is Pareto optimal", which is not possible to express in a constraint. Therefore, we also compare with a direct approach, named Max HV, which is not part of our framework and finds k solutions such that the hypervolume measure is maximized. By construction, an optimal solution is guaranteed to consist of a set of Pareto optimal solutions. All approaches are given the two extreme points as start solutions. We use Chuffed as backend solver.

Clearly the greedy approach can effectively find good representatives of the Pareto frontier quickly. The post hoc approaches find in general somewhat better solutions, using more time. The direct approach is competitive with the other approaches in terms of hypervolume for $k \leq 5$, but does not scale to higher values of k.

Case Study: Process Plant Layout Optimization

Consider again the plant layout problem introduced in Example 1, modelled using (a) the state-of-the-art model of the problem defined by Belov et al. (2018), and (b) the definition of diversity based on the relative positions of the rack as discussed in Example 3. The diversity annotations are:

```
:: diverse(rel_pos_rack, swaps);
:: div_aggregator(min);
:: intra_div_constraint(within_five_percent);
```

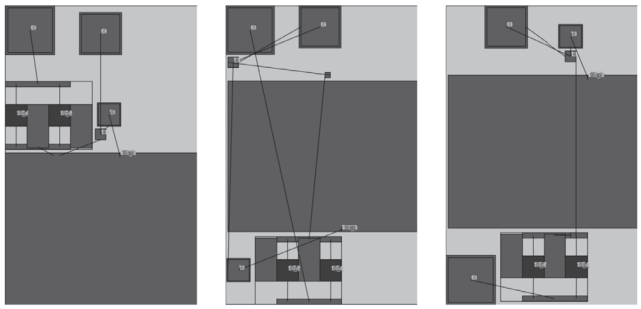

Figure 2: Three diverse solutions to the process plant layout problem.

where `within_five_percent` constrains the objective to be within five percent of the optimal value. For demonstration purposes, we use a subset of a real world instance with 20 (out of more than 200 in total) pieces of equipment. While the full instance cannot be solved to optimality even within 2 weeks, our small instance takes less than one minute to solve to optimality using Gurobi on 4 threads. Figure 2 shows a top-down view of three plant designs, where the large dark grey area in each solution is the pipe rack, and the smaller rectangles are other pieces of equipment. All solutions are generated with the greedy approach, using the above diversity annotations. Clearly, the user is presented with three very different designs.

Related Work and Conclusion

The diversity of solutions has been studied in the context of constraint programming (CP) (Hebrard et al. 2005; Hebrard, O'Sullivan, and Walsh 2007; Petit and Trapp 2015), mixed integer programming (MIP) (Glover, Løkketangen, and Woodruff 2000), Boolean satisfiability (SAT) (Nadel 2011) and answer set programming (ASP) (Eiter et al. 2009), among others. The key design principle of our work is to separate modelling of diversity problems from how they are solved, while in most previous work modelling and solving are conflated. An exception is work by Hebrard et al. (2005), which introduces several classes of diversity (and similarity) problems for constraint satisfaction, as well as two generic ways of solving such problems. Perhaps their most important problem class is MAXDIVERSEkSET, which aims to find k solutions, such that the minimum distance between any two solutions is maximized, for a given distance measure. The general diversity problem from Definition 1 is a generalization of MAXDIVERSEkSET, as we consider (multi-objective) optimization problems, support the definition and combination of different distance measures, and of constraint solutions with user-defined intra- and inter-diversity constrains. Similarly, the two solving algorithms proposed by Hebrard et al. (2005) are special cases of the greedy and exact approaches discussed in this paper. The work by Petit and Trapp (2015) generalizes Hebrard et al.'s greedy algorithm to single objective problems. To find solutions that are diverse as well as near optimal, the ratio of diversity and loss in optimality is maximized in each iteration. Their algorithm allows threshold values on optimality and diversity to be specified, which is similar in idea to our intra- and inter-diversity constraints. Their algorithm is, however, implemented directly into a solver without a clear separation between modelling and solving. Our work clearly distinguishes itself from this work, and all the other ones we are aware of, by providing an easy-to-use modelling framework where various diversity components can be defined in an expressive modelling language directly in the model, independently of how the problem is solved.

Work has been done for finding diverse solutions in specific solving technologies. This includes, for example, papers by Hebrard et al. (2005) and Hebrard, O'Sullivan, and Walsh (2007), which introduce global constraints and propagation algorithms for CP, and work by Glover, Løkketangen, and Woodruff (2000), which is specific for MIP. Although the aim of our work is to be solver-independent, future work should include how these methods can be utilized by our framework when using a certain solving technology.

In population based methods, such as evolutionary algorithms, maintaining diversity in the population pool is needed to prevent premature convergence to a sub-optimal solution, and can be achieved by so called niching methods (Li et al. 2017). Niching can also be used for finding several diverse solutions (see e.g. Kruisselbrink et al. 2009). Whether ideas from niching methods can be reused in our framework, is an interesting research direction.

To the best of our knowledge, there is no previous work that combines Pareto optimality with solution diversity. However, there are many algorithms in the literature for solving MCOPs in general. Many such algorithms are designed for enumerating the *complete* Pareto frontier, such as the widely studied ϵ-constraint method (Y. Haimes, Lasdon, and A. Wismer 1971) and a recently proposed decision diagram method (Bergman and Ciré 2016). More related to our focus on diversity is previous work on finding representative *subsets* of the Pareto frontier. An overview of such algorithms for continuous optimization problems was recently done (Burachik, Kaya, and Rizvi 2019), and whether any of them can be used in our framework should be investigated in future work. Two studies that focus on discrete problems are by Schwind et al. (2016) and by Masin and Bukchin (2008). The former considers the problem of finding a subset of a given size k of the Pareto frontier, such that a measure of representativity is maximized, and prove the decision version of the problem to be Σ_2^{P}-complete. The latter finds a representative set by iteratively adding the worst represented point on the Pareto frontier. Notably, their algorithm is a special case of Algorithm 1.

Conclusion We define a modelling framework for solving diversity problems and a number of solving methods to tackle these problems. This allows a modeller to add measures of diversity to their model in a straightforward way, and to collect a diverse set of solutions. Experiments show the approach is applicable to large real world problems. It is the first modelling framework we are aware of to allow diversity to be directly specified as part of the model.

Acknowledgements This work was partly sponsored by the Australian Research Council grant DP180100151. The authors would like to thank the anonymous reviewers for their helpful comments, and Ilankaikone Senthooran for providing instances for the case study in the experimental evaluation of this work.

References

Belov, G.; Czauderna, T.; de la Banda, M. G.; Klapperstück, M.; Senthooran, I.; Smith, M.; Wybrow, M.; and Wallace, M. 2018. Process plant layout optimization: Equipment allocation. In Hooker, J. N., ed., *Proceedings of the Twenty-Fourth International Conference on Principles and Practice of Constraint Programming, CP 2018*, volume 11008 of *Lecture Notes in Computer Science*, 473–489. Springer.

Bergman, D., and Ciré, A. A. 2016. Multiobjective optimization by decision diagrams. In Rueher, M., ed., *Proceedings of the Twenty-second International Conference on Principles and Practice of Constraint Programming, CP 2016*, volume 9892 of *Lecture Notes in Computer Science*, 86–95. Springer.

Burachik, R.; Kaya, C.; and Rizvi, M. 2019. Algorithms for generating pareto fronts of multi-objective integer and mixed-integer programming problems. Preprint, http://arxiv.org/abs/1903.07041.

Eiter, T.; Erdem, E.; Erdoğan, H.; and Fink, M. 2009. Finding similar or diverse solutions in answer set programming. In Hill, P. M., and Warren, D. S., eds., *Proceedings of the Twentyfifth International Conference in Logic Programming, ICLP 2009*, volume 5649 of *Lecture Notes in Computer Science*, 342–356. Springer.

Glover, F.; Løkketangen, A.; and Woodruff, D. L. 2000. Scatter search to generate diverse MIP solutions. In Laguna, M., and Velarde, J. L. G., eds., *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, volume 12, 299–317. Springer.

Hebrard, E.; Hnich, B.; O'Sullivan, B.; and Walsh, T. 2005. Finding diverse and similar solutions in constraint programming. In Veloso, M. M., and Kambhampati, S., eds., *Proceedings of the Twentieth AAAI Conference on Artificial Intelligence*, 372–377. AAAI Press / The MIT Press.

Hebrard, E.; O'Sullivan, B.; and Walsh, T. 2007. Distance constraints in constraint satisfaction. In Veloso, M. M., ed., *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, IJCAI 2007*, 106–111.

Kruisselbrink, J. W.; Aleman, A.; Emmerich, M. T. M.; IJzerman, A. P.; Bender, A.; Bäck, T.; and van der Horst, E. 2009. Enhancing search space diversity in multi-objective evolutionary drug molecule design using niching. In Rothlauf, F., ed., *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2009*, 217–224. ACM.

Li, X.; Epitropakis, M. G.; Deb, K.; and Engelbrecht, A. P. 2017. Seeking multiple solutions: An updated survey on niching methods and their applications. *IEEE Trans. Evolutionary Computation* 21(4):518–538.

Masin, M., and Bukchin, Y. 2008. Diversity maximization approach for multiobjective optimization. *Operations Research* 56(2):411–424.

Nadel, A. 2011. Generating diverse solutions in SAT. In Sakallah, K. A., and Simon, L., eds., *Proceedings of the Fourteenth International Conference on Theory and Applications of Satisfiability Testing, SAT 2011*, volume 6695 of *Lecture Notes in Computer Science*, 287–301. Springer.

Nethercote, N.; Stuckey, P. J.; Becket, R.; Brand, S.; Duck, G. J.; and Tack, G. 2007. MiniZinc: Towards a standard CP modelling language. In Bessiere, C., ed., *Proceedings of the Thirteenth International Conference on Principles and Practice of Constraint Programming, CP 2007*, volume 4741 of *Lecture Notes in Computer Science*, 529–543. Springer.

Petit, T., and Trapp, A. C. 2015. Finding diverse solutions of high quality to constraint optimization problems. In Yang, Q., and Wooldridge, M. J., eds., *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015*, 260–267. AAAI Press.

Schwind, N.; Okimoto, T.; Clement, M.; and Inoue, K. 2016. Representative solutions for multi-objective constraint optimization problems. In Baral, C.; Delgrande, J. P.; and Wolter, F., eds., *Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning, KR 2016*, 601–604. AAAI Press.

Stuckey, P. J., and Tack, G. 2013. MiniZinc with functions. In Gomes, C. P., and Sellmann, M., eds., *Proceedings of the Tenth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR 2013*, volume 7874 of *Lecture Notes in Computer Science*, 268–283. Springer.

Vanhoucke, M.; Demeulemeester, E.; and Herroelen, W. 2001. An exact procedure for the resource-constrained weighted earliness-tardiness project scheduling problem. *Annals OR* 102(1-4):179–196.

Y. Haimes, Y.; Lasdon, L.; and A. Wismer, D. 1971. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-1:296–297.

Zitzler, E.; Thiele, L.; Laumanns, M.; Fonseca, C. M.; and da Fonseca, V. G. 2003. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Trans. Evolutionary Computation* 7(2):117–132.