Approximate Pareto sets of minimal size for multi-objective optimization problems
Cristina Bazgan, Florian Jamain, Daniel Vanderpooten

To cite this version:

HAL Id: hal-01505524
https://hal.archives-ouvertes.fr/hal-01505524
Submitted on 11 Apr 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Approximate Pareto sets of minimal size for multi-objective optimization problems

Cristina Bazgan\textsuperscript{a,b}, Florian Jamain\textsuperscript{a}, Daniel Vanderpooten\textsuperscript{a}

\textsuperscript{a}PSL, Université Paris-Dauphine, LAMSADE UMR 7243
Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France
\textsuperscript{b}Institut Universitaire de France

Abstract

We are interested in a problem introduced by Vassilvitskii and Yannakakis [12], the computation of a minimum set of solutions that approximates within an accuracy $\varepsilon$ the Pareto set of a multi-objective optimization problem. We mainly establish a new $3$-approximation algorithm for the bi-objective case. We also propose a study of the greedy algorithm performance for the tri-objective case when the points are given explicitly, answering an open question raised by Kolm and Papadimitriou in [9].

Keywords: Multi-objective optimization, Pareto set, non-dominated points, approximation algorithm, greedy algorithm.

1. Introduction

In multi-objective optimization, in opposition to single objective optimization, there is typically no optimal solution i.e. one that is best for all the objectives. Therefore, the standard situation is that any solution can always be improved on at least one objective. The solutions of interest, called efficient solutions, are such that any solution which is better on one objective is necessarily worse on at least one other objective. In other words, a solution is efficient if its corresponding vector of objective values is not dominated by any other vector of objective values corresponding to a feasible solution. These vectors, associated to efficient solutions, are called non-dominated points. For many multi-objective optimization problems, one of the main difficulties is the large cardinality of the set of non-dominated points (or Pareto set). Indeed, it is well-known, in particular, that most multi-objective combinatorial optimization problems are intractable, in the sense that they admit families of instances for which the number of non-dominated points is exponential in the size of the instance [4]. Thus, instead of producing the full set of non-dominated points, we may prefer to provide an approximation of this set. This idea is represented by the concept of an $\varepsilon$-Pareto set, which is a set $P_{\varepsilon}$ of solutions that approximately dominates every other solutions, i.e. such that for every solution $s$, it contains a solution $s'$ that is better within a factor $1+\varepsilon$ than $s$ in all the objectives. The existence of $\varepsilon$-Pareto sets of polynomial size is well-known [10] and polynomial time algorithms that produce $\varepsilon$-Pareto sets have been developed and improved for many multi-objective optimization problems, including Multi-objective Shortest Path [7, 13, 11], Multi-objective Knapsack [5, 1]. However, note that there may exist many $\varepsilon$-Pareto sets, some of which can have very small size and some others very large size. An interesting problem introduced by [12] and continued in [3] is the efficient construction of $\varepsilon$-Pareto sets of size as small as possible. This paper focuses on the same issue.

In the following section, we define the basic concepts, formalize the problem and recall some results of previous related works. Then, in section 3, we mainly propose a new polynomial time 3-approximation algorithm of the size of a smallest $\varepsilon$-Pareto set for the bi-objective case. In section 4, we analyze the performance of the greedy algorithm when the points of the objectives space are given explicitly in the input and the number of objectives is three, answering an open question raised in [9]. We conclude with some possible extensions to this work.

2. Preliminaries

In this paper, we consider multi-objective optimization problems where we try to minimize several objectives, i.e. $\min_{x \in S} (f_1(x), \ldots, f_p(x))$, where $f_1, \ldots, f_p$ are $p \geq 2$ objective functions and $S$ is the set of feasible solutions. In the case where some or all objective functions are to be maximized, our results are directly extendable.

We distinguish the decision space $X$ which contains the set of feasible solutions of the instance and the criterion space $Y \subseteq \mathbb{R}^p$ which contains the criterion vectors or simply points. We denote by $Z = f(S) \subseteq Y$ the set of the images of feasible solutions called feasible points.

We denote by $y_i$ the coordinate on criterion $f_i$ of a point $y \in Y$ for $i = 1, \ldots, p$. We say that a point $y$ dominates another point $y'$ if $y$ is at least as good as $y'$ in all the objectives, i.e. $y_i \leq y_i'$ for all $i = 1, \ldots, p$. A feasible solution $x \in S$ is called efficient if there is no other feasible solution $x' \in S$ such that $f(x) \neq f(x')$ and $f(x')$ dominates $f(x)$. If $x$ is efficient, $z = f(x)$ is called a non-dominated point in the criterion space. We denote by $P$ the set of non-dominated points, called Pareto set.

Given a constant $c \geq 1$, a point $y$ dominates another point $y'$ if $y$ is at least as good as $y'$ up to a factor of $c$ in all the
objectives, i.e. $y_i \leq cy'_i$. For any rational $\varepsilon > 0$, an $\varepsilon$-Pareto set $P_\varepsilon$ is a subset of feasible points such that for all $z \in P$, there exists $z' \in P_\varepsilon$ such that $z' (1 + \varepsilon)$-dominates $z$. In the context of $\varepsilon$-Pareto sets, the central relation is the $(1 + \varepsilon)$-dominance relation, denoted by $\leq_\varepsilon$.

For a given instance $I$, there may exist several $\varepsilon$-Pareto sets, and these may have different sizes. It is shown in [10] that, for every classical multi-objective optimization problem, an $\varepsilon$-Pareto set of size polynomial in the input size and $1/\varepsilon$ always exists. Moreover its computation is related to the computation of the following routine $GAP_\varepsilon$.

Given an instance $I$ of a given problem, a point $y$ and a rational $\delta \geq 0$, the routine $GAP_\delta(y)$ either returns a feasible point that dominates $y$ or reports that there does not exist any feasible point $z$ such that $z_i \leq \frac{y_i}{1+\delta}$ for all $i = 1, \ldots, p$.

We say that routine $GAP_\delta(y)$ runs in polynomial time (resp. fully polynomial time) when $\delta > 0$ its running time is polynomial in $|I|$ and $|y|$ (resp. $|I|$, $|y|$, $|\delta|$ and $1/\delta$). An $\varepsilon$-Pareto set is computable in polynomial time (resp. fully polynomial time) if and only if the routine $GAP_\varepsilon$ runs in polynomial time (resp. fully polynomial time) [10].

Since an $\varepsilon$-Pareto set of polynomial size can still be quite large, Vassilvitskii and Yannakakis investigate in [12] the determination of $\varepsilon$-Pareto sets of minimal size. These authors also propose generic algorithms to deal with this problem. An algorithm is called generic if it does not depend on any particular problem and makes use of general purpose routines for which only the implementation is specific to the problem ($GAP_\varepsilon$ is such a general purpose routine). In such algorithms it is only required to have bounds on the minimum and maximum values of the objective functions. Assuming in the following that the objective functions take positive rational values whose numerators and denominators have at most $m$ bits, any feasible point has a value between $2^{-m}$ and $2^m$ and moreover the difference between the values of any two solutions is at least $2^{-2m}$ for any criterion. From [10], $opt_\varepsilon$ is polynomial in the input size and $1/\varepsilon$.

In order to use generic algorithms, Diakonikolas and Yannakakis introduced in [3] two other general purpose routines called $Restrict_\varepsilon$ and $DualRestrict_\varepsilon$ for the bi-objective case.

Given an instance $I$, a bound $b$ and a rational $\delta \geq 0$, the routine $Restrict_\varepsilon(f_1, f_2 \leq b)$ either returns a feasible point $z$ satisfying $z_2 \leq b$ and $z_1 \leq (1 + \delta)$, $\min f_1(x) : x \in S$ and $f_2(x) \leq b)$ or correctly reports that there does not exist any feasible point $z$ such that $z_2 \leq b$.

Given an instance $I$, a bound $b$ and a rational $\delta \geq 0$, the routine $DualRestrict_\varepsilon(f_1, f_2 \leq b)$ either returns a feasible point $z$ satisfying $z_2 \leq b(1 + \delta)$ and $z_1 \leq \min f_1(x) : x \in S$ and $f_2(x) \leq b)$ or correctly reports that there does not exist any feasible point $z$ such that $z_2 \leq b$.

We say that routine $Restrict_\varepsilon(f_1, f_2 \leq b)$ or $DualRestrict_\varepsilon(f_1, f_2 \leq b)$ runs in polynomial time (resp. fully polynomial time when $\delta > 0$) if its running time is polynomial in $|I|$ and $|b|$ (resp. $|I|$, $|b|$, $1/\delta)$.

Routines $Restrict_\varepsilon(f_1, f_2 \leq b)$ and $DualRestrict_\varepsilon(f_2, f_1 \leq b')$ are polynomially equivalent as proved in [3].

In the routines considered in this paper we assume that the error $\delta$ is a rational number, otherwise it is approximated from below by a rational number. We denote by $P_\varepsilon$ a smallest $\varepsilon$-Pareto set and by $opt_\varepsilon$ its cardinality. It follows from [10] that $opt_\varepsilon$ is polynomial in the input size and $1/\varepsilon$.

We are interested in generic algorithms that compute in polynomial time an $\varepsilon$-Pareto set of minimal size. For the bi-objective case, a generic algorithm that computes an $\varepsilon$-Pareto set of size at most $3opt_\varepsilon$ was established in [12] using routines $GAP_\varepsilon$. Moreover, if the routine $GAP_\varepsilon$ runs in polynomial time (resp. fully polynomial time) then the algorithm also runs in polynomial time (resp. fully polynomial time). Then, it is shown in [3] that an $\varepsilon$-Pareto set of size at most $2opt_\varepsilon$ is computable in polynomial time if there exists routines $Restrict_\varepsilon$ computable in polynomial time for both objectives. These approximation results are tight for the class of problems admitting such routines. An algorithm that computes an $\varepsilon$-Pareto set of size at most $k opt_\varepsilon$ is called a $k$-approximation algorithm.

3. Two objectives

We first present a hardness result for the Bi-objective Knapsack problem then we propose a new generic algorithm that approximates the size of a smallest $\varepsilon$-Pareto set to a factor 3, which is much simpler and, in some cases, more efficient than the one presented in [12].

3.1. Approximation hardness for Bi-objective Knapsack

Diakonikolas and Yannakakis [3] showed that the size of a smallest $\varepsilon$-Pareto set of Bi-objective Shortest Path and Bi-objective Spanning Tree cannot be approximated within a factor better than 2 in polynomial time, unless $P = NP$. These results are tight since these two problems admit a routine $Restrict_\varepsilon$ that runs in polynomial time, and thus an $\varepsilon$-Pareto set of size at most $2opt_\varepsilon$ is computable in polynomial time as shown in [3]. Vassilvitski and Yannakakis [12] showed that the size of a smallest $\varepsilon$-Pareto set of an artificial variant of Knapsack, called Bi-objective 2-Type-Knapsack, cannot be approximated within a factor better than 3 in polynomial time, unless $P = NP$. This result is also tight since this problem has a routine $GAP_\varepsilon$ that runs in polynomial time, and thus an $\varepsilon$-Pareto set of size at most $3opt_\varepsilon$ is computable in polynomial time as shown in [12].

In this part, we investigate the status of the classical version, called Bi-objective Knapsack, with as input a set $Q$ of items, a capacity $c$ and for each item $i$ two values $v_1(i)$, $v_2(i)$ and a weight $w(i)$. Values and weights are positive rationals. A solution is a nonempty subset $Q'$ of items with total values $v_1(Q') = \sum_{i \in Q'} v_1(i)$, $v_2(Q') = \sum_{i \in Q'} v_2(i)$ and a total weight $w(Q') = \sum_{i \in Q'} w(i) \leq c$. The goal is to maximize the values. First, note that the size of a smallest $\varepsilon$-Pareto set of Bi-objective Knapsack is approximable in polynomial time to a factor $3$ since this problem admits an FPTAS, which is equivalent to the existence of a polynomial time routine $GAP_\varepsilon$ [5]. We prove that the size of a smallest $\varepsilon$-Pareto set of Bi-objective Knapsack is not approximable in polynomial time within a factor better than $3$, if $P \neq NP$. 

2
Theorem 1. For Bi-objective Knapsack the size of a smallest \(s\)-Pareto set cannot be approximated within a factor better than 3 in polynomial time, unless \(P = NP\).

**Proof:** We construct a gap-preserving reduction from the Partition problem. Thus, from any instance \(I\) of Partition, we construct an instance \(I'\) of Bi-objective Knapsack such that if the answer of \(I\) is ‘yes’ then the size of the smallest \(s\)-Pareto set of \(I'\) is 1 and if the answer of \(I\) is ‘no’ then the size of the smallest \(s\)-Pareto set of \(I'\) is 3. Recall that in Partition, the input is a set \(N\) of \(n\) positive integers \(a_1, \ldots, a_n\), and we have to determine if it is possible to partition \(N\) into two subsets with equal sum. Starting with such an instance we construct an instance of Bi-objective Knapsack as follows. Let \(b = \sum_{i=1}^{n} a_i/2\). For each \(i = 1, \ldots, n\), we have one item \(i\) with values \(v_1(i) = v_2(i) = a_i\) and weight \(w(i) = a_i\). In addition, we have two special items \(\alpha\) and \(\beta\) with \(v_1(\alpha) = (1 + \epsilon)\beta\), \(v_2(\alpha) = 0\), \(w(\alpha) = b\) and \(v_1(\beta) = 0\), \(v_2(\beta) = (1 + \epsilon)b\), \(w(\beta) = b\). The capacity of the knapsack is \(b\).

Note that if a solution contains a special item, it cannot contain any feasible point with a special item getting the same value on each criterion, let \(z^∗\) be the point with the largest value \(v^∗\) on each criterion. \(z^∗\) dominates all other such points.

If \(I\) is a ‘yes’ instance, we have \(v^∗ = b\). Thus, \(z^∗(1 + \epsilon)\)-dominates both \(z^ε\) and \(z^β\), and \(z^∗\) is an \(s\)-Pareto set. If \(I\) is a ‘no’ instance, we have \(v^∗ < b\). Thus \(z^ε\) and \(z^β\) make part of any \(s\)-Pareto set and \(\{z^ε, z^β, z^∗\}\) is a smallest \(s\)-Pareto set. □

Remark that we can generalize the previous result, proving that for \(p\)-objective Knapsack with \(p \geq 2\) the size of a smallest \(s\)-Pareto set cannot be approximated within a factor better than \(p + 1\) in polynomial time, unless \(P = NP\).

3.2. A new 3-approximation algorithm

We propose in this section a new 3-approximation algorithm, based on another routine called SoftRestrict.

Given a positive rational bound \(b\) and a parameter \(\delta > 0\), the routine \(\text{SoftRestrict}(f_1, f_2 \leq b)\) either returns a feasible point \(z\) satisfying \(z_2 \leq 1 + \delta b\) and \(z_1 \leq (1 + \delta)\min(f_1(x): x \in S)\) and \(f_2(x) \leq b\) or correctly reports that there does not exist any feasible point \(z\) such that \(z_2 \leq b\).

We say that a routine \(\text{SoftRestrict}(f_1, f_2 \leq b)\) runs in polynomial time (resp. fully polynomial time when \(\delta > 0\)) if its running time is polynomial in \(|I|\) and \(|b|\) (resp. \(|I|\), \(|b|\), \(|I|\), \(|b|\)) and \(1/\delta\).

Remark that a routine \(\text{SoftRestrict}(f_1, f_2 \leq b)\), with a strict constraint, can easily be simulated by a routine \(\text{SoftRestrict}(f_1, f_2 \leq b')\) using \(b' = b - 2^{-nm}\).

Such a routine was proposed for several problems. For instance, for the Bi-objective Spanning Tree problem, the running time of the routine presented in [8] is \(O(nm^2r(|(n - 1)/\delta|, [(n - 1)/\delta]))\) where \(n\) is the number of vertices in the graph, \(m\) the number of edges in the graph and \(r(a, b)\) is the time to multiply polynomials of maximum degrees less than or equal to \(a\) and \(b\). For a Bi-objective Single Machine Scheduling problem, the running time of the routine presented in [2] is \(O(n^2R/(\delta^3L))\) where \(n\) is the number of jobs, \(L\) and \(R\) are respectively lower and upper bounds on the first coordinate of an optimal solution.

For Bi-objective Bipartite Matching, the running time of the routine presented in [6] is \(O(m^{\Theta(1/\delta^2)})\) where \(m\) is the number of edges in the graph.

Our approximation algorithm has the same approximation ratio as the algorithm presented in [12] but is much simpler, both in its description and in its proof, owing to the use of the routine \(\text{SoftRestrict}\) instead of \(\text{GAP}\). Its running time is comparable to the one of [12] and better under some conditions.

Before presenting and analyzing this new 3-approximation algorithm, we first compare the two routines \(\text{GAP}\) and \(\text{SoftRestrict}\).

**Proposition 1.** The routines \(\text{SoftRestrict}\) and \(\text{GAP}\) are polynomially equivalent.

**Proof:** We first show that we can answer to \(\text{GAP}\) using \(\text{SoftRestrict}(f_1, f_2 \leq y_2/(1 + \delta))\). Indeed, if \(\text{SoftRestrict}(f_1, f_2 \leq y_2/(1 + \delta))\) returns NO or returns a feasible point \(z\) such that \(z_1 > y_1\), we return NO and if \(\text{SoftRestrict}(f_1, f_2 \leq y_2/(1 + \delta))\) returns a feasible point \(z\) with \(z_1 \leq y_1\) we return \(z\).

We give in the following an algorithm that computes the function \(\text{SoftRestrict}(f_1, f_2 \leq b)\) using a polynomial number of calls to \(\text{GAP}\) where \(\delta = \sqrt{1 + \delta} - 1\). We first call \(\text{GAP}_g((1 + \delta)^2m, (1 + \delta)b)\). If it returns NO, then we also return NO for \(\text{SoftRestrict}(f_1, f_2 \leq b)\). Otherwise, we partition the objective space by defining intervals, on the first objective, from \(2^{-m}/(1 + \delta)\) to \(2^m\) such that the ratio between the upper and lower bounds of each interval is \(1 + \delta\). We perform a binary search on the upper bounds of the previous intervals calling \(\text{GAP}_g(a, (1 + \delta)\beta - b)\) for some \(a\) until one finds a value \(a^∗\) such that (i) \(\text{GAP}_g(a^∗(1 + \delta), (1 + \delta)b)\) returns a feasible point \(z^∗\) and (ii) \(\text{GAP}_g(a^∗, (1 + \delta)b)\) returns NO. Then we return \(z^∗\).

The number of subdivisions on the first coordinate is \(2m/\log(1 + \delta) = \Theta(4m/\delta)\). Hence, the number of calls to \(\text{GAP}_g\) is \(\Theta(\log(m/\delta)) = \Theta(\log(m/\delta))\).

**Corollary 1.** Consider the class of bi-objective problems that possess a fully polynomial time routine \(\text{SoftRestrict}\) with \(\delta > 0\) for both objectives. Then, for any \(\epsilon > 0\), there is no polynomial time generic algorithm using \(\text{SoftRestrict}\) that computes an \(s\)-Pareto set of size less than or equal to \(3\text{opt}_s\).

**Proof:** Follows from Proposition 1 and the fact that the same result holds for the routine \(\text{GAP}\) [12]. □

**Algorithm description.** We first describe briefly the idea of the algorithm. We compute \(f_1^\text{min}\) and \(f_2^\text{min}\) which represent lower bounds on the minimum values on the first and second objectives using \(\text{SoftRestrict}\). The algorithm iteratively generates a sequence of points \(q^1, q^2, \ldots, q^t, q^∗\). Points \(q^1, \ldots, q^t\) are selected in decreasing order according to \(f_1\) and increasing order according to \(f_2\). Point \(q^1\) is selected so as to \((1 + \epsilon)\)-dominate the feasible points that have an optimal second coordinate while getting the best possible value on \(f_1\). The algorithm stops when it generates a point \(q^∗\) that \((1 + \epsilon)\)-dominates the feasible points
that have a first coordinate equal to \( f_1^{\text{min}} \). Routines \( \text{SoftRestrict}_5(f_2, f_1 \leq b) \) and \( \text{SoftRestrict}_6(f_2, f_1 \leq b) \) are alternatively used to construct points \( r' \) and points \( q' \) respectively. Point \( r' \) is a point with a smallest second coordinate that we can determine with the routine \( \text{SoftRestrict}_5 \) that is not \((1 + \delta)/(1 + \delta)\)-dominated by the points \( q_j \) with \( j < i \). Point \( q' \) is a point with a smallest first coordinate that we can determine with routine \( \text{SoftRestrict}_6 \) that \((1 + \epsilon)\)-dominates point \( r' \). A formal description of this algorithm is given in Algorithm 1.

In order to obtain a 3-approximation algorithm, we consider in the following that \( \delta \leq \sqrt{1 + \epsilon} - 1 \). Before analyzing this algorithm, we illustrate its behavior in Figure 1 where 3 points \( q_1', q_2', q_3' \) are selected by Algorithm 1 in order to cover the part of the objective space with first objective value at least \( f_1^{\text{min}} \), whereas only one point, \( p^{1*} \), is sufficient.

Algorithm analysis. We show now that Algorithm 1 produces a 3-approximation of the size of a smallest \( \epsilon \)-Pareto set. Let \( Q = \{ q_1', \ldots, q_s' \} \) and \( R = \{ r_1', \ldots, r_s' \} \) be the sets of feasible points produced by the algorithm. We show in the following that set \( Q \) is an \( \epsilon \)-Pareto set, then that its size is at most three times the size of \( P^*_\epsilon \), an \( \epsilon \)-Pareto set of minimal size. The proof is essentially the same as the one in [3] for the 2-approximation algorithm. We first show some preliminary results regarding points in \( Q \) and \( R \).

**Lemma 1.** For all \( i = 2, \ldots, s \) we have (i) \( r_i' < q_i^{1^{-1}}/(1 + \epsilon) \) and (ii) for each feasible point \( z \) with \( z_1 < q_i^{1^{-1}}/(1 + \epsilon) \), we have \( z_2 \geq \max[f_2^{1^{-1}}, r_i'/(1 + \delta)] \).

**Proof:** This results from the definition of the routine \( \text{SoftRestrict}_5 \) and steps 10-12 and 15 of the algorithm. \( \square \)

**Lemma 2.** For all \( i = 1, \ldots, s \) we have (i) \( q_i^1 \leq (1 + \delta)f_2 \) and (ii) for each feasible point \( z \) with \( z_1 \leq f_2 \), we have \( z_1 \geq q_i^1/(1 + \delta) \).

**Proof:** This results from the definition of the routine \( \text{SoftRestrict}_6 \) and steps 10-12 of the algorithm. \( \square \)

We can now prove the following result.

**Proposition 2.** Set \( Q \) is an \( \epsilon \)-Pareto set.

**Proof:** We show that the points in \( Q \) cover all the feasible points by partitioning the range of feasible values on \( f_1 \). More precisely, we show that:

(i) Point \( q_i^1 \) \((1 + \epsilon)\)-dominates all the feasible points with an \( f_1 \) value greater than or equal to \( q_i^1/(1 + \epsilon) \).

(ii) For each \( i = 2, \ldots, s \) point \( q_i^1 \) \((1 + \epsilon)\)-dominates all the feasible points that have their \( f_1 \) value in the interval \([q_i^1/(1 + \epsilon), q_i^{1^{-1}}/(1 + \epsilon)]\).

(iii) There is no feasible point with a \( f_1 \) value smaller than \( q_i^1/(1 + \epsilon) \).

(i) Let \( z \) be a feasible point with \( z_1 \geq q_i^1/(1 + \epsilon) \). We need to show that \( z \) is \((1 + \epsilon)\)-dominated by \( q_i^1 \), i.e. that \( z_2 \geq q_i^1/(1 + \epsilon) \). From steps 2-4 we get where \( q_i^1 \leq r_i^1/(1 + \epsilon) \leq f_2^{\text{min}}/(1 + \epsilon) \) and thus \( q_i^1/(1 + \epsilon) \leq z_2 \).

(ii) Let \( z \) be a feasible point satisfying \( q_i^1/(1 + \epsilon) \leq z_1 < q_i^{1^{-1}}/(1 + \epsilon) \). We need to show that \( z \) is \((1 + \epsilon)\)-dominated by \( q_i^1 \), i.e. that \( z_2 \geq q_i^1/(1 + \epsilon) \). From Lemma 1-(ii) we have \( z_2 \geq \max[f_2^{1^{-1}}, r_i'/(1 + \delta)] \). Furthermore from Lemma 2-(ii) we have \( q_i^1 \leq (1 + \delta)f_2^{1^{-1}} \). Hence, from the definition of \( f_2^{1^{-1}} \) (step 11), we get \( q_i^1 \geq \max[f_2^{1^{-1}}, r_i'/(1 + \delta)] \leq (1 + \epsilon)z_2 \).

(iii) The stopping condition of the algorithm (step 8) is \( f_1 = q_i^{1^{-1}}/(1 + \epsilon) \leq f_1^{\text{min}} \).

We show now that the size of \( Q \) is at most three times the size of an optimal \( \epsilon \)-Pareto set.
Proposition 3. Set \( Q \) is such that \( |Q| \leq 3 \text{opt}_c \).

Proof : Let \( P_k = \{p^1, \ldots, p^k\} \) be an optimal \( \varepsilon \)-Pareto set, where its points \( p^i \) for \( i = 1, \ldots, k \) are in increasing order of their coordinates on \( f \) and decreasing order of their coordinates on \( f_1 \). We have to show that \( |Q| \leq 3k \). For this purpose, we show by induction on \( i \) that if the algorithm selects a feasible point \( q^{i-2} \), then there must exist a point \( p^i \) in \( P_k \), if the algorithm selects a point \( q^{i-1} \) then \( p^i_1(1+\delta) \geq q^i_1 \) and if the algorithm selects a point \( q^i \) then \( p^i_1 > (1+\delta)q^i_1 \).

Initialization \((i = 1)\). The first statement trivially holds. To prove the second statement we first show that \( p^2_1 \leq f_2^2 \). Since the feasible point \( r^1 \) must be \((1+\varepsilon)\)-dominated by a point of \( P_k \) and in particular by point \( p^1 \), we have \( p^2_1 \leq r^2_1(1+\varepsilon) \). From the definitions of \( f_2 \) and \( f_2 \) (step 11) and using that \( \delta < \sqrt{1+\varepsilon}-1 \), we have \( r^2_1(1+\varepsilon) = f_2^2(1+\delta) \leq f_2^2 \). Using that \( p^2_1 \leq f^2_2 \) \((\text{steps 13 and 14})\) Lemma 1(ii) we obtain \( r^1_1(1+\delta) \geq q^1_1 \). Since we have \( q^1_1 \leq r^1_1 \) \((\text{steps 13 and 14})\) Lemma 1(iii) \((\text{steps 13 and 14})\) implies \( q^i_1 < q^i_2(1+\delta)/(1+\varepsilon) \) and so that the third statement is a consequence of the second one, considering that \( \delta < \sqrt{1+\varepsilon}-1 \).

Induction step. Assume the result is true until index \( i - 1 \), we prove it for index \( i \). First, if the algorithm selects a point \( q^{i-2} \), we show that \( p^i_1 \leq f^i_{i-1} \). By the termination condition of the algorithm \((\text{step 8})\), we have \( q^{i-1} > q^{i-2} \) \((\text{step 11})\) and by the induction hypothesis that \( p^{i-1}_{r^i} > (1+\delta)q^{i-1}_{r} \), it follows that \( p^{i-1}_{r^i} > (1+\delta)(1+\varepsilon)q^{i-1}_{r} \). Thus, point \( p^{i-1}_{r^i} \) does not \((1+\varepsilon)\)-dominate the feasible points that have a minimum first coordinate, and so \( P_k \) must contain another point \( p^{i-1}_{\text{r}} \). To prove the second statement we first show that \( p^i_2 \leq f^i_{i-1} \). Since the feasible point \( r^{i-2} \) must be \((1+\varepsilon)\)-dominated by a point of \( P_k \) and in particular by point \( p^i \), we have \( p^i_2 \leq r^i_2(1+\varepsilon) \). From the definitions of \( f_2 \) and \( f_2 \) \((\text{step 11})\) and using that \( \delta < \sqrt{1+\varepsilon}-1 \), we have \( r^i_2(1+\varepsilon) = f_2^i(1+\delta) \leq f_2^i \). Using that \( p^i_2 \leq f^i_{i-1} \) \((\text{steps 13 and 14})\) Lemma 2(ii) we obtain \( p_1^i(1+\delta) \geq q_1^i \). Since we have \( q^i_1 \leq r^i_1 \) \((\text{steps 13 and 14})\) Lemma 1(i) \((\text{steps 13 and 14})\) implies \( q^i_1 < q^i_2(1+\delta)/(1+\varepsilon) \) and so that the third statement is a consequence of the second one, considering that \( \delta < \sqrt{1+\varepsilon}-1 \).

Combining the previous results, we obtain the main result of this section.

Theorem 2. Algorithm 1 computes an \( \varepsilon \)-Pareto set of size less than or equal to \( 3\text{opt}_c \) \((\text{steps } 1, 2, 3)\) routine calls to \( \text{SoftRestrict}_c \), when \( \delta \leq \sqrt{1+\varepsilon}-1 \).

Proof : The result is a direct consequence of Propositions 2 and 3. Since the algorithm uses \( 2|Q| \) times the routine \( \text{SoftRestrict}_c \), the number of routine calls is bounded by \( 6\text{opt}_c \).

Comparison to existing algorithms. For the class of problems admitting a routine \( \text{GAP} \) that runs in polynomial time, the algorithm of Vassilvitskii and Yannakakis presented in [12] was the only one ensuring some guarantee on the size of the returned \( \varepsilon \)-Pareto set. This algorithm, called \( \text{ZIGZAG} \), is a generic algorithm based on the routine \( \text{GAP} \), that establishes a \( \text{3-approximation} \) of the size of a smallest \( \varepsilon \)-Pareto set and it needs \( O(\text{opt}_c \cdot \log(m/e)) \) routine calls. Since algorithm \( \text{ZIGZAG} \) and our algorithm run in polynomial time for the same class of problems and give the same approximation ratio of a smallest \( \varepsilon \)-Pareto set we can compare them with regard to their running times. The running time of a generic algorithm is defined as the product between the number of routine calls and the running time of the routine called. This way, the running time of algorithm \( \text{ZIGZAG} \) is \( O(\text{opt}_c \cdot \log(m/e)) \cdot T_{\text{GAP}} \), with \( \delta = \sqrt{1+\varepsilon}-1 \) and the running time of our algorithm \( \text{SoftGreedy} \) is \( O(\text{opt}_c) \cdot T_{\text{SoftRestrict}_c} \), with \( \delta = \sqrt{1+\varepsilon}-1 \) where \( T_{\text{GAP}} \) and \( T_{\text{SoftRestrict}_c} \) are the running times of the routines \( \text{GAP}_c \) and \( \text{SoftRestrict}_c \), respectively.

The running times of algorithms \( \text{SoftGreedy} \) and \( \text{ZIGZAG} \) are comparable since we can solve the routine \( \text{SoftRestrict}_c \) using \( O(\log(m/e)) \) calls to \( \text{GAP}_c \) \((\text{see Proposition 1})\). Moreover, if we can solve faster the runtime \( \text{SoftRestrict}_c \), the running time of algorithm \( \text{SoftGreedy} \) will be smaller than the one of algorithm \( \text{ZIGZAG} \). Especially, if the best known algorithm to solve \( \text{GAP}_c \), \( \text{SoftRestrict}_c \), in the same time, we gain the time of the binary searches. It is the case for Bi-objective \( \text{Spanning Tree} \) since the algorithm in [8] that solves \( \text{SoftRestrict}_c \) is the best known to solve \( \text{GAP}_c \).

4. More than two objectives

Computing small \( \varepsilon \)-Pareto sets for problems with \( p \geq 3 \) objectives raises specific difficulties.

A first type of difficulties is related to the power or availability of the routines. In [12], Vassilvitski and Yannakakis showed that any generic algorithm based on routine \( \text{GAP}_c \) cannot establish a \( \text{c-approximation} \) of the size of a smallest \( \varepsilon \)-Pareto set for any constant \( c \). Using \( \text{SoftRestrict}_c \) instead of \( \text{GAP}_c \) leads to the same result since the two routines are polynomially equivalent (see Proposition 1, whose proof for 2 objectives can be generalized straightforwardly for \( p \) objectives).

Therefore, in order to obtain stronger results, it is necessary to resort to routines involving less tolerance, such as the extended versions, for \( p \) objectives, of \( \text{Restrict}_c \) or \( \text{DualRestrict}_c \). First remark that extended \( \text{Restrict}_c \) is more demanding than extended \( \text{DualRestrict}_c \) since the tolerance \( \delta \) is represented once only in the objective to be minimized in \( \text{Restrict}_c \) whereas it is present in the \( p - 1 \) constraints in \( \text{DualRestrict}_c \). This is confirmed by known results [6], where it is shown that, for \( p \geq 3 \), there is no polynomial-time routine \( \text{Restrict}_c \) for classical problems like \( \text{Multi-objective Shortest Path} \), \( \text{Multi-objective Spanning Tree} \), \( \text{Multi-objective Perfect Matching} \), whereas, for instance, there exists a polynomial-time routine \( \text{DualRestrict}_c \) for \( \text{Multi-objective Spanning Tree} \). Therefore, an interesting problem is to devise a generic procedure to generate an \( \varepsilon \)-Pareto set of smallest size for a \( p \)-criterion problem, using \( \text{DualRestrict}_c \).

A second type of difficulties is related to the way of exploring objective spaces with more than two objectives. It should be noticed that, in the bi-objective case, non-dominated points can
be sorted such that their objective values are increasing in one objective and decreasing in the other objective. This order is used in algorithms generating small $\varepsilon$-Pareto sets, including our algorithm SoftGreedy. The non-existence of such an order for $p \geq 3$ makes the exploration of the objective space much more difficult.

Owing to the previous difficulties, we focus in this section on the simple case where the feasible points are given explicitly in the input. In this case, we can easily filter out the dominated points and thus we consider in the following that the input contains non-dominated points only.

In this context, when $p = 3$, the problem of finding an $\varepsilon$-Pareto set of smallest size is $c$-approximable, for some constant $c$ [9]. Moreover, for any number of criteria, there exists an $O(\log n)$-approximation since our problem reduces to Set Cover, for which the greedy algorithm provides an $O(\log n)$-approximation. An interesting open question, raised by Kolton and Papadimitriou in [9], is to know whether the greedy algorithm performs better on the very specific instances of our problem. We provide an negative answer in Theorem 3. Recall that the greedy algorithm iteratively selects the point that covers the largest number of non-covered points.

Note that for 2 criteria, the greedy algorithm gives a 2-approximation, but it is not really satisfying because the algorithm of Diakonikolas and Yannakakis [3] finds an optimal approximation, but it is not really satisfying because the algorithm of Diakonikolas and Yannakakis [3] finds an optimal solution when the points are given explicitly in the input.

Theorem 3. For $p \geq 3$ objectives, when the feasible points are given explicitly in the input, the solution set produced by the greedy algorithm for Set Cover has a size $\Theta(\log n) \cdot opt_c$ in the worst case.

Proof: We prove the result for $p = 3$. The result clearly extends to $p \geq 4$ since we can consider the same points extended with the last $p-3$ coordinates to 0. In order to prove it, we give a family of instances where the algorithm produces a $\Theta(\log n)$-approximation. Let $a, b, c$ be three nonnegative integers, $n = 2^\ell$ and consider $N = 2^\ell + \ell - 3 = n + \log n - 3$ non-dominated points in the criterion space, defined as follows.

For all $i = 1, \ldots, \log n - 1$ let $\mathcal{I}_i = (1, \frac{b}{1 + e^{i - \log n - 1} - 1}, c(1 + e^{i - \log n - 1})), \Gamma_j$ be a set of $\frac{1}{3n}$ points lying uniformly on the line from $(a, \frac{b}{1 + e^{i - \log n - 1} - 1}, c(1 + e^{i - \log n - 1}))$ to $(a(1 + e^{i - \log n - 1} - 1), c(1 + e^{i - \log n - 1}))$ if $i \neq \log n - 1$ and be the singleton $(a, \frac{b}{1 + e^{i - \log n - 1} - 1}, c(1 + e^{i - \log n - 1}))$ if $i = \log n - 1$, and $\Gamma_z$ be a set of $\frac{1}{3n}$ points lying uniformly on the line from $(a, \frac{b}{1 + e^{i - \log n - 1} - 1}, c(1 + e^{i - \log n - 1}))$ to $(a(1 + e^{i - \log n - 1} - 1), b, c(1 + e^{i - \log n - 1}))$ if $i \neq \log n - 1$ and be the singleton $(a, \frac{b}{1 + e^{i - \log n - 1} - 1}, b, c(1 + e^{i - \log n - 1}))$ if $i = \log n - 1$.

Let $\xi = \bigcup \{\mathcal{I}_i\}$ and $\Gamma = \bigcup \Gamma_j \cup \Gamma_z$. We can set $a, b, c$ sufficiently large to have the following $(1 + \varepsilon)$-dominance relations:

(i) for any $\mathcal{I}_i, \mathcal{I}_j \in \xi$, $\mathcal{I}_i \succeq \mathcal{I}_j$

(ii) for any $i, j = 1, \ldots, \log n - 1$, for any $z \in \Gamma_j \cup \Gamma_z$, $\mathcal{I}_i \preceq z$ if and only if $i = j$

(iii) for any $z, \mathcal{I}_i \in \Gamma, z \preceq \mathcal{I}_i$

(iv) for any $z, \mathcal{I}_i \in \Gamma, z \preceq \mathcal{I}_i$

(v) for any $z \in \Gamma, \mathcal{I}_i \in \Gamma_z, z \preceq \mathcal{I}_i$ and $\mathcal{I}_i \not\preceq z$

We show in the following that the greedy algorithm selects the points $\mathcal{I}_i, i = 1, \ldots, \log n - 1$ in this order. The proof is by induction on $i$.

Initialization ($i = 1$). Note that $|\Gamma| = |\Gamma'| = \frac{n}{3} - 1$. From (v) it follows that any point in $\Gamma \cup \Gamma'$ ($1 + \varepsilon$)-dominates at most $\frac{n}{3} + \log n - 2$ points. From (i) and (ii) point $\mathcal{I}_i$ ($1 + \varepsilon$)-dominates exactly the points in $\xi \cup \Gamma_j \cup \Gamma_z$, where $|\xi \cup \Gamma_j \cup \Gamma_z| = \frac{n}{3} + \log n - 1$. In particular point $\mathcal{I}_i$($1 + \varepsilon$)-dominates $\frac{n}{3} + \log n - 1$ points. Therefore, point $\mathcal{I}_i$ is the first point selected by the greedy algorithm at step $i$.

Induction step. Assuming that the first $i - 1$ points selected by the greedy algorithm are $\mathcal{I}_1, \ldots, \mathcal{I}_{i-1}$, we prove that the next one is point $\mathcal{I}_i$. The points ($1 + \varepsilon$)-dominated by $\{\mathcal{I}_1, \ldots, \mathcal{I}_{i-1}\}$ are exactly the points in $\xi \cup (\bigcup_{j=1}^{\log n - 1}(\Gamma_j \cup \Gamma_z))$. Therefore, any point in $\Gamma \cup \Gamma'$ ($1 + \varepsilon$)-dominates exactly $|\bigcup_{j=1}^{\log n - 1}(\Gamma_j \cup \Gamma_z)| = \frac{n}{3} - 1$ points that are not already covered. Point $\mathcal{I}_i$ exactly ($1 + \varepsilon$)-dominates, among the non-covered points, the points in $\Gamma_j \cup \Gamma_z'$ where $|\Gamma_j \cup \Gamma_z'| = \frac{n}{3}$. Thus point $\mathcal{I}_i$ is selected by the greedy algorithm at step $i$.

Observe now, that the first point of $\Gamma_1$ ($1 + \varepsilon$)-dominates all the points in $\xi$. Thus, from (iii) and (iv), it follows that a set constituted by the first point of $\Gamma_1$ and any point in $\Gamma'$ ($1 + \varepsilon$)-dominate the $N$ points. Therefore, the greedy algorithm returns a set of points of size $\log n - 1$ while an optimal set of points contains only two points.

5. Conclusions

We investigated the problem of computing small $\varepsilon$-Pareto sets for multiobjective problems. While the situation is clear for bi-objective problems, it remains challenging for problems involving at least three objectives. In this latter case, even when the points are given explicitly in the input, it is NP-hard to determine an $\varepsilon$-Pareto set of minimal size, but there exists a 100-approximation of the size of a smallest $\varepsilon$-Pareto set [9]. A natural open question is to narrow the large gap between lower and upper bounds.

Acknowledgments

This work was supported by the French ANR project ANR-09-BLAN-0361 “Guaranteed Efficiency for PAREto optimal solutions Determination (GUEPARD)’’.
References