Efficiently Computing Succinct Trade-off Curves

Sergei Vassilvitskii[1] and Mihalis Yannakakis[2]

[1] Stanford University, Stanford, CA;
sergei@cs.stanford.edu
[2] 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 efficeintly 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 come 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 travelled, 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 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) posesses 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.

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 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? These are the questions we investigate in this paper. 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 points than k points.

We also discuss 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* 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.

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 $O(k \log(m/\epsilon))$ *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: the problem is equivalent to the Set Cover problem. Furthermore, 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). 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.

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 $O((4m/\epsilon)^{d-1})$. Furthermore, for every fixed d, there is an algorithm for constructing a $P_\epsilon(x)$ in time polynomial in $|x|$ and $1/\epsilon$ (i.e., a fully polynomial time approximation scheme - FPTAS) if and only if there is a subroutine GAP that solves the following problem in time polynomial in $|x|$ and $1/\epsilon$: given x and d-vector b, either return a solution whose vector dominates b or report that there does not exist any solution whose vector is better than b by more than a $(1+\epsilon)$ factor in all of the coordinates. For simplicity, we will usually drop the instance x from the notation and use $\text{GAP}_\epsilon(\text{b})$ to denote the solution returned by the subroutine. To make the presentation easier, we will also say that GAP returns YES if it returns a solution, and returns NO otherwise.

We will assume from now on that the routine GAP exists, and will present our algorithms using this subroutine as a black box. We say that such an algorithm 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 polynomial time.

3 Two Objectives

3.1 Lower Bound

Theorem 1. *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. In particular, there is a biobjective problem with a polynomial time* GAP *procedure that cannot be approximated within a factor better than 3 unless P=NP.*

Proof. Suppose we have minimization objectives (the same holds for maximization or mixed objectives). Here we exploit the fact that there are points b on which $\text{GAP}_\delta(\text{b})$ is not uniquely defined. Consider the following set of points $p = (p_x, p_y)$, $q = \left(p_x(1+\epsilon)+1, \frac{p_y}{1+\epsilon}\right)$ and $r = \left(p_x(1+\epsilon)+1, \frac{p_y-1}{(1+\epsilon)}\right)$. Let $P = \{p, q\}$ and $Q = \{p, q, r\}$. The smallest ϵ-Pareto set for P consists of only one point, while the smallest ϵ-Pareto set for Q must include at least two points.

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, we notice that for the points remaining $\text{GAP}_\delta(\text{b})$ can return NO. But then, using the GAP_δ function as a black box, we cannot say whether

or not r is part of the solution, and thus are forced to take at least two points, even when we are presented with the set P.

We can make a symmetric observation if instead of q and r we have $q' = \left(\frac{p_x}{1+\epsilon}, p_y(1+\epsilon)+1\right)$ and $r' = \left(\frac{p_x-1}{(1+\epsilon)}, p_y(1+\epsilon)+1\right)$. Here again using the GAP_δ routine we cannot decide 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 $P = \{p, q, r, q', r'\}$.

We can turn this into an NP-hardness proof by defining a suitable biobjective problem so that the points r, r' are present iff a given instance of the Partition problem has a solution. Then it is NP-hard to determine whether the smallest ϵ-Pareto set has 1 point or needs 3 points. Finally, if we wish, we can expand the solution set replicating this configuration a number of times k on the plane far apart from each other, and show that, for any k, it is NP-hard to determine whether the smallest ϵ-Pareto set has k points or needs $3k$ points. (Details omitted.)

Note however, that the lower bound above is brittle, for 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 below, 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,PY2]. 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 vaues, the plane is partitioned arithmetically into squares of size $\log(1+\epsilon')$ ($\approx \epsilon/2$ for small ϵ). Proceed to call $\text{GAP}_{\epsilon'}$ 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. To compute a 3-approximation to P^*_ϵ we will proceed in two phases. We will first compute an ϵ'-Pareto set for a particular $\epsilon' < \epsilon$ and then delete points from this set until we are left with a small ϵ-Pareto set.

We begin again by partitioning the space into rectangles, this time with a coordinate ratio of $(1+\epsilon') = \sqrt[4]{1+\epsilon}$. Look at the set of corner points, as a set of

points on the x-y plane. The algorithm consists of two repeating steps. ZAG(b) returns a corner point p with minimum y value, such that $\mathrm{GAP}_{\epsilon'}(\mathrm{p}) = \mathrm{YES}$ and $x(p) \leq x(b)$, where $x(p)$ is simply the x-coordinate of p. The second step, ZIG(b) returns a corner point p with minimum x value, such that $\mathrm{GAP}_{\epsilon'}(\mathrm{p}) = \mathrm{YES}$ and $y(p)/(1+\epsilon') \leq y(b)$.

In the first phase, the algorithm will discover the Pareto set by scanning the range of possible values in order of decreasing x (increasing y) coordinate. Let p be the point that has the maximum possible value in both objectives. Clearly $\mathrm{GAP}_{\epsilon'}(\mathrm{p}) = \mathrm{YES}$. We then find the point $\mathrm{q_1} = \mathrm{ZIG(ZAG(p))}$. The point q_i is found as $\mathrm{ZIG(ZAG(q_{i-1}/(1+\epsilon')))}$. The first phase terminates when the ZAG step returns NO. We then return the set $Q = \{q_1, q_2, \ldots, q_m\}$.

Lemma 1. *The* ZIGZAG *algorithm above returns a set Q that $(1+\epsilon')^2$-covers the set P of all solution points.*

Proof. The algorithm maintains several invariants. First, the x coordinates of points $q_1, \ldots, q_m$ form a strictly decreasing sequence, while the y coordinates form a strictly increasing sequence. With that in mind, the claim below implies by induction that Q is a $(1+\epsilon')^2$-cover.

Claim. The point q_i $(1+\epsilon')^2$-covers all of the points in P with their x-coordinate between $x(q_{i-1})/(1+\epsilon')$ and $x(q_i)/(1+\epsilon')$.

Proof. Suppose there exists some point p in the solution space, and q_{i-1}, $q_i \in Q$ such that $x(q_i)/(1+\epsilon') \leq x(p) \leq x(q_{i-1})/(1+\epsilon')$, and q_i does not $(1+\epsilon')^2$-cover p. In that case $y(p) < y(q_i)/(1+\epsilon')^2$. But then the y value of $\mathrm{ZAG(q_{i-1})}(1+\epsilon')^2$ must be less than $y(p)$ by definition of $\mathrm{GAP}_{\epsilon'}$, and further, $\mathrm{y(ZIG(ZAG(q_{i-1})))} \leq \mathrm{y(ZIG(q_{i-1}))}$. Therefore $y(q_i)/(1+\epsilon')^2 \leq y(p)$, a contradiction.

Thus Q forms an $(1+\epsilon')^2$-cover of P.

Lemma 2. *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. We charge those points in Q that are $(1+\epsilon)$-covered by p^* to the point p^*. We prove that every point p^* is charged with at most 11 points of Q. We omit the details of the proof.

In the second phase we reduce the set Q to make it a small ϵ-Pareto set. Note that if R is any $(1+\epsilon')^2$-cover of the points in Q, then for any point in the original solution space, there is some point in R that $(1+\epsilon')^2(1+\epsilon')^2 = (1+\epsilon)$-covers it, in other words R is an ϵ-Pareto set. Further, since there is a total order on points in Q, we can compute the smallest $(1+\epsilon')^2$-cover by a simple greedy algorithm. At the end of phase 1 the points in Q are already sorted by their x value. Given q_1 we now find a point q_i with the smallest x coordinate such that $q_i/(1+\epsilon')^2 \leq q_1$ in all dimensions. Add q_i to R, remove points $q_1, \ldots, q_{i-1}$ and repeat until all of the points are covered.

Lemma 3. *The size of the smallest $(1+\epsilon')^2$-cover of Q is no more than $3 \cdot |P^*_\epsilon|$.*

Proof. Consider any point p^* in P^*_ϵ again, and let q be a point of Q that $(1+\epsilon')^2$ covers p^*. If we add q to our cover, the points in Q that are $(1+\epsilon)$ covered by p^* but are not $(1+\epsilon)$ covered by q are split into two disjoint groups: those above q and those below q, each of which can always be $(1+\epsilon')^2$-covered by a single point.

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 can be implemented as binary searches on $O(m/\epsilon)$ points. Therefore, the runtime of the first part is bounded by $O(k \log(m/\epsilon))$ $\mathrm{GAP}_{\epsilon'}$ calls. The greedy algorithm as described is linear, and its time is subsumed by that of the first phase. Therefore the overall runtime is $O(k \log(m/\epsilon))$ $\mathrm{GAP}_{\epsilon'}$ calls.

Theorem 2. *The* ZIGZAG *algorithm with the cleanup step as described above computes a 3-approximation to the smallest ϵ-Pareto set in time $O(k \log(m/\epsilon))$ $\mathrm{GAP}_{\epsilon'}$ calls.*

Suppose that we are allowed to return an ϵ'-Pareto set for a value $\epsilon' > \epsilon$ and let δ be such that $(1+\epsilon') = (1+\epsilon)(1+\delta)^4$. We can proceed in a way similar to above: first divide the grid with the coordinate ratio of $(1+\delta)$, and use ZIGZAG with GAP_δ. In the cleanup phase, we look for the smallest $(1+\delta)^2(1+\epsilon)$ cover of the points from the first phase.

Theorem 3. *For $(1+\epsilon') = (1+\epsilon)(1+\delta)^4$ we can find an ϵ'-Pareto set R such that $|R| < |P^*_\epsilon|$ in time $O(k \log(m/\delta))$ GAP_δ calls.*

Proof. Since a $(1+\epsilon)(1+\delta)^2$ cover of a $(1+\delta)^2$-Pareto set will form a $(1+\epsilon)(1+\delta)^4 = (1+\epsilon')$-Pareto set, the only step we need to prove is that the size of the smallest such cover is no bigger than the size of the smallest ϵ-Pareto set. We proceed the same as above, by picking a point $p^* \in P^*_\epsilon$. Then there's a point $q \in Q$ such that $q(1+\delta)^2$ covers p^*. That implies, that all of the points that are $(1+\epsilon)$-covered by p^* are $(1+\delta)^2(1+\epsilon)$-covered by q. Therefore the smallest $(1+\delta)^2(1+\epsilon)$ cover of all of the points in Q is of size no bigger than $|P^*_\epsilon|$

3.3 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 well as possible the Pareto curve. That is, we wish to find a set S of k solutions that minimizes the value of the ratio r such that S r-covers the whole set P of solutions. For $k = 1$, this solution is the "knee" of the Pareto set.

As shown at the beginnining of the section, finding the knee (and more generally the best k points) is NP-hard even in simple cases. However, we can approximate the optimal ratio r, within any degree of accuracy $1+\delta$. For the knee case it is easy to do this using $O(\log^2(m/\delta))$ GAP_δ calls by a simple binary search(we omit the details). For general k, we use our above algorithm for the $\epsilon' > \epsilon$ case to show the following (proof omitted).

Theorem 4. *We can approximate the smallest ratio $1+\epsilon$ for which the ϵ-Pareto set has at most k points to a factor of $1+\delta$ in time $O(k \log^2(m/\delta))$ GAP_δ calls.*

3.4 Applications

The results here 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 bounds can be proved, 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 b on the cost of the path, minimize the length of the path subject to the bound on the cost. There exists an FPTAS for RSP with running time $O(en/\epsilon)$, where n is the number of nodes and e the number of edges [ESZ].

The RSP routine can be used directly to implement both the ZIG and the ZAG steps in the algorithm. Hence we can compute the smallest ϵ-Pareto set for bicriteria shortest paths in time $O(\frac{enk}{\epsilon})$ where k is the size of the smallest ϵ-Pareto set.

4 Three Objectives

4.1 Lower Bound

Theorem 5. *Any 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. Just like in the case of 2 objectives we will exploit the fact that GAP_δ(b) is not uniquely defined for some points b. Again 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. Consider a point $p = (p_x, p_y, p_z)$, and let $q_i = \left(p_x(1+\epsilon)^i, p_y(1+\epsilon)^{k-i}, \frac{p_z}{1+\epsilon}\right)$ for $i = 0 \ldots k$. Let $P = \{p, q_0, \ldots, q_k\}$. Clearly, $\{p\}$ is an ϵ-Pareto set for P. Let $r_i = \left(p_x(1+\epsilon)^i, p_y(1+\epsilon)^{k-i}, \frac{p_z-1}{(1+\epsilon)}\right)$. Notice that p does not $(1+\epsilon)$-cover any of the r_is and none of the q_is cover p. So if we let $Q = \{p, q_0, \ldots q_k, r_0, \ldots, r_k\}$, the smallest ϵ-Pareto set for Q will have $\Theta(k)$ points. But again GAP_δ cannot

distinguish between the two cases, since for all b where GAP_δ(b) can return r_i it can either return q_i or return NO. Therefore, we cannot conclude if the size of the optimal solutoin is one point, or $\Theta(k)$ points for arbitrary k. Again, we can turn this 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 $\delta > 0$ such that $(1+\epsilon') > (1+\epsilon)^2(1+\delta)^4$; it is convenient to pick a δ such that $(1+\epsilon)$ is a power of $(1+\delta)$. As before, we will be working with a geometric grid of the space of objective values (equivalently, an arithmetic grid in the log-log scale). 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).

Assume for concreteness again that all objectives are to be minimized; the algorithm is similar in the other cases. 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 $x(r) \leq x(q)(1+\epsilon)^2(1+\delta)^2$, $y(r) \leq y(q)(1+\epsilon)(1+\delta)$ and $z(r) \leq z(q)(1+\epsilon)(1+\delta)$; 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 minimal (undominated) points in the corresponding sets.
Update $Q \leftarrow Q \cup \{s(p), t(p)\}$.

To simplify notation, we blur below the distinction between the selected set Q of corner points and the corresponding set $\mathrm{GAP}_\delta(Q) = \{\mathrm{GAP}_\delta(\mathrm{q}) \mid \mathrm{q} \in \mathrm{Q}\}$ of feasible solution points derived from it; note that every point $q \in Q$ is dominated by $\mathrm{GAP}_\delta(q)$.

Theorem 6. *The set Q computed by the above algorithm forms a ϵ'-Pareto set, and $|Q| \leq 4|P^*_\epsilon|$.*

Proof. Let P^*_ϵ be the optimal ϵ-Pareto set. We will charge each point of Q to a point of P^*_ϵ so that every point $p^* \in P^*_\epsilon$ is charged with at most 4 points of Q. The details of the proof are quite involved and are omitted from this extended abstract.

Observe that given p, it is easy to find $s(p)$ 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 $p \leftarrow \min_x \min_z C/Q$.

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 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.

Lemma 4. *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 of points undominated in x and y. These points are sorted in increasing order by their x coordinate, as $q_1, \ldots, q_l$. The x-y projection of the corner points that are eligible (i.e. not covered by Q) is the region below a rectilinear curve that passes through a translation of the points q_j. 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 posible 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 $\text{GAP}_\delta\left(\frac{x(q_{j+1})}{(1+\epsilon)^2(1+\delta)^3}, \frac{y(q_j)}{(1+\epsilon)^2(1+\delta)^2}, z\right)$ 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. When we add a new point to Q, we may eliminate some of the elements of Q (if they become dominated), 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 immediately jump to the next z value where we will be guaranteed to find a point in C/Q. Once we find the z value, we limit our search to an appropriate interval to seek the next point from C/Q. We can find the point using an algorithm similar to the ZIGZAG algorithm presented for the 2-d case.

Theorem 7. *The algorithm to compute the ϵ'-Pareto set Q can be implemented to run in time $O(|Q| \log(m/\delta))$ GAP_δ 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. Let $(1+\epsilon') = (1+\epsilon)(1+\delta)^2$.

Theorem 8. *For any $\epsilon' > \epsilon$ we can compute an ϵ'-Pareto set Q such that $|Q| \leq |(\log(d \log(m/\delta))|P^*_\epsilon|$ 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. Break up the solution space using a geometric grid of size $\sqrt{1+\delta}$ and call $\text{GAP}_{\sqrt{1+\delta}}$ 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 $\log n$ approximate 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 (unless P=NP) the above algorithm is the best possible in very high dimensions, even if the points are given explicitly as input.

Theorem 9. *The task of computing the smallest ϵ-Pareto set on d objectives is NP-hard to approximate to within $\log d$ even if all the solution points are given explicitly.*

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 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. 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.

Observe that the reduction above 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 10. *Let* $(1+\epsilon') < (1+\epsilon)^{\log^* d/3}$. *Even if all the solution points are given explicitly in the input, it is NP-hard to compute an* ϵ'*-Pareto set whose size is within a* $\log^* d$ *factor of the smallest* ϵ*-Pareto set.*

Proof. We will use a reduction from assymetric k-center along with a recent result by Chuzhoy et al. [CG+] to finish the proof. In the assymetric k-center problem we are given a set of points V with distances, $dist(u,v)$ that must satisfy the triangle inequality, but may be assymetric: i.e. $dist(u,v) \neq dist(v,u)$. We are asked to find a subset $U \subseteq V$, $|U| = k$, that minimizes $dist^* = \max_{v \in V} \min_{u \in U} dist(u,v)$.

Let us chose a value $dist'$ which will specify later, and encode the asymmetric k-center problem as follows. For each $v_i \in V$ create a point p_i such that, the ith coordinate of p_i is 1, and the jth coordinate is $(1+\epsilon)^{-\lceil dist_{ij}/dist' \rceil}$. Notice that 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 do a binary search on $dist'$ to find 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 $\log^* n$ even when using $(\frac{1}{3}\log^* n)k$ centers.

References

[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. To appear in *Proc. STOC 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. September 2002.

[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.

[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.