Efficiently Computing Succinct Trade-off Curves

Sergei Vassilvitskii
Stanford University
Stanford, CA
sergei@cs.stanford.edu

Mihalis Yannakakis*
Columbia University
New York, NY
mihalis@cs.columbia.edu

Abstract

Trade-off (aka Pareto) curves are typically used to represent the trade-off among different objectives in multiobjective optimization problems. Although trade-off curves are exponentially large for typical combinatorial optimization problems (and infinite for continuous problems), it was observed in [PY1] that there exist polynomial size ϵ approximations for any $\epsilon > 0$, and that under certain general conditions, such approximate ϵ-Pareto curves can be constructed in polynomial time. In this paper we seek general-purpose algorithms for the *efficient approximation* of trade-off curves *using as few points as possible*. In the case of two objectives, we present a general algorithm that efficiently computes an ϵ-Pareto curve that uses at most 3 times the number of points of the smallest such curve; we show that no algorithm can be better than 3-competitive in this setting. If we relax ϵ to any $\epsilon' > \epsilon$, then we can efficiently construct an ϵ'-curve that uses no more points than the smallest ϵ-curve. With three objectives we show that no algorithm can be c-competitive for any constant c unless it is allowed to use a larger ϵ value. We present an algorithm that is 4-competitive for any $\epsilon' > (1+\epsilon)^2 - 1$. We explore the problem in high dimensions and give hardness proofs showing that (unless P=NP) no constant approximation factor can be achieved efficiently even if we relax ϵ by an arbitrary constant.

1 Introduction

When evaluating different solutions from a design space, it is often the case that more than one criterion comes into play. For example, when choosing a route to drive from one point to another, we may care about the time it takes, the distance traveled, the complexity of the route (e.g. number of turns), etc. When designing a (wired or wireless) network, we may consider its cost, its capacity (the load it can carry), its coverage, etc. When solving computational problems we care about their use of resources such as time, memory, and processors.

Such problems are known as *multicriteria* or *multiobjective* problems. The area of multiobjective optimization has been extensively investigated for many years with a number of conferences and books (e.g. [Cli, Ehr]). In such problems we are interested in the trade-off between the different objectives. This is captured by the *trade-off* or *Pareto curve*, the set of all feasible solutions whose vector of the various objectives is not dominated by any other solution. The trade-off curve represents the range of reasonable possibilities in the design space. Typically we have a small number of objectives (2, 3, ...) and we wish to plot the trade-off curve to get a sense of the

*Work partially supported by NSF Grant CCF-04-30946.

design space. Unfortunately, often the trade-off curve has exponential size for discrete optimization problems even for two objectives (and it is typically infinite for continuous problems).

Recently we started a systematic study of multiobjective optimization based on an approximation that circumvents the aforementioned exponential size problem [PY1, PY2]. The approach is based on the notion of an *ϵ-Pareto curve* (for $\epsilon > 0$), which is a set of solutions that approximately dominate every other solution. More specifically, for every solution s, the ϵ-Pareto curve contains a solution s' that is within a factor $(1+\epsilon)$ of s, in all of the objectives. Such an approximation was studied before for certain specific problems, most notably for multicriteria shortest paths, where Hansen [Han] and Warburton [Wa] showed how to construct an ϵ-Pareto curve in polynomial time.

It was shown in [PY1] that every multiobjective optimization problem with a fixed number of polynomially computable objective functions (as is commonly the case) possesses an ϵ-Pareto curve of size polynomial in the size of the instance and $1/\epsilon$, for every $\epsilon > 0$. Generally, however such an approximate curve may not be constructible in polynomial time. A necessary and sufficient condition for its efficient computability is the existence of an efficient algorithm for the following multiobjective version of the *Gap* problem: Given a vector of values b, either compute a solution that dominates b, or determine that there is no solution that is better than b by at least a factor of $1+\epsilon$ in all objectives. Several classes of problems (including specifically shortest paths, spanning trees, matching, and others) are shown in [PY1, PY2] to satisfy this property and hence have polynomially constructible ϵ-Pareto sets. These include a variety of problems with both linear and nonlinear objective functions.

Although the theorem and construction of [PY1] yield a polynomial size ϵ-Pareto set, the set is not exactly "small": for d objectives, it has size roughly $(m/\epsilon)^{d-1}$, and the construction requires $(m/\epsilon)^d$ calls to the *Gap* routine. Here m is the number of bits used to represent the values in the objective functions. (We give the precise definitions of the framework and the parameters in the next section.)

Note that an ϵ-Pareto set is not unique: many different subsets may qualify and it is quite possible that some are very small while others are very large (without containing any redundant points). Having a small approximate Pareto set gives a succinct outline of the trade-offs involved and is important for many reasons. For example, often the representative set of solutions is investigated further by humans to assess the different choices and pick a suitable one, based on factors that are perhaps not quantifiable.

Suppose that we are willing to tolerate a certain approximation ϵ to the Pareto set, and our problem instance has a small ϵ-Pareto set. Can we find one? Furthermore, can we find one, while spending time that is proportional to the size of the small computed set, rather than the worst case set?

Or, dually, suppose that we can only select a limited number k of solutions. How shall we pick them to represent as well as possible the spectrum of all possible solutions? Clearly, the best choice is a set that provides the best approximation to the Pareto set, i.e., a set that forms an ϵ-Pareto set with the minimum ϵ. Can we find an optimal or nearly optimal such set of k points efficiently? These are the questions we investigate in this paper.

Of course, if for the particular multiobjective problem at hand it is not possible to compute in polynomial time *any* ϵ-Pareto curve, then of course we cannot expect to find an approximately smallest one. That is, the questions make sense only for problems where some ϵ-Pareto curve can be constructed in polynomial time. By the abovementioned characterization, this is the case if and only if there is a (polynomial-time) *Gap* routine. We seek general algorithms that apply in all polynomial cases, i.e. whenever a *Gap* routine as above is available.

In the next section, we define the framework. In Section 3 we study the case of two objectives. We present a general algorithm that for any $\epsilon > 0$ computes an ϵ-Pareto set that has size at most 3 times k, the size of the smallest ϵ-Pareto set. This algorithm uses only $O(k \log(m/\epsilon))$ calls to a *Gap* routine (this is the dominant factor in the running time). We show a matching lower bound on the approximation ratio, i.e. there is no general algorithm that can do better than 3. However, if we relax ϵ to any $\epsilon' > \epsilon$, then we can efficiently construct an ϵ'-curve that uses no more than k points.

We also address the dual problem: Given a bound, k, on the number of points we are willing to have, how good of an approximation (how small of an ϵ) can we get? For example, if $k = 1$, this is the so-called *knee* (or sweet spot) problem: if we pick one point to minimize the ratio for all objectives, what should that compromise point be, and what is the ratio? We show that the ratio can be approximated arbitrarily closely. Furthermore, for the knee problem, we show that we can find an approximately optimal knee for any number of objectives.

In Section 4 we study the case of three objectives. We show that no general algorithm can be within any constant factor c of the smallest ϵ-Pareto set unless it is allowed to use a larger ϵ-value. We present an algorithm that achieves a factor of 4 for any $\epsilon' > (1+\epsilon)^2 - 1$ ($\approx 2\epsilon$ for small ϵ). Furthermore, our algorithm again uses only $O(k \log(m/(\epsilon' - 2\epsilon))$ *Gap* calls.

Finally in Section 5 we discuss the case of an arbitrary number of objectives. We show that even if the solution points are given to us explicitly in the input, we cannot efficiently approximate the size of the smallest ϵ-Pareto curve: We show that the problem is equivalent to the Set Cover problem, and can be approximated efficiently only within a logarithmic factor. Furthermore, we show that no constant factor approximation can be efficiently achieved, even if we relax ϵ by an arbitrary constant.

2 Preliminaries

A multiobjective optimization problem has a set of *instances*, every instance x has a set of solutions $S(x)$. There are d objective functions, $f_1, \ldots, f_d$, each of which maps every instance x and solution $s \in S(x)$ to a positive rational number $f_j(x,s)$. The problem specifies for each objective whether it is to be maximized or minimized. We say that a d-vector u *dominates* another d-vector v if it is at least as good in all the objectives, i.e. $u_j \geq v_j$ if f_j is to be maximized ($u_j \leq v_j$ if f_j is to be minimized); the domination is strict if at least one of the inequalities is strict. Similarly, we define domination between any solutions according to the d-vectors of their objective values. Given an instance x, the *Pareto set* $P(x)$ is the set of undominated d-vectors of values of the solutions. As usual we are also interested in solutions that realize these values, but we will often blur the distinction and refer to the Pareto set also as a set of solutions that achieve these values.

We say that a d-vector u *c-covers* another d-vector v if u is at least as good as v up to a factor of c in all the objectives, i.e. $u_j \geq v_j/c$ if f_j is to be maximized ($u_j \leq cv_j$ if f_j is to be minimized). The definitions is illustrated for two dimensions in Figure 1, where part (a) shows the case of two minimization objectives and (b) the case of two maximization objectives: point u c-covers all points in the shaded region Given an instance x and $\epsilon > 0$, an *ϵ-Pareto set* $P_\epsilon(x)$ is a set of d-vectors of values of solutions that $(1+\epsilon)$-cover all vectors in $P(x)$; i.e., for every $u \in P(x)$, there exists a $u' \in P_\epsilon(x)$ such that u' $(1+\epsilon)$-covers u. For a given instance, there may exist many ϵ-Pareto sets, and they may have very different sizes.

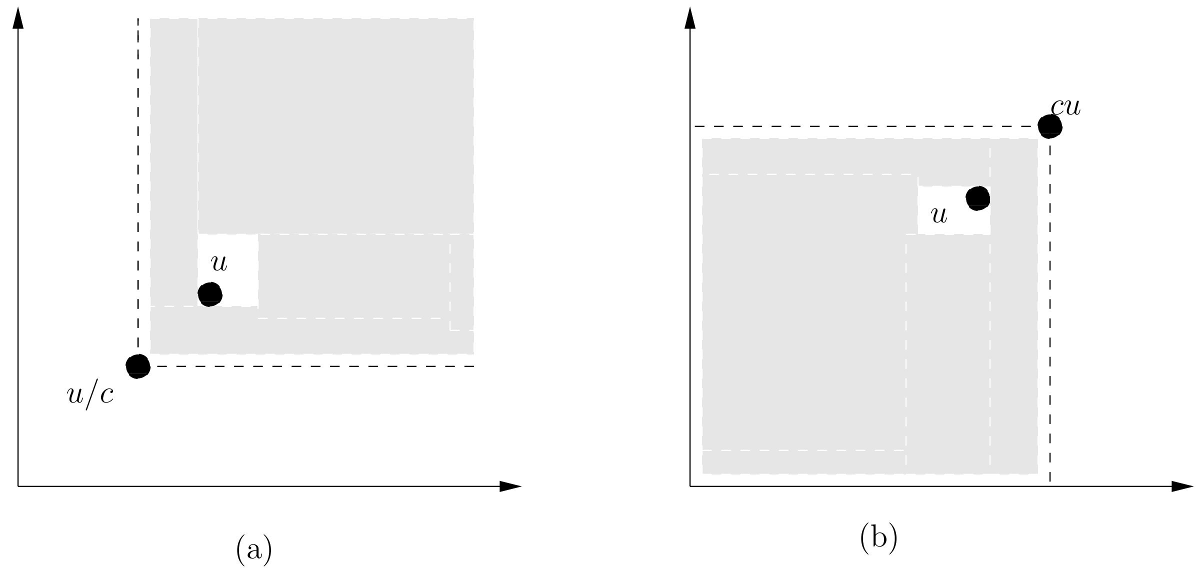

Figure 1: c-coverage: (a) minimization objectives; (b) maximization objectives

To study the complexity of the relevant computational problems, we assume as usual that instances and solutions are represented as strings, that solutions are polynomially bounded and polynomially recognizable in the size of the instance, and that the objective functions are polynomially computable. In particular this means that each value $f_j(x, s)$ is a positive rational whose numerator and denominator have at most m bits, where $m \leq p(|x|)$, for some polynomial p. It is shown in [PY1] that for every multiobjective problem in this framework, for every instance x and $\epsilon > 0$ there exists an ϵ-Pareto set $P_\epsilon(x)$ of size at most $O((4m/\epsilon)^{d-1})$.

Hence, for every multiobjective optimization problem with a fixed number d of objectives, for every instance x and every $\epsilon > 0$ there always exists an approximate ϵ-Pareto set $P_\epsilon(x)$ of polynomial size in the size $|x|$ of the instance x and $1/\epsilon$. The issue is thus one of efficient computability. We say that a multiobjective problem Π has a polynomial time approximation scheme (respectively, a fully polynomial time approximation scheme) if there is an algorithm, which, given instance x and a rational number $\epsilon > 0$, constructs an ϵ-Pareto set $P_\epsilon(x)$ in time polynomial in the size $|x|$ of the instance x (respectively, in time polynomial in the size $|x|$ of the instance, the size $|\epsilon|$ of ϵ (i.e. number of bits in the representation of ϵ), and in $1/\epsilon$).

Let MPTAS (resp. MFPTAS) denote the class of multiobjective problems that have a polynomial time (resp. fully polynomial time) approximation scheme. A necessary and sufficient condition was shown in [PY1], which relates the efficient computation of an ϵ-Pareto set for a multiobjective problem Π with a fixed number d of objectives to the following *GAP Problem*: given an instance x of Π, a (positive rational) d-vector b, and a rational $\epsilon > 0$, either return a solution whose vector dominates b or report that there does not exist any solution whose vector is better than b by at least a $(1 + \epsilon)$ factor in all of the coordinates. The definition is illustrated in Figure 2 for (a) two minimization (and (b) two maximization objectives: GAP has to either return a point that dominates b or declare that there is no point in the shaded region.

As shown in [PY1], a problem Π is in MPTAS (resp. MFPTAS) if and only if there is a

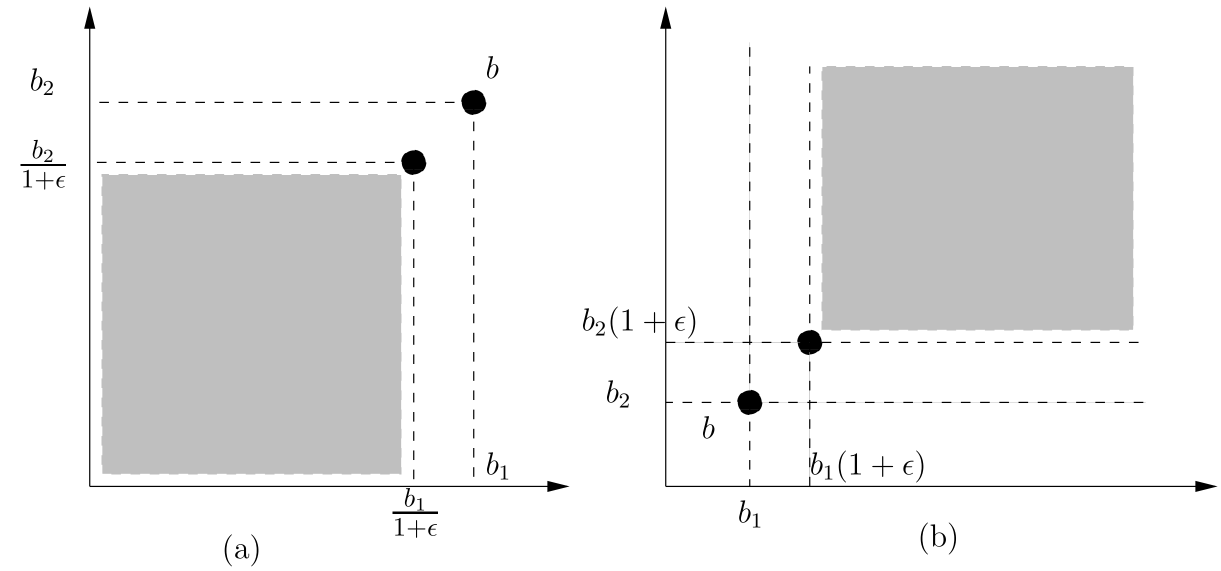

Figure 2: GAP problem for (a) minimization objectives; (b) maximization objectives

subroutine GAP that solves the GAP problem for Π in time polynomial in $|x|$ and $|b|$ (resp. in $|x|, |b|, |\epsilon|$ and $1/\epsilon$). The one direction of this equivalence is quite simple: if we can construct an ϵ-Pareto set $P_\epsilon(x)$ in (fully) polynomial time, then the following simple algorithm solves the GAP problem: construct an ϵ-Pareto set $P_\epsilon(x)$ and check if the given vector b is dominated by any point of $P_\epsilon(x)$; if so, then return the corresponding solution, else return NO. The other direction is sketched in the next section.

For simplicity, we will usually drop the instance x from the notation and use $\mathrm{GAP}_\epsilon(\mathrm{b})$ to denote the solution returned by the GAP subroutine. To make the presentation easier, we will also say that GAP returns YES and that b is a YES point if GAP returns a solution; otherwise we'll say that GAP returns NO and b is a NO point.

We will assume from now on that the polynomial time routine GAP exists, and will present our algorithms using this subroutine as a black box. We say that an algorithm that uses a GAP routine as a black box to access the solutions of the multiobjective problem is *generic*, as it is not geared to a particular problem, but applies to all of the problems for which we can construct an ϵ-Pareto set in (fully) polynomial time. All that our algorithms need to know about the input instance is bounds on the minimum and maximum possible values of the objective functions. For example, if the objective functions are rationals whose numerators and denominators have at most m bits, then an obvious lower bound on the objective values is 2^{-m} and an obvious upper bound is 2^m; however, for specific problems better bounds may be available. Based on the bounds, our algorithms call the routine $GAP_\delta(b)$ for certain tolerances δ and points b, and uses the returned results to compute an approximate Pareto set. In our algorithms the tolerance δ depends only on ϵ, and not for example on the size of the objective values in the input instance or the results of previous GAP calls. In general, however, we can allow a generic algorithm to call the GAP routine with arbitrary δ and b as long as $1/\delta$ and the size of b are polynomially bounded in $1/\epsilon$ and the size of the input, so that the overall algorithm runs in polynomial time.

Note that the GAP problem is monotonic in the input vector b: if a vector b dominates another vector b', and there is a solution that dominates b, then there is also a solution (namely the same solution) that dominates also b'. It will be often convenient to assume that the GAP routine is also monotone in b, i.e. if it returns a solution on a vector b that dominates b', then it returns also a solution on b'. Note that, if a multiobjective problem is in MPTAS (resp. MFPTAS), then it has a polynomial time (resp. fully polynomial time) monotone GAP routine; for instance, the routine described above is monotone.

In a general multiobjective problem we may have some minimization and some maximization objectives. It will be often convenient in the algorithms and the proofs to assume that all objective are of a particular type, e.g. all are minimization objectives, so that we do not have to consider all possible combinations. This can be done without loss of generality for the following reasons. Suppose that we have a problem Π with d objectives, $f_1, \ldots, f_d$, where the first t objectives are maximization and the rest minimization objectives. Consider the problem Π' which has the same set of instances and feasible solutions, and whose first t objectives are the reciprocals of those of Π, while the remaining $d - t$ objectives are the same as Π. All objectives of Π' are minimization objectives. It follows easily from the definitions that for every instance there is a straightforward 1-1 correspondence between the Pareto sets of the two problems, and between ϵ-Pareto sets.

3 Two Objectives

We have a biobjective problem with an associated GAP routine that runs in (fully) polynomial time. We are given an instance and an ϵ, and we wish to construct an ϵ-Pareto set of as small size as possible. This Section is organized as follows. Section 3.1 concerns lower bounds on the size of the ϵ-Pareto set that can be efficiently constructed, as compared to the smallest ϵ-Pareto set. In particular, we show that no generic algorithm can guarantee an approximation ratio better than 3. In Section 3.2 we present a generic algorithm that guarantees ratio 3. In Section 3.3 we show that for any $\epsilon' > \epsilon$, we can compute efficiently an ϵ'-Pareto set that contains no more points than the smallest ϵ-Pareto set. In Section 3.4, we use the latter algorithm to approximate the dual problem: Given an instance and a number k, find the best k solutions that approximate the Pareto curve as closely as possible. We give an algorithm that is $(1 + \theta)$-competitive for any $\theta > 0$. For the special case of $k = 1$ (i.e. the 'knee' case) we give an algorithm that works for any number of objectives. In Section 3.5 we discuss briefly the implications for some concrete multiobjective problems.

We use the following notation in this section. Consider the plane whose coordinates correspond to the two objectives. Every solution is mapped to a point on this plane. We use x and y as the two coordinates of the plane. If p is a point, we use $x(p), y(p)$ to denote its coordinates; that is, $p = (x(p), y(p))$.

3.1 Lower Bound

To prove a lower bound, we take advantage of the fact that on some inputs the *Gap* routine can return either YES or NO. In particular, we will present two Pareto sets which are indistinguishable from each other using the *Gap* procedure as a black box, yet whose smallest ϵ-Pareto sets are of different sizes.

Theorem 1. *a. There is no generic algorithm that approximates the size of the smallest ϵ-Pareto set to a factor better than 3 in the biobjective case.*

b. In particular, there is a biobjective problem with a polynomial time GAP *procedure that cannot be approximated within a factor better than 3 in polynomial time unless P=NP.*

Proof. a. Suppose we have minimization objectives (the same holds for maximization or mixed objectives). Fix any rational $\epsilon > 0$. Consider the following set of points p, $q = \left(x(p)(1+\epsilon)+1, \frac{y(p)}{1+\epsilon}\right)$ and $r = \left(x(p)(1+\epsilon)+1, \frac{y(p)-1}{(1+\epsilon)}\right)$. Let $P = \{p, q\}$ and $Q = \{p, q, r\}$. Note that point p $(1+\epsilon)$-covers point q. Therefore, the smallest ϵ-Pareto set for P consists of only one point, p. However, p does not $(1+\epsilon)$-cover r (because of the y coordinate), and neither q nor r $(1+\epsilon)$-cover p (because of the x coordinate). Therefore, the smallest ϵ-Pareto set for Q must include at least two points.

Let $x(p) = y(p) = M$ be very large integers (exponential in the size of the input). Suppose that we have a polynomial time generic algorithm. We will argue that such an algorithm cannot tell the difference between instances P and Q. Here we exploit the fact that there are points b on which $\text{GAP}_\delta(\text{b})$ is not uniquely defined. Consider the points b where $\text{GAP}_\delta(\text{b})$ can return r; these are the points which r dominates in both objectives. Now if we throw out the points where GAP_δ can also return q, i.e. the points which both q and r dominate in both objectives, we notice that for the points remaining $\text{GAP}_\delta(\text{b})$ can return NO as long as $(1+\delta)y(r) > y(q)$, i.e. as long as $1/\delta < M - 1$. Since we have a polynomial generic algorithm, $1/\delta$ has to be polynomially bounded in the size of the input, hence polynomial in $\log M$. Therefore, using the GAP_δ function as a black box, the algorithm cannot say whether or not r is part of the solution, and thus it is forced to take at least two points, even when it is presented with the set P.

We can make a symmetric observation if instead of q and r we have $q' = \left(\frac{x(p)}{1+\epsilon}, y(p)(1+\epsilon)+1\right)$ and $r' = \left(\frac{x(p)-1}{(1+\epsilon)}, y(p)(1+\epsilon)+1\right)$. Here again using the GAP_δ routine the algorithm cannot detect if the point r' is in the solution space or not. Combining the two bad cases, we see that we cannot tell if the size of the optimal solution is one point, as it is if $P = \{p, q, q'\}$ or if it is three points, as it is when $P = \{p, q, r, q', r'\}$.

b. We can turn the above argument into an NP-hardness proof by defining a suitable biobjective problem so that the points r, r' are present iff a given instance of an NP-complete problem has a solution. Consider the following biobjective problem Π, a variant of the Knapsack problem, which we will call 2-Type-Knapsack problem.[1]. We are given a set $U = U_1 \cup U_2$ of items, partitioned into a set U_1 of items of type 1 and a set U_2 of items of type 2; each item $u \in U$ has a positive rational size $s(u)$ and value $v(u)$. We are also given a size bound C and a value bound D. A solution is a nonempty subset R of items of the same type such that, either all items are of type 1 and the total size $s(R) = \sum_{u \in R} s(u)$ is at most C, or all items are of type 2 and their total value $v(R) = \sum_{u \in R} v(u)$ is at least D. The goal is to minimize size and maximize value.

We have to show two things for the 2-Type Knapsack problem: first, that the GAP problem for it can be solved in fully polynomial time, and second, that it it is NP-hard to approximate the smallest ϵ-Pareto curve with factor less than 3, for any $\epsilon > 0$.

Claim. The GAP problem for the 2-Type Knapsack problem can be solved in fully polynomial time.

[1] For the standard Knapsack problem with the two objectives, size and value, it is easy to show by a similar argument a factor of 2 hardness, i.e., it is NP-hard to determine for a given instance whether one point suffices to $(1+\epsilon)$-cover the size-value trade-off curve or whether two points are needed. The same property can be shown for many other common problems.

Proof. We are given an instance of the 2-type Knapsack problem consisting of sets U_1, U_2 of items of type 1 and 2 respectively, a size bound C and a value bound D. We are also given a point $b = (b_1, b_2)$ and an ϵ, and wish to either (i) find a solution with size at most b_1 and value at least b_2, or (ii) determine that there is no solution with size at most $b_1(1+\epsilon)$ and value at least $b_2/(1+\epsilon)$. To do this we solve approximately two instances I_1, I_2 of the Knapsack problem, using a fully polynomial time approximation scheme (FPTAS) [IK] with approximation error ϵ. In the first instance I_1, we have the set of items U_1 and an upper bound $min(C, b_1)$ on the capacity of the knapsack, and we wish to find a subset of U_1 with (approximately) maximum total value that fits in the knapsack, i.e. has size at most $min(C, b_1)$. If the solution found by the FPTAS algortihm has value at least b_2 then our algorithm for GAP returns this solution. If not, it continues to solve the second Knapsack instance. In the second instance I_2, we have the set U_2 of items and a lower bound $max(D, b_2)$ on the desired value, and we wish to find a subset of items with (approximately) minimum total size that has total value at least $max(D, b_2)$. If the solution found by the FPTAS algorithm has size at most b_1 then our algorithm for GAP returns this solution. If not, then the algorithm returns No.

Clearly, if the algorithm returns a set S of items, then the set S is a solution to the 2-Type Knapsack problem and dominates the point b (has size at most b_1 and value at least b_2). Conversely, suppose that the 2-Type Knapsack problem has a solution R with size at most $b_1/(1+\epsilon)$ and value at least $b_2(1+\epsilon)$. If R consists of type 1 items, then $size(R) \leq min(C, b_1)$, hence R is a solution to the first instance I_1 of the Knapsack problem. Since $value(R) \geq b_2(1+\epsilon)$, and the PTAS returns a solution S which is within $1+\epsilon$ of optimal, it follows that $value(S) \geq b_2$, hence our algorithm will return S. If R consists of type 2 items, then $value(R) \geq max(D, b_2)$, hence R is a solution to the second instance I_2 of the Knapsack problem. Since $size(R) \leq b_1/(1+\epsilon)$, and the PTAS returns a solution S which is within $1+\epsilon$ of optimal, it follows that $size(S) \leq b_1$, hence our algorithm will return S. Thus, in either case the algorithm will return a feasible solution that dominates $b = (b_1, b_2)$. It follows that the algorithm solves the GAP problem in fully polynomial time. □

We prove now that for any $\epsilon > 0$, it is NP-hard to approximate the size of the minimum ϵ-Pareto set for Π with ratio smaller than 3. In fact, it is NP-hard to determine whether one point suffices or we need at least 3 points.

The reduction is from the Partition problem (a special case of the Knapsack problem) [GJ]. Recall that in the Partition problem, we are given a set N of n positive integers $a_1, \ldots, a_n$, and we wish to determine whether it is possible to partition N into two subsets with equal sum. Given an instance of the Partition problem, construct an instance of the biobjective 2-Type Knapsack problem Π as follows. Let $B = \sum a_i/2$ and let $H > 2(1+\epsilon)(B+1)$. For each $i = 1, \ldots, n$, we have one item of type 1 with size Ha_i and value $(1+\epsilon)a_i$ and one item of type 2 with size $a_i/(1+\epsilon)$ and value a_i/H. In addition we have a special item p of type 2 with size $B+1$ and value $B-1$. Let $C = HB$ and $D = B/H$.

We claim that if the instance of the Partition problem has a solution then the smallest ϵ-Pareto set must contain at least 3 points, while if there is no solution then one point suffices. Suppose first that the instance of the Partition problem has no solution, i.e. there is no subset I of $\{1, \ldots, n\}$ such that $\sum_{i \in I} a_i = B$. We claim then that the solution consisting of the special item p is a $(1+\epsilon)$-Pareto set. It suffices to show that every solution R of Π has size at least $(B+1)/(1+\epsilon)$ and value at most $(1+\epsilon)(B-1)$. Let R be a solution of Π and let I be the subset of $\{1, \ldots, n\}$ corresponding to the items of R. If R is a set of items of type 1, then since R is nonempty, its size is at least $H > B+1$. Since its size is at most $C = HB$, it follows that $\sum_{i \in I} a_i \leq B$. Since the Partition problem has no solution, $\sum_{i\in I} a_i \le B-1$ and thus $v(R) = (1+\epsilon)\sum_{i\in I} a_i \le (1+\epsilon)(B-1)$. If R is a set of items of type 2 then its value is at most $B-1+2B/H < B$. If R includes p, then its size is at least $B+1$. If R does not include p then $v(R) \ge B/H$ implies that $\sum_{i\in I} a_i \ge B$. Since the Partition instance has no solution, it follows that $\sum_{i\in I} a_i \ge (B+1)$ hence $s(R) \ge (B+1)/(1+\epsilon)$.

Suppose that the instance of the Partition problem has a solution, i.e. there is a subset I of $\{1,\dots,n\}$ such that $\sum_{i\in I} a_i = B$. Let R (respectively R') be the set of items of type 1 (resp. type 2) that correspond to the indices in I. Then R is mapped to the point $r = (s(R), v(R)) = (HB, (1+\epsilon)B)$ and R' is mapped to the point $r' = (s(R'), v(R')) = (B/(1+\epsilon), B/H)$. Point r is not $(1+\epsilon)$-covered by any solution consisting of items of type 2 (because their value is too low), and point r' is not $(1+\epsilon)$-covered by any solution that includes the special item p or which consists of items of type 1 (because their size is too high). In addition, the solution $\{p\}$ that contains only the special item p is not $(1+\epsilon)$-covered by any other solution that does not include p, because solutions consisting of items of type 1 have too high size and solution consisting only of other items of type 2 have too low value. Therefore, any ϵ-Pareto set must contain at least 3 points. $\square$

The proof of the theorem showed that a generic algorithm cannot determine whether there is an ϵ-Pareto set with 1 point or whether 3 points are needed. If we wish, we can modify the construction to show that, for any k, it is impossible for a generic algorithm (and it is NP-hard for some problems) to determine whether the smallest ϵ-Pareto set has k points or needs $3k$ points. The modification involves the replication of the bad configuration of the proof k times. Specifically, instead of one point p, we have k points $p_1,\dots,p_k$, where $p_i = ((2+\epsilon)^i x(p), (2+\epsilon)^{k-i} y(p))$, and similarly we have k points corresponding to each one of q, q', r, r', obtained by multiplying the x-coordinate by $(2+\epsilon)^i$ and the y-coordinate by $(2+\epsilon)^{k-i}$. Note that no point with index i is $(1+\epsilon)$-covered by any point with a different index. Thus, if the points corresponding to r, r' are not present then there is a ϵ-Pareto set with k points, while if they are present then we need $3k$ points.

Note finally that the lower bound above is brittle, in the sense that if the algorithm is allowed to return an ϵ'-Pareto set for any $\epsilon' > \epsilon$, the proof no longer holds. In fact we will show in Section 3.3, that for any $\epsilon' > \epsilon$ there is an algorithm that finds an ϵ'-Pareto set $P_{\epsilon'}$, of size no bigger than the optimal ϵ-Pareto set.

3.2 2-Objective Algorithms

We assume for concreteness that both objectives are to be minimized; the algorithm is similar in the other cases. We recall here the original algorithm of [PY1]. To compute an ϵ-Pareto set, and in fact prove a polynomial bound on its size, consider the following scheme. Divide the space of objective values geometrically into rectangles, such that the ratios of the large to the small coordinates is $(1+\epsilon')=\sqrt{1+\epsilon}$ in all dimensions; equivalently if we switch to a log-log scale of the objective values, the plane is partitioned arithmetically into squares of size $\log(1+\epsilon')$ ($\approx \epsilon/2$ for small ϵ). If $\sqrt{1+\epsilon}$ is not rational, then we let ϵ' be a rational that approximates $\sqrt{1+\epsilon}-1$ from below, and which has representation size $O(|\epsilon|)$ (i.e. number of bits in the numerator and denominator). Proceed to call $\mathrm{GAP}_{\epsilon'}$ on all of the rectangle corner points, and keep an undominated subset of all points returned. It is easy to see that this forms an ϵ-Pareto set. (To prove that this set cannot be too large, note that we can discard points until there is at most one remaining in each of the rectangles.) If m is the maximum number of bits in the numerator and denominator of the objective functions, then the ratio of the largest to the smallest possible objective value is 2^{2m}, hence the number of subdivisions in each dimension is $2m/\log(1+\epsilon') \approx 4m/\epsilon$ for small ϵ.

This algorithm gives no guarantees on the size of the ϵ-Pareto set it returns with respect to P^*_ϵ, the smallest ϵ-Pareto set. It is possible to modify the algorithm in a simple manner so that it computes a ϵ-Pareto set of size at most $7|P^*_\epsilon|$, however we will instead give a better algorithm that achieves an approximation ratio of 3, and which furthermore is more efficient in the number of calls to the GAP routine.

To compute a 3-approximation to P^*_ϵ we will proceed in two phases. In the first phase we will compute an ϵ''-Pareto set for a particular $\epsilon'' < \epsilon$. In the second phase we will delete points from this set until we are left with a small ϵ-Pareto set.

We describe the first phase now. Let $\epsilon' > 0$ be a rational that approximates $\sqrt[4]{1+\epsilon} - 1$ from below with a representation of size $O(|\epsilon|)$. Thus, $(1+\epsilon')^4 \leq 1+\epsilon < (1+\epsilon')^5$. Let $\delta \leq \epsilon'$ be a rational, also of size $O(|\epsilon|)$, such that $(1+\epsilon)(1+\delta) < (1+\epsilon')^5$. Note that we can pick ϵ', δ so that $1/\epsilon' = O(1/\epsilon)$ and $1/\delta = O(1/\epsilon)$.

We will begin again by partitioning the plane of objective values into rectangles, using a finer grid now with ratio $1+\epsilon'$. We will use the term "corner points" to refer to the points of the grid (i.e. the corner points of the rectangles). For the lower left corner of the grid we pick a point x_0, y_0 that has coordinates less than the smallest possible values of the objective functions. This determines then the lines of the grid, which we extend one level beyond the maximum possible values of the objective functions. That is, the upper right corner of the grid has coordinates $x_0(1+\epsilon')^h, y_0(1+\epsilon')^v$, for some numbers h, v such that $x_0(1+\epsilon')^{h-1}$, $y_0(1+\epsilon')^{v-1}$ are at least as great as the maximum possible objective values. Clearly, the number of horizontal and vertical lines of the grid is $O(m/\log(1+\epsilon))$, which is $O(m/\epsilon)$ for small ϵ. However, we will not construct explicitly the whole grid: our algorithm will only generate a subset of the points of the grid as it needs them.

The first phase of the algorithm consists of two repeating steps that use two operations, ZAG and ZIG, applied to corner points b of the grid. It is simpler to define first the operations under the assumption that the GAP routine is monotone, although we will show that the algorithm works correctly and guarantees ratio 3 even if the GAP routine is not monotone. The operation ZAG(b) returns a corner point p on the same vertical line as b with minimal y value, such that GAP$_\delta(p)$ = YES, if there is such a point; if there is no such point then ZAG(b) returns NO. The operation ZIG(b) is defined symmetrically: ZIG(b) returns a corner point p on the same horizontal line as b with minimal x value, such that GAP$_\delta(p)$ = YES, if there is such a point; if there is no such point then ZIG(b) returns NO.

We implement ZAG and ZIG as follows. Suppose that GAP$_\delta(b)$ = YES. Note that GAP returns NO on all the points on the leftmost line and the bottom line of the grid. To compute ZAG(b) we do a binary search on the set of corner points that lie below b on the same vertical line. Similarly, to compute ZIG(b) we do a binary search on the set of grid points that lie left of b on the same horizontal line. In this case, ZIG and ZAG return a point.

Suppose that GAP$_\delta(b)$ = NO. If GAP$_\delta$ returns NO also at the highest point of the vertical line through b, then ZAG(b) returns NO. Otherwise, compute ZAG(b) by doing a binary search on the set of corner points that lie above b on the same vertical line. Similarly, ZIG(b) returns NO if GAP$_\delta$ returns NO at the rightmost point of the horizontal line through b; otherwise compute ZAG(b) using a binary search on the set of corner points that lie to the right of b on the same horizontal line.

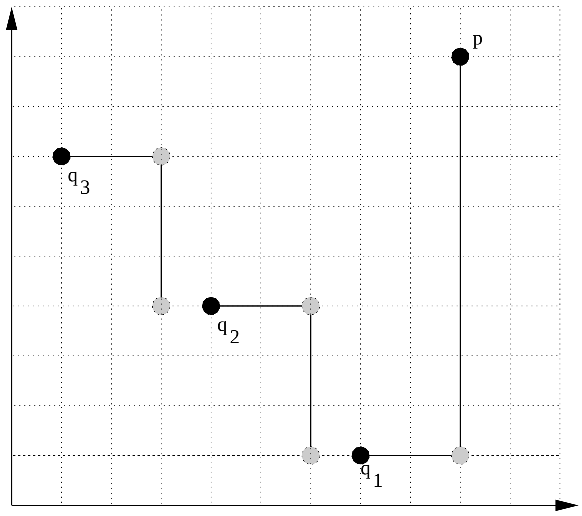

Figure 3: Schematic performance of the ZigZag algorithm

Clearly, if the GAP routine is monotone, then the above implementations of ZIG and ZAG return the required minimal values according to the definitions. We will use the above implementations of ZIG and ZAG even if GAP is not monotone. In this case the procedures have the following properties: Procedure ZAG(b) returns NO iff GAP_δ returns NO on point b and on the highest point of the vertical line through b. Otherwise, ZAG(b) returns a corner point p that lies on the same vertical line as b (i.e. $x(p) = x(b)$), such that $\mathrm{GAP}_\delta(p) = \mathrm{YES}$ and $\mathrm{GAP}_\delta(p') = \mathrm{NO}$, where $p' = (x(p), y(p)/(1+\epsilon'))$ is the corner point immediately below p. Procedure ZIG(b) returns NO iff GAP_δ returns NO on point b and on the rightmost point of the horizontal line through b. Otherwise, ZAG(b) returns a corner point p such that $y(p) = y(b)$, $\mathrm{GAP}_\delta(p) = \mathrm{YES}$ and $\mathrm{GAP}_\delta(p') = \mathrm{NO}$, where $p' = (x(p)/(1+\epsilon'), y(p))$, is the corner point immediately to the left of p.

The first phase of the algorithm computes a set Q of corner points. It is as follows.

Phase 1: ZIGZAG:
p = top right corner point of the grid
If GAP_δ(p) = NO then halt.
q_1 = ZIG(ZAG(p))
$Q = \{q_1\}$
$q'_1 = (x(q_1)/(1+\epsilon'), y(q_1))$.
$i = 1$
While (ZAG(q'_i) $\neq$ NO) do the following.
{ $q_{i+1} = \mathrm{ZIG}(\mathrm{ZAG}(q'_i))$
  $Q = Q \cup \{q_{i+1}\}$
  $q'_{i+1} = (x(q_{i+1})/(1+\epsilon'), y(q_{i+1}))$
  $i = i + 1$
}

The set of points computed by the algorithm is shown schematically in Figure 3.

Consider the top right corner point p of the grid. If $\mathrm{GAP}_\delta(\mathrm{p}) = \mathrm{NO}$, then there are no feasible solutions because the coordinates of p exceed by at least $1+\epsilon' \geq 1+\delta$ the maximum possible values of the objectives. So in this case we can just terminate the algorithm. So assume that the solution set is nonempty and hence $\mathrm{GAP}_\delta(\mathrm{p}) = \mathrm{YES}$. This implies that ZAG($p$) will return a YES point (i.e. a point for which GAP_δ returns YES), and thus ZIG(ZAG(p)) will also return a YES point q_1. The corner point q_1' immediately to the left of q_1 is a NO point by the definition of the procedure ZIG. Similarly, in every iteration of the loop, since $\mathrm{ZAG}(q_i') \neq \mathrm{NO}$, $\mathrm{ZIG}(\mathrm{ZAG}(q_i'))$ will return a YES point q_{i+1}. The loop terminates when $\mathrm{ZAG}(q_i') = \mathrm{NO}$, i.e. if the top corner point on the vertical line through q_i' is a NO point.

Let $Q = \{q_1, \ldots, q_r\}$ be the set of points computed by the above ZIGZAG algorithm. We shall prove that the set Q of points $(1+\epsilon')(1+\delta)$-covers the set P of all solution points. We prove first some useful properties.

Lemma 2. *The x coordinates of the points $q_1, \ldots, q_r$ of Q form a strictly decreasing sequence, and the y coordinates form a strictly increasing sequence.*

Proof. Since q_i' is a NO point immediately to the left of q_i, $\mathrm{ZAG}(q_i')$ has strictly smaller x-coordinate and strictly larger y-coordinate than q_i. Hence, the same is true for $q_{i+1} = \mathrm{ZIG}(\mathrm{ZAG}(q_i'))$ which is equal to or lies to the left of $\mathrm{ZAG}(q_i')$. □

Lemma 3. *1. The point q_1 $(1+\epsilon')(1+\delta)$-covers all of the solution points in P that have x-coordinate at least $x(q_1)/(1+\epsilon')(1+\delta)$.*

2. For each $i = 2, \ldots, r$, the point q_i $(1+\epsilon')(1+\delta)$-covers all of the solution points in P that have their x-coordinate between $x(q_i)/(1+\epsilon')(1+\delta)$ and $x(q_{i-1})/(1+\epsilon')(1+\delta)$.

3. There are no solution points with x-coordinate smaller than $x(q_r)/(1+\epsilon')(1+\delta)$.

Proof. 1. Let p be the top right corner point of the grid as in the algorithm, and let s be the corner point immediately below ZAG(p); $x(s) = x(p)$ and $y(s) = y(q_1)/(1+\epsilon')$. By the definition of the procedure ZAG, s is a NO point. Suppose that there exists a solution point t with $x(t) \geq x(q_1)/(1+\epsilon')(1+\delta)$ such that t is not $(1+\epsilon')(1+\delta)$-covered by q_1. Then we must have $y(t) < y(q_1)/(1+\epsilon')(1+\delta)$, and hence $y(t) < y(s)/(1+\delta)$. By our definition of the rightmost line of the grid, we have also $x(t) \leq x(p)/(1+\epsilon') \leq x(s)/(1+\delta)$. Therefore, $\mathrm{GAP}_\delta(s)$ cannot return NO, a contradiction.

2. Suppose that there exists a solution point t whose x-coordinate satisfies $x(q_i)/(1+\epsilon')(1+\delta) \leq x(t) < x(q_{i-1})/(1+\epsilon')(1+\delta)$ and such that t is not $(1+\epsilon')(1+\delta)$-covered by q_i. Then we must have $y(t) < y(q_i)/(1+\epsilon')(1+\delta)$. Let s be the corner point immediately below $\mathrm{ZAG}(q_{i-1}')$. From the definition of ZAG, s is a NO point. The coordinates of s are $x(s) = x(q_{i-1}') = x(q_{i-1})/(1+\epsilon')$ and $y(s) = y(q_i)/(1+\epsilon')$. Thus, $x(t) \leq x(s)/(1+\delta)$ and $y(t) < y(s)/(1+\delta)$. Therefore, $\mathrm{GAP}_\delta(s)$ cannot return NO, a contradiction.

3. Suppose that there is a solution point t with x-coordinate $x(t) < x(q_r)/(1+\epsilon')(1+\delta)$. Let s be the top corner point in the vertical line of q_r'. By the termination condition of the algorithm, s is a NO point. The x-coordinate of s is $x(s) = x(q_r') = x(q_r)/(1+\epsilon')$, hence $x(t) < x(s)/(1+\delta)$. From the definition of the top line of the grid, we have also $y(t) \leq y(s)/(1+\delta)$. Therefore, again $\mathrm{GAP}_\delta(s)$ cannot return NO, a contradiction. □

The preceding lemma implies immediately now the claimed property for the set Q.

Lemma 4. *The* ZIGZAG *algorithm above returns a set Q that $(1+\epsilon')(1+\delta)$-covers the set P of all solution points.*

We bound now the size of the computed set Q, so that we can bound later on the complexity of the algorithm.

Lemma 5. *The size of Q returned by* ZIGZAG *above, is no more than 11 times the size of the smallest ϵ-Pareto set, P^*_ϵ.*

Proof. Let P^*_ϵ be the smallest ϵ-Pareto set, and let p^* be a point in P^*_ϵ. Since all the points of Q are YES points, i.e. are dominated by some solution point, all points of Q are $(1+\epsilon)$-covered by the points in P^*_ϵ. We charge those points in Q that are $(1+\epsilon)$-covered by p^* to the point p^*.

Let s be a point in Q that $(1+\epsilon')(1+\delta)$-covers p^*. Let q be a point in Q that is $(1+\epsilon)$-covered by p^*. Since the points in Q are incomparable (do not dominate each other), either $q = s$ or $x(q) < x(s)$ or $y(q) < y(s)$. If $x(q) < x(s)$, then since $x(q) \geq x(p^*)/(1+\epsilon)$ it follows that $x(q) \geq x(s)/(1+\epsilon)(1+\epsilon')(1+\delta) > x(s)/(1+\epsilon')^6$, i.e. $x(q) = x(s)/(1+\epsilon')^i$ where i=1,2,3,4 or 5. Since the points of Q do not dominate each other, it follows that there are at most 5 points $q \in Q$ with $x(q) < x(s)$ that are $(1+\epsilon)$-covered by p^*. Similarly, if $y(q) < y(s)$ then we can conclude that $y(q) = y(s)/(1+\epsilon')^i$ where i=1,2,3,4 or 5, and thus there at most 5 points q of Q with $y(q) < y(s)$. Therefore, there are at most 11 points of Q that are $(1+\epsilon)$-covered by p^*. Hence $|Q| \leq 11|P^*_\epsilon|$. ☐

In the second phase we reduce the set Q to a subset Q' that $(1+\epsilon)$-covers all the solution points, and return the set of solution points $R = \mathrm{GAP}_\delta(Q') = \{\mathrm{GAP}_\delta(q) | q \in Q'\}$. Since the points of R dominate the corresponding points in Q', it follows that R is an ϵ-Pareto set.

Let $c = (1+\epsilon)/(1+\epsilon')(1+\delta)$. Note that if Q' is any c-cover of the points in Q, then for any point in the original solution space, there is some point in Q that $(1+\epsilon')(1+\delta)$-covers it, and hence there is a point in Q' that $(1+\epsilon)$-covers it. We will compute the smallest c-cover Q' of Q. This can be done easily by a simple greedy algorithm. Recall that the points of the set $Q = \{q_1, \ldots, q_r\}$ computed in Phase 1 are sorted in decreasing order of their x-coordinate and increasing order of y-coordinate.

In words, the greedy algorithms works as follows: Starting from q_1 we find a point $q_i \in Q$ with the maximum i (i.e. smallest x- and largest y-coordinate) that c-covers q_1. Since q_1 has the largest x-coordinate among all points of Q, we only need to check the y-coordinate: q_i is the last point whose y-coordinate is at most $cy(q_1)$. Include q_i into the set Q', remove all the points that are c-covered by q_1, and iterate. Note that q_i c-covers all points $q_1, \ldots, q_i$ and perhaps some points q_j with $j > i$, namely those points whose x-coordinate is at least $x(q_i)/c$.

Phase 2: Cleanup:
$Q' = \emptyset$
$i = 1$
$v = y(q_1)$
While $(i \leq r)$ do the following:
{ if $(i < r$ and $y(q_{i+1}) \leq cv)$ then $i = i + 1$
  else if $(Q' \neq \emptyset$ and $x(q_i) \geq x(q)/c)$ then $i = i + 1$
  else { $Q' = Q' \cup \{q_i\}$
    $q = q_i$
    if $(i < r)$ then $v = y(q_{i+1})$
    $i = i + 1$
  }
}

Return $R = \{\mathrm{GAP}_\delta(q) | q \in Q'\}$.

Lemma 6. *The set Q' computed by the Cleanup algorithm is a minimum c-cover of Q.*

Proof. It is easy to see that Q' is a c-cover of Q. To show that it has minimum size, consider any subset Q'' of Q that c-covers Q. It is easy to show by induction that for $i = 1, \ldots, r$, the set Q'' contains at least as many points as Q' among the first i points $q_1, \ldots, q_i$ of Q. For the basis, $i = 1$, suppose that $q_1 \in Q'$. Then q_2 does not c-cover q_1 because its y-coordinate is too large. Hence the same is true for all other points of Q as well, and Q'' must also include q_1.

For the induction step, suppose that the claim is not true for i, but holds for smaller indices. Then $q_i \in Q' - Q''$. Since $q_i \notin Q''$, it is c-covered by some other point q_j of Q''. If $j > i$, then $y(q_j) \leq cy(q_i)$, which implies that $y(q_{i+1}) \leq cy(q_i) \leq cv$, where v is the value of the variable at iteration i, and thus q_i should not have been added to Q'. If $j < i$, then, by the induction hypothesis, the set Q' must include at least some point among $q_j, \ldots, q_{i-1}$; that point also c-covers q_i and hence q_i should not have been added to Q' in this case too.

□

Lemma 7. *The size of the smallest c-cover of Q is no more than $3 \cdot |P^*_\epsilon|$.*

Proof. First observe that $c = (1+\epsilon)/(1+\epsilon')(1+\delta) \geq (1+\epsilon')^2$. Consider any point p^* in P^*_ϵ, and let s be a point of Q that $(1+\epsilon')(1+\delta)$ covers p^*. By the analysis of Lemma 4, there are at most 11 points of Q that are $(1+\epsilon)$-covered by p^*: point s, at most 5 points that are above and to the left of s that have x-coordinate $x(s)/(1+\epsilon')^i$ where $1 \leq i \leq 5$, and at most 5 points that are below and to the right of s and have y-coordinate $y(s)/(1+\epsilon')^i$ where $1 \leq i \leq 5$.

Three points suffice to c-cover all the points of Q that are $(1+\epsilon)$-covered by p^*: Point s covers itself, the points above it with x-coordinate $x(s)/(1+\epsilon')^i$, $i = 1, 2$, and the points below it with y-coordinate $y(s)/(1+\epsilon')^i$, $i = 1, 2$. Then we only need to take (at most) one more point above s and one point below s to c-cover the other points that are not c-covered by s.

Therefore, there is a subset of Q with at most $3 \cdot |P^*_\epsilon|$ elements that c-covers all points of Q. □

It follows from Lemmas 6 and 7 that the size of the set R computed by the algorithm is $|R| = |Q'| \leq 3|P^*_\epsilon|$.

We now proceed to analyze the running time of the algorithm. Let k be the total number of points in the optimal ϵ-Pareto set, $k = |P^*_\epsilon|$. Recall that m denotes the number of bits in the objective functions. To avoid clutter in the expressions below, we will use ϵ in place of $\log(1+\epsilon)$, which is a valid approximation for small ϵ (for large ϵ simply drop this factor). In the end of the first phase we produce $O(k)$ points. To find each point, we called one execution of ZIG and one of ZAG. Both of these are implemented as binary searches on $O(m/\epsilon)$ points. Therefore, the runtime of the first phase is bounded by $O(k \log(m/\epsilon))$ GAP_δ calls. The greedy algorithm of Phase 2 is linear, and its time is subsumed by that of the first phase. Therefore the overall runtime is $O(k \log(m/\epsilon))$ GAP_δ calls. We summarize the properties of the algorithm in the following theorem.

Theorem 8. *The ZIGZAG-&-Cleanup algorithm as described above computes a 3-approximation to the smallest ϵ-Pareto set in time $O(k \log(m/\epsilon))$ GAP_δ calls, where $1/\delta = O(1/\epsilon)$.*

3.3 Algorithm for Relaxed ϵ

The lower bound on the competitive ratio is brittle, and even a small relaxation to ϵ allowes us to circumvent it. Intuitively, the lower bound comes from the area where *Gap* can return either YES or NO. However, we can reduce the size of this area if we are required to produce a ϵ'-Pareto set for $\epsilon' > \epsilon$ and have time on the order of $(\epsilon' - \epsilon)^{-1}$.

Suppose that we are allowed to return an ϵ'-Pareto set for a value $\epsilon' > \epsilon$. Let δ be a suitably small rational and $c = (1+\delta)^j$ a power of $1+\delta$ such that $(1+\epsilon') \geq c(1+\delta)^2 = (1+\delta)^{j+2}$ and $1+\epsilon \leq c/(1+\delta) = (1+\delta)^{j-1}$. For example, we can choose δ to be a rational with $O(|\epsilon| + |\epsilon'|)$ representation such that $(1+\delta)^4$ approximates from below $(1+\epsilon')/(1+\epsilon)$. Define $c = (1+\delta)^j$ where j satisfies $(1+\delta)^{j-2} < 1+\epsilon \leq (1+\delta)^{j-1}$. Obviously, $1+\epsilon \leq c/(1+\delta)$, and $(1+\epsilon') \geq (1+\epsilon)(1+\delta)^4 > (1+\delta)^{j+2}$.

Divide the space of the objective values using a grid with the coordinate ratio of $(1+\delta)$. The algorithm is given below.

Relaxed ZIGZAG:
$p =$ top right corner point of the grid
If $\mathrm{GAP}_\delta(p) =$ NO then halt.
$u_1 = \mathrm{ZAG}(p)$
$v_1 = (x(u_1), cy(u_1))$
$q_1 = \mathrm{ZIG}(v_1))$
$Q = \{q_1\}$
$w_1 = (x(q_1)/[c(1+\delta)], y(q_1)/(1+\delta))$.
$i = 1$
While $(\mathrm{ZAG}(w_i) \neq \text{NO}$) do the following.
{ $u_{i+1} = \mathrm{ZAG}(w_i)$
  $v_{i+1} = (x(u_{i+1})(1+\delta), cy(u_{i+1}))$
  $q_{i+1} = \mathrm{ZIG}(v_{i+1})$
  $Q = Q \cup \{q_{i+1}\}$
  $w_{i+1} = (x(q_{i+1})/[c(1+\delta)], y(q_{i+1})/(1+\delta))$
  $i = i+1$
}
Return $R = \mathrm{GAP}_\delta(Q) = \{\mathrm{GAP}_\delta(q) | q \in Q\}$

The operation of the algorithm is illustrated in Figure 4. For simplicity we used in the figure $c = (1+\delta)^2$.

As in the previous section, if $\mathrm{GAP}_\delta(p) =$ NO, then there are no feasible solutions, and there is nothing to compute. So assume there exist solutions. It is easy to see that all the points defined in the algorithm are well-defined, i.e., when a point is assigned the value returned by the ZIG or ZAG procedure, then the procedure indeed returns a point. Let $Q = \{q_1, \ldots, q_r\}$ be the set of corner points computed by the algorithm. The following two lemmas and their proofs are similar to corresponding lemmas from the previous subsection.

Lemma 9. *The x coordinates of the points $q_1, \ldots, q_r$ of Q form a strictly decreasing sequence, and the y coordinates form a strictly increasing sequence.*

Proof. Consider two successive elements q_i, q_{i+1} of Q. Since the point immediately to the left of q_i is a NO point, it follows that w_i is also a NO point. The point u_{i+1} is thus strictly above w_i. Hence

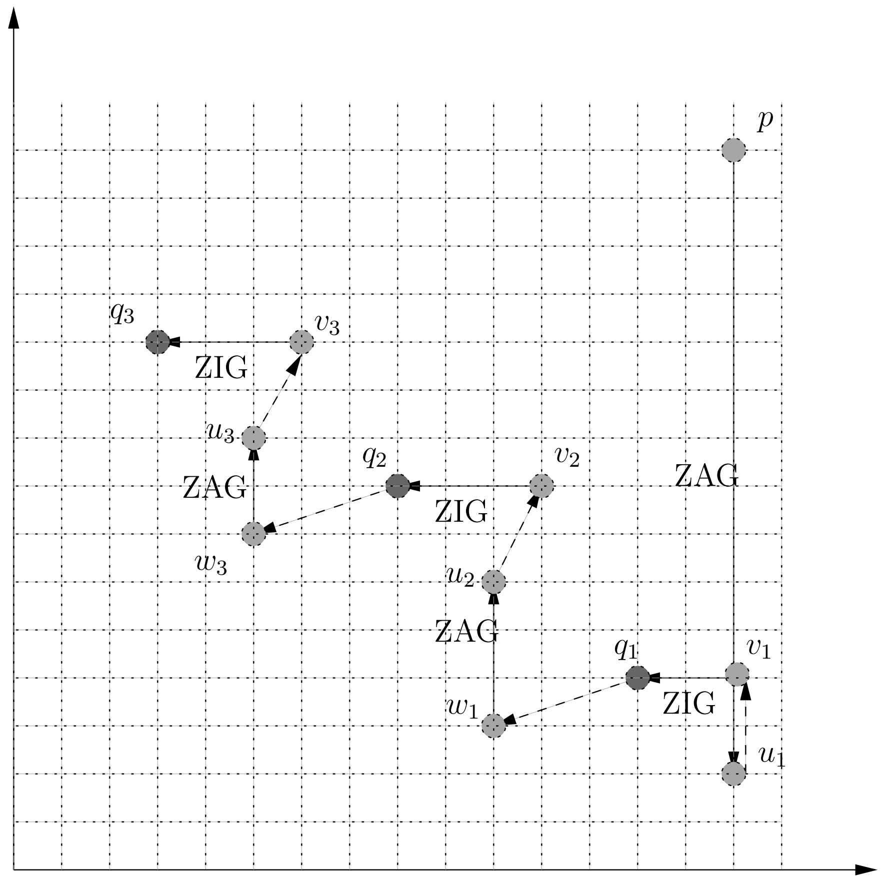

Figure 4: Illustration of the Relaxed ZIGZAG Algorithm

$y(q_{i+1}) \geq cy(u_{i+1}) \geq cy(q_i)$. Since u_{i+1} is a YES point, it follows that v_{i+1} is also a YES point. Hence $q_{i+1} = \text{ZIG}(v_{i+1})$ exists and is at or to the left of v_{i+1}. Thus, $x(q_{i+1}) \leq x(v_{i+1}) \leq x(q_i)/c$. $\square$

Lemma 10. *1. The point q_1 $(1+\epsilon')$-covers all of the solution points in P that have x-coordinate at least $x(q_1)/(1+\epsilon')$.*

2. For each $i = 2, \ldots, r$, the point q_i $(1+\epsilon')$-covers all of the solution points in P that have their x-coordinate between $x(q_i)/(1+\epsilon')$ and $x(q_{i-1})/(1+\epsilon')$.

3. There are no solution points with x-coordinate smaller than $x(q_r)/(1+\epsilon')$.

Proof. 1. Let p be the top right corner point of the grid as in the algorithm, and let s be the corner point immediately below $u_1 = \text{ZAG}(p)$; $x(s) = x(p)$ and $y(s) = y(u_1)/(1+\delta)$. By the definition of the procedure ZAG, s is a NO point. Suppose that there exists a solution point t with $x(t) \geq x(q_1)/(1+\epsilon')$ such that t is not $(1+\epsilon')$-covered by q_1. Then we must have $y(t) < y(q_1)/(1+\epsilon') = cy(u_1)/(1+\epsilon') \leq y(u_1)/(1+\delta)^2$, and hence $y(t) < y(s)/(1+\delta)$. By our definition of the rightmost line of the grid, we have also $x(t) \leq x(s)/(1+\delta)$. Therefore, $\text{GAP}_\delta(s)$ cannot return NO, a contradiction.

2. Suppose that there exists a solution point t whose x-coordinate satisfies $x(q_i)/(1+\epsilon') \leq x(t) < x(q_{i-1})/(1+\epsilon')$ and such that t is not $(1+\epsilon')$-covered by q_i. Then we must have $y(t) < y(q_i)/(1+\epsilon')$. Let s be the corner point immediately below u_i. From the definition of ZAG, s is a NO point. The coordinates of s are $x(s) = x(u_i) = x(q_{i-1})/c(1+\delta)$ and $y(s) = y(u_i)/(1+\delta)$. Thus, $x(t) < x(q_{i-1})/(1+\epsilon') \leq x(q_{i-1})/c(1+\delta)^2 \leq x(s)/(1+\delta)$ and $y(t) < y(q_i)/(1+\epsilon') = cy(u_i)/(1+\epsilon') \leq y(u_i)/(1+\delta)^2 = y(s)/(1+\delta)$. Therefore, $\text{GAP}_\delta(s)$ cannot return NO, a contradiction.

3. Suppose that there is a solution point t with x-coordinate $x(t) < x(q_r)/(1+\epsilon')$. Let s be the top corner point in the vertical line of w_r. By the termination condition of the algorithm, s is a NO point. The x-coordinate of s is $x(s) = x(w_r) = x(q_r)/c(1+\delta)$, hence $x(t) < x(q_r)/(1+\epsilon') \leq (q_r)/c(1+\delta)^2 = x(s)/(1+\delta)$. From the definition of the top line of the grid, we have also $y(t) \leq y(s)/(1+\delta)$. Therefore, again $\text{GAP}_\delta(s)$ cannot return NO, a contradiction. $\square$

We show now that any ϵ-Pareto set must contain at least $r = |Q|$ points.

Theorem 11. *Let $P^*_\epsilon = \{p_1, \ldots, p_k\}$ be an optimal ϵ-Pareto set, where its points p_i are ordered in increasing order of their y- and decreasing order of their x-coordinate. Then $k \geq r$ and $x(q_i) < x(p_i)(1+\delta)^2$ for every $i = 1, \ldots, r$.*

Proof. We prove by induction that if there is a q_i (i.e. $i \leq r$), then there is also a p_i (i.e., $i \leq k$) and $x(q_i) < x(p_i)(1+\delta)^2$.

Basis ($i = 1$). Consider the solution point $t_1 = \text{GAP}_\delta(u_1)$. We have $y(t_1) \leq y(u_1) = y(q_1)/c \leq y(q_1)/[(1+\delta)(1+\epsilon)]$. Point t_1 is covered by some element of P^*_ϵ; point p_1 has the smallest y-coordinate among the points of P^*_ϵ, therefore, $y(p_1) \leq y(t_1)(1+\epsilon) \leq y(q_1)(1+\delta)$. Since the point immediately to the left of q_1 is a NO point (by the definition of ZIG), we must have $x(p_1) > x(q_1)/(1+\delta)^2$.

Induction step. Suppose the claim holds for indices smaller than i; we will prove it for i. The proof is similar to the basis case. Consider the solution point $t_i = \text{GAP}_\delta(u_i)$. We have $y(t_i) \leq y(u_i) = y(q_i)/c \leq y(q_i)/[(1+\delta)(1+\epsilon)]$. Also, $x(t_i) \leq x(u_i) = x(w_{i-1}) = x(q_{i-1})/c(1+\delta) \leq x(q_{i-1})/(1+\epsilon)(1+\delta)^2$. All the previous points $p_1, \ldots p_{i-1}$ of P^*_ϵ have x-coordinate strictly greater than $x(q_{i-1})/(1+\delta)^2 \geq x(t_i)(1+\epsilon)$ by the induction hypothesis, and therefore they do not $(1+\epsilon)$-cover point t_i. Hence P^*_ϵ must have at least i points, and t_i is $(1+\epsilon)$-covered by one of the remaining points $p_i, \ldots$. Point p_i has the smallest y-coordinate among them. Therefore, $y(p_i) \leq y(t_i)(1+\epsilon) \leq y(q_i)/(1+\delta)$. Since the point immediately to the left of q_i is a NO point, we must have $x(p_i) > x(q_i)/(1+\delta)^2$. □

Combining the previous lemmas we have the following theorem.

Theorem 12. *For any $\epsilon' > \epsilon > 0$, we can construct an ϵ'-Pareto set R whose size is bounded by the size k of the smallest ϵ-Pareto set P^*_ϵ. The time complexity of the algorithm is $O(k \log(m/\delta))$ GAP_δ calls, where $1/\delta = O(1/(\epsilon' - \epsilon))$.*

3.4 Computing the best k solutions

Now let's consider the dual problem: We want to compute a set of k solutions that collectively approximate as closely as possible the Pareto curve. That is, we wish to find a set S of k solutions that minimizes the value of the ratio ρ such that S ρ-covers the whole set P of solutions. For $k = 1$, this solution is the "knee" of the Pareto set. It is a point which in some sense offers the best compromise between the different objectives.

Finding the knee (and more generally the best k points) is NP-hard even in simple cases. However, as we will show, we can approximate the optimal ratio, within any degree of accuracy $1+\theta$ in time polynomial in the input and $1/\theta$, provided that we have the GAP routine. In fact, in the knee ($k = 1$) case the result holds for any number of objectives.

We give first the algorithm for the $k = 1$ case (and any number d of objectives), and then present the algorithm for arbitrary k (and 2 objectives).

Theorem 13. *For any multiobjective optimization problem with a polynomial time GAP routine, and for any $\theta > 0$, we can compute a solution point that approximates the minimum ratio achieved by the knee within a factor $1+\theta$ in time $O(d \log(m/\delta))$ GAP_δ calls, where $1/\delta = O(1/\theta)$ and d is the number of objectives.*

Proof. We give the algorithm and proof for an arbitrary number d of objective functions $f_1, \ldots, f_d$. Assume again without loss of generality that all the objectives are to be minimized. For each $i = 1, \ldots, d$, let c^*_i be the minimum value of f_i achieved by any solution of the given instance. Let $\delta > 0$ be a suitable small rational such that $(1+\delta)^4$ approximates from below $1+\theta$.

Compute first for each $i = 1, \ldots, d$ a value c_i such that $c^*_i \leq c_i < c^*_i(1+\delta)^2$. We do this using a similar method to ZIG (and ZAG): Take a point $p = (p_1, \ldots, p_d)$ that exceeds the maximum possible values of all the objective functions by a factor $(1+\delta)$. For each $i = 1, \ldots, d$ do the following. Consider the line through p parallel to the ith axis as being geometrically subdivided with ratio $(1+\delta)$, i.e. subdivided by "grid" (corner) points with the ith coordinate equal to $p_i/(1+\delta)^j$, for $j = 0, 1, \ldots, 2m/\log(1+\delta) + 1$. Point p is a YES point (i.e. $\mathrm{GAP}_\delta(p)$=YES) while the lowest grid point on the line is a NO point. Do a binary search along the grid points of the line to find a minimal YES grid point, i.e. a YES point such that the one below it is a NO point. Let c_i be the ith coordinate $p_i/(1+\delta)^j$ of the minimal YES point. Then, by the definition of GAP we know that $c^*_i \leq c_i$ and $c^*_i > c_i/(1+\delta)^2$.

Let ρ^* be the minimum ratio achieved by the optimal knee solution point s^*. Since all solution points are ρ^*-covered by s^*, we must have $s^*_i \leq \rho^* c^*_i$ for all $i = 1, \ldots, d$. Let c be the point $(c_1, \ldots, c_d)$. Of course c may not be a solution point. If c is a YES point, then return $s = \mathrm{GAP}_\delta(c)$. Clearly, $s < s^*(1+\delta)^2$, hence s approximates the minimum ratio within a factor $(1+\delta)^2 \leq 1+\theta$.

Assume that c is a NO point. Consider the set of points $c(1+\delta)^j$, $j = 0, 1, \ldots, 2m/\log(1+\delta)+1$. The last point is clearly a YES point. Do a binary search among these points to find a minimal YES point, i.e. a YES point $c(1+\delta)^j$ such that $c(1+\delta)^{j-1}$ is a NO point. Return $t = \mathrm{GAP}_\delta(c(1+\delta)^j)$.

Since $c(1+\delta)^{j-1}$ is a NO point, $c(1+\delta)^{j-2}$ is not dominated by s^*, hence for at least one coordinate i we have $s_i^* > c_i(1+\delta)^{j-2}$. Thus, $\rho^* c_i^* > c_i(1+\delta)^{j-2}$, hence $\rho^* > (1+\delta)^{j-2}$. On the other hand, for all coordinates i we have, $t_i \leq c_i(1+\delta)^j < c_i^*(1+\delta)^{j+2}$. Therefore, the computed solution point t ρ-covers all solution points, where $\rho = (1+\delta)^{j+2} < \rho^*(1+\delta)^4 \leq \rho^*(1+\theta)$.

The time complexity of the algorithm is clearly $O(d\log(m/\delta))$ GAP_δ calls, and $1/\delta = O(1/\theta)$. □

We address now the case of an arbitrary number k of points and 2 objectives. Let $\rho^* = 1+\epsilon^*$ be the minimum ratio that can be achieved by k solution points. We will compute a set R of at most k solution points that achieves ratio $\rho \leq \rho^*(1+\theta)$.

Theorem 14. *We can approximate the smallest ratio $\rho^* = 1+\epsilon^*$ for which the ϵ^*-Pareto set has at most k points to a factor of $1+\theta$ in time $O(k\log^2(m/\delta))$ GAP_δ calls, where $1/\delta = O(1/\theta)$.*

Proof. Let $\delta > 0$ be a suitable small rational such that $(1+\delta)^4$ approximates from below $1+\theta$. Consider the following set of candidate ratios $\rho_i = (1+\delta)^i$, $i = 0, 1, \ldots, 2m/\log(1+\delta)+1$. Clearly, when i is the maximum value $2m/\log(1+\delta)+1$, then any single solution point ρ_i-covers all other solution points, i.e. any one point suffices. Do a binary search among the candidate ratios ρ_i to identify a minimal value $i*$ of $i \geq 4$ such that the relaxed ZIGZAG algorithm of the last subsection returns a ϵ'-Pareto set with at most k points when it is called with parameters $1+\epsilon' = \rho_{i*}$ and δ, with $c = (1+\delta)^{i*-2} = (1+\epsilon')/(1+\delta)^2$. Our algorithm returns this ϵ'-Pareto set R for the minimal such $i*$.

Clearly the algorithm returns a set R of at most k points. The approximation ratio ρ of R with respect to the Pareto curve is at most $\rho_{i*} = (1+\delta)^{i*}$. If $i* = 4$, then $\rho \leq 1+\theta$.

So suppose that $i* > 4$. Then the Relaxed ZIGZAG algorithm returns for $i = i*-1$, i.e., for $c = (1+\delta)^{i*-3}$, a set with more than k elements. By the results of the previous subsection we know that the smallest ϵ-Pareto set for $1+\epsilon \leq c/(1+\delta) = (1+\delta)^{i*-4}$ has more than k points. Therefore, $\rho^* = 1+\epsilon^* > (1+\delta)^{i*-4} \geq \rho/(1+\delta)^4$, and hence $\rho \leq \rho^*(1+\theta)$.

The number of calls to the Relaxed ZIGZAG algorithm is $2m/\log(1+\delta) = O(m/\delta)$ (for small δ). In order to limit the running time, we do not let the calls run to completion if they try to generate more than k points, but terminate them as soon as they try to generate a $(k+1)$th point. With this modification, each call takes time $O(k\log(m/\delta))$ GAP_δ calls. Thus, the total running time is $O(k\log^2(m/\delta))$ GAP_δ calls. □

3.5 Applications

Our results can be applied to all of the problems which have the required GAP routine, e.g. the classes of problems shown in [PY1, PY2], including multiobjective flows and other convex problems, shortest path, spanning tree, matching, and cost-time trade-offs in query evaluation.

In some cases, better complexity bounds can be obtained by using a sharper routine than GAP. We discuss briefly the case of shortest paths, with two objectives, cost and length. A stronger variant of the GAP problem in this case is the well-studied *Restricted Shortest Path* (RSP) problem: given a bound on the cost of the path, minimize the length of the path subject to the bound on the cost. This problem has been studied in a number of papers [Wa, Has, LR, GR+, ESZ]. The problem is

NP-hard, but it has a fully polynomial time approximation scheme. The best current algorithms approximate the optimal restricted path within factor $1+\epsilon$ in time $O(en/\epsilon)$ for acyclic graphs [ESZ], and time $O(en(\log\log n + 1/\epsilon))$ for general graphs [LR], where n is the number of nodes and e the number of edges.

One call to the RSP routine can be obviously used to solve the GAP problem. Moreover, we can use the RSP routine directly to implement (with one call) the ZIG and the ZAG operations in the algorithm. Hence, we can approximate within a factor of 3 the smallest ϵ-Pareto set for bicriteria shortest paths in time $O(enk(\log\log n + 1/\epsilon))$ (or $O(enk/\epsilon$ for acyclic graphs), where k is the size of the smallest ϵ-Pareto set. Also, for any $\epsilon' > \epsilon$, we can compute in time $O(enk(\log\log n + 1/(\epsilon' - \epsilon)))$ an ϵ'-Pareto set whose size is no more than k, the size of the smallest ϵ-Pareto set.

4 Three Objectives

We will first show in Section 4.1 that for 3 (and more) objectives it is impossible to guarrantee a constant factor approximation with the same ϵ. Then in Section 4.2, we show that if we relax ϵ to a larger value, $\epsilon' > (1+\epsilon)^2 - 1$, then we can achieve a factor 4 approximation.

4.1 Lower Bound

Theorem 15. *Any polynomial generic algorithm computing the smallest ϵ-Pareto set for a problem with more than two objective functions cannot be c-competitive for any constant c.*

Proof. Suppose again that we have minimization objectives; the same arguments hold for maximization or mixed objectives. Fix any rational $\epsilon > 0$ and pick any (rational) $\epsilon' > \epsilon$. Just like in the case of 2 objectives we will exploit the fact that $\mathrm{GAP}_\delta(\mathrm{b})$ is not uniquely defined for some points b. Again we will construct two sets, P and Q such that GAP_δ cannot distinguish between them, and the size of the optimal ϵ-Pareto set for P has one point, and for Q has arbitrarily many points.

We will use x, y, z for the three coordinates corresponding to the three objectives. Consider a point $p = (x(p), y(p), z(p))$, and let $q_i = \left(x(p)(1+\epsilon')^i, y(p)(1+\epsilon')^{k-i}, \frac{z(p)}{1+\epsilon}\right)$ for $i = 0 \ldots k$. Let $P = \{p, q_0, \ldots, q_k\}$. Clearly, $\{p\}$ is an ϵ-Pareto set for P. Let $r_i = \left(x(p)(1+\epsilon')^i, y(p)(1+\epsilon')^{k-i}, \frac{z(p)-1}{(1+\epsilon)}\right)$ for $i = 1, \ldots, k$, and let $Q = \{p, q_0, \ldots q_k, r_0, \ldots, r_k\}$, Notice that p is not $(1+\epsilon)$-covered by any of the q_i's, r_i's, hence p must belong to any ϵ-Pareto set for Q. Furthermore, r_i is not $(1+\epsilon)$-covered by any other point of Q except only for q_i. Therefore, every ϵ-Pareto set for Q must contain at least one of q_i, r_i for all $i = 1, \ldots, k$. Hence, the smallest ϵ-Pareto set for Q will have $k+1$ points.

Suppose that we have a polynomial generic algorithm. Let $x(p), y(p), z(p) = M$ be large integers (exponentially large in $1/\epsilon$ and the input). Again GAP_δ for $\delta > 1/(M-1)$, cannot distinguish between the two cases, since for all b where $\mathrm{GAP}_\delta(\mathrm{b})$ can return r_i it can either return q_i or return NO. Therefore, we cannot conclude if the size of the optimal solution is one point, or $k+1$ points for arbitrary k. Again, we can turn this argument into an NP-hardness proof by specifying a suitable 3-objective problem. □

In order to beat the lower bound above, we are forced to search for algorithms which will return ϵ'-Pareto sets, for $\epsilon' > \epsilon$ when the original problem has 3 or more objectives.

4.2 Three Objectives Algorithm

We will present an algorithm that is 4-competitive and returns an ϵ'-Pareto set for $(1+\epsilon') > (1+\epsilon)^2$. Choose a suitable small rational $\delta > 0$ such that $(1+\epsilon') > (1+\epsilon)^2(1+\delta)^4$. For small ϵ, ϵ', we can pick δ so that $1/\delta = O(1/(\epsilon' - 2\epsilon))$.

It is convenient for the algorithm and the proof to have $(1+\epsilon)$ be a power of $(1+\delta)$. This can be assumed without loss of generality. To see this, pick $\hat{\delta}$ such that $(1+\epsilon') > (1+\epsilon)^2(1+\hat{\delta})^6$, and let $1+\hat{\epsilon}$ be the smallest power of $(1+\hat{\delta})$ that is at least as large as $(1+\epsilon)$. Then $(1+\epsilon') > (1+\hat{\epsilon})^2(1+\hat{\delta})^4$. Apply the algorithm that we will describe with $\hat{\epsilon}$ in place of ϵ and $\hat{\delta}$ in place of δ. Then the algorithm will construct an ϵ'-Pareto set of size at most 4 times the size of the smallest $\hat{\epsilon}$-Pareto set. But, since $\hat{\epsilon} \geq \epsilon$, the smallest $\hat{\epsilon}$-Pareto set is no larger than the smallest ϵ-Pareto set. Therefore, the constructed ϵ'-Pareto set has the required property.

Assume for concreteness again that all objectives are to be minimized; the algorithm is similar in the other cases. As before, we will be working with a geometric grid of the space of objective values (equivalently, an arithmetic grid in the log scale). In this case we will use an asymmetric grid; it turns out that this is convenient as the algorithm treats the dimensions in a certain order, and it helps us to ensure a factor 4 approximation. (We have an alternative algorithm that uses a symmetric grid which achieves a factor 6.) We let the ratio of the grid in the x dimension be $(1+\delta)$ and in the y, z dimensions be $(1+\epsilon)(1+\delta)$. Let C be the set of all corner points of the grid where GAP_δ returns a solution. (We will not be computing these points explicitly). We assume in this subsection that the GAP_δ routine is monotonic, i.e., if $p \geq q$ and $\mathrm{GAP}_\delta(p) = \mathrm{NO}$ then also $\mathrm{GAP}_\delta(q) = \mathrm{NO}$.

We will outline first the algorithm, then prove its correctness, and finally sketch an efficient implementation.

The algorithm computes a set of corner points Q such that $\mathrm{GAP}_\delta(Q) = \{\mathrm{GAP}_\delta(q) \mid q \in Q\}$ is an ϵ'-Pareto set of size at most 4 times the size of the optimal ϵ-Pareto set P^*_ϵ. We say that a corner point $r \in C$ is *ineligible* at some time during the algorithm if there is a point $q \in Q$ such that $x(q) \leq x(r)(1+\epsilon)^2(1+\delta)^2$, $y(q) \leq y(r)(1+\epsilon)(1+\delta)$ and $z(q) \leq z(r)(1+\epsilon)(1+\delta)$; the conditions are asymmetric because of the asymmetry in the grid ratios. The corner point r is called *eligible* otherwise, and we let C/Q denote the set of eligible corner points. For a set S of points, we use $\min_x S$ to denote the subset of points of S that have minimum x coordinate, similarly define $\min_y S$ and $\min_z S$.

$Q = \emptyset$
While $C/Q \neq \emptyset$ do the following:
{ Find the point $p \leftarrow \min_y \min_x \min_z C/Q$.
  $S(p) = \{s \in C : x(s) \leq x(p)(1+\epsilon)(1+\delta), y(s) \leq y(p), z(s) \leq z(p)(1+\epsilon)(1+\delta)\}$.
  $T(p) = \{t \in C : x(t) \leq x(p), y(t) \leq y(p)(1+\epsilon)(1+\delta), z(t) \leq z(p)(1+\epsilon)(1+\delta)\}$.
  Let $s(p) \in \min_y S(p)$ and $t(p) \in \min_x T(p)$ be points in the corresponding sets.
  Update $Q \leftarrow Q \cup \{s(p), t(p)\}$.
}
Return $R = \mathrm{GAP}_\delta(Q) = \{\mathrm{GAP}_\delta(\mathrm{q}) \mid \mathrm{q} \in \mathrm{Q}\}$

Remark: The algorithm does not compute explicitly the sets of points C/Q, $S(p)$, $T(p)$. It only maintains the set Q and computes in each iteration the points p, $s(p)$, $t(p)$. After proving the correctness of the algorithm, we will describe how these points can be computed efficiently.

Theorem 16. *Let P^*_ϵ be the optimal ϵ-Pareto set. The set R computed by the above algorithm forms a ϵ'-Pareto set, and $|R| \leq 4|P^*_\epsilon|$.*

Proof. We will charge each point of Q (and R) to a point of the optimal ϵ-Pareto set P^*_ϵ so that every point $p^* \in P^*_\epsilon$ is charged with at most 4 points of Q.

Note that if a corner point r satisfies $r \geq p^*(1+\delta)$, then $GAP_\delta(r)$ returns a solution, hence $r \in C$. Let $\hat{p^*}$ denote the corner point obtained by rounding up each coordinate of $p^*(1+\delta)$ to the grid. Then $\hat{p^*} \in C$. Note that if p^* $(1+\epsilon)$-covers a corner point p then $\hat{p^*} \leq p(1+\epsilon)(1+\delta)$. The reason is that $p^* \leq p(1+\epsilon)$ implies that $p^*(1+\delta) \leq p(1+\epsilon)(1+\delta)$; the right hand side is a corner point (since p is), thus rounding up the left hand side to the nearest corner point $\hat{p^*}$ preserves the inequality.

Let $Z^-(p^*)$ be the set of corner points in C that are $(1+\epsilon)$-covered by p^* and have a lower z value than $\hat{p^*}$, and let $Z^+(p^*)$ be the set of corner points in C that are $(1+\epsilon)$-covered by p^* and have an equal or higher z value than $\hat{p^*}$. We will charge a pair of points $s(p), t(p)$ of Q to p^* if p is $(1+\epsilon)$-covered by p^*. We will argue that at most one pair of points is charged to p^* from a point $p \in Z^-(p^*)$ and at most one pair from a point $p \in Z^+(p^*)$.

Consider first $Z^-(p^*)$. Since the ratio in the z direction is $(1+\epsilon)(1+\delta)$, all points of $Z^-(p^*)$ must have the same z coordinate, specifically $z(\hat{p^*})/(1+\epsilon)(1+\delta)$. Look at the first time (if any) that a point of $Z^-(p^*)$ is selected as the point p by the algorithm. We will show that $s(p)$ $(1+\epsilon)(1+\delta)$-covers all of the remaining eligible points q in $Z^-(p^*)$ (i.e. not already covered by Q).

As noted above, every other point $q \in Z^-(p^*)$ has $z(q) = z(p)$. This implies that $x(p) \leq x(q)$, since p was selected instead of q. Suppose that $y(\hat{p^*}) > y(p)$. Then $y(p) \leq y(\hat{p^*})/(1+\epsilon)(1+\delta)$ because of the grid ratio along the y dimension, and we know that $y(\hat{p^*})/(1+\epsilon)(1+\delta) \leq y(q)$ because q is $(1+\epsilon)$-covered by p^*; hence $y(p) \leq y(q)$. Thus, if $y(\hat{p^*}) > y(p)$ then every other remaining corner point in $Z^-(p^*)$ is dominated by p. Since $p \in S(p)$ by the definition, the point $s(p)$ $(1+\epsilon)(1+\delta)$-covers p, and hence also all of the remaining points q in $Z^-(p^*)$.

Suppose that $y(\hat{p^*}) \leq y(p)$. Then $\hat{p^*} \in S(p)$, and therefore $y(s(p)) \leq y(\hat{p^*}) \leq y(q)(1+\epsilon)(1+\delta)$. Since p dominates q in the other two coordinates x, z, it follows that $s(p)$ $(1+\epsilon)(1+\delta)$-covers all of the remaining points q of $Z^-(p^*)$ in this case also.

Now consider $Z^+(p^*)$. Let p be the first point in $Z^+(p^*)$ (if any) selected by the algorithm. Then every other remaining eligible point q has $z(q) \geq z(p)$. We distinguish two cases, depending on the x coordinates of p and $\hat{p^*}$.

Suppose that $x(p) \leq x(\hat{p^*})$. If $y(\hat{p^*}) > y(p)$ then all eligible corner points q that are $(1+\epsilon)$-covered by p^* must have $y(q) \geq y(p)$; since $p \in S(p)$, it follows that $y(s(p)) \leq y(p) \leq y(q)$ in this case. If $y(\hat{p^*}) \leq y(p)$ then $\hat{p^*} \in S(p)$, and therefore $y(s(p)) \leq y(\hat{p^*}) \leq y(q)(1+\epsilon)(1+\delta)$. Further, $z(s(p)) \leq z(p)(1+\epsilon)(1+\delta) \leq z(q)(1+\epsilon)(1+\delta)$. And, $x(s(p)) \leq x(p)(1+\epsilon)(1+\delta) \leq x(\hat{p^*})(1+\epsilon)(1+\delta) \leq x(q)(1+\epsilon)^2(1+\delta)^2$. Thus, $s(p)$ makes ineligible every remaining point q of $Z^+(p^*)$. Note furthermore that $x(\hat{p^*}) \leq x(p^*)(1+\delta)^2$, and therefore $s(p)$ $(1+\epsilon)^2(1+\delta)^3$-covers all solution points (not just corner points) that are $(1+\epsilon)$-covered by p^*.

Suppose that $x(p) > x(\hat{p^*})$. Since $z(p) \geq z(\hat{p^*})$, we must have $y(p) < y(\hat{p^*})$. For, otherwise $\hat{p^*}$ would dominate p, and either $\hat{p^*}$ was already made ineligible by Q, in which case p was also ineligible, or else the algorithm should have selected $\hat{p^*}$ in place of p because it has at least as small z coordinate and strictly smaller x coordinate. We will show that $t(p)$ $(1+\epsilon)(1+\delta)$-covers every remaining point $q \in Z^+(p^*)$. Note that $\hat{p^*} \in T(p)$, hence $x(t(p)) \leq x(\hat{p^*}) \leq x(q)(1+\epsilon)(1+\delta)$. Since $y(p) < y(\hat{p^*})$ and because of the ratio in the y dimension we have $y(t(p)) \leq y(p)(1+\epsilon)(1+\delta) \leq y(\hat{p^*}) \leq y(q)(1+\epsilon)(1+\delta)$. Finally, $z(p) \leq z(q)$ for still eligible q in $Z^+(p^*)/Q$, since p was selected by the algorithm before q, hence $z(t(p)) \le z(q)(1+\epsilon)(1+\delta)$. Thus, $t(p)$ will $(1+\epsilon)(1+\delta)$-cover $Z^+(p^*)$. Since the algorithm took both $s(p)$ and $t(p)$, we can charge these two points to cover all of those in $Z^+(p^*)$.

Overall we have charged 4 points of Q to cover all of the corner points $(1+\epsilon)$-covered by p^*.

At the end of the algorithm every corner point $r \in C$ is ineligible, i.e., there is a point $q \in Q$ that is within a factor $(1+\epsilon)^2(1+\delta)^2$ in the x dimension and within $(1+\epsilon)(1+\delta)$ in the y, z dimensions. For every solution point u, let $\hat{u}$ be the corner point obtained by rounding $u(1+\delta)$ up to the nearest corner point. Then $\hat{u}$ is in C and it is within a factor $(1+\delta)^2$ of u in the x dimension and within $(1+\epsilon)(1+\delta)^2$ in the y, z dimensions. Since $(1+\epsilon') > (1+\epsilon)^2(1+\delta)^4$, it follows that every solution point u is $(1+\epsilon')$-covered by some point of Q, and hence also by some solution point of R. □

We discuss now the implementation of the algorithm. For a given (corner) point p, let $S'(p)$ be the subset of $S(p)$ where the inequalities on x and z are satisfied with equality, i.e., $S'(p) = \{s \in C : x(s) = x(p)(1+\epsilon)(1+\delta), y(s) \le y(p), z(s) = z(p)(1+\epsilon)(1+\delta)\}$, and let $T'(p) = \{t \in C : x(t) \le x(p), y(t) = y(p)(1+\epsilon)(1+\delta), z(s) = z(p)(1+\epsilon)(1+\delta)\}$ be the subset of $T(p)$ where the inequalities on y and z are satisfied with equality. By the monotonicity of the GAP routine we have $\min_y S(p) = \min_y S'(p)$ and $\min_x T(p) = \min_x T'(p)$. Note that the points of $S'(p)$ lie on a line parallel to the y axis, and similarly the points of $T'(p)$ lie on a line parallel to the x axis. Thus, we can compute $s(p)$ and $t(p)$ in $O(\log(m/\delta))$ GAP_δ calls using the binary search technique, as in the 2-objective case.

The question remains of how to efficiently find $p \leftarrow \min_y \min_x \min_z C/Q$. An obvious solution is to scan through the z values from smallest to largest, and at each z value use the 2 objective algorithm to find p. Ignore the point if it is already covered (made ineligible) by another point in Q and continue. However, this involves both a linear scan through all z values (of cost at least $O(m/\delta)$) and potentially many points p which are covered by others in Q. The following lemma provides a more efficient method.

Lemma 17. *We can compute $p \leftarrow \min_y \min_x \min_z C/Q$ using $O(\log(m/\delta))$ GAP_δ calls.*

Proof. Since we consider points in increasing z value, we only need to consider the x-y projection of the points in Q and maintain the frontier F of points undominated in x and y. These points are sorted in increasing order by their x coordinate, as $q_1, \ldots, q_l$. A remaining point is ineligible iff its projection is dominated by one of the points $q'_i = \left(\frac{x(q_i)}{(1+\epsilon)^2(1+\delta)^2}, \frac{y(q_i)}{(1+\epsilon)(1+\delta)}\right)$. Thus, the set of ineligible points is the region of points that lie at or above and to the right of a rectilinear curve that passes through the points q'_i (see Figure 5). The set of eligible points is the set of YES corner points that lie in the region *strictly* below this curve; equivalently, it is the set of YES points that lie at or below and to the left of the rectilinear curve obtained by shifting left by a factor $1+\delta$ (the ratio along the x dimension) and down by a factor $(1+\epsilon)(1+\delta)$ (the ratio along the y dimension). The convex corners of the region are $c_j = \left(\frac{x(q_{j+1})}{(1+\epsilon)^2(1+\delta)^3}, \frac{y(q_j)}{(1+\epsilon)^2(1+\delta)^2}\right)$, $j = 0, \ldots, l$. (We let $y(c_0), x(c_l)$ be the maximum possible values of the objectives, and omit the points whose values are below the minimum.)

Observe that every eligible point dominates one of the c_j's. For each j let h_j = minimum z such that $\mathrm{GAP}_\delta\left(\frac{x(q_{j+1})}{(1+\epsilon)^2(1+\delta)^3}, \frac{y(q_j)}{(1+\epsilon)^2(1+\delta)^2}, z\right)$ returns YES. We can compute h_j via a binary search in $O(\log(m/\delta))$ GAP_δ calls for each j. We maintain the h_j's in a priority queue H, breaking ties according to the index j (equivalently, according to the x-coordinate of c_j). When we add a new

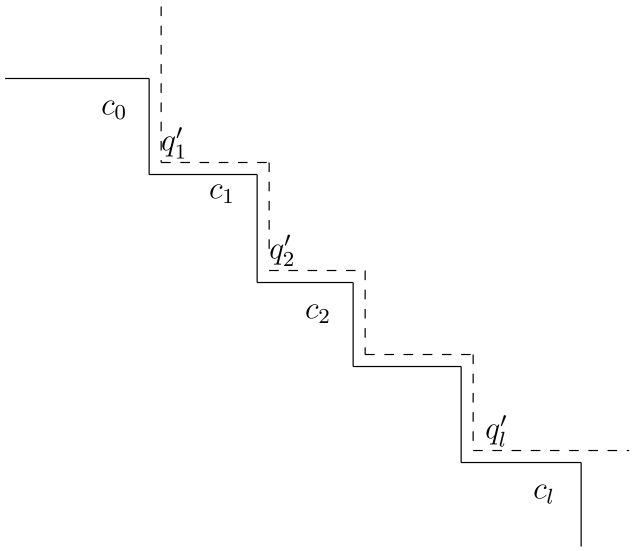

Figure 5: Frontier of ineligible points in x-y projection

point to Q, we may eliminate some of the elements of the frontier F (if they become dominated on the $x-y$ plane by the new point), and we will create at most two more intervals. The computation of two more h_j values can be done in the time allotted.

Now, instead of doing a linear scan through the z values, we can perform an Extract_min operation on the priority queue H to obtain the next smallest $z = h_j$ value where we will be guaranteed to find a point in C/Q. Once we find the smallest $z = h_j$ value and the index j, we limit our search to the set of points that lie on the hyperplane $z = h_j$ and whose x-y projection dominates c_j. We seek the YES corner point in this quarter-plane that has minimum x coordinate, with ties broken according to the y coordinate. This is a 2-dimensonal problem now, and we can find the desired point p of C/Q by calling $ZAG(ZIG(c_j))$. ☐

Theorem 18. *The algorithm to compute the ϵ'-Pareto set of size at most $4k$, where k is the size of the smallest ϵ-Pareto set, can be implemented to run in time $O(k \log(m/\delta))$ GAP_δ calls, where $1/\delta = O(1/(\epsilon' - 2\epsilon))$.*

Proof. By Theorem 16, the set R, and hence also the set Q, has at most $2k$ points. Since every iteration adds 2 points to Q, it follows that the algorithm performs at most $2k$ iterations. In each iteration the algorithm computes the point p and the points $s(p)$ and $t(p)$. By Lemma 17 and the discussion preceding the lemma, these points can be computed in $O(\log(m/\delta))$ GAP_δ calls. Thus, the total number of GAP calls is $O(k \log(m/\delta))$. The rest of the time is dominated by the GAP calls. ☐

5 d Objectives

We have shown that for $d \geq 3$ objectives we are forced to compute an $\epsilon' > \epsilon$-Pareto set, if we are to have a guarantee on its size. In fact, we can easily find a $\log n$-competitive algorithm for the problem, if we are willing to spend time that grows with the dth power of the number of bits. Let δ be such that $(1+\epsilon') \geq (1+\epsilon)(1+\delta)^2$.

Theorem 19. *For any $\epsilon' > \epsilon$ we can compute an ϵ'-Pareto set Q such that $|Q| \leq (d\log(m/\delta))|P^*_\epsilon|$ using $O((m/\delta)^d)$ GAP calls.*

Proof. The algorithm will proceed in two stages. In the first stage, we will compute a δ-Pareto set, by using the original algorithm of [PY1]. Break up the solution space using a geometric grid of size $\sqrt{1+\delta}$ and call $\mathrm{GAP}_{\sqrt{1+\delta}}$ on all of the corner points, while keeping an undominated subset R. Note that $|R| \leq O(m/\delta)^{d-1}$. Now we can phrase the problem as a set cover problem. Let the universe be all of the points in R, and for each $r \in R$ associate a set $S_r = \{$ the points that $(1+\epsilon)(1+\delta)$-cover $r\}$. The smallest set cover, will comprise an $(1+\epsilon)(1+\delta)^2$-Pareto set, Q. Since we can compute a $\log n$ approximation for the Set Cover problem on a universe of size n, the result follows. □

Unfortunately, the algorithm above is the best that we know of for $d > 3$. There are two aspects in which this algorithm is inferior to the ones we presented earlier (even for fixed d): The approximation ratio is not constant, and the running time grows with m rather than $\log m$.

We show that, even if all the solution points are given explicitly as input, we cannot do better than the Set Cover problem in high dimensions.

Theorem 20. *Even if all the solution points are given explicitly in the input, for any $\epsilon > 0$, we cannot approximate the smallest ϵ-Pareto set on d objectives in polynomial time to within $c\ln d$ for any constant $c < 1$ unless NP is contained in $\mathrm{DTIME}(n^{\log\log n})$.*

Proof. We will prove this via a gap-preserving reduction from *SetCover*. In a set cover instance we are given a universe of elements U with n elements and subsets $S_1, \ldots, S_l \subseteq U$. We are then asked to select a minimum number of subsets S_i such that their union is U. It is well known that the Set Cover problem is hard to approximate [LY, Fei].

Our reduction is as follows: for each element $u_i \in U$ add a point p_i in the solution space, whose ith coordinate is $1/(1+\epsilon)$ and all other coordinates are at ∞. For each set S_j we add a point q_j such that the ith coordinate of q_j is 1 if $u_i \in S_j$ and $(1+\epsilon)^3$ otherwise. Finally, we add a point r with value $(1+\epsilon)$ in all dimensions.

Let P_ϵ be the smallest ϵ-Pareto set. Since r cannot be $(1+\epsilon)$-covered by any other points, r must be part of the final solution. Since every point p_i is $(1+\epsilon)$-covered in P_ϵ, the sets corresponding to the q_is must form a valid set cover. Finally it is easy to see that this approximation is gap preserving and the theorem follows from the hardness of the Set Cover problem. □

Observe that the above reduction breaks down if we are allowed to relax the ϵ value, since for $(1+\epsilon') = (1+\epsilon)^2$ the ϵ'-Pareto set will always contain just the single point r. We show below that in high dimensions we cannot achieve a constant factor approximation to the ϵ-Pareto set, even if we are allowed to relax ϵ by an arbitrary constant.

Theorem 21. *Let $(1+\epsilon') < (1+\epsilon)^{\log^* d/3}$. Even if all the solution points are given explicitly in the input, it is impossible to compute in polynomial time an ϵ'-Pareto set whose size is within a $\log^* d$ factor of the smallest ϵ-Pareto set unless $NP \subseteq \mathrm{DTIME}(n^{\log\log n})$.*

Proof. We will use a reduction from asymmetric k-center along with a recent result by Chuzhoy et al. [CG+] to finish the proof. In the asymmetric k-center problem we are given a set of nodes V with distances, $dist(u, v)$ that must satisfy the triangle inequality, but may be asymmetric: i.e. $dist(u, v) \neq dist(v, u)$. We are asked to find a subset $U \subseteq V$, $|U| = k$, that minimizes $dist^* = \max_{v \in V} \min_{u \in U} dist(u, v)$. It is shown in [CG+] that this problem cannot be approximated in polynomial time within any constant factor unless P=NP, and cannot be approximated within a factor $\log^* n - \alpha$, for some constant α unless $NP \subseteq \mathrm{DTIME}(n^{\log\log n})$. Furthermore, even if we are allowed to use $k \log^* n$ centers we cannot approximate the optimal distance $dist^*$ within $\log^* n/2 - \alpha$ unless $NP \subseteq \mathrm{DTIME}(n^{\log\log n})$ [Chu]. We remark also that the distances $dist(u, v)$ in the construction of [CG+] are distances in an unweighted directed graph, i.e. they are small integers or ∞.

Let us choose a value $dist'$ which we will specify later, and encode the asymmetric k-center problem as follows. We have one coordinate (objective) for each node $i \in V$. For each node $i \in V$ create a point p_i such that, the ith coordinate of p_i is 1, and the jth coordinate, for each $j \neq i$, is $(1 + \epsilon)^{[dist_{ij}/dist']}$.

Notice that if $dist_{ij} \leq dist'$ then p_i $(1 + \epsilon)$ covers p_j. That is, for every coordinate l, $(p_i)_l \leq (1 + \epsilon)(p_j)_l$. This is clearly true for $l = i$ and $l = j$. For the other coordinates, the triangle inequality implies $dist_{ij} + dist_{jl} \geq dist_{il}$. Therefore, $(p_i)_l = (1 + \epsilon)^{[dist_{il}/dist']} \leq (1 + \epsilon)^{[dist_{ij}/dist']} \cdot (1 + \epsilon)^{[dist_{jl}/dist']} \leq (1 + \epsilon)(p_j)_l$.

Conversely, if p_i $(1 + \epsilon)$-covers p_j then $(p_i)_j = (1 + \epsilon)^{[dist_{ij}/dist']} \leq (1 + \epsilon)(p_j)_j = (1 + \epsilon)$, hence $dist_{ij} \leq dist'$.

Thus, if $dist' = dist^*$ then the smallest ϵ-Pareto set will contain precisely k points and correspond to the optimal solution. In a similar way if we can compute a $(1 + \epsilon)^a$-Pareto set of size less than ck then we can approximate $dist^*$ to a factor of a while using less than ck centers. Thus, if we try every pairwise distance $dist_{ij}$ for $dist'$ (or we do a binary search) to find the lowest distance that still uses fewer than ck centers, we can obtain an (a, c) approximation to the asymmetric k-center problem. However, this problem is hard to approximate to a factor of $\log^* n/3$ even when using $k \log^* n$ centers. □

Note that the theorem implies also the hardness of approximation for the dual problem of computing the best k points that approximate the Pareto curve as closely as possible (with the minimum ratio $1 + \epsilon$). The theorem implies that the optimal ratio cannot be approximated within a power $\log^* n/3$ of the ratio, and this holds even if we use $k \log^* n$ points.

Conversely, we can reduce the dual problem of finding the best k points to an asymmetric k-center problem.

Theorem 22. *Suppose that there is an ϵ-Pareto curve that contains k points. Then for any $\delta > 0$, we can compute k points which approximate the Pareto curve with ratio $(1 + \epsilon')^{O(\log^* k)}$ using $O((m/\delta)^d)$ GAP calls, where $1 + \epsilon' = (1 + \epsilon)(1 + \delta)^2$.*

Proof. Suppose first that we are given explicitly a set P of points in d dimensions and a parameter k. Assume without loss of generality that all objectives are minimization objectives. Construct an instance of the asymmetric k-center problem that contains a node u for each point $u \in P$ and define the distances between the nodes as follows: $dist_{uv} = \max(\max_i\{\log(u_i/v_i)\}, 0)$. Note that by definition, point u $2^{dist_{uv}}$-covers point v. It is easy to see that the distances satisfy the triangle inequality: For every triple of points (nodes) u, v, w and every coordinate i, the inequalities $u_i \leq 2^{dist_{uv}} v_i$ and $v_i \leq 2^{dist_{vw}} w_i$ imply that $u_i \leq 2^{dist_{uv}+dist_{vw}} w_i$, hence $dist_{uw} \leq dist_{uv} + dist_{vw}$.

The asymmetric k-center problem can be approximated with ratio $O(\log^* k)$ [Ar, PV]. We can use this algorithm to find a set of k centers, which correspond to a set of k points of P. If the optimal ratio that can be achieved with k points in the original Pareto problem is $1+\epsilon$, then the optimal distance in the asymmetric k-center problem is $dist^* = \log(1+\epsilon)$. Every node is within distance $O(\log^* k \log(1+\epsilon))$ from one of the k centers, hence every point of P is $(1+\epsilon)^{O(\log^* k)}$-covered by one of the selected k points.

For a general multiobjective problem where the solution points are not given explicitly, we impose a geometric $\sqrt{1+\delta}$ grid, call $\mathrm{GAP}_{\sqrt{1+\delta}}$ at the grid points, and then apply the above algorithm to the set of points returned. Then the set of k points computed by the algorithm provide a $(1+\epsilon')^{O(\log^* k)}$-cover of the Pareto curve, where $1+\epsilon' = (1+\epsilon)(1+\delta)^2$. □

6 Conclusions

An approximate Pareto curve is a convenient way to represent the trade-offs between different objectives over the whole range of solutions of a multiobjective problem. It is important to do this using as few solutions (points) as possible, while ensuring a desired degree of approximation to the Pareto curve. In this paper we have raised this succinctness issue as an important goal in multiobjective optimization, and have taken the first steps in addressing it algorithmically. Of course, if it is impossible to compute in polynomial time an ϵ-Pareto set of any size, then it is certainly impossible to compute the most succinct one or even approximate it within any polynomial factor. For this reason, we focused on the class of multiobjective problems for which some ϵ-Pareto set can be constructed in polynomial (or fully polynomial) time; this class is characterized by the existence of an efficient routine for a GAP problem associated with the multiobjective optimization problem. We investigated general-purpose efficient algorithms that work for the whole class of such multiobjective problems. In the design of the algorithms we paid special attention to (1) minimize the number of calls that they make to the GAP routine since this will be in general the dominant term in the running time, and (2) ensure that the time depends on the number k of points in the minimum ϵ-Pareto set, so that if the given instance has a very small ϵ-Pareto set (i.e. k is very small), then we can find it more quickly and do not incurr the worst-case complexity.

We studied two dual versions of the problem, corresponding to the two parameters of interest: the number k of solutions in the computed approximate Pareto curve, and the degree $1+\epsilon$ of the approximation to the Pareto curve that is achieved. In the first version, we are given an error tolerance ϵ and wish to compute a minimum set of points that forms an ϵ-Pareto curve. For two objectives we have a tight result: we can approximate the size of the smallest ϵ-Pareto set within a factor of 3, and the factor 3 is best possible in general. Furthermore, if we relax ϵ to any larger $\epsilon' > \epsilon$, then we can compute an ϵ'-Pareto set of size no larger than the smallest ϵ-Pareto set. For three objectives, we showed that it is impossible in general to guarantee a constant factor approximation for the same ϵ. We gave an efficient algorithm that guarantees a factor of 4 if we relax ϵ to $\epsilon' > (1+\epsilon)^2 - 1$ ($\approx 2\epsilon$ for small ϵ). For an arbitrary number d of objectives, a (nonconstant factor) approximation can be obtained for any $\epsilon' > \epsilon$ using a reduction to the Set Cover problem. However, even if the whole set of solution points is given explicitly, then the minimum ϵ-Pareto set cannot be efficiently approximated better than a logarithmic factor, and furthermore, it cannot be approximated within a constant factor even if we relax ϵ by an arbitrary constant.

In the dual version of the problem, we are given a bound k on the number of solutions that we are willing to select, and we wish to pick an optimal set of k solution points that minimizes the approximation ratio $\rho = 1 + \epsilon$ to the Pareto curve. These are the points that represent best the whole spectrum of solutions. We showed that for $k = 1$ (the "knee" problem), we can find efficiently a point whose ratio is arbitrarily close to the optimal ratio, for any number of objectives. For arbitrary k we gave an algorithm that achieves this in the biobjective case. For general number d of objectives and general k, finding a good approximation is intractable even if the whole set of points is explicitly given; in fact one cannot guarantee any constant power of the optimal ratio ρ, and this fact holds even if we are allowed to use more than k points (any constant factor more).

The problem of constructing an approximately minimum ϵ-Pareto for the case that the whole set of points is given explicitly was studied recently and independently by Koltun and Papadimitriou [KP], motivated by a database application. (Note the major difference with our setting: in a typical multiobjective optimization problem there are exponentially many solution points and they are not given explicitly). For $d = 2$ dimensions, [KP] showed that the problem can be solved optimally by a greedy algorithm (similar to the Cleanup phase of Section 3.2). For $d = 3$ dimensions, they showed that the problem is NP-hard and they gave a constant factor approximation algorithm. For arbitrary d, they observed also that the problem can be reduced to the Set Cover problem (but they did not observe the reverse reduction).

There is a number of open problems and further directions of research that deserve to be investigated. We mention some of the immediate ones related to the subject of this paper. For 2 objectives our results are tight. For 3 objectives probably they are not (as far as we know). Can we extend the factor 4 algorithm to work for any $\epsilon' > \epsilon$? Can we improve the factor 4? What is the best ratio achievable? The case of 4 (or more) objectives is completely open. Can we achieve a constant factor approximation for some relaxed ϵ', eg. for some $\epsilon' \leq (1+\epsilon)^c - 1$ for some constant c (i.e. $\epsilon' \leq c\epsilon$ for small ϵ)? Similar questions apply to the dual version of the problem.

The above questions concern the whole classes MPTAS and MFPTAS of multiobjective problems for which we can construct ϵ-Pareto sets in polynomial and fully polynomial time for any ϵ. It is of interest to investigate these questions for specific important multiobjective problems. Individual concrete problems in the classes may admit specialized, problem-specific algorithms that achieve better approximations than are possible for the class as a whole. Finally similar questions can be investigated for the multiobjective analogue of APX (call it MAPX) of problems for which an ϵ-Pareto set is constructible in polynomial time for some ϵ, but not all ϵ.

References

[Ar] A. F. Archer. Two $O(\log^* k)$-approximation algorithms for the asymmetric k-center problem. *Proc. 8th Conf. on Integer programming and Combinatorial Optimization*, pp. 1-14, 2001.

[CJK] T. C. E. Cheng, A. Janiak, and M. Y. Kovalyov. Bicriterion Single Machine Scheduling with Resource Dependent Processing Times. *SIAM J. Optimization*, 8(2), pp. 617–630, 1998.

[CG+] J. Chuzhoy, S. Guha, E. Halperin, S. Khanna, G. Kortsartz, S. Naor Asymmetric k-center is $\log^* n$-hard to Approximate. *Proc. 36th ACM STOC*, pp. 21-27, 2004.

[Chu] J. Chuzhoy, private communication, 2004.

[Cli] J. Climacao, Ed. *Multicriteria Analysis*. Springer-Verlag, 1997.

[Ehr] M. Ehrgott. *Multicriteria optimization*. Springer-Verlag, 2000.

[ESZ] F.Ergun, R.Sinha, and L.Zhang. An improved FPTAS for Restricted Shortest Path. *Information Processing Letters* 83(5):237-293, 2002.

[Fei] U. Feige. A threshold of $\ln n$ for approximating set cover. *JACM*, 45(4), pp. 634–652, 1998.

[GJ] M. R. Garey, D. S. Johnson. *Computers and Intractability.* W. H. Freeman, 1979.

[GR+] A. Goel, K. G. Ramakrishnan, D. Kataria, and D. Logothetis. Efficient computation of delay-sensitive routes from one source to all destinations. *Proc. IEEE INFOCOM*, 2001.

[GG+] S. Guha, D. Gunopoulos, N. Koudas, D. Srivastava, and M. Vlachos. Efficient Approximation of Optimization Queries Under Parametric Aggregation Constraints. *Proc. 29th VLDB*, 2003.

[Han] P. Hansen. Bicriterion Path Problems. *Proc. 3rd Conf. Multiple Criteria Decision Making Theory and Application*, pp. 109–127, Springer Verlag LNEMS 177, 1979.

[Has] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1), pp. 36–42, 1992.

[IK] O. H. Ibarra, C. E. Kim. Fast approximation algorithms for the knapsack and sum of subsets problem. *JACM*, 22(4), pp. 463–468, 1975.

[KP] V. Koltun, C. H. Papadimitirou. Approximately dominating representatives. *Proc. Intl. Conf. Database Theory*, Springer Verlag, pp. 204-214, 2005.

[LR] D. H. Lorenz, D. Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 28(5), pp. 213–219, 2001.

[LY] C. Lund, M. Yannakakis. On the hardness of approximating minimization problems. *JACM*, 41(5), pp. 960–981, 1994.

[PV] R. Panigrahy, S. Vishwanathan. An $O(\log^* n)$ approximation algorithm for the asymmetric p-center problem. *J. of Algorithms*, 27(2), pp. 259-268, 1998.

[PY1] C.H.Papadimitriou, M.Yannakakis. On the Approximability of Trade-offs and Optimal Access of Web Sources. In *Proceedings 41st IEEE Symp. on Foundations of Computer Science*, 2000.

[PY2] C.H.Papadimitriou, M.Yannakakis. Multiobjective Query Optimization. In *Proceedings of PODS* 2001.

[RM+] R. Ravi, M.V. Marathe, S.S. Ravi, D.J. Rosenkrantz, and H.B. Hunt. Many Birds with One Stone: Multi-objective Approximation Algorithms. *Proc. 25th STOC*, pp. 438-447, 1993.

[Wa] A. Warburton. Approximation of Pareto Optima in Multiple-Objective Shortest Path Problems. *Operations Research*, 35, pp. 70-79, 1987.