Representative Solutions for Bi-Objective Optimisation

Emir Demirović
School of Computing and Information Systems
University of Melbourne
Melbourne, Australia
emir.demirovic@unimelb.edu.au

Nicolas Schwind
National Institute of Advanced
Industrial Science and Technology
Tokyo, Japan
nicolas-schwind@aist.go.jp

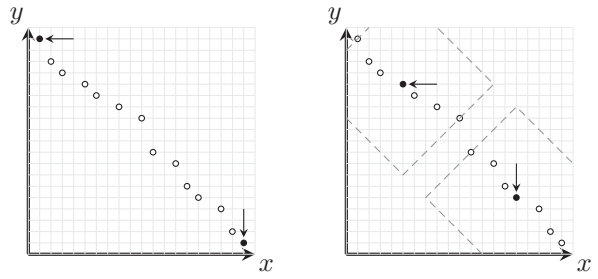
Abstract

Bi-objective optimisation aims to optimise two generally competing objective functions. Typically, it consists in computing the set of nondominated solutions, called the Pareto front. This raises two issues: 1) time complexity, as the Pareto front in general can be infinite for continuous problems and exponentially large for discrete problems, and 2) lack of decisiveness. This paper focusses on the computation of a small, "relevant" subset of the Pareto front called the *representative set*, which provides meaningful trade-offs between the two objectives. We introduce a procedure which, given a precomputed Pareto front, computes a representative set in polynomial time, and then we show how to adapt it to the case where the Pareto front is not provided. This has three important consequences for computing the representative set: 1) does not require the whole Pareto front to be provided explicitly, 2) can be done in polynomial time for bi-objective mixed-integer linear programs, and 3) only requires a polynomial number of solver calls for bi-objective problems, as opposed to the case where a higher number of objectives is involved. We implement our algorithm and empirically illustrate the efficiency on two families of benchmarks.

Introduction

Bi-objective optimisation aims to optimise two competing objective functions. For instance, in supply-chain management, the goal is to simultaneously minimise delivery time and delivery cost (Trisna et al. 2016). In the design of wireless sensor networks, both energy consumption and data collection time must be kept at a minimum (Caillouet, Li, and Razafindralambo 2011). An "ideal" option generally does not exist and trade-offs must be made.

The set of nondominated solutions, called the *Pareto front*, is exponentially large in the general case. This brings forth two main issues. First, enumerating the Pareto front can be computationally infeasible. Second, providing a large number of options might hinder decision making, as providing an overwhelming number of choices can be counterproductive (Iyengar and Lepper 2000; Shafir, Simonson, and Tversky 1993; Dhar 1997).



(a) Diverse Solutions (b) Representative Solutions.

Figure 1: Two notions of Pareto front filtering when two solutions are to be selected in each case. The dashed lines illustrate the radius of the corresponding representative set

A reasonable alternative is then to consider a subset of the Pareto front. Two main notions of such Pareto front filtering have been introduced in the literature. Both notions depend on a number k which is assumed to be "small" and given in input of the problem. The first one, called *diversity* (or uniformity) (Sayin 2000; Hebrard et al. 2005; Petit and Trapp 2015; Vaz et al. 2015; Masin and Bukchin 2008), aims to select k solutions from the Pareto front that are "spread out" as much as possible. The second one, called *representativity* (or coverage) (Sayin 2000; Vaz et al. 2015; Schwind et al. 2016), considers a set S of k solutions from the Pareto front with minimum *radius*, where the radius of S is the maximal distance between any solution p from the Pareto front and the solution from S that is the closest to p; such a set S is called a *representative set* (cf. Fig. 1).

In this paper, we focus on the computation of a representative set of solutions in the bi-objective case. In particular, we pose the following research questions:

1. Is it possible to efficiently compute a representative set without computing the whole complete Pareto front?
2. Can a representative set be computed even if the Pareto front is infinite?

Our contribution provides a positive answer to both questions. This is done constructively: we provide a generic procedure for bi-objective optimisation that computes a rep-

resentative set using only a polynomial number of *solver calls*. The notion of a solver call depends on the structure of the considered optimisation problem and refers to either a sub-procedure that solves the single-objective version of the problem, e.g. interior-point method for linear programs, or an oracle for problems whose decision counterpart is NP-complete. In general, the Pareto front is infinite for linear programs and exponentially large for discrete optimisation problems. Considering only a polynomial number of solver calls is a clear advantage over enumerating the Pareto front when computing a representative set. As a proof-of-concept, we implement our procedure and empirically show that it computes the representative set in reasonable time.

In the next section, we position our paper with respect to related work and provide further motivation. We then follow up with preliminaries and our main contributions.

Related Work

We assume that the reader is familiar with the complexity classes P and NP (see (Papadimitriou 2003) for more details). Higher complexity classes are defined using *oracles*, i.e. abstract machines that solve problems from a particular class in constant time. In particular, P^{NP} (resp. $\mathsf{NP}^{\mathsf{NP}}$) is the class of decision problems that are solved in polynomial time by a deterministic (resp. non-deterministic) Turing machine using an oracle for NP.

In (Schwind et al. 2016), the authors introduced the notion of representative solutions for multi-objective constraint optimisation problems. They proved that the decision problem underlying the computation of a representative set of k solutions (k being given in unary form) is $\mathsf{NP}^{\mathsf{NP}}$-hard in the general multi-objective case. They also introduced a procedure that first enumerates the Pareto front, then solves the NP-hard task of computing a represented set of k solutions. Our aim is to show the computational advantages of focussing on only two objectives. This is motivated by the fact that *when the Pareto front is given in input*, computing a representative set is an NP-hard task in the general multi-objective case (Schwind et al. 2016), but can be done in polynomial time in the bi-objective case. Our goal is then to investigate whether the restriction to two objectives also leads to a computational shift in the case when the Pareto front is not explicitly given. As presented in the following sections, this is indeed the case.

To the best of our knowledge, computing a representative set for the dedicated bi-objective case has not been addressed in the literature. Instead, the research efforts have been focussed on developing *heuristics* to compute (not necessarily representative) subsets of the Pareto front for bi- and multi-objective problems. In this context, heuristic approaches compute (Pareto optimal) solutions that *aim* to cover the Pareto front, but do not *guarantee* the optimality of the set with respect to the desired filtering notion.

There is a number of works aiming to filter the Pareto front by selecting only a few Pareto optimal solutions and obtain a "good" representation (Eusébio, Figueira, and Ehrgott 2014; Vaz et al. 2015; Schwind et al. 2016; Raith and Sedeño-Noda 2017; Masin and Bukchin 2008; Ceyhan, Köksalan, and Lokman 2019). How well a Pareto front is represented by means of a few solutions depends on the context. As mentioned in the introduction, there are two main such quality measures (Fig. 1b): diversity (or uniformity) and representativity (or coverage). There is also a third measure called ϵ-indicator considered in (Zitzler et al. 2003; Vaz et al. 2015), which is closely related to the one of representativity. However, this measure consists in computing a restricted set of solutions that are "close enough" to the Pareto front, but that are not necessarily Pareto optimal. An orthogonal diversity measure can be considered over decision variables rather than objectives, i.e., computing near-optimal solutions with a structure of diverse nature (Adomavicius and Kwon 2014). The authors in (Petit and Trapp 2019) provide a detailed survey on related diversity measures. One of the motivations for diversity is that it might not be easy to capture the true preferences of users in the form of an objective function, and thus providing a variety of solutions, which can be further filtered offline and online, is beneficial. In our work, in contrast, we assume that qualitative measures are possible for two objectives, for instance production time versus cost. In these cases, the objectives functions can be used to succintly explain the differences among solutions and their trade-offs to the user. Note that a discrimination based on both decision variables and objectives is not incompatible. Indeed, one could consider computing a representative set, which provides meaningful trade-offs between two objectives, and afterwards search for diverse solutions among those, in an attempt to capture other important solution features that were not given explicitly.

In (Bazgan, Jamain, and Vanderpooten 2017), the authors introduce the notion of ϵ-Pareto front (ϵ-PF), which is a set of solutions that dominate each Pareto optimal solution in an ϵ-relaxed sense. The authors discuss algorithms for generating the smallest ϵ-PF with additional properties and variations. This notion is related but is not directly comparable to the representative set, as the goals are different: it aims to *approximate* the Pareto front rather than compute a *representation* with a few solutions.

Focussing on the bi-objective optimisation, network flow problems are solved in (Eusébio, Figueira, and Ehrgott 2014; Raith and Sedeño-Noda 2017) by computing only a subset of the Pareto front. However, the quality measure is again different from the representativity measure. In (Raith and Sedeño-Noda 2017), so-called "extreme" solutions are selected, i.e. the Pareto optimal solutions that lie on the boundaries of the convex hull of the solution space. In (Eusébio, Figueira, and Ehrgott 2014), the notion of representativity is the same as the one considered here, but only heuristics are provided: a subset of the Pareto front is computed in an online fashion, and does not necessarily result in a representative set. The number of returned solutions is not decided as an upstream step; instead, the user is required to provide the desired distance betweens solutions in a semi-interactive process. For representative solutions of multi-objective problems with continuous variables, the authors in (Karasakal and Köksalan 2009) propose a heuristic to sample a surface that approximates the Pareto front.

Closely related to our work is (Vaz et al. 2015): the au-

thors introduce two complete algorithms to compute a representative set of k solutions for bi-objective optimization problems, under the assumption that the Pareto front is available in input. Their most efficient algorithm is based on Dynamic Programming and runs in $\mathcal{O}(k \cdot |PF| + |PF| \cdot log\,|PF|)$, where $|PF|$ is the size of the Pareto front. Similarly to our procedure presented in the next section, the first step is to re-order the Pareto front in an increasing order according to their values projected to first objective. Then, roughly speaking, their algorithm iteratively computes the radius of a representative set of size k' where k' ranges over $\{1, \ldots, k\}$. This departs from our method which, in a nutshell, consists in computing a representative set of a minimum number of k' solutions given a *fixed* radius R, and doing so iteratively by performing a dichotomic search over the radius values ranging between two bounds until some optimality conditions are met and k' coincides with the value k required in input. Most importantly, in (Vaz et al. 2015) the authors focussed on the case where the Pareto front is given in input, and did not consider the case where it is characterised succinctly, e.g. by means of constraints.

Two constraint programming approaches have been proposed for multi-objective optimisation. In (Gavanelli 2002), the author introduces a data structure for storing the Pareto front. This is used in an anytime depth-first search algorithm that eventually enumerates the whole Pareto front. The approach has been refined into a so-called *global constraint* (Schaus and Hartert 2013), which is suitable for constraint programming solvers, and used in a large neighbourhood search framework. The idea is to maintain a list of nondominated solutions and to iteratively select one solution from the set, relax and optimise it so as to improve the hypervolume surrounding the maintained nondominated set of solutions. While related, the nature of our paper is very different to (Schaus and Hartert 2013): they consider a *heuristic* to *approximate* the Pareto front with *many* solutions, while we consider a *complete* algorithm for *representing* the Pareto front with *few* solutions.

The most popular algorithms for enumerating bi-objective Pareto frontiers are the ϵ-method (Haimes 1971) and the *two-phase* approach (Ulungu and Teghem 1995). The ϵ-method systematically enumerates the complete Pareto front for bi-objective problems: it computes an initial solution by lexicographically minimising f_x and f_y and iteratively computes lexicographical solutions that minimise the first and then the second objective function while requiring the value according to the first objective function to be greater than previously computed. In the two-phase algorithm, the decision variables are assumed to be binary. It consists of two steps: 1) to compute the nondominated solutions that lie on the convex hull of the Pareto front, and 2) to compute the missing solutions by considering the space in between two consecutive nondominated solutions of the convex hull. The first stage is generic, but the second one is problem-specific. Indeed, in our experiments, the ϵ-method gave better results than the two-phase approach with a generic second-stage implementation. A refinement of the two-phase approach (Stidsen, Andersen, and Dammann 2014) may have a better performance depending on the problem. We refer to (Stidsen, Andersen, and Dammann 2014) for a survey of others techniques and references for problem-specific variants.

In (Bergman and Cire 2016), a binary-decision diagram (BDD) approach is applied to compute the Pareto front for multi-objective optimisation. The strength of the method is that once a combinatorial problem is represented as a BDD, a multi-objective shortest path algorithm can be used to compute an optimal solution. The main drawback is that not all problems admit a compact BDD representation.

Preliminaries

We are given a fixed set V of integer variables and a set of constraints C. An assignment of all variables V to a value is called a *solution* if it satisfies all constraints from C. We assume that checking whether a given assignment is a solution can be done in polynomial time. For simplicity, we use C^* as the (succintly characterised) set of solutions. We are also given two objective functions f_x, f_y of the form $f : C^* \rightarrow \mathbb{Q}$, which are to be minimized simultaneously. A solution $p \in C^*$ *dominates* another solution $p' \in C^*$ whenever $f_x(p) \leqslant f_x(p')$ and $f_y(p) \leqslant f_y(p')$. We say that p *strictly dominates* p' if p dominates p' and p' does not dominate p. And p is said to be *Pareto optimal* if there is no solution p' that strictly dominates p. The set PF denotes the *Pareto front*, i.e., the set of all Pareto optimal solutions.

The Manhattan distance (or $L1$-norm) between two solutions $p, p' \in C^*$ is defined as $dist(p, p') = |f_x(p) - f_x(p')| + |f_y(p) - f_y(p')|$. In the following, we focus on the Manhattan distance and simply refer to it as the "distance".

The *radius* of a set of Pareto optimal solutions $S \subseteq PF$, denoted as $\Omega(S)$, is defined as

$$\Omega(S) = \max_{p' \in PF} \min_{p \in S} dist(p, p').$$

We say that a set $S \subseteq PF$ is *optimally representative of* PF (*representative* for short) if S has a minimal radius among all sets $S' \subseteq PF$ such that $|S'| = |S|$. When $|S| = k$ and S is a representative set, we also say that S is a k-representative set. Intuitively, a representative set S is such that every Pareto optimal solution is as close as possible to some solution of S. Our ultimate goal is given a bi-objective optimisation problem and positive integer k, to find a k-representative set of solutions.

Computing a Representative Set When the Pareto Front is Available

We introduce a procedure for computing a k-representative set of solutions, given k and assuming that the Pareto front is also given e.g. when it has been computed as an upstream step. The advantage of our procedure is that it can then be adapted to the case where the Pareto front is not given explicitly but by means of a set of variables V and a set of constraints C. This will be shown in the next section, roughly speaking, by substituting some of the operations with solver calls. The signature of our procedure is as follows:

Definition 1 ($RepSet(PF, k)$)

- ***Input:*** *Pareto front PF, integer k.*

- ***Output:*** *A k-representative set of solutions.*

The key step of our procedure is, given a *fixed* radius R, to compute a k'-representative set S such that $\Omega(S) \leqslant R$ while minimizing k'. This step is denoted hereafter by RS-$R(PF, R)$, where R is a radius and PF is a *sorted* Pareto front, i.e. a Pareto front where all solutions are sorted in a non-decreasing order according to f_x.

The following helper methods are used in RS-$R(PF, R)$:

- $furthest\_within\_radius(p, PF, R)$, or $fwr(p, PF, R)$ for short: given a Pareto front PF, a solution $p \in PF$, and a radius R, it is defined as

$$fwr(p, PF, R) = \arg\max_{p' \in PF}\{f_x(p') \mid dist(p, p') \leqslant R\}.$$

  Note that the method is guaranteed to produce a solution: the reference solution p can be returned.

- $filter(p, PF, R)$: given a Pareto front PF, a solution $p \in PF$ and a radius R, it computes a new Pareto front PF', which contains solutions from PF that are distant from p by at least R, i.e. it is defined as

$$filter(p, PF, R) = \{p' \mid p' \in PF \wedge dist(p, p') > R\}.$$

- $p^x_{min}(PF)$: given a Pareto front PF, it is defined as

$$p^x_{min}(PF) = \arg\min_{p \in PF}\{f_x(p)\}.$$

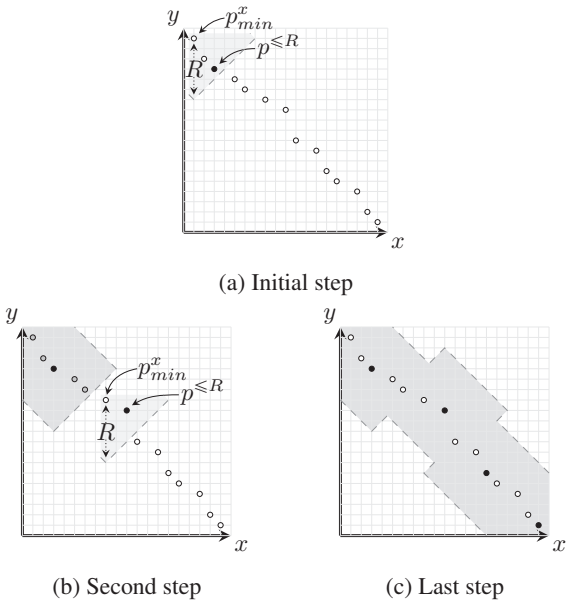
Algorithm 1: RS-$R(PF, R)$

input: Sorted Pareto front PF, radius R
output: A k'-representative set S such that $\Omega(S) \leqslant R$ and k' is minimal

```
1 begin
2     S ← ∅
3     while PF ≠ ∅ do
4         p^{≤R} ← fwr(p^x_min(PF), PF, R)
5         S ← S ∪ {p^{≤R}}
6         PF ← filter(p^{≤R}, PF, R)
7     return S
```

The method RS-$R(PF, R)$ is described in Algorithm 1. The representative set S is initially set to empty (line 2). The first solution $p^{\leqslant R}$ to be added to S is defined as the most distant solution from the leftmost solution $p^x_{min}(PF)$ within the distance range R (lines 4 and 5). Then all solutions within the range of R from the selected solution $p^{\leqslant R}$ are removed from the Pareto front (line 6), and the process is repeated until the Pareto front becomes empty. The method is illustrated in Fig. 2 (a-c) through an example. In the figures, $p^{\leqslant R}$ and p^x_{min} denote respectively the solutions $p^{\leqslant R}$ and $p^x_{min}(PF)$ at the corresponding step (lines 4 and 5 in Algorithm 1), i.e. when PF is updated accordingly (line 6 from the previous iteration in Algorithm 1).

Proposition 1 *Algorithm 1 is correct, i.e. it computes a k'-representative set RS-$R(PF, R)$ of solutions S such that $\Omega(S) \leqslant R$ and k' is minimal.*



(a) Initial step

(b) Second step

(c) Last step

Figure 2: Illustration of Algorithm 1

Proposition 2 *Algorithm 1 runs in $k' \cdot log(|PF|))$ time, where k' is the cardinality of the smallest representative set S with $\Omega(S) \leqslant R$.*

We are now ready to describe our main procedure $RepSet(PF, k)$, which uses the method RS-$R(PF, R)$ iteratively by performing a dichotomic search on the radius between two bounds. At each such iteration step, 1) the radius R is increased or decreased depending on whether the value k' resulting from the computation in the previous step is lower or greater than the required number k; and 2) the two bounds are updated accordingly. The procedure $RepSet(PF, k)$ terminates once the two bounds meet.

The following helper methods are used:

- $p^x_{max}(PF)$: given a Pareto front PF, it is defined as

$$p^x_{max}(PF) = \arg\max_{p \in PF}\{f_x(p)\}.$$

- $sort(PF)$: given a Pareto front PF, it sorts the solutions from PF in a non-decreasing order according to f_x.

The procedure $RepSet(PF, k)$ is described in Algorithm 2. We assume integer-valued objectives. Initially, the Pareto front is sorted in a non-decreasing order according to f_x (line 2). This step allows one to reduce the time complexity of the subsequent methods. The bounds are initially set in lines 3 and 4 for the purpose of efficiency, i.e. in such a way that the initial radius defined in line 7 implies the representative set computed at the first iteration step (line 8) to contain at most k solutions. Given these bounds, the algorithm performs a classical dichotomic search.

Proposition 3 *Algorithm 2 is correct, i.e. it computes a k-representative set $RepSet(PF, k)$.*

Proposition 4 *Algorithm 2 runs in $\mathcal{O}(k \cdot log(|PF|) \cdot log(R_{UB}) + |PF| \cdot log(|PF|))$, where $R_{UB} = \lfloor \frac{dist(p^x_{min}, p^x_{max})}{k} \rfloor$.*

Algorithm 2: $RepSet(PF, k)$

input: Pareto front PF of an bi-objective optimisation problem with integer-valued objectives, integer k
output: A k-representative set

1 **begin**
2 $PF \leftarrow sort(PF)$
3 $ub \leftarrow \lfloor \frac{dist(p^x_{min}(PF), p^x_{max}(PF))}{k} \rfloor$
4 $lb \leftarrow 0$
5 $S_{best} \leftarrow \varnothing$
6 **while** $lb \leqslant ub$ **do**
7 $middle \leftarrow \lfloor \frac{lb+ub}{2} \rfloor$
8 $S \leftarrow RS\text{-}R(PF, middle)$
9 **if** $|S| > k$ **then**
10 $lb \leftarrow middle + 1$
11 **else**
12 $S_{best} \leftarrow S$
13 $ub \leftarrow middle - 1$
14 **return** S_{best}

For rational objective functions, the algorithm needs to be modified to use dichotomic search defined on rational numbers (Kwek and Mehlhorn 2003). In this case, the complexity would substitute R_{UB} with $max(p, q)$, where p and q are the maximum denominator and numerator out of all considered rational numbers. Alternatively, the floor functions can be removed from lines 3 and 7, an up-to-precision small $\epsilon > 0$ can be used instead of the constant 1 in lines 10 and 13, and the procedure could stop once the lower and upper bounds are within a predefined distance (line 6).

As in (Vaz et al. 2015), the obtained complexity is dominated by the complexity of sorting. The advantage of our algorithm, however, is that it can be extended to the implicitly-defined Pareto front, as shown in the next section.

Computing a Representative Set in Constraint-Based Bi-Objective Optimisation

We adapt the procedure $RS\text{-}R(PF, R)$ presented in the previous section to the case when the Pareto front is not available, but instead succinctly characterised by means of a bi-objective optimisation problem. The adapted procedure, denoted now by $RS\text{-}R2(\langle X, C, f_x, f_y\rangle, R)$, follows the same scheme as presented in Algorithm 1, but the helper functions fwr, $filter$, and p^x_{min} used in defining the method $RS\text{-}R$ must be updated since the Pareto front is no longer available. The following helper methods are used in $RS\text{-}R_2(\langle X, C, f_x, f_y\rangle, R)$. These methods are similar as presented in the previous section, but updated to new signatures.

Helper method $fwr2(p^{ref}, \langle X, C, f_x, f_y\rangle, R)$**:** Given a Pareto optimal solution $p^{ref} = (p^{ref}_x, p^{ref}_y)$, a bi-objective optimisation problem $\langle X, C, f_x, f_y\rangle$, and a radius R, the task is to compute the Pareto optimal solution $p^{\leqslant R}$:

$$p^{\leqslant R} = \arg\max_{p' \in PF} \{f_x(p') \mid dist(p^{ref}, p') \leqslant R\}.$$

The Pareto front PF is not given. We do not need to compute it entirely, but rather solve a series of optimisation problems. Prior to this, let us discuss necessary subproblems.

To compute $p^{\leqslant R}$, we only need to consider as candidates the solutions p' that are not dominated by other solutions p such that $dist(p^{ref}, p) > R$. Ensuring Pareto optimality is one of the main challenges. This is done by initially computing a bound for the second objective f_y for Pareto optimal solutions p' such that $dist(p^{ref}, p') \leqslant R$. The bound is used to compute the Pareto optimal solution $p^{>R}$ that is the closest one to $p^{\leqslant R}$ but *outside* the range of p^{ref} w.r.t. the radius R. Lastly, we compute the solution $p^{\leqslant R}$ by searching for the Pareto optimal solution that is maximal w.r.t. f_x while remaining within the range of p^{ref} w.r.t. the radius R and nondominated by $p^{>R}$. The bound is computed as:

$$y_{bound} = \min_{p \in C*} f_y(p) \tag{1}$$

$$p^{ref}_y - f_y(p) + f_x(p) - p^{ref}_x \leqslant R \tag{2}$$

Eq. 2 constrains the solution to be within the range of p^{ref} w.r.t. the radius R. Please note that the distance is defined as a norm, but we simplified the equation by taking into account that solutions are computed in increasing f_x value. The Pareto optimal solution $p^{>R}$ that is the closest one to $p^{\leqslant R}$ but outside the range of $p^{\leqslant R}$ w.r.t. the radius R, is computed using lexicographical optimisation:

$$p^{>R} = \arg\min_{p \in C*} f_x(p) : \arg\min_{p \in C*} f_y(p) \tag{3}$$

$$p^{ref}_y - f_y(p) + f_x(p) - p^{ref}_x > R \tag{4}$$

$$f_y(p) < y_{bound} \tag{5}$$

Please note that the solution $p^{>R}$ is Pareto optimal, since we have that $p^{>R}_y < y_{bound}$ and $p^{>R}$ is minimised. The last step is to compute the Pareto optimal solution $p^{\leqslant R}$ using lexicographical optimisation:

$$p^{\leqslant R} = \arg\min_{p \in C*} f_y(p) : \arg\min_{p \in C*} f_x(p) \tag{6}$$

$$p^{ref}_y - f_y(p) + f_x(p) - p^{ref}_x \leqslant R \tag{7}$$

$$f_x(p) < p^{>R}_x \tag{8}$$

Eq. 7 constrains the resulting solution $p^{\leqslant R}$ to be within the range of p^{ref} w.r.t. the radius R, while Eq. 8 ensures the solution is not dominated by $p^{\leqslant R}$.

Note that minimising f_x or f_y within or outside of radius R is not enough to guarantee Pareto optimality. As previously discussed, the Pareto optimal solution $p^{>R}$, which is the closest Pareto optimal solutions to p^ref that is outside of radius R, is used in the computation of $p^{\leqslant R}$, the further Pareto optimal solution from p^{ref} (with greater f_x) within radius R. To see that importance of the solution $p^>R$, consider a bi-optimisation problem that implicitly defines points $(2, 4), (3, 3), (4, 2), (4, 0)$ and let $p^{ref} = (2, 4)$ and $R = 4$. The solution $p^{\leqslant R} = (3, 3)$, but merely minimising f_y within

radius R yields $(4, 2)$, which is not correct since it is dominated by $p^{>R} = (4, 0)$. However, to compute $p^{>R}$ we need *ybound* (Eq. 1-2), i.e., the minimum f_y within the radius. Consider another example with $(2, 4), (3, 1), (4, 1), (5, 0)$ and $p^{ref} = (2, 4)$ and $R = 4$. Minimising f_x outside radius R produces $(4, 1)$, which is dominated by $(3, 1)$, but the correct solution $p^{>R}$ is $(5, 0)$. Once *ybound* is enforced, $p^{>R} = (5, 0)$ is correctly computed.

As in the explicit fwr case, the method $fwr2$ is guaranteed to produce a solutions since the reference solution p^{ref} can be returned.

Helper method $filter2(p^{ref}, \langle X, C, f_x, f_y\rangle, R)$**:** Given a Pareto optimal solution p^{ref}, a bi-objective optimisation problem $\langle X, C, f_x, f_y\rangle$, and a radius R, the task is to solve a restriction $\langle X, C', f_x, f_y\rangle$ of the problem, i.e. $C \subseteq C'$, so that to discard the Pareto optimal solutions from $C*$ that are within the range of p^{ref} w.r.t. the radius R. The updated set C' of constraints is defined as $C' = C \cup c_1 \cup c_2$, where c_1, c_2 are defined as follows:

$$c_1 : p_y^{ref} - f_y(p) + f_x(p) - p_x^{ref} > R \tag{9}$$

$$p^{\leqslant R} = fwr2(p^{ref}, \langle X, C, f_x, f_y\rangle, R) \tag{10}$$

$$c_2 : f_y(p) < p_y^{\leqslant R} \tag{11}$$

Eq. 11 enforces that the solutions are not dominated by $p^{\leqslant R}$, while Eq. 9 ensures that the solutions are outside the range of p^{ref} w.r.t. the radius R.

Helper method $p2^x_{min}(\langle X, C, f_x, f_y\rangle)$**:** Given a bi-objective optimisation problem $\langle X, C, f_x, f_y\rangle$, the task is to compute the Pareto optimal solution $p2^x_{min}$ with a minimum value for f_x. This is done by a lexicographical problem:

$$p2^x_{min} = \underset{p \in C*}{\arg\min}\ f_x(p) : \underset{p \in C*}{\arg\min}\ f_y(p) \tag{12}$$

The procedure $RS\text{-}R2(\langle X, C, f_x, f_y\rangle, R)$, which computes the representative set given a bi-objective optimisation problem $\langle X, C, f_x, f_y\rangle$, can be analogously defined as in Algorithm 1 using the helper methods defined in this section. In a similar fashion, $RepSet2(PF, k)$ can be derived from Algorithm 2 by using $RS\text{-}R2$. Note that none of the steps in the algorithm requires to compute the Pareto front PF of the bi-objective optimisation problem $\langle X, C, f_x, f_y\rangle$, but the resulting output is the k-representative set.

Proposition 5 *The procedure* $RepSet2(\langle X, C, f_x, f_y\rangle, k)$ *computes a* k*-representative set by using* $\mathcal{O}(k \cdot log(R_{UB}))$ *solver calls, where* $R_{UB} = \lfloor \frac{dist(p^x_{min}, p^x_{max})}{k} \rfloor$.

Note that R_{UB} represents a value that is exponential in the input size, but the overall complexity remains polynomial as it relies on $log(R_{UB})$. The proof of the proposition is analogous to Prop. 2, but individual elements of the Pareto front are computed using solver calls rather than accessed in constant time.

An interesting consequence of Prop. 5 is that focussing on the bi-objective case to compute representative solutions leads to a computational shift in comparison to the general multi-objective case. Indeed, let us consider the decision problem related to the computation of representative solutions introduced in (Schwind et al. 2016):

Definition 2 (DP2 **(Schwind et al. 2016))** *Given a multi-objective optimisation problem* $P = \langle X, C, F\rangle$ *where* F *is a set of objective functions, and two integers* α*,* k *where* k *is bounded by a polynomial in the size of* P*, does there exist* $S \subseteq PF$ *such that* $|S| = k$ *and* $\Omega(S) \leqslant \alpha$*?*

Proposition 6 ((Schwind et al. 2016)) DP2 *is* $\mathsf{NP}^{\mathsf{NP}}$*-hard.*

Proposition 7 *If* $P = \langle X, C, F\rangle$ *and* F *consists of at most two objective functions,* DP2 *is in* P^{NP}.

This result shows that, even though there may be an exponential number of Pareto optimal solutions, computing the representative set only requires at most a polynomial number of solver calls, regardless of the size of the Pareto front. Thus, computing a representative set is done more efficiently than enumerating the Pareto front. This also shows that for linear bi-objective programs, representative solutions can be computed in *polynomial time* despite infinite-sized Pareto fronts. To the best of our knowledge, this is the first result of this kind for bi-objective optimisation, as previous works compute either an approximate set or the complete Pareto front. We illustrate the practicality of our algorithm in the next section.

Experimental Results

We implemented our algorithms and performed a numerical study. The goal was to verify the theoretical result and evaluate the empirical performance. The computational study has a supporting role and is meant as a proof-of-concept rather than a rigorous comparison of solving paradigms and techniques, which is out of the scope of the paper. To the best of our knowledge, this is the first time a theoretical result (Prop. 5) and algorithm have been provided for computing provably optimal representative solutions without resorting to a complete Pareto front enumeration. Possibly better results could be achieved with further algorithmic improvements, in particular by exploiting problem-specific features, but this does not demean our main results: a theoretical result and principled algorithms to compute representative solutions for bi-objective optimisation.

Our code and benchmarks are available online: *bitbucket.org/EmirD/representative-solutions-for-bi-objective-optimisation*. The input format for our program is an MPS file. MiniZinc (Nethercote et al. 2007) users can convert MZN/DZN files to the MPS format using the built-in conversion feature of MiniZinc.

Computational Setting and Benchmarks

Experiments were performed on a machine with an i7-7700HQ CPU @ 2.80GHz processor and 32 GB of RAM, running one instance at a time with a time limit of ten hours. We used Gurobi as the optimisation solver and experimented with two benchmark families with varying instance size:

- *Resource-constrained project scheduling problems with weighted earliness and tardiness objectives*, labelled as RCPSP-wet in the MiniZinc Challenge 2016 and 2017. The instances are naturally bi-objective as the objective consists of two parts: earliness and tardiness. For ease of presentation we partitioned the benchmarks in three groups: RCPSP-30, -60, and -90, containing six small, three medium, and three large instances. We excluded the large *j90-10* benchmark as none of the techniques could produce meaningful results within ten hours.
- Generated large bi-objective *set covering* benchmarks. Similar instances were used in other single- and multi-objective works (Musliu 2006; Bergman and Cire 2016). The characteristics of the benchmarks are as follows: the number of columns $n = \{1000, 2000\}$, number of rows $\frac{n}{5}$, each row can be covered by ten ($n = 1000$) or twenty ($n = 2000$) randomly selected columns, and each column has a random weight from the interval $[1, 1000]$ for each objective. The task is to cover each row while minimising the cost of the selected columns as a bi-objective problem. We created twenty benchmarks, ten for each value of n.

In the following, we consider the implicit Pareto front case. For the explicit case, the k-representative solutions can be computed in seconds, as the dominating complexity factor is sorting, and is thus not further considered.

Results and Discussion

We compare our approach with two techniques that compute the Pareto front: the ϵ-method (Haimes 1971) and the two-phase approach (Ulungu and Teghem 1995). Although somewhat old, these techniques are the two most widely used generic methods for Pareto front enumeration in bi-objective optimisation for integer programming according to a recent survey (Stidsen, Andersen, and Dammann 2014). As the time required to compute a representative set given a Pareto front is negligible for bi-objective problems, we compare against generating the complete Pareto front. In further text, we discuss the results in comparison to the ϵ-method since it was an order-of-magnitude faster than the two-phase approach in our experiments. The discrepancy is likely because we used a generic implementation of the second phase in the two-phase approach, which is typically tailored to particular applications. We would like to note that we are not aware of other techniques that directly compute the representative set: this is our main contribution.

The results are given in Table 1. We consider values $[1, 2, 3, 4, 5]$ for k, the number of desired representative solutions. There is a correlation between k and the runtime of our approach. Similarly, the number of iterations are kept low (recall that each iteration solves a decision problem). These observations are in accordance with the computational complexity (Prop. 5).

Our method uses a fewer number of solver calls compared to Pareto front enumeration for the considered benchmarks. This results in better performance. The conclusion holds irrespective of k, the size of the representative set. The gain in time is roughly proportional to the difference in the number of solver calls. Furthermore, the number of intermediate solutions generated throughout the search is typically considerably smaller than the number of Pareto optimal solutions. These results are in line with our main claim, i.e., computing the representative solutions can be done efficiently without resorting to the complete Pareto front.

				time (seconds)			# of NP-hard calls		additional stats		
Bench	#	k	$\|PF\|$	ϵ-PF	k-Rep	k-Relax	ϵ-PF	k-Rep	#int	#unique	#iter
SC(n=1,000)	10	1	703	678	72	1	1,407	83	34	19	17
		2			115	3		198	76	41	16
		3			170	4		307	117	60	16
		4			222	5		406	154	85	15
		5			264	7		495	187	99	15
SC(n=2,000)	10	1	2,095	13,541	460	4	4,191	88	36	22	18
		2			535	8		210	81	49	17
		3			795	14		321	122	76	16
		4			1,088	17		432	164	100	16
		5			1,382	22		530	200	124	16
RCPSP-30	6	1	66	56	13	1	133	43	18	12	9
		2			22	3		96	37	24	8
		3			30	4		137	52	31	8
		4			43	5		183	70	40	7
		5			56	6		221	83	44	7
RCPSP-60	3	1	244	2,632	511	14	489	52	21	16	11
		2			391	29		125	49	36	10
		3			615	57		166	63	48	9
		4			788	64		220	84	65	9
		5			898	70		300	114	83	9
RCPSP-90*	2	1	507	9,334	514	144	1,015	64	26	20	13
		2			1,551	346		133	51	38	11
		3			2,041	706		214	82	59	11
		4			2,101	872		256	97	74	10
		5			3,210	1,036		337	128	93	10

Table 1: Comparison of the ϵ-method and our k-representative approach. The column $'\#'$ shows the number of benchmarks; $|PF|$ is the size of the PF; k-Relax refers to the k-representative solution of the linear relaxation; $\#int$ and $\#unique$ is the number of intermediate and unique solutions found during the search; $\#iter$ denotes the number of iterations. Results are averaged. *The Pareto front could not be generated for one benchmark (*j-90-19*) within ten hours, but we could compute the representative solutions.

During the search, the same Pareto optimal solution can be computed more than once, i.e., the number of unique solutions is smaller than the number of intermediate solutions. This indicates an overlap between the iterations.

We considered the linear relaxation of the benchmarks, i.e. the integrality constraints of the variables are relaxated to continous variables. In these instances, the Pareto front is of infinite size, but our approach can compute the k-representative set. The lower run time for the relaxed instances is expected, given that in general linear programs are easier to solve than integer programs.

Conclusion

We provide a novel algorithm for computing the representative solutions for bi-objective optimisation. We show an improved complexity result over the general multi-objective case. Moreover, the numerical study illustrates the practicality of our approach. For future work, we would reduce potential redundancy in subsequent iterations of our algorithm by taking into account previously computed solutions, and incorporate other properties of Pareto optimal solutions, such as those proposed in (Bazgan, Jamain, and Vanderpooten 2017), to further refine the representative set.

References

Adomavicius, G., and Kwon, Y. 2014. Optimization-based approaches for maximizing aggregate recommendation diversity. *INFORMS Journal on Computing* 26(2):351–369.

Bazgan, C.; Jamain, F.; and Vanderpooten, D. 2017. Discrete representation of the non-dominated set for multi-objective optimization problems using kernels. *European Journal of Operational Research* 260(3):814–827.

Bergman, D., and Cire, A. A. 2016. Multiobjective optimization by decision diagrams. In *Proc. of the 22nd International Conference on Principles and Practice of Constraint Programming (CP'16)*, 86–95. Springer.

Caillouet, C.; Li, X.; and Razafindralambo, T. 2011. A multi-objective approach for data collection in wireless sensor networks. In *Proc. of the 10th International Conference on Ad-hoc, Mobile, and Wireless Networks (ADHOC-NOW'11)*, 220–233.

Ceyhan, G.; Köksalan, M.; and Lokman, B. 2019. Finding a representative nondominated set for multi-objective mixed integer programs. *European Journal of Operational Research* 272(1):61–77.

Dhar, R. 1997. Consumer Preference for a No-Choice Option. *Journal of Consumer Research* 24(2):215–231.

Eusébio, A.; Figueira, J.; and Ehrgott, M. 2014. On finding representative non-dominated points for bi-objective integer network flow problems. *Computers and Operations Research* 48:1–10.

Gavanelli, M. 2002. An algorithm for multi-criteria optimization in CSPs. In *Proc. of the 15th European Conference on Artificial Intelligence (ECAI'02)*, volume 2, 136–140.

Haimes, Y. 1971. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE transactions on systems, man, and cybernetics* 1(3):296–297.

Hebrard, E.; Hnich, B.; O'Sullivan, B.; and Walsh, T. 2005. Finding diverse and similar solutions in constraint programming. In *Proc. of the 20th National Conference on Artificial Intelligence (AAAI'05)*, 372–377.

Iyengar, S. S., and Lepper, M. R. 2000. When choice is demotivating: Can one desire too much of a good thing? *Journal of personality and social psychology* 79(6):995.

Karasakal, E., and Köksalan, M. 2009. Generating a representative subset of the nondominated frontier in multiple criteria decision making. *Operations research* 57(1):187–199.

Kwek, S., and Mehlhorn, K. 2003. Optimal search for rationals. *Information Processing Letters* 86(1):23–26.

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

Musliu, N. 2006. Local search algorithm for unicost set covering problem. In *Proc. of the 19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE)'06*, 302–311.

Nethercote, N.; Stuckey, P. J.; Becket, R.; Brand, S.; Duck, G. J.; and Tack, G. 2007. Minizinc: Towards a standard cp modelling language. In *Proc. of the 13th International Conference on Principles and Practice of Constraint Programming (CP'07)*, 529–543. Springer.

Papadimitriou, C. H. 2003. *Computational complexity*. John Wiley and Sons Ltd.

Petit, T., and Trapp, A. C. 2015. Finding diverse solutions of high quality to constraint optimization problems. In *Proc. of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, 260–267.

Petit, T., and Trapp, A. C. 2019. Enriching solutions to combinatorial problems via solution engineering. *INFORMS Journal on Computing* 31(3):429–444.

Raith, A., and Sedeño-Noda, A. 2017. Finding extreme supported solutions of biobjective network flow problems: An enhanced parametric programming approach. *Computers and Operations Research* 82:153 – 166.

Sayin, S. 2000. Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming. *Mathematical Programming* 87:543–560.

Schaus, P., and Hartert, R. 2013. Multi-objective large neighborhood search. In *Proc. of the 19th International Conference on Principles and Practice of Constraint Programming (CP'13)*, 611–627.

Schwind, N.; Okimoto, T.; Clement, M.; and Inoue, K. 2016. Representative solutions for multi-objective constraint optimization problems. In *Proc. of the 15th International Conference of Principles of Knowledge Representation and Reasoning (KR'16)*, 601–604.

Shafir, E.; Simonson, I.; and Tversky, A. 1993. Reason-based choice. *Cognition* 49(1-2):11–36.

Stidsen, T.; Andersen, K. A.; and Dammann, B. 2014. A branch and bound algorithm for a class of biobjective mixed integer programs. *Management Science* 60(4):1009–1032.

Trisna, T.; Marimin, M.; Arkeman, Y.; and Sunarti, T. 2016. Multi-objective optimization for supply chain management problem: A literature review. *Decision Science Letters* 5(2):283–316.

Ulungu, E. L., and Teghem, J. 1995. The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences* 20(2):149–165.

Vaz, D.; Paquete, L.; Fonseca, C. M.; Klamroth, K.; and Stiglmayr, M. 2015. Representation of the non-dominated set in biobjective discrete optimization. *Computers and Operations Research* 63:172–186.

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 Transactions on evolutionary computation* 7(2):117–132.