Approximation of Multiobjective Optimization Problems

Ilias Diakonikolas

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2011

ABSTRACT

Approximation of Multiobjective Optimization Problems

Ilias Diakonikolas

We study optimization problems with multiple objectives. Such problems are pervasive across many diverse disciplines – in economics, engineering, healthcare, biology, to name but a few – and heuristic approaches to solve them have already been deployed in several areas, in both academia and industry. Hence, there is a real need for a rigorous investigation of the relevant questions.

In such problems we are interested not in a single optimal solution, but in the tradeoff between the different objectives. This is captured by the *tradeoff* or *Pareto* curve, the set of all feasible solutions whose vector of the various objectives is not dominated by any other solution. Typically, we have a small number of objectives and we wish to plot the tradeoff curve to get a sense of the design space. Unfortunately, typically the tradeoff curve has exponential size for discrete optimization problems even for two objectives (and is typically infinite for continuous problems). Hence, a natural goal in this setting is, given an instance of a multiobjective problem, to efficiently obtain a "good" approximation to the entire solution space with "few" solutions. This has been the underlying goal in much of the research in the multiobjective area, with many heuristics proposed for this purpose, typically however without any performance guarantees or complexity analysis.

We develop efficient algorithms for the succinct approximation of the Pareto set for a large class of multiobjective problems. First, we investigate the problem of computing a minimum set of solutions that approximates within a specified accuracy the Pareto curve of a multiobjective optimization problem. We provide approximation algorithms with tight performance guarantees for bi-objective problems and make progress for the more challenging case of three and more objectives. Subsequently, we propose and study the notion of the approximate convex Pareto set; a novel notion of approximation to the Pareto set, as the appropriate one for the convex setting. We characterize when such an approximation can be efficiently constructed and investigate the problem of computing minimum size approximate convex Pareto sets, both for discrete and convex problems. Next, we turn to the problem of approximating the Pareto set as efficiently as possible. To this end, we analyze the Chord algorithm, a popular, simple method for the succinct approximation of curves, which is widely used, under different names, in a variety of areas, such as, multiobjective and parametric optimization, computational geometry, and graphics.

Table of Contents

List of Figures

List of Tables

To my mother, Elli Diakonikola

who has always made decisions

having her children as her only objective.

Chapter 1

Introduction

Suppose we want to drive from New York to Boston. We care about the travel time, the total distance, the simplicity of the route (e.g. number of turns), etc. Which route should we choose?

Decision making involves the evaluation of different alternative solutions from a design space, and the selection of a solution that is "best" according to the criteria of interest. In most situations there are usually more than one criteria that matter. For example, in network design we are concerned with its cost, capacity, reliability; in investments we care about return and risk; in radiation therapy we care about the effects on the tumor on the one hand, and healthy organs on the other; and so forth. These are problems of *multiobjective optimization*, a research area in the interface of operations research and microeconomics that has been under intense study since the 1950s, with many papers, conferences and books, see e.g. [Cli, Ehr, EG, FGE, Mit].

This type of *multicriteria* or *multiobjective* problems arise across many diverse disciplines, in engineering, in economics and business, healthcare, manufacturing, biology, and others. There is an extensive literature on this subject in operations research and other areas, and heuristic approaches to solve such problems have already been deployed, in both academia and industry.

Due to the ubiquity of such problems, the following question immediately arises:

- *What does it mean to "solve" a multiobjective problem? What is the right "solution concept"?*

The basic ingredients of a single-criterion optimization problem are its set of instances, solutions and objective function (to be minimized or maximized). The objective function is given by an efficient algorithm f, which given an instance x and a feasible solution s, computes its value $f(x, s)$.

We seek, given x, to find $\mathrm{argmin} f(x, s)^1$.

In a *multiobjective optimization problem* we have instead $d \geq 2$ objective functions (all defined on the same set of feasible solutions). In such problems, there is typically no solution that is uniformly best in all the objectives; hence, it is not immediately obvious what a computational solution should entail in this setting.

One approach to multicriteria problems is to impose a global utility function g that combines in some monotone way the different criteria and thereby treat the problem as a single objective optimization problem. This approach is fine if (i) one can determine ahead of time a global function g that faithfully reflects the preferences of the decision maker, and (ii) we can optimize g. However, it is typically hard to tell in advance which utility function to pick without first getting a sense of the design space and the *trade-offs* involved, and moreover the preferences of the decision maker may not be formalized or even quantifiable. Furthermore, different users may have different preferences with unknown utility functions.

This is why, in multiobjective optimization, usually one cares not only in a single optimal solution, but in a more complicated object, the set of Pareto-optimal solutions or *Pareto set*. These are the solutions that are not dominated by other solutions, that is to say, a solution is Pareto-optimal if there does not exist another solution that is simultaneously better in all criteria. The Pareto set represents the range of reasonable "optimal" choices in the design space and captures the intuitive notion of a "trade-off."; it is precisely the set of optimal solutions for all possible global utility functions that depend monotonically on the different objectives.

The common approach in the more computationally–oriented multiobjective optimization is thus to compute the Pareto set, that is then presented to the decision maker. A decision maker, presented with the Pareto set, can select a solution that corresponds best to her preferences; as previously mentioned, different users generally may have different preferences and select different solutions.

Computing the Pareto set seems to be conceptually the right idea, but it is often computationally intractable since, even for the simplest problems, its size can be exponential (even for the case of two objectives) and is infinite for continuous problems. Furthermore, for even the simplest problems (shortest path, spanning tree, matching) and even for two objectives, determining whether a

[1]We assume for the sake of the exposition that the objective function is to be minimized.

point belongs to the Pareto set is NP-hard. We thus need a *solution concept* that circumvents these problems.

A good way to define a meaningful computational problem related to multiobjective optimization involves *approximation*. Ideally, we want to compute *efficiently* and present to the decision makers a *small* set of solutions (as small as possible) that represents as well as possible the whole range of choices, i.e. that provides a good approximation to the Pareto set. Indeed this is the underlying goal in much of the research in the multiobjective area, with many heuristics proposed, usually however without any performance guarantees or complexity analysis as we do in theoretical computer science. In this thesis, we define and systematically investigate the relevant computational questions. We provide efficient algorithms to compute *succinct* approximations to the Pareto set for wide classes of multiobjective problems.

1.1 Approximation of the Pareto Set

The most natural notion of approximation to the Pareto set is based on the concept of an ϵ-Pareto set [PY1]: An ϵ-Pareto set, $\epsilon > 0$, is a set of solutions that approximately dominate all other solutions. That is, for every other solution, the set contains a solution that is at least as good approximately (within a factor $1 + \epsilon$) in all objectives. Such an approximation was studied before for certain problems, e.g. multiobjective shortest paths, for which Hansen [Han] and Warburton [Wa] showed how to construct an ϵ-Pareto set in polynomial time (for fixed number of objectives). Note that typically in most real-life multiobjective problems the number of objectives is small. In fact, the great majority of the multiobjective literature concerns the case of two objectives.

This notion of approximation is rather attractive for the following reason: Under very general conditions [PY1], there exists a polynomially succinct (in the input size and $1/\epsilon$) ϵ-approximate Pareto set. A polynomially small ϵ-Pareto set always exists, but it is hard to construct in general. In [PY1] a necessary and sufficient condition for its efficient computability is given, which implies that for several important combinatorial problems – including multiobjective versions of shortest path, spanning tree and matching – there is a PTAS for constructing an ϵ-Pareto set. (It should be emphasized that the Pareto set is not given to us explicitly. The whole point is to approximate it without constructing the entire set.)

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 redundant points). Construction of a polynomial-size approximate Pareto set is useful, but not good enough in itself: For example, if we plan a trip, we want to examine just a few possible routes, not a polynomial number in the size of the map. In fact, even in cases where the Pareto set has polynomial size, we may still want a very small number of solutions that provide a good approximation. Having a *small* approximate Pareto set gives a succinct outline of the trade-offs involved and is important for many reasons. For example, the selected representative solutions are investigated more thoroughly by humans, hence there should be a small limit on the number of solutions to be examined. For example, in radiotherapy planning, different plans will be assessed by the physician to select one that provides the best balance [CHSB]. Obviously, there is a small limit on the number of plans that can be examined.

The above discussion motivates the following questions.

Consider a multiobjective problem with a fixed number of objectives, for example shortest path with cost and time objectives. For a given instance, and error tolerance ϵ, we would like to compute a *minimum cardinality* set of solutions that form an ϵ-Pareto set.

- *Can we do it in polynomial time? If not, how well can we approximate the smallest (i.e. minimum cardinality) ϵ-Pareto set?*

Dually, given an integer k, we would like to compute a set of k solutions that provide the best approximation to the Pareto set, i.e. form an ϵ-Pareto set for the minimum possible error ϵ.

- *Can we do it in polynomial time? If not, how well can we approximate the optimal error?*

We study the relevant computational problems in Chapter 3.

1.2 Objective Space: Convex or Discrete?

In several problems, the set of solution values in the objective space (and/or decision space) is convex, i.e. if v, v' are the vectors of objective values for two solutions, then every convex combination of v, v' is also the vector of values for some solution.

This is the case for example for *Multiobjective Linear Programming* (MOLP): minimize (or maximize) a collection of linear functions subject to a set of linear constraints. In this case, the Pareto curve is a polygonal line for two objectives (a polyhedral surface for more). Although there is an infinite set of Pareto points, a finite set of points, namely the vertices of the curve, suffice to represent the curve; every other Pareto point is a convex combination of vertices. Indeed, MOLP has been studied thoroughly and several algorithms (e.g. Multiobjective Simplex) have been developed to generate all the vertices of the Pareto set, see e.g. [Ehr, Ze]. Another example is the design of optimal policies for Markov decision processes with multiple objectives corresponding to the probabilities of satisfaction of a set of properties of the execution or to given discounted rewards [EKVY, CMH]; in effect, these problems can be reduced in both cases to MOLP problems.

Convexity can arise in various other ways, in both continuous and discrete problems, even if it is not present originally. In several applications, solutions that are dominated by convex combinations of other solutions may be regarded as inferior and thus not desirable. The multicriteria literature uses sometimes the term "efficient" for a Pareto solution and "supported efficient" for a solution that is not strictly dominated by the convex combination of other solutions. Thus, sometimes only supported efficient solutions are sought. These are the solutions whose values lie on the Pareto set of the convex hull of all solution points. This "boundary" set can be represented by its extreme points which we call the *convex Pareto set* of the instance. Of course, if the objective space is convex, then the boundary set coincides with the Pareto set.

The most common approach to the generation of Pareto points (called weighted-sum method) is to give weights $w_i \geq 0$ to the different objective functions f_i (assume for simplicity that they are all minimization objectives) and then optimize the linear combining function $\sum_i w_i f_i$; this approach assumes availability of a subroutine $Comb$ that optimizes such linear combinations of the objectives. This is done for a sequence of weight tuples and then the "dots are connected" to form a representation of the boundary (lower envelope) of the objective space. For any set of nonnegative weights, the optimal solution is clearly in the Pareto set, actually in the convex Pareto set. In fact, the convex Pareto set is precisely the set of optimal solutions for *all* possible such weighted linear combinations of the objectives. Of course, we cannot try all possible weights; we must select carefully a finite set of weights, so that the resulting set of solutions provides a good representation of the space. Thus, the representation that is obtained is actually an approximate representation of the Pareto set of the convex hull.

In some applications we may actually want to depict this convex trade-off curve; for example [VV] studies network routing with multiple QoS criteria (these are essentially multiobjective shortest path problems) and the associated trade-off curves, both the exact Pareto curve and the convex curve. Another case of convexification is when the decision is randomized, not deterministic (pure), i.e. the decision is a probability distribution over the set of solutions, and the figures of merit are the expected values of the objective function. Randomization has the effect of taking the convex hull of the set of solution points. Note that this holds for all types of objectives (both linear and nonlinear).

In the case of combinatorial optimization problems with linear objectives the convex Pareto set is closely related to *parametric optimization* [Gus, Meg1, Meg2]. For example, consider the parametric $s-t$ shortest path problem where each edge e has cost $c_e + \lambda d_e$ that depends on the parameter λ. The length of the shortest path is a piecewise linear concave function of λ whose pieces correspond to the vertices of the convex Pareto curve for the bi-objective shortest path problem with cost vectors c, d on the edges.

The size (number of vertices) of the convex Pareto set may be *much* smaller than the size of the Pareto set. For example, the Pareto set for the bi-objective shortest path problem has exponential size in the worst-case, and the same is true for essentially all common combinatorial problems (bi-objective knapsack, spanning trees, etc.) In contrast, the convex Pareto set for bi-objective shortest paths has quasi-polynomial size, $n^{\Theta(\log n)}$ (upper and lower bound) [Gus, Car]; for bi-objective spanning trees it has polynomial size [Chan, Gus]. On the other hand, for bi-objective minimum cost flows (and for LP) the convex Pareto set, which coincides with the vertices of the Pareto set, has exponential size in the worst-case [Ru]. Hence, in some cases, even computing the convex Pareto set can be an intractable problem.

Regarding the approximation of the convex Pareto set, the notion of ϵ-Pareto set is *not* the right one, as it can be very wasteful. For example consider the case of two objectives, and suppose that the Pareto set P is just a straight line segment ab. The two vertices a, b are not enough to form an ϵ-Pareto set P_ϵ: we may need to add many points along (or close to) the segment ab so that every point of P is almost dominated by a point of P_ϵ. This is obviously redundant and ignores the convexity of the setting.

A natural definition of approximation in the convex setting is the following: A set S of solution points is a *ϵ-convex Pareto set* if for every solution point p there is a convex combination of members of S that almost dominates p in all objectives (i.e., is within a factor $1+\epsilon$ of p or better). Such a set of points S can be arbitrarily smaller than the smallest ϵ-Pareto set.

Since any ϵ-Pareto set is also an ϵ-convex Pareto set, it follows that a polynomially succinct ϵ-convex Pareto set always exists (under the general conditions of [PY1]). A basic computational question is then the following:

Consider a multiobjective problem with a fixed number of objective functions.

• *Is an ϵ-convex Pareto set polynomial time computable ? Is there a general necessary and sufficient condition for efficient constructibility ?*

Analogously to the nonconvex case, for every instance there is a unique convex Pareto set, but not a unique ϵ-convex Pareto set; there are usually many different (nonredundant) such sets and they could have very different sizes. Hence, the following question arises:

Consider a multiobjective problem for which *some* ϵ-convex Pareto set (not necessarily the smallest one) is computable in polynomial time.

• *Can we efficiently construct an ϵ-convex Pareto set of (approximately) minimum cardinality ? How well can we approximate the optimum size ?*

We study these questions in Chapters 4 and 5.

We note that there is extensive literature in the optimization and management areas on the approximation of convex Pareto curves, both in terms of general methods and for specific (continuous and combinatorial) problems; see e.g. [EG, RW] for some references. The bulk of the literature concerns bi-objective problems; only a very small fraction considers 3 or more objectives. There are several methods proposed in the multicriteria literature which compute a sequence of solution points by optimizing weighted linear combinations of the objectives, and then "connect the dots". The underlying goal of the methods is basically the same, to obtain a good approximation of the convex Pareto curve with few points. Usually, however, there are no quantitative guarantees on the approximation error that is achieved by the methods and the size of the computed set. Also in many cases the methods try explicitly to get an even distribution of the points on the surface. However, for the purposes of minimizing the approximation error with a limited number of points, it is better to have an uneven distribution, with a denser representation in the areas of high curvature, and fewer points in the flat areas. The problem is of course that we are not given explicitly the Pareto surface (the whole point is to construct a good, succinct representation), but can only access it indirectly.

The problem of computing an approximate convex Pareto curve of minimum cardinality has some similarities to the problem of computing minimal approximations to polytopes and convex surfaces (which has been studied in computational geometry, e.g. [ABRSY, Cl, MitS]), but some important differences also, the main ones being (i) the reference surface (the convex Pareto curve) is given implicitly, not explicitly, (ii) we have an asymmetric dominance relation here, as opposed to proximity. Also the metrics of proximity used in geometry are usually different than the ratio measure here (which is common in the analysis of approximation algorithms). Some of the techniques developed there however are still useful in our context.

1.3 Minimizing the Computational Effort

In the previous sections, we discussed the problem of computing succinct approximations to the Pareto set. We did not put any restrictions on the running time of our algorithms (as long as it is polynomial). A different goal is to construct an ϵ-approximation to the Pareto set as efficiently as possible.

Consider for example a multiobjective problem with an efficient Comb routine (for minimizing monotone linear combinations of the objectives). (As shown in Chapter 4, the existence of such a routine characterizes the efficient constructibility of an ϵ-convex Pareto set.) Typically, the Comb routine is a nontrivial piece of software. Each call (query) takes a substantial amount of time, thus we want to make the "best use" of the calls – to achieve as good a representation of the space as possible. Ideally, one would like to achieve the smallest possible error ϵ with the fewest number of calls to the routine. That is,

- *Given $\epsilon > 0$, we want to compute an ϵ-convex Pareto set for the instance at hand using as few Comb calls as possible.*

We emphasize that this metric is quite different from the one of the previous sections. The problems described in the previous sections placed no restriction on the number of queries, as long as they are polynomially many (so that the overall algorithms run in polynomial time). An important point that should be stressed here is that, as in the case of online algorithms, we do not have complete information about the input, i.e. the (convex) Pareto curve is *not* given explicitly, but can only access it indirectly through calls to the Comb routine; in fact, the whole purpose is to obtain an approximate knowledge of the curve.

Both of the metrics are important and reflect different aspects of the use of the approximation in the decision making process. Consider a problem, say with two objectives, suppose we make several calls, say N, to the Comb routine, compute a number of solution points, connect them and present the resulting curve to the decision maker to visualize the range of possibilities, i.e., get an idea of the true convex Pareto curve. (The process may not end there, e.g., the decision maker may narrow the range of interest, followed by computation of a better approximation for the narrower range, and so forth). In this scenario, we want to achieve as small an error ϵ as possible, using as small a number N of calls as we can, ideally, as close as possible to the minimum number that is absolutely needed for the instance. In this setting, the cost of the algorithm is measured by the number of calls (i.e., the computational effort), and the performance ratio is as usual the ratio of the cost to the optimum cost.

Consider now a scenario where the decision maker does not just inspect visually the curve, but will look more closely at a set of solutions to select one; for instance a physician in the radiotherapy example will consider carefully a small number of possible treatments in detail to decide which one to follow. Since human time is much more limited than computational time (and more valuable, even small constant factors matter a lot), the primary metric in this scenario is the number n of selected solutions that is presented to the decision maker for closer investigation, while the computational time, i.e. the number N of calls, is less important and can be much larger (as long as it is feasible of course).

Motivated by this question, in Chapter 6 we analyze the performance of the *Chord* algorithm, a simple, natural greedy algorithm for the approximation of the convex Pareto set. The algorithm and variants of it have been reinvented many times and used often, under various names, for multiobjective problems [AN, CCS, CHSB, FBR, RF, Ro, YG] as well as several other types of applications involving the approximation of convex curves. We focus on the bi-objective case; although the algorithm can be defined (and has been used) for more objectives, most of the literature concerns the bi-objective case, which is already rich enough, and covers also most of the common uses of the algorithm. We prove sharp upper and lower bounds on the performance ratio of the Chord algorithm, both in the worst case and in a reasonable average case setting. We also show an information-theoretic lower bound, which says that no algorithm can have constant ratio in this setting.

The Chord algorithm is quite natural, it has been often reinvented and is commonly used for a number of other purposes. As pointed out in [Ro], an early application was by Archimedes who used it to approximate a parabola for area estimation [Ar]. In the area of *parametric optimization*, the algorithm is known as the "Eisner-Severance" method after [ES]. Recall that parametric optimization is closely related to bi-objective optimization. For example, in the parametric shortest path problem, each edge e has cost $c_e + \lambda d_e$ that depends on a parameter λ. The length of the shortest path is a piecewise linear function of λ whose pieces correspond to the vertices of the convex Pareto curve for the bi-objective shortest path problem with cost vectors c, d on the edges. A call to the Comb routine for the bi-objective problem corresponds to solving the parametric problem for a particular value of the parameter.

The Chord algorithm is also useful for the approximation of convex functions, and for the approximation and smoothening of convex and even non-convex curves. In the context of smoothening and compressing curves and polygonal lines for graphics and related applications, the Chord algorithm is known as the Ramer-Douglas-Peucker algorithm, after [Ra, DP] who independently proposed it.

Previous work has analyzed the Chord algorithm (and variants) for achieving an ϵ-approximation of a function or curve with respect to vertical and Hausdorff distance, and proved bounds on the cost of the algorithm as a function of ϵ: for all convex curves of length L, (under some technical conditions on the derivatives) the algorithm uses at most $O(\sqrt{L/\epsilon})$ calls to construct an ϵ-approximation , and there are curves (for example, a portion of a circle) that require $\Omega(\sqrt{L/\epsilon})$ calls [Ro, YG].

Note however that these results do not tell us what the performance ratio is, because for many instances, the optimal cost OPT_ϵ may be much smaller than $\sqrt{L/\epsilon}$, perhaps even a constant. For example, if P is a convex polygonal line with few vertices, then the Chord algorithm will perform very well for $\epsilon = 0$; in fact, as shown by [ES] in the context of parametric optimization, if there are N breakpoints, then the algorithm will compute the exact curve after $2N - 1$ calls. (The problem is of course that in most bi-objective and parametric problems, the number N of vertices is huge, or even infinite for continuous problems, and thus we have to approximate.)

1.4 Organization of the Dissertation

In Chapter 2, we provide the required background from Multiobjective Optimization. We formally define the (approximate) solution concepts studied throughout this thesis and review the relevant bibliography.

In Chapter 3, we investigate the problem of computing a minimum set of solutions that approximates within a specified accuracy ϵ the Pareto curve of a multiobjective optimization problem. We show that for a broad class of bi-objective problems (containing many important widely studied problems such as shortest paths, spanning tree, matching and many others), we can compute in polynomial time an ϵ-Pareto set that contains at most twice as many solutions as the minimum such set. Furthermore we show that the factor of 2 is tight for these problems, i.e., it is NP-hard to do better. We present upper and lower bounds for three or more objectives, as well as for the dual problem of computing a specified number k of solutions which provide a good approximation to the Pareto curve.

In Chapters 4 and 5 we study the succinct approximation of convex Pareto curves of multiobjective optimization problems. We propose the concept of ϵ-convex Pareto (ϵ-CP) set as the appropriate one for the convex setting, and observe that it can offer arbitrarily more compact representations than ϵ-Pareto sets in this context. In Chapter 4, we characterize when an ϵ-CP can be constructed in polynomial time in terms of an efficient routine Comb for optimizing (exactly or approximately) monotone linear combinations of the objectives.

In Chapter 5, we investigate the problem of computing minimum size ϵ-convex Pareto sets, both for discrete (combinatorial) and continuous (convex) problems, and present general algorithms using a Comb routine. For bi-objective problems, we show that if we have an exact Comb optimization routine, then we can compute the minimum ϵ-CP for continuous problems (this applies for example to bi-objective Linear Programming and Markov Decision Processes), and factor 2 approximation to the minimum ϵ-CP for discrete problems (this applies for example to bi-objective versions of polynomial-time solvable combinatorial problems such as Shortest Paths, Spanning Tree, etc.). If we have an approximate Comb routine, then we can compute factor 3 and 6 approximations respectively to the minimum ϵ-CP for continuous and discrete bi-objective problems. We also present upper and lower bounds for three or more objectives.

In Chapter 6, we turn to the problem of constructing an ϵ-convex Pareto set using as few calls to the Comb routine as possible. To this end, we consider the Chord algorithm, a popular, simple method for the succinct approximation of curves, which is widely used, under different names, in a variety of areas, such as, multiobjective and parametric optimization, computational geometry, and graphics. We analyze the performance of the Chord algorithm, as compared to the optimal approximation that achieves a desired accuracy with the minimum number of points. We prove sharp upper and lower bounds, both in the worst case and average case setting.

Finally, in Chapter 7 we conclude this thesis and highlight the main open problems for future work.

The results of Chapters 3-5 are joint work with Mihalis Yannakakis [DY1, DY2] and those of Chapter 6 are joint work with Constantinos Daskalakis and Mihalis Yannakakis [DDY].

Chapter 2

Background

2.1 Basic Definitions

2.1.1 Multiobjective Optimization Problems

We describe a general framework of a *multiobjective optimization problem* Π to which our results are applicable. Our framework includes all (discrete) combinatorial problems of interest (whose corresponding decision problem is in NP), but also contains many other problems (e.g. problems whose solution set is continuous and contains solutions with irrational coordinates, thus not computable exactly). Our model of computation is the standard discrete Turing machine model.

A multiobjective optimization problem Π has a set $\mathcal{I}_\Pi$ of *valid instances*, represented by strings over a fixed finite alphabet Σ, and every instance $I \in \mathcal{I}_\Pi$ has an associated set of *feasible solutions* $\mathcal{S}(I)$. The latter set is typically called *solution space* or *decision space*. As usual, it is assumed that, given a string over Σ, one can determine in polynomial time if the string is a valid instance of the problem. Unlike usual discrete problems, for a given instance I, the set $\mathcal{S}(I)$ may well be infinite; solutions here are in general real-valued vectors of finite dimension, polynomial in the size $|I|$ of the instance. If a solution has only rational coefficients, then it can be represented by a finite string over Σ; in general of course it cannot.

There are d objective functions, $f_1, f_2, \dots, f_d$, each of which maps every instance $I \in \mathcal{I}_\Pi$ and solution $s \in \mathcal{S}(I)$ to a real number $f_j(I, s)$. The problem specifies for each objective whether it is to be maximized or minimized. We denote by $\mathbf{f} = [f_1, f_2, \dots, f_d]$ the d-vector of objective values. For $s \in \mathcal{S}(I)$, we denote $\mathbf{f}(s) \equiv [f_1(s), f_2(s), \ldots, f_d(s)]$. As usual for a computational context, it is assumed that the objective functions are polynomially computable. That is, there is an algorithm which, for every instance I and solution $s \in \mathcal{S}(I)$ with finite representation, computes $f_j(I,s)$ and runs in time polynomial in the size $|I|$ of the instance I and $|s|$ of the solution s. We restrict the objective values to be positive, as is commonly the case in the context of approximation. We further *assume* that there exists a polynomial $p_\Pi(\cdot)$ such that for every instance $I \in \mathcal{I}_\Pi$ and any solution $s \in \mathcal{S}(I)$ it holds $2^{-m} \leq f_j(I,s) \leq 2^m$, where $m = p_\Pi(|I|)$. (We remark that the latter assumption is satisfied by all *discrete combinatorial problems* whose decision version is in NP. For such problems, feasible solutions are *polynomially bounded* in the size of the instance. Since each f_j is polynomially computable, each value $f_j(I,s)$ is a rational number whose numerator and denominator have at most m bits, where $m \leq p(|I|)$, for some polynomial p.)

For a given instance I, we say that a point $p \in \mathbb{R}^d_+$ is a *solution point* if there exists a solution $s \in \mathcal{S}(I)$ such that $p = \mathbf{f}(s)$. The *objective space* (for this instance) is the set $\mathfrak{X}(I) = \bigcup_{s \in \mathcal{S}(I)} \mathbf{f}(s)$; the subset of $\mathbb{R}^d_+$ containing the images (under $\mathbf{f}$) of feasible solutions.

2.1.2 Pareto Set and Approximations

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. Given a (not necessarily finite) set of points $S \subseteq \mathbb{R}^d_+$, the *Pareto set* of S, denoted by $P(S)$, is the set of undominated points in S. The *convex Pareto set* of S, denoted by $CP(S)$, is the *minimal* set of points in S (i.e. containing no redundant points) whose *convex combinations* dominate all vectors in S[1]. Thus, the convex Pareto set $CP(S)$ can be obtained from $P(S)$ by omitting the points of $P(S)$ that are dominated by convex combinations of others. Note that for any set S, the Pareto set and convex Pareto set are unique. A point $p \in S$ belongs to $P(S)$ (resp. $CP(S)$) iff it is not *strictly* dominated by any other point in S (resp. convex combination of points in S).

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 a set $S \subseteq \mathbb{R}^d_+$ and $\epsilon > 0$, an *ϵ-Pareto set* of S, denoted by $P_\epsilon(S)$, is a set of points in S that

[1]Observe that if a set dominates $P(S)$, then it also dominates S.

$(1+\epsilon)$-cover all vectors in S[2]. An *ϵ-convex Pareto set* of S, denoted by $CP_\epsilon(S)$, is a set of points in S, whose *convex combinations* $(1+\epsilon)$-cover all points in S. Clearly, an ϵ-Pareto set is also an ϵ-convex Pareto set, but the inverse does not generally hold.

At this point, we should stress a crucial distinction between exact and approximate Pareto set (resp. exact and approximate convex Pareto set). As opposed to the exact Pareto set (resp. exact convex Pareto set), approximate Pareto sets (resp. approximate convex Pareto sets) are not necessarily unique. More specifically, for a given set S and error tolerance ϵ, there may exist many ϵ-Pareto sets (resp. ϵ-convex Pareto sets) *without redundant points*, and they may have very different sizes. This fact follows as a consequence of the approximate dominance relation.

The above definitions apply to any set $S \subseteq \mathbb{R}^d_+$. We will be particularly interested in the case that the set S corresponds to the set of solution points in the objective space of a multiobjective problem. Let Π be a d-objective optimization problem in the above defined framework. Fix an instance $I \in \mathcal{I}_\Pi$. We define domination between any solutions $s, s' \in \mathcal{S}(I)$ according to the d-vectors of their objective values. The *Pareto set* for the instance I, denoted by $P(I)$, is defined as $P(I) \doteq P(\mathfrak{X}(I))$, i.e. it is the set of undominated solution points. Similarly, the *convex Pareto set* for the instance I, denoted by $CP(I)$, is defined as $CP(I) \doteq CP(\mathfrak{X}(I))$, i.e. it is the *minimal* set of solution points whose *convex combinations* dominate all vectors in $\mathfrak{X}(I)$.

Consider a d-objective optimization problem Π. A combining function $h : \mathbb{R}^d_+ \to \mathbb{R}_+$ for Π is called *monotone* if (i) h is monotonic in each coordinate; (ii) if the i-th objective function f_i of Π is to be maximized, then h is monotone increasing in its i-th coordinate and (iii) if the i-th objective function f_i of Π is to be minimized, then h is monotone decreasing in its i-th coordinate. In decision-making, we are usually interested in optimizing some *unknown* monotone combining function of the objectives $h : \mathbb{R}^d_+ \to \mathbb{R}_+$. A point is in the Pareto set iff it is optimal under *some* monotone combining function of the objectives. If the combining function is linear (or even quasi-concave), then an optimal point can always be found in the convex Pareto set. In fact, a solution point (d-vector) q is in the convex Pareto set iff it is the *unique* optimum under *some* linear monotone combining function.

Given $\epsilon > 0$, an *ϵ-Pareto set* for the instance I, $P_\epsilon(I)$, is an ϵ-Pareto set for $\mathfrak{X}(I)$, i.e. a set of solution points that $(1+\epsilon)$-cover all vectors in $\mathfrak{X}(I)$. Similarly, an *ϵ-convex Pareto set* for

[2]Observe again that if a set $(1+\epsilon)$-covers $P(S)$, then it also $(1+\epsilon)$-covers S.

the instance I, $CP_\epsilon(I)$, is an ϵ-convex Pareto set for $\mathfrak{X}(I)$, i.e. a set of solution points whose *convex combinations* $(1 + \epsilon)$-cover all vectors in $\mathfrak{X}(I)$. In this context, we are also interested in actual solutions that realize these values, but we will often blur the distinction and refer to the exact or approximate (convex) Pareto set also as a set of solutions that achieve these values - with the understanding that for each solution point in the associated set, we pick exactly one corresponding solution. For a given instance, there may exist many ϵ-(convex) Pareto sets, and they may have very different sizes. Ideally, we would like to efficiently compute one with the smallest possible size.

In a general multiobjective problem we may have both minimization and maximization objectives. Throughout this thesis, it will be convenient in the algorithms and the proofs to assume that all objectives are of a particular type, e.g. all are minimization objectives, so that we do not have to consider all possible combinations. All our results can be extended for the case of maximization or mixed objectives.

We say that an algorithm that uses a routine as a black box to access the solutions of the multiobjective problem is *generic* (or general purpose), as it is not geared to a particular problem, but applies to all of the problems for which the particular routine is available. All that such an algorithm needs to know about the input instance is bounds on the minimum and maximum possible values of the objective functions. Based on the bounds, the algorithm calls the given routine for certain values of its parameters, and uses the returned results to compute an approximate (convex) Pareto set.

2.2 Previous Work

In this section, we briefly discuss the previous work most relevant to the results in this thesis. It is shown in [PY1] that for every multiobjective optimization problem in the aforementioned framework, for every instance I and $\epsilon > 0$, there exists an ϵ-Pareto set (thus, also an ϵ-convex Pareto set) of size $O((4m/\epsilon)^{d-1})$, i.e. polynomial for fixed d. (We note that the exponential dependence on the number of objectives is inherent.) An approximate (convex) Pareto set always exists, but it may not be constructible in polynomial time. We say that the problem of computing an ϵ-(convex) Pareto set for a multiobjective problem Π has a *polynomial time approximation scheme* (PTAS) if there is an algorithm that for every instance I and $\epsilon > 0$ constructs an ϵ-(convex) Pareto set in time polynomial in the size $|I|$ of the instance I. We say that the problem of computing an ϵ-(convex) Pareto set for a multiobjective problem Π has a *fully polynomial time approximation scheme* (FPTAS) if there is an algorithm that for every instance I and $\epsilon > 0$ constructs an ϵ-(convex) Pareto set in time polynomial in $|I|$, the representation size $|\epsilon|$ of ϵ, and in $1/\epsilon$.

There is a simple necessary and sufficient condition [PY1], which relates the efficient computability of an ϵ-Pareto set for a multiobjective problem Π, with a fixed number of objectives d, to the following *GAP Problem*: given an instance I of Π, a (positive rational) d-vector b, and a rational $\delta > 0$, 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 $(1+\delta)$ factor in all of the coordinates. As shown in [PY1], there is an PTAS (resp. FPTAS) for constructing an ϵ-Pareto set iff there is a subroutine GAP that solves the GAP problem for Π in time polynomial in $|I|$ and $|b|$ (resp. in $|I|$, $|b|$, $|\delta|$ and $1/\delta$).

The one direction of this equivalence is quite simple: if we can construct an ϵ-Pareto set $P_\epsilon(I)$ in time polynomial in $|I|$, then the following simple algorithm solves the GAP problem: construct an ϵ-Pareto set $P_\epsilon(I)$ and check if the given vector b is dominated by any point of $P_\epsilon(I)$; if so, then return the corresponding solution, else return NO. Conversely, to compute a polynomial size ϵ-Pareto set, consider the following scheme: Divide the objective space geometrically into rectangles, such that the ratios of the large to the small coordinates is $(1+\epsilon') \leq \sqrt{1+\epsilon}$ in all dimensions; here ϵ' is a rational that approximates $\sqrt{1+\epsilon}-1$ from below and which has representation size $O(|\epsilon|)$. Proceed to call $\text{GAP}_{\epsilon'}$ on all of the rectangle corner points, and keep an undominated subset of all points returned. It is not hard to see that this is an ϵ-Pareto set of cardinality $O((4m/\epsilon)^{d-1})$.

As a corollary, a polynomial time algorithm for the GAP problem is clearly a sufficient condition for the polynomial constructibility of an ϵ-convex Pareto set. However, as shown in Chapter 4, it is not a necessary condition.

Vassilvitskii and Yannakakis [VY] give generic algorithms that compute small ϵ-Pareto sets and are applicable to all multiobjective problems possessing a (fully) polynomial GAP routine. They consider the following "dual" problems: Given an instance and an $\epsilon > 0$, construct an ϵ-Pareto set of as small size as possible. And dually, given a bound k, compute an ϵ-Pareto set with at most k points that has as small an ϵ value as possible. We summarize their results in Chapter 3.

2.3 Related Work

Trade-offs are present everywhere in life and science – in fact, one can argue that optimization theory studies the very special and degenerate case in which we happen to be interested in only one objective. As a consequence, the literature in multi-objective optimization is huge, and it is not possible to present a complete overview. We refer the reader to the book by Ehrgott [Ehr] and the survey collection by Ehrgott and Gandibleux [EG]. In the following, we attempt to present a brief summary of the literature that is most relevant to the results in this thesis.

Many algorithms for generating the Pareto set (or convex Pareto set) of various problems have been proposed in the optimization literature, see e.g. [NU, KW] for multiobjective knapsack problems and [Ze] for multiobjective linear programming. Since the worst-case size of the Pareto set can be exponential for most problems, these algorithms do not run in polynomial time. Moreover, a variety of heuristics have been proposed for the approximation of the Pareto set [Coh, DasDen], however none of these methods provides any guarantees on the approximation performance or running time.

One of the typical ways to handle multiple criteria in TCS has been to optimize one subject to bounds on the others, see e.g. [Rav+] for an early example in bi-criteria network design. The problem of efficiently approximating the Pareto set for the multi-objective shortest path problem was considered in the pioneering work by Hansen [Han] and Warburton [Wa] who showed that there is an FPTAS for constructing an ϵ-Pareto set. In [SOD] the authors give conditions for the existence of an FPTAS for the computation of ϵ-Pareto sets for several combinatorial problems, including network flow and scheduling problems. A recent sequence of papers [ABG, BMP, MR] study the approximability of the multi-objective traveling salesman problem (with triangle inequalities).

The results in this thesis are concerned with the design of efficient algorithms to approximate the Pareto set. These algorithms are guaranteed to run in polynomial time independent of the size of the (exact) Pareto set. A different but related line of work [BV04, KN07, RT09] establishes that the *expected size* of the Pareto set for combinatorial problems with linear objective functions is polynomial in an appropriate smoothed input-model. Consider a multiobjective problem with linear objective functions. Suppose we are given a worst-case instance of the problem and then randomly perturb the coefficients in the objective functions. The goal is to show that the expected size of the Pareto set (resp. convex Pareto set) in this model is bounded by a polynomial in the size of the input and the perturbation. (The degree of the polynomial increases with the number d of objectives.) Several heuristics for multiobjective problems run in output-polynomial time in the size of the Pareto set (resp. convex Pareto set). Hence, such a result would provide a plausible explanation for the observed practicality of such heuristics on real-world instances.

Chapter 3

Succinct Approximate Pareto Sets

3.1 Introduction

As argued in Chapter 1, construction of a polynomial-size approximate Pareto set is useful, but not good enough in itself: For example, if we plan a trip, we want to examine just a few possible routes, not a polynomial number in the size of the map. More generally, in typical multicriteria situations, the selected representative solutions are investigated more thoroughly by the decision maker (designer, physician, corporation, etc.) to assess the different choices and pick the most preferable one, based possibly on additional factors that are perhaps not formalized or not even quantifiable. We thus want to select as small a set as possible that achieves a desired approximation.

In [VY] the problem of constructing a minimum ϵ-Pareto set was raised formally and investigated in a general framework. It was shown that for all bi-objective problems with a polynomial-time GAP routine, one can construct an ϵ-Pareto set that contains at most 3 times the number of points of the smallest such set; furthermore, the factor 3 is best possible in the sense that for some problems it is NP-hard to do better. Note that although the factor 3 of [VY] is best possible in general for two objectives, one may be able to do better for specific problems.

We show in this chapter, that for an important class of bi-objective problems (containing many widely studied natural ones such as shortest paths, spanning tree, matching, knapsack, scheduling problems and others) we can obtain a 2-approximation, and furthermore the factor of 2 is tight for them, i.e., it is NP-hard to do better. Our algorithm is a general algorithm that relies on a routine for a stronger version of the Gap problem, namely a routine that solves approximately the following

Restricted problem: Given a (hard) bound b_1 for one objective, compute a solution that optimizes approximately the second objective subject to the bound. Many problems (e.g. shortest paths, etc.) have a polynomial time approximation scheme for the Restricted problem. For all such problems, a 2-approximation to the minimum ϵ-Pareto set can be computed in polynomial time. Furthermore, the number of calls to the Restricted routine (and an associated equivalent dual routine) is linear in the size OPT_ϵ of the optimal ϵ-Pareto set.

The bi-objective shortest path problem is probably the most well-studied multiobjective problem. It is the paradigmatic problem for dynamic programming (thus can express a variety of problems), and arises itself directly in many contexts. One area is network routing with various QoS criteria (see e.g. [CX2, ESZ, GR+, VV]). For example, an interesting proposal in a recent paper by Van Mieghen and Vandenberghe [VV] is to have the network operator advertise a portfolio of offered QoS solutions for their network (a trade-off curve), and then users can select the solutions that best fit their applications. Obviously, the portfolio cannot include every single possible route, and it would make sense to select carefully an "optimal" set of solutions that cover well the whole range. Other applications include the transportation of hazardous materials (to minimize risk of accident, and population exposure) [EV], and many others; we refer to the references, e.g. [EG] contains pointers to the extensive literature on shortest paths, spanning trees, knapsack, and the other problems. Our algorithm applies not only to the above standard combinatorial problems, but more generally to any bi-objective problem for which we have available a routine for the Restricted problem; the objective functions and the routine itself could be complex pieces of software without a simple mathematical expression.

Organization. The structure of this chapter is as follows: We present in Section 3.2 our general lower and upper bound results for bi-objective problems, as well as applications to specific problems. In Section 3.3 we present some results for $d = 3$ and more objectives. Here we assume only a GAP routine; i.e. these results apply to all problems with a polynomial time constructible ϵ-Pareto set. It was shown in [VY] that for $d = 3$ it is in general impossible to get any non-trivial approximation: for any parameter k, there exist instances with $O(k)$ points in which we cannot efficiently distinguish (given the GAP routine) whether the optimal ϵ-Pareto set has 1 point or (at least) k points. This means that one has to relax ϵ, i.e compute an ϵ'-Pareto set for some $\epsilon' > \epsilon$ and compare its size to the smallest ϵ-Pareto. Combining results from [VY] and [KP] we show that for *any* $\epsilon' > \epsilon$ we can construct an ϵ'-Pareto set of size $c\mathrm{OPT}_\epsilon$, i.e. within a constant factor c of the size OPT_ϵ of the optimal ϵ-Pareto set. For general d, the problem can be reduced to a Set Cover problem whose VC dimension and codimension are at most d, and we can construct an ϵ'-Pareto set of size $O(d \log \mathrm{OPT}_\epsilon) \cdot \mathrm{OPT}_\epsilon$.

We also study the *Dual* problem: For a specified number k of points, find k points that provide the best approximation to the Pareto curve, i.e. that form an ϵ-Pareto set with the minimum possible ϵ. In [VY] it was shown that for $d = 2$ objectives the problem is NP-hard, but we can approximate arbitrarily well (i.e. there is a PTAS) the minimum approximation ratio $\rho^* = 1 + \epsilon^*$. We show that for $d = 3$ this is not possible, in fact one cannot get any multiplicative approximation (unless P=NP). We exploit a relationship of the Dual problem to the asymmetric k-center problem and techniques from the latter problem to show that the Dual problem can be approximated (for $d = 3$) within a constant power, i.e. we can compute k points that cover every point on the Pareto curve within a factor $\rho' = (\rho^*)^c$ or better in all objectives, for some constant c. (It follows from our results that $c \leq 9$.) For small ρ^*, i.e. when there is a set of k points that provides a good approximation to the Pareto curve, constant factor and constant power are related, but in general of course they are not.

3.2 Two Objectives

3.2.1 Preliminaries

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, $p = \big(x(p), y(p)\big)$.

We consider the class of bi-objective problems Π for which we can approximately minimize one objective (say the y-coordinate) subject to a "hard" constraint on the other (the x-coordinate). Our basic primitive is a polynomial time (or fully polynomial time) routine for the following *Restricted problem* (for the y-objective): Given an instance $I \in \mathcal{I}_\Pi$, a (positive rational) bound C and a parameter $\delta > 0$, either return a solution point $\tilde{s}$ satisfying $x(\tilde{s}) \leq C$ and $y(\tilde{s}) \leq (1+\delta) \cdot \min\{y$ over all solutions $s \in \mathcal{S}(I)$ having $x(s) \leq C\}$ or correctly report that there does not exist any solution s such that $x(s) \leq C$. For simplicity, we will drop the instance from the notation and use Restrict$_\delta$ $(y, x \leq C)$ to denote the solution returned by the corresponding routine. If the routine does not return a solution, we will say that it returns NO. We say that a routine Restrict$_\delta$ $(y, x \leq C)$ runs in polynomial time (resp. fully polynomial time) if its running time is polynomial in $|I|$ and $|C|$[1] (resp. $|I|$, $|C|$, $|\delta|$ and $1/\delta$). The Restricted problem for the x-objective is defined analogously. We will also use the Restricted routine with strict inequality bounds; it is easy to see that they are polynomially equivalent.

Note that in general the two objectives could be nonlinear and completely unrelated. Moreover, it is possible that a bi-objective problem possesses a (fully) polynomial Restricted routine for the one objective, but not for the other. The considered class of bi-objective problems is quite broad and contains many well-studied natural ones, most notably the bi-objective shortest path and spanning tree problems (see Section 3.2.4 for a more detailed list of applications).

The structure of this section is as follows: In Section 3.2.2, we show that, even if the given bi-objective problem possesses a fully polynomial Restricted routine *for both objectives*, no generic algorithm can guarantee an approximation ratio better than 2. (This lower bound applies *a fortiori* if the Restricted routine is available for one objective only.) Furthermore, we show that for two such natural problems, namely, the bi-objective shortest path and spanning tree problems, it is NP-hard to do better than 2. In Section 3.2.3 we give a matching upper bound: we present an efficient 2-approximation algorithm that applies to all of the problems that possess a polynomial Restricted routine for one of the two objectives. In Section 3.2.4 we discuss some applications.

3.2.2 Lower Bound

To prove a lower bound for a generic procedure, we present two Pareto sets which are indistinguishable in polynomial time from each other using the Restricted routine as a black box, yet whose smallest ϵ-Pareto sets are of different sizes.

Proposition 3.2.1. *Consider the class of bi-objective problems that possess a fully polynomial Restricted routine for both objectives. Then, for any $\epsilon > 0$, there is no polynomial time generic algorithm that approximates the size of the smallest ϵ-Pareto set P^*_ϵ to a factor better than 2.*

[1]For a rational number C, we denote by $|C|$ the bit complexity of its representation as a ratio of relatively prime integers.

Proof. Fix a rational $\epsilon > 0$ and consider the following set of points: $p = (x(p), y(p))$, $q = \Big(x(p)\frac{1+2\epsilon}{1+\epsilon}, \frac{y(p)}{1+\epsilon}\Big)$, $r = \Big(\frac{x(p)}{1+\epsilon}, y(p)\frac{1+2\epsilon}{1+\epsilon}\Big)$, $p_q = \Big(x(p)+1, y(p)(1-\frac{1}{x(p)})\Big)$ and $p_r = \Big(x(p)(1-\frac{1}{y(p)}), y(p)+1\Big)$, where $x(p), y(p) > 1 + \frac{1}{\epsilon}$ (see Figure 3.1). Let $P = \{p, q, r, p_q, p_r\}$ and $P' = \{q, r, p_q, p_r\}$ be the feasible (solution) sets corresponding to two input instances. Note that p $(1+\epsilon)$-covers all the points, p_q does not $(1+\epsilon)$-cover r (due to the x coordinate) and p_r does not $(1+\epsilon)$-cover q (due to the y coordinate). It is easy to see that the smallest ϵ - Pareto set for P consists of only one point (namely point p), while the smallest ϵ - Pareto set for P' *must* include two points.

We can show that a generic algorithm *is guaranteed* to tell the difference between P and P' only if $1/\delta$ is exponential in the size of the input. The argument is very similar to the proof of Theorem 1 in [VY]. Let $x(p) = y(p) = M$, where M is an integer value exponential in the size of the input and $1/\epsilon$. By exploiting the fact that, in some cases, our primitive is *not uniquely defined*, we can argue that a polynomial time generic algorithm cannot distinguish between instances P and P'. More specifically, a generic algorithm *is guaranteed* to tell the difference between P and P' only if the tolerance δ is inverse exponential in the size of the input.

First, note that both points q and r can be efficiently computed by appropriately using the given routine; these two points suffice to $(1+\epsilon)$-cover the feasible set in both cases. Distinguishing between the two instances means determining whether p is part of the solution. Assume that we use the operation Restrict$_\delta(x, y \le C)$, where $C \in [y(p), y(p_r))$. It is easy to see that this is the only "meaningful" operation using this routine as a black box. Then, even if p is part of the solution, by definition, Restrict$_\delta$ can return p_q as long as $x(p_q) \le (1+\delta)x(p)$ or equivalently $\delta \ge \frac{1}{M}$. But since we want a polynomial time algorithm, $\frac{1}{\delta}$ has to be polynomial in $\lg M$; hence, the latter constraint must hold. By symmetry, the same property holds for the Restrict$_\delta(y, \cdot)$ routine. Therefore, using each of these routines as a black box, a polynomial time algorithm cannot determine if p is part of the solution, and it is thus forced to take at least two points, even when it is presented with the set P. Note that the above configuration can be replicated to show that it is impossible for a generic algorithm to determine whether the smallest ϵ-Pareto set has k points or $2k$ points are needed. ■

In fact, we can prove something stronger (assuming P $\neq$ NP) for the bi-objective shortest path (*BSP*) and spanning tree (*BST*) problems. In the *BSP* problem, we are given a (directed or undirected) graph, positive rational "costs" and "delays" for each edge and two specified nodes s and t. The

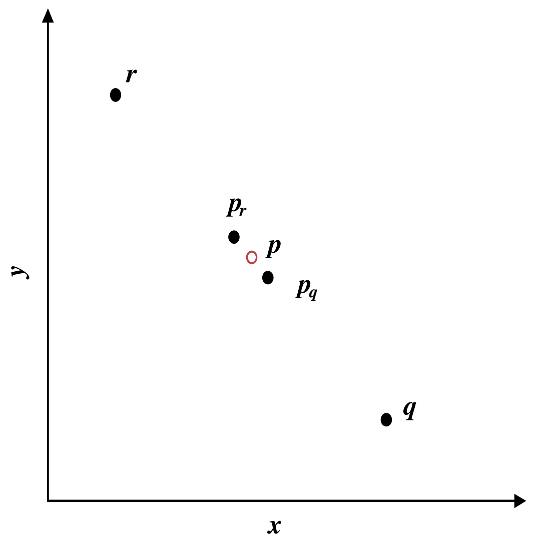

Figure 3.1: A polynomial time generic algorithm cannot determine if p is a solution of the given instance.

set of feasible solutions is the set of $s - t$ paths. The objectives (to be minimized) are linear, i.e. the "total weight" of a path equals the sum of the weights of its edges. The *BST* problem is defined analogously. These problems are well-known to possess polynomial Restricted routines for *both* objectives [LR, GR]. We show the following:

Theorem 3.2.2. *a. For the BSP problem, for any k from $k = 1$ to a polynomial, it is NP-hard to distinguish the case that the minimum size OPT_ϵ of the optimal ϵ-Pareto set is k from the case that it is $2k - 1$.*
b. The same holds for the BST problem for any fixed k.

Proof. The reductions are from the Partition problem [GJ]; we are given a set A of n positive integers $A = \{a_1, a_2, \ldots, a_n\}$, and we wish to determine whether it is possible to partition A into two subsets with equal sum.

a. For simplicity, we first prove the theorem for $k = 1$ and then generalize the construction. Given an instance of the Partition problem, we construct an instance of the *BSP* problem as follows: Let G be a graph with $n + 1$ nodes v_i, $i \in [n + 1]$ and $2n$ edges $\{e_j, e'_j\}$, $j \in [n]$. We attach the pair of (parallel) edges $\{e_i, e'_i\}$ from v_i to v_{i+1}, $i \in [n]$ and set $s \equiv v_1$ and $t \equiv v_{n+1}$. We

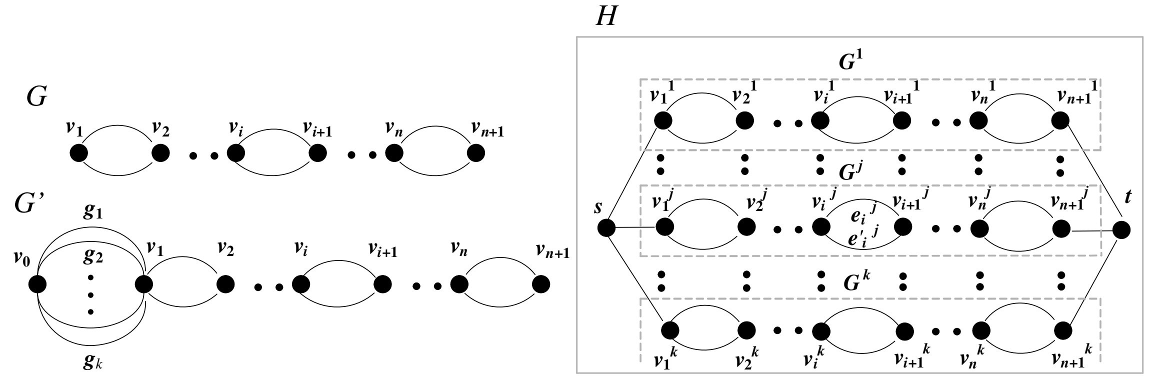

Figure 3.2: Graphs in the reduction of Theorem 3.2.2.

now specify the two cost functions $c(\cdot)$ and $d(\cdot)$ on the edges: $c(e_i) = d(e'_i) = S + 2\epsilon a_i n$ and $d(e_i) = c(e'_i) = S$, where $S = \sum_{i=1}^{n} a_i$.

Clearly, this simple transformation defines a bijection between subsets of $[n]$ and $s - t$ paths in G; the set $J \subseteq [n]$ is mapped to the $s - t$ path $\mathcal{P}_J = \bigcup_{i \in J}\{e_i\} \cup \bigcup_{i \notin J}\{e'_i\}$. Since $c(\mathcal{P}_J) = nS + 2\epsilon n(\sum_{i \in J} a_i)$ and $d(\mathcal{P}_J) = nS + 2\epsilon n(\sum_{i \notin J} a_i)$, each $s - t$ path $\mathcal{P}$, satisfies the equation $c(\mathcal{P}) + d(\mathcal{P}) = 2(1+\epsilon)nS$; hence, all feasible solutions are undominated.

Now observe that two solution points suffice to $(1+\epsilon)$-cover the feasible set; just pick the ("extreme") points $r = ((1+2\epsilon)Sn, Sn)$, $l = (Sn, (1+2\epsilon)Sn)$, corresponding to the $s - t$ paths $\mathcal{P}_{[n]} = \bigcup_{i=1}^{n}\{e_i\}$ and $\mathcal{P}_{\emptyset} = \bigcup_{i=1}^{n}\{e'_i\}$ respectively. Indeed, r $(1+\epsilon)$-covers all the points having cost (x-coordinate) at least $(1+\epsilon)Sn$ (since $(1+2\epsilon)/(1+\epsilon) < 1+\epsilon$). Equivalently, it $(1+\epsilon)$-covers all the solution points having delay (y-coordinate) up to $(1+\epsilon)Sn$ (since all the solutions lie on the line segment $x + y = 2(1+\epsilon)nS$). Moreover, the point l $(1+\epsilon)$-covers all the solution points having y-coordinate at least $(1+\epsilon)Sn$.

Since for each feasible solution $\mathcal{P}$ it holds $\min\{c(\mathcal{P}), d(\mathcal{P})\} \geq nS$ (and the "extreme" paths have cost or delay equal to nS), it follows that there exists an ϵ-Pareto set containing (exactly) one point if and only if there exists a path in G with coordinates $((1+\epsilon)Sn, (1+\epsilon)Sn)$. It is immediate to verify that such a path exists if and only if there is a solution to the original instance of the Partition problem.

Note that the above part of the proof does not rule out the possibility of an efficient *additive* approximation algorithm, i.e. an algorithm that outputs an ϵ-Pareto set of cardinality at most

$\mathrm{OPT}_\epsilon + \alpha$, where α is an absolute constant. We can rule this out as follows: Intuitively, we can think of the Pareto set of G as a "cluster". To prove the theorem for $k > 1$, the goal is to construct an instance of the problem such that the corresponding Pareto set consists of k such clusters that are "$(1+\epsilon)$-far" from each other, i.e. no point in a cluster $(1+\epsilon)$-covers any point in a different cluster.

For the *BSP* problem, we can generalize the proof to hold for any $k = \mathrm{poly}(n, \sum_{i=1}^{n} \log(a_i))$ and for all $\epsilon > 0$. This can be achieved by exploiting the combinatorial structure of the problem; we essentially replicate the graph G k times and appropriately scale the weights.

Formally, consider k (disjoint) copies of the graph G, $G^j = (V^j, E^j)$, $j \in [k]$, with $V^j = \bigcup_{i=1}^{n+1}\{v_i^j\}$ and $E^j = \bigcup_{i=1}^{n}\{e_i^j, e_i'^j\}$. Add a (source) node s, a (sink) node t; for each j add an edge from s to v_1^j and one from v_{n+1}^j to t. That is, construct the graph $H = (V_H, E_H)$ (see Figure 3.2) with

$$V_H = \{s,t\} \cup \bigcup_{j=1}^{k} V^j \text{ and } E_H = \bigcup_{j=1}^{k}\{(s, v_1^j) \cup E^j \cup (v_{n+1}^j, t)\}$$

Assign zero cost and delay to each edge incident to s or t[†] and set:

$$\begin{aligned}(1+2\epsilon)^{2(j-1)}c(e_i^j) &= d(e_i'^j)/(1+2\epsilon)^{2(j-1)} = S + 2\epsilon a_i n \\ (1+2\epsilon)^{2(j-1)}c(e_i'^j) &= d(e_i^j)/(1+2\epsilon)^{2(j-1)} = S\end{aligned}$$

From the above equations, it follows that for each $s-t$ path $\mathcal{P}^j$ "using" graph G^j, $j \in [k]$, it holds:

$$(1+2\epsilon)^{2(j-1)}c(\mathcal{P}^j) + d(\mathcal{P}^j)/(1+2\epsilon)^{2(j-1)} = 2(1+\epsilon)nS$$

This implies that all feasible solutions are undominated. In particular, it is easy to see that the Pareto set for this instance is the union of k disjoint "clusters" with endpoints $l_j = \left(\frac{Sn}{(1+2\epsilon)^{2(j-1)}}, Sn(1+2\epsilon)^{2(j-1)+1}\right)$ and $r_j = \left(\frac{Sn}{(1+2\epsilon)^{2(j-1)-1}}, Sn(1+2\epsilon)^{2(j-1)}\right)$, $j \in [k]$. The solution points in each cluster lie on the line segment $l_j r_j$. (The objective space for this instance is illustrated in Figure 3.3.)

Now notice that no solution point corresponding to an $s-t$ path using graph G^j is $(1+\epsilon)$-covered by any point corresponding to an $s-t$ path using graph G^l for $j \neq l$. Indeed, due to the structure of the Pareto set, it suffices to check that, for each $j \in [k-1]$, the points l_j and r_{j+1} do not $(1+\epsilon)$-cover each other. This holds by construction: r_{j+1} is a factor of $(1+2\epsilon)$ to the left and

[†]For simplicity, we allow zero weights on the edges, since there does not exist any $s-t$ path with zero total cost or delay. This can be easily removed by appropriate perturbation of the weights.

$(1+2\epsilon)$ above l_j. Therefore, any two clusters are "$(1+\epsilon)$-far" from each other. Thus, any ϵ-Pareto set for this instance must contain at least k points.

As in the case of $k=1$, for all $j \in [k]$, the solution points l_j and r_j $(1+\epsilon)$-cover the (solution points in the) jth cluster. Thus, $2k$ solution points suffice to $(1+\epsilon)$-cover the feasible set. Also, the jth cluster is $(1+\epsilon)$-covered by one point if and only if there exists an $s-t$ path in H with coordinates $m_j = \left(\frac{(1+\epsilon)Sn}{(1+2\epsilon)^{2(j-1)}}, (1+\epsilon)Sn(1+2\epsilon)^{2(j-1)}\right)$. Similarly, this holds if and only if the original Partition instance is a Yes instance. So, if there exists a partition of the set A, the smallest ϵ-Pareto set contains exactly k points. Otherwise, the smallest such set must contain $2k$ points.

To finish the proof, we observe that there exists an ϵ-Pareto set with (at most) $2k-1$ points if and only if there exists an ϵ-Pareto set with exactly k points. Indeed, the former statement holds if and only if *some* cluster is $(1+\epsilon)$-covered by *one* point, i.e. if and only if there exists an $s-t$ path in H with coordinates m_j for some $j \in [k]$, which in turn holds if and only if the original Partition instance is a Yes instance. The latter holds if and only if the smallest ϵ-Pareto set contains exactly k points.

b. In the *BST* problem, we are given an undirected graph with positive rational "costs" and "delays" on each edge. The set of feasible solutions is the set of spanning trees; the goal is to minimize cost and delay. For $k=1$, the NP-hardness proof for the *BST* problem is identical to the $k=1$ proof for the *BSP* problem (since any $s-t$ path in the "basic" graph G is also a spanning tree of G and vice-versa).

For $k>1$, we again use G as a basic building block to construct an instance of *BST* whose Pareto set consists of k clusters that are "$(1+\epsilon)$-far" from each other. We give a construction that works for any *fixed* k and for sufficiently small ϵ; in fact $\epsilon = O(1/k)$ suffices. Consider the graph G' obtained from G by adding one node v_0 connected to v_1 with k parallel edges g_i, $i \in [k]$. Subtract the value S from all the weights of G and set: $c(g_i) = \left\{2-(1+2\epsilon)^{2i}\right\}Sn$, $d(g_i) = (1+2\epsilon)^{2i}Sn$. Intuitively, these edges play the role of *offsets*. Clearly, as long as $(1+2\epsilon)^{2k} < 2$, all the weights are in the interval $(0, 2Sn)$.

A spanning tree $\mathcal{T}$ of G' consists of an edge g_i for some $i \in [k]$ and a path $\mathcal{P}$ from v_1 to v_{n+1}. Every edge g_i satisfies $c(g_i)+d(g_i) = 2Sn$, and every path $\mathcal{P}$ from v_1 to v_{n+1} satisfies $c(\mathcal{P})+d(\mathcal{P}) = 2\epsilon Sn$. Thus, all the solution points (spanning trees) $\mathcal{T}$ lie on the line $c(\mathcal{T})+d(\mathcal{T}) = 2(1+\epsilon)Sn$.

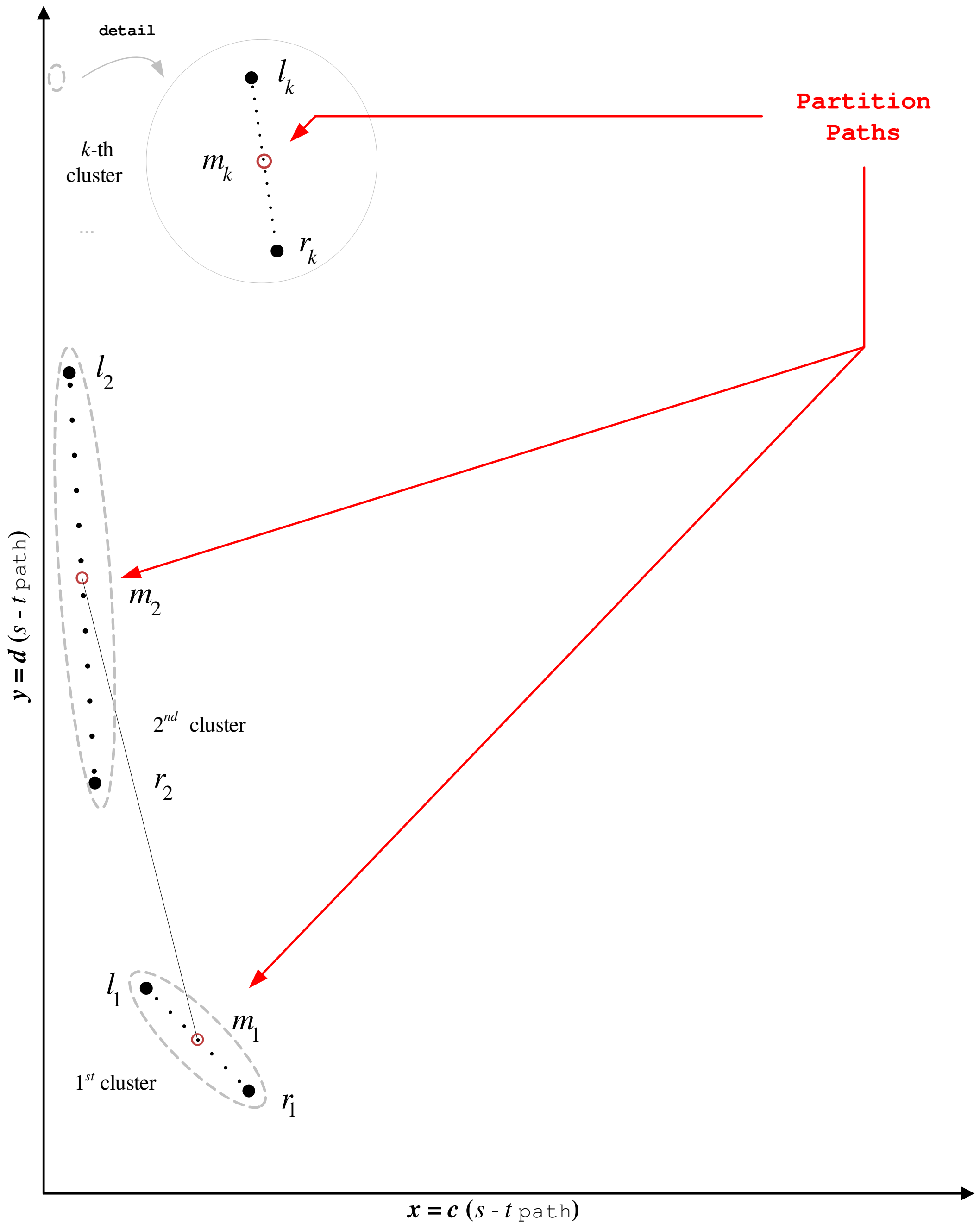

Figure 3.3: Pareto set for graph H of Theorem 3.2.2.

The Pareto set of G' consists of k clusters, where the i-th cluster, $i \in [k]$, contains all the spanning trees that consist of the edge g_i and a path from v_1 to v_{n+1}. We claim that the k clusters are "$(1+\epsilon)$-far" from each other, i.e. no point of any cluster $(1+\epsilon)$-covers any point of another cluster. The leftmost point of the i-th cluster is $l_i = (c(g_i), d(g_i) + 2\epsilon Sn)$ and the rightmost point is $r_i = (c(g_i) + 2\epsilon Sn, d(g_i))$. To show the claim, it suffices to show that $x(l_i) > (1+\epsilon)x(r_{i+1})$ and that $y(r_{i+1}) > (1+\epsilon)y(l_i)$. The first inequality is equivalent to $[2-(1+2\epsilon)^{2i}]Sn > (1+\epsilon)[2-(1+2\epsilon)^{2i+2}+2\epsilon]Sn$, which can be rewritten as $[(1+\epsilon)(1+2\epsilon)^2-1](1+2\epsilon)^{2i} > 4\epsilon+2\epsilon^2$; the inequality is clearly true since $(1+\epsilon)(1+2\epsilon)^2-1 > 4\epsilon+2\epsilon^2$. The second statement, $y(r_{i+1}) > (1+\epsilon)y(l_i)$, follows from the valid inequality $(1+2\epsilon)^2 > (1+\epsilon)(1+2\epsilon)$.

We now claim that the solution points l_i and r_i suffice to $(1+\epsilon)$-cover the i-th cluster. To show this, it suffices to see that the (solution) point with y-coordinate $y(l_i)/(1+\epsilon)$ lies to the right of the (solution) point with x-coordinate $x(r_i)/(1+\epsilon)$. Indeed, since all feasible points q satisfy $x(q) + y(q) = 2(1+\epsilon)Sn$, the previous statement amounts to verifying that $2(1+2\epsilon)Sn = x(r_i) + y(l_i) \leq 2(1+\epsilon)^2 Sn$. Hence, two points suffice for each cluster and therefore there always exists an ϵ-Pareto set with $2k$ points.

Now, suppose that there is one point q that covers all the points in the i-th cluster. Then q must have $x(q) \leq (1+\epsilon)x(l_i) = (1+\epsilon)c(g_i)$ and $y(q) \leq (1+\epsilon)y(r_i) = (1+\epsilon)d(g_i)$. Since $x(q) + y(q) = 2(1+\epsilon)Sn = (1+\epsilon)[c(g_i) + d(g_i)]$, the point q must have coordinates exactly $((1+\epsilon)c(g_i), (1+\epsilon)d(g_i))$. Such a point exists if and only if there exists a subset of A with sum $(1+2\epsilon)^{2i}S/2$. Hence, there exists an ϵ-Pareto set with k points if and only if there exist k subsets of A with sums $(1+2\epsilon)^{2i}S/2$, $i \in [k]$.

To complete the proof, we use the fact that the following *variant* of the Subset Sum problem is NP-hard: Given $A = \{a_1, a_2, \ldots, a_n\}$ with the property that (i) either there exist k subsets $A_i \subseteq A$, $i \in [k]$, such that $\sum_{x \in A_i} x = \gamma^i S/2$ or (ii) *no* such subset exists, decide which one of the two cases holds (for any fixed integer k and rational $\gamma > 1$ such that $\gamma^k < 2$). (This can be shown by a reduction from the Partition problem.) Therefore, it is NP-hard to decide if the smallest ϵ-Pareto set for the instance has k points or $2k$ points are needed and the proof is complete. ■

Remark 3.2.3. For $k = 1$ the theorem says that it is NP-hard to decide if one point suffices or we need at least 2 points for an ϵ-approximation. We proved that the theorem holds also for more general k to rule out additive and asymptotic approximations. We can easily modify the proof so that the graphs in the reductions are simple. For the *BSP* problem, this can be achieved by inserting a new ("dummy") node in the "middle" of each parallel edge (subdividing the weights arbitrarily). For the *BST* problem, this does not suffice, because all the additional nodes must be covered (by a spanning tree). Let w_i be the node inserted in the middle of $e_i = (v_i, v_{i+1})$. The problem is solved by setting $c((v_i, w_i)) = d((v_i, w_i)) = 0$, $c((w_i, v_{i+1})) = c(e_i)$ and $d((w_i, v_{i+1})) = d(e_i)$. By scaling the weights of the Partition instance we can see that the NP-hardness holds even in the case where all the edge weights are restricted to be positive integers. Similar hardness results can be shown for several other related problems (see Section 3.2.4).

3.2.3 Two Objectives Algorithm

We have a bi-objective problem with an associated Restricted routine for the y-objective that runs in polynomial (or 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. In this subsection, we present a generic algorithm that guarantees ratio 2. By the result of the previous subsection, this factor is optimal. Recall that the algorithm in [VY] works for all problems in MPTAS and is a factor 3 approximation. (The analysis of the latter algorithm is tight for the class of problems considered here.) In Section 3.2.3.1, we show that a straightforward greedy approach cannot guarantee a ratio better than 3 in our setting. We next make a crucial observation that is exploited in Section 3.2.3.2 to achieve the optimal factor.

3.2.3.1 The Greedy Approach

We remark that if the underlying problem has polynomial time *exact* Restricted routines for both objectives (i.e. Restrict$_\delta$ for $\delta = 0$), then we can efficiently compute the *optimal* ϵ-Pareto set by a simple greedy algorithm. The algorithm is similar to the one given in [KP, VY] for the (special) case where all the solution points are given explicitly in the input. We denote by $x_{\min}$, $y_{\min}$ the minimum values of the objectives in each dimension. The greedy algorithm proceeds by iteratively selecting points $q_1, \dots, q_k$ in decreasing x (increasing y) as follows: We start by computing a point q'_1 having minimum y coordinate among all feasible solutions (i.e. $y(q'_1) = y_{\min}$); q_1 is then selected to be the *leftmost* solution point satisfying $y(q_1) \leq (1 + \epsilon)y(q'_1)$. During the jth iteration ($j \geq 2$) we initially compute the point q'_j with minimum y-coordinate among all solution points s having

$x(s) < x(q_{j-1})/(1+\epsilon)$ and select as q_j the leftmost point which satisfies $y(q_j) \leq (1+\epsilon)y(q'_j)$. The algorithm terminates when the last point selected $(1+\epsilon)$-covers the leftmost solution point(s) (i.e. the point(s) q having $x(q) = x_{\min}$). It follows by an easy induction that the set $\{q_1, q_2, \dots, q_k\}$ is an ϵ-Pareto set of minimum cardinality. (This exact algorithm is applicable to bi-objective linear programming and all problems reducible to it, for example bi-objective flows, the bi-objective *global* min-cut problem [AZ] and several scheduling problems [CJK]. For these problems we can compute an ϵ-Pareto set of minimum cardinality.)

If we have approximate Restricted routines, one may try to modify the greedy algorithm in a straightforward way to take into account the fact that the routines are not exact. However, as shown below, this modified greedy algorithm is suboptimal, in particular it does not improve on the factor 3 that can be obtained from the general GAP routine. More care is required to achieve a factor 2, matching the lower bound.

Suppose that we have a (fully) polynomial Restrict$_\delta$ routine (even for both objectives). Consider the following scheme, where δ is the "uncertainty parameter" - $\delta < \epsilon$, but $1/\delta$ must be polynomially bounded in the size of the input and $1/\epsilon$, so that the overall algorithm runs in polynomial time:

```
**Algorithm Greedy**
Compute $y_{\min}$ and $x_{\min}$.
$\bar{y}_1 = y_{\min}(1+\epsilon)$;
$q_1 = \text{Restrict}_\delta(x, y \leq \bar{y}_1)$;
$Q = \{q_1\}; i = 1$;
**While** $(x_{\min} < x(q_i)/(1+\epsilon))$ **do**
{ $q'_{i+1} = \text{Restrict}_\delta(y, x < x(q_i)/(1+\epsilon))$;
  $\bar{y}_{i+1} = (1+\epsilon) \cdot \max\{\bar{y}_i, y(q'_{i+1})/(1+\delta)\}$;
  $q_{i+1} = \text{Restrict}_\delta(x, y \leq \bar{y}_{i+1})$;
  $Q = Q \cup \{q_{i+1}\}$;
  $i = i + 1$; }
**Return** $Q$.
```

Table 3.1: Pseudo-code for the greedy algorithm.

Since the Restricted routines are now approximate, in order to guarantee that the output set of points is an ϵ-Pareto set, we had to appropriately modify the algorithm based on the parameter δ. More specifically, note that the point q'_{i+1} can have y-coordinate up to $(1+\delta)$ times the minimum y over all points s satisfying $x(s) < x(q_i)/(1+\epsilon)$. In other words, there may exist a solution point $\tilde{s}$ satisfying $x(\tilde{s}) < x(q_i)/(1+\epsilon)$ and $y(\tilde{s}) = y(q'_{i+1})/(1+\delta)$. (The algorithm has "no way of knowing this" unless it uses a value of δ with $1/\delta$ exponential in the size of the input.) This "uncertainty" forces the algorithm to select as point q_{i+1} the leftmost point that satisfies $y(q_{i+1}) \leq (1+\epsilon)y(q'_{i+1})/(1+\delta)$. Due to this "weakness", we have the following:

Theorem 3.2.4. *For any $\delta > 0$, with $1/\delta$ polynomial in the size of the input and $1/\epsilon$, there exist instances on which the greedy algorithm above outputs a set Q such that $|Q| = 3k-1$, where $k = \mathrm{OPT}_\epsilon$.*

Proof. Denote by $P^*_\epsilon = \{p^*_1, \ldots, p^*_k\}$ the optimal set, where its points p^*_i, $i \in [k]$ are ordered in decreasing order of their x-coordinate, and $Q = \{q_1, \ldots, q_r\}$ the set selected by the greedy algorithm. By exploiting the *uncertainty* introduced by the parameter δ, we describe an adversarial scenario such that $r = 3k-1$.

The idea is the following: Consider the call $q'_{i+1} = \mathrm{Restrict}_\delta(y, x < x(q_i)/(1+\epsilon))$. By definition, we have $\tilde{y} \leq y(q'_{i+1}) \leq (1+\delta)\tilde{y}$, where $\tilde{y} = \min\{y(s) \mid x(s) < x(q_i)/(1+\epsilon)\}$. Suppose that the routine returns a point q'_{i+1} satisfying $\tilde{y} = y(q'_{i+1})$. Call this condition (†). If q'_{i+1} satisfies this condition, the optimal point p^*_j $(1+\epsilon)$-covering q'_{i+1} can have y-coordinate up to $(1+\epsilon)y(q'_{i+1})$, while the algorithm is forced to select a point q_{i+1} with y-value at most $(1+\epsilon)y(q'_{i+1})/(1+\delta)$.

We refer the reader to Figure 3.4 for an illustration. In the instance presented there, the rightmost optimal point p^*_1 $(1+\epsilon)$-covers all the solution points that are $(1+\epsilon)$-covered by the set $\{q_1, q_2\}$, while, for $j \geq 2$, the optimal point p^*_j $(1+\epsilon)$-covers all the solution points that are $(1+\epsilon)$-covered by the set $\{q_{3j}, q_{3j+1}, q_{3j+2}\}$. This proves the desired claim. In the following, we explain the situation in detail.

Consider the first point $q_1 \in Q$ selected by the algorithm. By the definition of the Restricted routine and the fact that q'_1 must be $(1+\epsilon)$-covered by p^*_1, it follows that $x(p^*_1) \geq x(q_1)/(1+\delta)$. Now suppose that the following scenario occurs: $x(p^*_1) = x(q_1)/(1+\delta)$, $x(q_1)/[(1+\epsilon)(1+\delta)] \leq x(q_2) < x(q_1)/(1+\epsilon)$ and there are no solutions with x-coordinate in the interval $\big[x(q_2)/(1+\epsilon), x(q_2)\big)$. Then, the point p^*_1 $(1+\epsilon)$-covers all solutions that are $(1+\epsilon)$-covered by the set $\{q_1, q_2\}$. Notice that the algorithm only "loses" one additional point here; we have that $x(q_2) < x(p^*_1)$. This is due

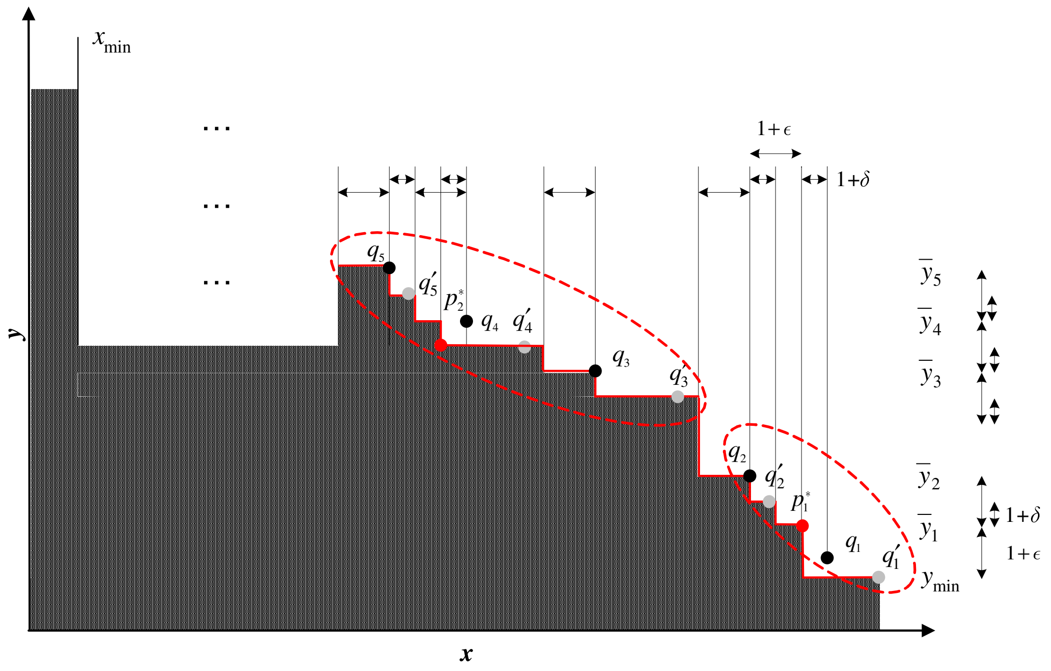

Figure 3.4: Illustration of the worst-case performance of the greedy approach. There are no solution points in the shaded region.

to the fact that we can exactly compute the minimum y-coordinate. However, since this does not hold for the next iterations, the algorithm can "lose" two additional points for each optimal point.

Now suppose that the points $\{p_2^*, q_3, q_3', q_4, q_4'\}$ satisfy the following scenario: q_3' satisfies condition (†), $y(q_3) = [(1+\epsilon)/(1+\delta)]y(q_3')$, $y(q_4') = (1+\delta)y(q_3)$, $x(q_4) = (1+\delta)x(p_2^*)$ and $y(p_2^*) = y(q_4')$. It is easy to see that these conditions are simultaneously realizable. (Observe that p_2^* $(1+\epsilon)$-covers q_3'.) Finally, if $x(q_4)/[(1+\epsilon)(1+\delta)] \leq x(q_5) < x(q_4)/(1+\epsilon)$ and there are no solutions with x-coordinates in the interval $\big[x(q_5)/(1+\epsilon), x(q_5)\big)$, the point p_2^* $(1+\epsilon)$-covers all the solutions $(1+\epsilon)$-covered by the set $\{q_3, q_4, q_5\}$.

By replicating the above described configuration, it follows inductively that p_{i+1}^* $(1+\epsilon)$-covers all the solutions $(1+\epsilon)$-covered by $\{q_{3i}, q_{3i+1}, q_{3i+2}\}$. This completes the proof. ■

In fact, one can show that the greedy algorithm guarantees a factor 3, i.e. the above described adversarial scenario represents a worst-case instance for the algorithm. Let us now try to understand why the greedy approach fails to guarantee a factor 2 in the aforementioned scenario. The problem is that, due to the uncertainty introduced by δ, the point p_2^* can lie arbitrarily to the left of q_3. Thus, the only invariant that the greedy algorithm can guarantee is $x(q_4) \leq (1+\delta)x(p_2^*)$.

We can overcome this obstacle by exploiting an additional structural property of the considered class of bi-objective problems. In particular, our generic algorithm will also use a polynomial routine for the following *Dual Restricted problem* (for the x-objective): Given an instance, a (rational) bound D and $\delta > 0$, either return a solution $\tilde{s}$ satisfying $y(\tilde{s}) \leq (1+\delta)D$ and $x(\tilde{s}) \leq \min\{x(s)$ over all solutions s having $y(s) \leq D\}$ or correctly report that there does not exist any solution s such that $y(s) \leq D$. Similarly, we drop the instance from the notation and use DualRestrict$_\delta(x, y \leq D)$ to denote the solution returned by the corresponding routine. If the routine does not return a solution, we will say that it returns NO. We say that the corresponding routine runs in polynomial time (resp. fully polynomial time) if its running time is polynomial in $|I|$ and $|D|$ (resp. $|I|$, $|D|$, $|\delta|$ and $1/\delta$).

The following lemma establishes the fact that any bi-objective problem that possesses a (fully) polynomial Restricted routine for the one objective, also possesses a (fully) polynomial *Dual Restricted* routine for the other.

Lemma 3.2.5. *For any bi-objective optimization problem, the problems Restrict$_\delta(y, \cdot)$ and DualRestrict$_\delta(x, \cdot)$*

are polynomially equivalent.

Proof. The proof of (both directions of) this equivalence uses binary search on the range of values of one objective with an application of the polynomial routine (for the other objective) at each step of the search. Let m be an upper bound on the number of bits in the objectives; recall that m is polynomially bounded in the size of the instance. Observe that (the absolute value of) the minimum possible difference between the objective values of any two solutions is at least 2^{-2m}.

First, we argue that a polynomial time algorithm for $\mathrm{Restrict}_\delta(y, x \leq C)$ can be used as a black box to obtain a polynomial time algorithm for $\mathrm{DualRestrict}_\delta(x, y \leq D)$.

Given an upper bound D and a (rational) error tolerance $\delta > 0$, the following algorithm computes the function $\mathrm{DualRestrict}_\delta(x, y \leq D)$:

1. If $\mathrm{Restrict}_\delta(y, x \leq 2^m)$ returns a solution s_0 having $y(s_0) > (1+\delta)D$ or returns "NO", then output "NO".

2. Otherwise, do a binary search on the parameter C in the range $[2^{-m}, 2^m]$ calling $\mathrm{Restrict}_\delta(y, x \leq C)$ in each step, until you find a value $\tilde{C}$ such that:

   (a) $\mathrm{Restrict}_\delta(y, x \leq \tilde{C})$ returns a solution $\tilde{s}$ satisfying $x(\tilde{s}) \leq \tilde{C}$ and $y(\tilde{s}) \leq (1+\delta)D$.

   (b) $\mathrm{Restrict}_\delta(y, x \leq \tilde{C} - 2^{-2m})$ either returns a solution s' having $x(s') \leq \tilde{C} - 2^{-2m}$ and $y(s') > (1+\delta)D$ or returns "NO".

   Output the solution $\tilde{s}$.

The number of calls to the routine $\mathrm{Restrict}_\delta(y, x \leq C)$ is $\Theta(m)$, so the overall algorithm runs in polynomial time. It remains to argue about the correctness. In case 1, either there are no feasible solutions or all solutions have y coordinate strictly greater than D. In case 2, all solutions s having $x(s) \leq \tilde{C} - 2^{-2m}$ also satisfy $y(s) > D$. Since there are no solutions with x coordinate strictly between $x(\tilde{s})$ and $\tilde{C} - 2^{-2m}$, it follows that $\tilde{C} \leq \min\{x$ over all solution points s having $y(s) \leq D\}$.

Conversely, given an upper bound C and a (rational) error tolerance $\delta > 0$, the following algorithm computes the function $\mathrm{Restrict}_\delta(y, x \leq C)$ using as a black box an algorithm for $\mathrm{DualRestrict}_\delta\ (x, y \leq D)$:

1. If $\mathrm{DualRestrict}_\delta(x, y \leq 2^m)$ returns a solution s_0 having $x(s_0) > C$ or returns "NO", then output "NO".

2. Otherwise, do a binary search on the parameter D in the range $[2^{-m}, 2^m]$ calling $\mathrm{DualRestrict}_\delta(x, y \leq D)$ in each step, until you find a value $\tilde{D}$ such that:

   (a) $\mathrm{DualRestrict}_\delta(x, y \leq \tilde{D})$ returns a solution $\tilde{s}$ satisfying $x(\tilde{s}) \leq C$ and $y(\tilde{s}) \leq (1 + \delta)\tilde{D}$.

   (b) $\mathrm{DualRestrict}_\delta(x, y \leq \tilde{D} - 2^{-2m})$ either returns a solution s' having $x(s') > C$ (and $y(s') \leq (1 + \delta)(\tilde{D} - 2^{-2m})$ or returns "NO".

   Output the solution $\tilde{s}$.

The justification is similar. The number of calls to the routine $\mathrm{DualRestrict}_\delta(x, y \leq D)$ is $\Theta(m)$, so the overall running time is polynomial. For the correctness, in case 1, either there are no feasible solutions or all solutions have x coordinate strictly greater than C. In case 2, all solutions s having $y(s) \leq \tilde{D} - 2^{-2m}$ also satisfy $x(s) > C$. Since there are no solutions with y coordinate strictly between $y(\tilde{s})$ and $\tilde{D} - 2^{-2m}$, it follows that $\tilde{D} \leq \min\{x$ over all solution points s having $x(s) \leq D\}$. ■

3.2.3.2 Algorithm Description

We first give a high-level overview of the 2-approximation algorithm. The algorithm iteratively selects a set of solution points $\{q_1, \ldots, q_r\}$ (in decreasing x) by judiciously combining the two routines. The idea is, in addition to the Restricted routine (for the y-coordinate), to use the Dual Restricted routine (for the x-coordinate) in a way that circumvents the problems previously identified for the greedy algorithm. More specifically, after computing the point q'_i in essentially the same way as the greedy algorithm, we proceed as follows. We select as q_i a point that: (i) has y-coordinate at most $(1 + \epsilon)y(q'_i)/(1 + \delta)$ and (ii) has x-coordinate *at most* the minimum x over all solutions s with $y(s) \leq (1 + \epsilon)y(q'_i)/(1 + \delta)^2$ for a suitable δ. This can be done by a call to the Dual Restricted routine for the x-objective. Intuitively this selection means that we give some "slack" in the y-coordinate to "gain" some slack in the x-coordinate. Also notice that, by selecting the point q_i in this manner, there may exist solution points with y-values in the interval

```
Algorithm 2–Approximation
If $\mathrm{Restrict}_{\delta_0 \leftarrow 1}(y, x \leq 2^m) = \mathrm{NO}$ then halt.
$q'_1 = \mathrm{Restrict}_{\delta}(y, x \leq 2^m)$;
$q_{\mathrm{left}} = \mathrm{DualRestrict}_{\delta_0 \leftarrow 1}(x, y \leq 2^m)$;    $x_{\min} = x(q_{\mathrm{left}})$;
$\bar{y}_1 = y(q'_1)(1+\delta)$;
$q_1 = \mathrm{DualRestrict}_{\delta}(x, y \leq \bar{y}_1)$;
$\bar{x}_1 = x(q_1)/(1+\epsilon)$;
$Q = \{q_1\}$; $i = 1$;
While $(\bar{x}_i > x_{\min})$ do
{ $q'_{i+1} = \mathrm{Restrict}_{\delta}(y, x < \bar{x}_i)$;
  $\bar{y}_{i+1} = [(1+\epsilon)/(1+\delta)] \cdot \max\{\bar{y}_i, y(q'_{i+1})/(1+\delta)\}$;
  $q_{i+1} = \mathrm{DualRestrict}_{\delta}(x, y \leq \bar{y}_{i+1})$;
  $\bar{x}_{i+1} = x(q_{i+1})/(1+\epsilon)$;
  $Q = Q \cup \{q_{i+1}\}$;
  $i = i + 1$; }
Return $Q$.
```

Table 3.2: Pseudo-code for factor-2 algorithm.

$((1+\epsilon)y(q'_i)/(1+\delta)^2, (1+\epsilon)y(q'_i)/(1+\delta)]$ whose x-coordinate is *arbitrarily* smaller than $x(q_i)$. In fact, the optimal point $(1+\epsilon)$-covering q_i can be such a point. However, it turns out that this is sufficient for our purposes and, if δ is chosen appropriately, this scheme can guarantee that the point q_{2i} lies to the left (or has the same x-value) of the i-th rightmost point of the optimal solution. We now proceed with the formal description of the algorithm. In what follows, the error tolerance is set to $\delta \doteq \sqrt[3]{1+\epsilon} - 1$ ($\approx \epsilon/3$ for small ϵ). (For the case that the Restricted routine is available for both objectives, we have a variant of this algorithm that achieves a ratio of 2 and is slightly more efficient in the sense that it uses error tolerance $\delta' \doteq \sqrt{1+\epsilon} - 1$.) If $\sqrt[3]{1+\epsilon}$ is not rational, we let δ be a rational that approximates $\sqrt[3]{1+\epsilon} - 1$ from below, i.e. $(1+\delta)^3 \leq (1+\epsilon)$, and which has representation size $|\delta| = O(|\epsilon|)$ (i.e. number of bits in the numerator and denominator). The set of points computed by the algorithm is shown in Figure 4.1.

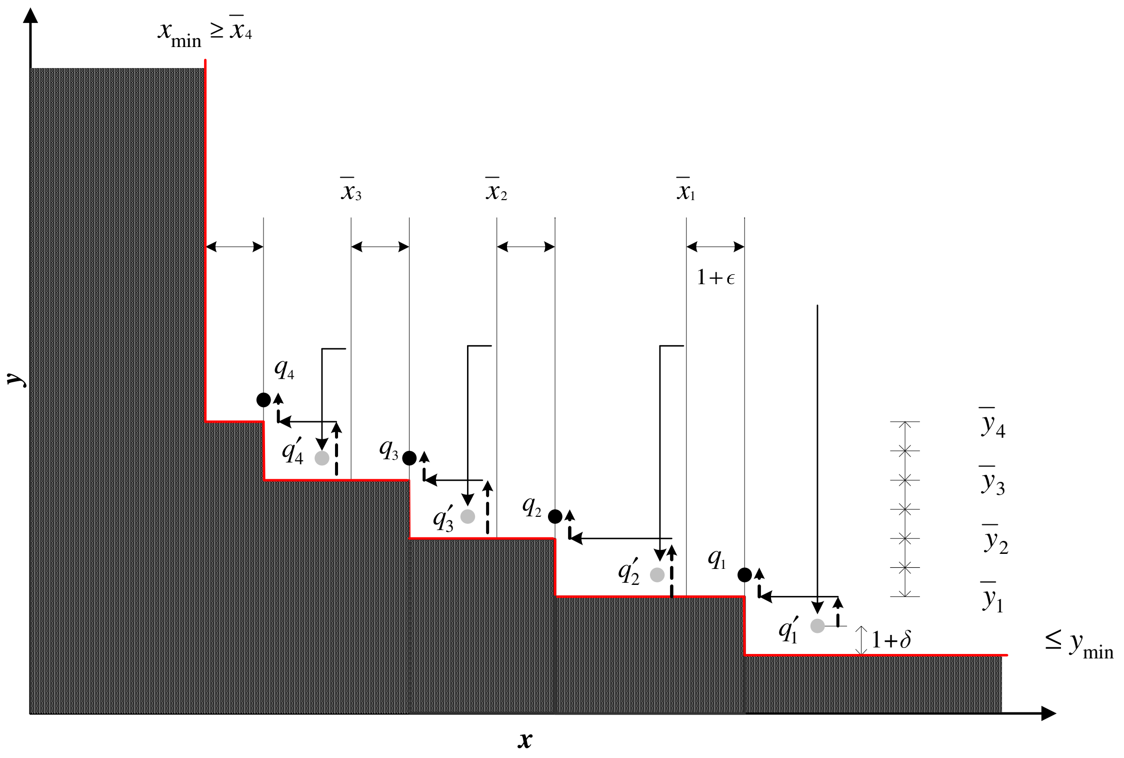

Figure 3.5: Schematic performance of the factor-2 algorithm. The scale is logarithmic in both dimensions. There are no solutions in the shaded region.

3.2.3.3 Algorithm Analysis

Recall that 2^m is an upper bound on the values of the objectives. Thus, if $\mathrm{Restrict}_{\delta_0 \leftarrow 1}(y, x \leq 2^m) =$ NO, there are no feasible solutions, in which case we can just terminate the algorithm. So, we can assume that the solution set is nonempty. In this case, the subroutine calls of lines 2 and 3 indeed return a solution; moreover, (*i*) the solution point q_{left} has minimum x-value among all feasible solutions and (*ii*) q'_1 has y-value *at most* $(1+\delta)y_{\min}$. Now observe that $y_{\min} \leq \bar{y}_i \leq \bar{y}_{i+1}$ and $\bar{x}_i > x_{\min}$ for all the values of i for which the body of the while loop is executed. It is thus easy to see that each subroutine call returns a point; so, all the points are well-defined.

Let $Q = \{q_1, q_2, \ldots, q_r\}$ be the set of solution points produced by the algorithm. We will prove that the set Q is an ϵ-Pareto set whose size is at most twice the optimum. We note the following simple properties.

Fact 3.2.6. *1. For each $i \in [r-1]$ it holds (i) $x(q'_{i+1}) < x(q_i)/(1+\epsilon)$ and (ii) for each solution point t with $x(t) < x(q_i)/(1+\epsilon)$, we have $y(t) \geq y(q'_{i+1})/(1+\delta)$.*
2. For each i $\in [r]$ it holds (i) $y(q_i) \leq (1+\delta)\bar{y}_i$ and (ii) for each solution point t with $y(t) \leq \bar{y}_i$ we have $x(t) \geq x(q_i)$.

Proof. The properties are just restatements of the definition of the two subroutines. ■

We can now prove the following lemmata (all properties used below refer to the above fact).

Lemma 3.2.7. *The x coordinates of the points $q_1, q_2, \ldots, q_r$ of Q form a strictly decreasing sequence.*

Proof. Consider two successive elements q_i, q_{i+1} of Q. For their x coordinates we will argue that $x(q_{i+1}) < x(q_i)/(1+\epsilon)$. First observe that $y(q'_{i+1}) \leq \bar{y}_{i+1}$. So, property 2-(*ii*) implies that $x(q_{i+1}) \leq x(q'_{i+1})$. Now from property 1-(*i*) we get $x(q'_{i+1}) < x(q_i)/(1+\epsilon)$ and the argument is complete. ■

The following lemma shows that Q is indeed an ϵ-Pareto set.

Lemma 3.2.8. *1. The point q_1 $(1+\epsilon)$-covers all of the solution points that have x-coordinate at least $x(q_1)/(1+\epsilon)$.*

2. For each $i \in [r] \setminus \{1\}$ the point q_i $(1+\epsilon)$-covers all of the solution points that have their x-coordinate in the interval $\big[x(q_i)/(1+\epsilon), x(q_{i-1})/(1+\epsilon)\big)$.

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

Proof. 1. Let t be a solution point with $x(t) \geq x(q_1)/(1+\epsilon)$. We need to show that t is $(1+\epsilon)$-covered by q_1. It clearly suffices to argue that $y(t) \geq y(q_1)/(1+\epsilon)$. Indeed, by property 2-*(ii)* we have $y(q_1) \leq (1+\delta)\bar{y}_1 = (1+\delta)^2 y(q_1')$ and the definition of q_1' implies that $y(t) \geq y(q_1')/(1+\delta)$, *for any solution point t.* By combining these facts we get that for any solution point t it holds $y(t) \geq y(q_1)/(1+\delta)^3 \geq y(q_1)/(1+\epsilon)$.

2. Let t be a solution point satisfying $x(q_i)/(1+\epsilon) \leq x(t) < x(q_{i-1})/(1+\epsilon)$; we will show that t is $(1+\epsilon)$-covered by q_i or equivalently that $y(t) \geq y(q_i)/(1+\epsilon)$. The proof is by contradiction. Suppose that there exists such a point t with $y(t) < y(q_i)/(1+\epsilon)$. By property 2-(*i*) and the definition of $\bar{y}_i$ this implies $y(t) < \max\{\bar{y}_{i-1}, y(q_i')/(1+\delta)\}$. Now since $x(t) < x(q_{i-1})/(1+\epsilon)$, property 1-(*ii*) gives $y(t) \geq y(q_i')/(1+\delta)$. Furthermore, since $x(t) < x(q_{i-1})$, by property 2-(*ii*) it follows that $y(t) > \bar{y}_{i-1}$. This provides the desired contradiction.

3. The termination condition of the algorithm is $x(q_r)/(1+\epsilon) \leq x_{\min}$. ■

Remark 3.2.9. We show in Lemma 3.2.10 below that the set Q is of cardinality $|Q| \leq 2\text{OPT}_\epsilon$. So, the algorithm could output this set of points. However, we observe that the set Q may contain "redundant" points: The y-coordinates of the points $q_1, \ldots, q_r$ do not necessarily form an increasing sequence. In fact, if $y(q_{i+1}) \leq (1+\delta)\bar{y}_i$, it may happen that $y(q_{i+1}) \leq y(q_i)$ (in which case the point y_i is redundant). (Note however that if $y(q_{i+1}) > (1+\delta)\bar{y}_i$, then by property 2-(*i*) we get $y(q_{i+1}) > y(q_i)$.) This observation can be further exploited for a post-processing step. For example, if $y(q_{2i}) \leq (1+\delta)\bar{y}_{2i-1}$, we can safely discard the point q_{2i-1} as implied by (the proof of) Lemma 3.2.8.

We now bound the size of the set of points Q in terms of the size of the optimal ϵ-Pareto set.

Lemma 3.2.10. *Let $P_\epsilon^* = \{p_1^*, p_2^*, \ldots, p_k^*\}$ be the optimal ϵ-Pareto set, where its points p_i^*, $i \in [k]$, are ordered in (strictly) increasing order of their y- and (strictly) decreasing order of their x-coordinate. Then, $|Q| = r \leq 2k$.*

Proof. We prove the following:

Claim 3.2.11. *If the algorithm selects a solution point q_{2i-1} (i.e. if $2i-1 \leq r$), then there must exist a point p_i^* in P_ϵ^* (i.e. it holds $i \leq k$) and if the algorithm selects a point q_{2i}, then $x(p_i^*) \geq x(q_{2i})$.*

The desired result follows directly from this. The claim is proved by induction on i.

Basis ($i = 1$). The first statement of the claim trivially holds. To show the validity of the second statement observe that for the rightmost point of P_ϵ^*, we must have $y\,(p_1^*) \leq y(q_1')(1+\epsilon) = \bar{y}_1(1+\epsilon)/(1+\delta) \leq \bar{y}_2$. The first inequality holds since the solution point q_1' must be $(1+\epsilon)$-covered by P_ϵ^* and in particular by the point of P_ϵ^* having the minimum y-coordinate. The two other inequalities follow from the definitions of $\bar{y}_1$ and $\bar{y}_2$. Now an application of property 2-(*ii*) gives $x(p_1^*) \geq x(q_2)$ and the base case is proved.

Induction step. Suppose that the claim holds for index $i-1$ (more specifically that $x(p_{i-1}^*) \geq x(q_{2i-2})$); we will prove it for i. We will prove each statement in turn.

Assume first that the algorithm selects a point q_{2i-1} (i.e. that $2i-1 \leq r$). We will show that P_ϵ^* contains a point p_i^* (i.e. that $i \leq k$). By the termination condition of the algorithm, our assumption implies that $x(q_{2i-2}) > (1+\epsilon)x_{\min}$. Therefore, by the induction hypothesis it follows that $x(p_{i-1}^*) > (1+\epsilon)x_{\min}$; that is, point p_{i-1}^* does *not* $(1+\epsilon)$-cover the leftmost solution point, which means there must exist a point p_i^* in the optimal set.

Now assume that the algorithm selects a point q_{2i}. We will show that $x(p_i^*) \geq x(q_{2i})$. First note that by property 1-(*i*) and the induction hypothesis $x(q_{2i-1}') < x(p_{i-1}^*)/(1+\epsilon)$. So, the point p_{i-1}^* does *not* $(1+\epsilon)$-cover the point q_{2i-1}' in the x-coordinate. Clearly, the latter point must be $(1+\epsilon)$-covered by a point in P_ϵ^*. Since the p_j^*'s are sorted in decreasing order of their x-coordinates, we conclude that p_i^* is the only eligible point for that purpose, i.e. q_{2i-1}' must be must $(1+\epsilon)$-covered by p_i^*. To complete the argument, we need the following fact:

Fact 3.2.12. *There does not exist any solution point t with $x(t) < x(q_{2i})$ such that t $(1+\epsilon)$-covers point q_{2i-1}'.*

Proof. We want to prove that for all solutions t having $x(t) < x(q_{2i})$ it holds $y(t) > (1+\epsilon)y(q_{2i-1}')$. For such a solution point t we have $y(t) > \bar{y}_{2i} \geq \bar{y}_{2i-1}(1+\epsilon)/(1+\delta) \geq (1+\epsilon)y(q_{2i-1}')$. The latter inequalities, in the order they appear, follow by applying property 2-(*ii*) and the definition of $\bar{y}_j$ (for $j = 2i-1, 2i$). ■

The above fact implies directly that $x(p_i^*) \geq x(q_{2i})$ and the proof is complete. ■

Thus far, we have proved that the set Q is an ϵ-Pareto set of size $|Q| \leq 2\mathrm{OPT}_\epsilon$. We now analyze the running time of the algorithm. Let k be the number of points in the smallest ϵ-Pareto set, $k = \mathrm{OPT}_\epsilon$. The algorithm involves $r \leq 2k$ iterations of the while loop; each iteration involves two calls to the subroutines. Therefore, the total running time is bounded by $4k$ subroutine calls. In summary, we proved the following theorem.

Theorem 3.2.13. *The above described algorithm computes a* 2*-approximation to the smallest* ϵ*-Pareto set in time* $O(\mathrm{OPT}_\epsilon)$ *subroutine calls, where* $1/\delta = O(1/\epsilon)$.

3.2.4 Applications

Our result can be applied to all of the problems which have a polynomial (or fully polynomial) time Restricted routine for one of the two objectives. It should be stressed that our algorithm is quite general; it does not assume for example linearity of the objectives. Applications include the shortest path problem [Han, Wa, ESZ, LR] and generalizations [EV, GR+, CX2, VV], cost-time trade-offs in query evaluation [PY2], matching [BBGS], spanning trees (and more generally matroid problems, see below) [GR, HL] and related problems [CX]. The aforementioned problems possess a polynomial Restricted routine for *both* objectives. In essence, for most of the aforementioned problems (with [PY2] being a notable exception), the two objectives are "the same" and we can efficiently optimize each of them separately. For several other problems [ABK1, ABK2, CJK, DJSS], the Restricted routine is available for one objective *only* (because it is NP-hard to separately optimize this objective). An example is the following classical scheduling problem: We are given a set of n jobs and a fixed number m of machines. Executing job j on machine i requires time p_{ij} and incurs cost c_{ij}. We are interested in the trade-off between makespan and cost. Minimizing the makespan is NP-hard, even for $m = 2$; hence, the Dual Restricted problem for this objective (equivalently, the Restricted problem for the cost objective) does not have a PTAS. If m is fixed, a fully polynomial time *Dual Restricted* routine for the cost objective is given in [ABK1]. (By Lemma 3.2.5 this implies an FPTAS for the Restricted problem for the makespan objective.)

For the bi-objective shortest path problem, a polynomial (resp. fully polynomial) Restricted routine corresponds to a polynomial (resp. fully polynomial) time approximation scheme for the

Restricted Shortest Path problem: given a bound on the cost of the path, minimize the delay of the path subject to the bound on the cost. This problem has been studied in a number of papers [Has, Wa, LR, ESZ]. The problem is NP-hard and 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 (directed) graphs [ESZ], and time $O(en(\log\log n + 1/\epsilon)$ for general (directed) graphs [LR], where n is the number of nodes and e is the number of edges. Moreover, the Dual Restricted problem also admits an FPTAS with the same time complexity. Thus, our algorithm runs in $O(en(\log\log n+1/\epsilon)\mathrm{OPT}_\epsilon)$ time for general graphs and $O(en\mathrm{OPT}_\epsilon/\epsilon)$ for acyclic graphs. The time complexity is comparable or better than previous algorithms [Han, Wa, TZ], which furthermore do not provide any guarantees on the size.

For the bi-objective spanning tree problem a polynomial Restricted routine corresponds to a polynomial time approximation scheme for the *Constrained Spanning Tree* (*CST*) problem [GR]: given a bound on the cost of the tree, minimize the weight of the tree subject to the bound on the cost. This problem is also NP-hard and is known to have a polynomial time approximation scheme [GR, HL]. (In fact, the aforementioned papers provide a PTAS for the more general problem of finding a minimum cost base of a matroid subject to a bound on the total length, as long as there is a polynomial time independence oracle for the matroid.) The best current algorithm for the problem [HL] has running time $O((1/\epsilon)^{1/\epsilon}n^3)$. As a corollary, our generic algorithm can compute a 2-approximation to the smallest ϵ-Pareto set in time $O((1/\epsilon)^{1/\epsilon}n^3\mathrm{OPT}_\epsilon)$. Whether such a 2-approximation can be computed in *fully* polynomial time is conditional on the existence of an FPTAS for the *CST* problem (which is an interesting open question). In contrast, by the results of [PY1, VY], a 3-approximation can be computed in fully polynomial time.

3.3 d Objectives

The results in this section use the GAP routine and thus apply to all problems in MPTAS.

3.3.1 Approximation of the optimal ϵ-Pareto set

Recall that for $d \geq 3$ objectives we are forced to compute an ϵ'-Pareto set, where $\epsilon' > \epsilon$, if we are to have a guarantee on its size [VY]. For any $\epsilon' > \epsilon$, a logarithmic approximation for the problem is given in [VY], by a straightforward reduction to the Set Cover problem. We can sharpen this result, by exploiting additional properties of the corresponding set system.

Theorem 3.3.1. *1. For any $\epsilon' > \epsilon$ there exists a polynomial time generic algorithm that computes an ϵ'-Pareto set Q such that $|Q| \leq O\big(d \log \mathrm{OPT}_\epsilon\big)\mathrm{OPT}_\epsilon$. The algorithm uses $O((m/\delta)^d)$ GAP_δ calls, where $1/\delta = O(1/(\epsilon' - \epsilon))$.*
2. For $d = 3$, the algorithm outputs an ϵ'-Pareto set Q satisfying $|Q| \leq c\mathrm{OPT}_\epsilon$, where c is a constant.

Definition 3.3.2. Consider the following problem $\mathcal{Q}(P, \epsilon)$: Given a set of n points $P \subseteq \mathbb{R}^d_+$ as input and $\epsilon > 0$, compute the smallest ϵ-Pareto set of P.

It should be stressed that, by definition, the set of points P is given *explicitly* in the input. (Note the major difference with our setting: for a typical multiobjective problem there are exponentially many solution points and they are not given explicitly.) This problem can be solved in linear time for $d = 2$ by a simple greedy algorithm. For $d = 3$ it is NP-hard and can be approximated within some (large) constant factor c [KP]. If d is arbitrary (i.e. part of the input, e.g. $d = n$), the problem is hard to approximate better than within a $\Omega(\log n)$ factor (unless $\mathrm{P} = \mathrm{NP}$) [VY].

The following fact, implicit in [VY], relates the approximability of $\mathcal{Q}$ with the problem of computing a small ϵ'-Pareto set for a multiobjective problem Π, given the GAP primitive. Let $\epsilon > 0$ be a given rational number. For any $\epsilon' > \epsilon$, we can find a $\delta > 0$ such that $1/\delta = O(1/(\epsilon' - \epsilon))$ satisfying $1 + \epsilon' \geq (1 + \epsilon)(1 + \delta)^2$.

Lemma 3.3.3. *Suppose that there exists an r-factor approximation algorithm for $\mathcal{Q}$. Then, for any $\epsilon' > \epsilon$, we can compute an ϵ'-Pareto set Q, such that $|Q| \leq r\mathrm{OPT}_\epsilon$ using $O((m/\delta)^d)$ GAP_δ calls.*

Proof. The algorithm proceeds in two phases; in the first phase, we compute a δ-Pareto set, by using the original algorithm of [PY1] and in the second phase we post-process the points produced by the latter algorithm by using the r-approximation algorithm for $\mathcal{Q}$ as a black box.

For the given instance $I \in \mathcal{I}_\Pi$, let $\mathfrak{X}(I)$ be the set of d-vectors of values of solutions in the objective space and fix an optimal ϵ-Pareto set $P^*_\epsilon = P^*_\epsilon(I)$. Let R be the δ-Pareto set produced in the first stage. We apply the r-approximation algorithm for $\mathcal{Q}$ on input R to produce a set $R' \subseteq R$ that $(1+\epsilon)(1+\delta)$-covers R. (Since $|R| \leq (m/\delta)^{d-1}$, it follows that the overall algorithm runs in polynomial time.) R' is clearly an ϵ'-Pareto set for the feasible set $\mathfrak{X}(I)$. We will argue that $|R'| \leq r\mathrm{OPT}_\epsilon$. Let R^* denote the smallest $(1+\epsilon)(1+\delta)$-cover for R using *only* points *from* R; we have $|R'| \leq r|R^*|$. The following simple claim completes the argument:

Claim 3.3.4. $|R^*| \leq \mathrm{OPT}_\epsilon$.

Proof. It suffices to show that there exists an $(1+\epsilon)(1+\delta)$-cover C for R of cardinality at most OPT_ϵ. Since R is a δ-Pareto set, for any solution point $s \in \mathfrak{X}(I)$, there exists a solution point $r \in R$ that $(1+\delta)$-covers s. C is constructed as follows: For each $s \in P^*_\epsilon$ pick an $r \in R$ that $(1+\delta)$-covers it. Then, $|C| \leq |P^*_\epsilon| = \mathrm{OPT}_\epsilon$. Every point $r \in R$ is $(1+\epsilon)$-covered by a point $s \in P^*_\epsilon$, which in turn is $(1+\delta)$-covered by a point $c \in C$. Therefore, C $(1+\epsilon)(1+\delta)$-covers all points of R. ■

■

Part 2 of Theorem 3.3.1 follows immediately from the fact that $\mathcal{Q}$ is constant factor approximable for $d = 3$ [KP] and Lemma 3.3.3. We consider the case of general d in the remainder.

To proceed, we need the following definition.

Definition 3.3.5. A set system is a pair $(U, \mathcal{R})$, where U is a set and $\mathcal{R}$ is a collection of subsets of U. For a set system $(U, \mathcal{R})$, we say that $X \subseteq U$ is *shattered* by $\mathcal{R}$ if for any $Y \subseteq X$, there exists a set $R \in \mathcal{R}$ with $X \cap R = Y$. The VC-dimension [VC] of the set system is the maximum size of any set shattered by $\mathcal{R}$. Let $T \subseteq U$ be a finite set and $r \in (1, \infty)$ be a parameter. A set $N \subseteq T$ is called an $1/r$-net for $(T, \mathcal{R})$ [HW], if $N \cap S \neq \emptyset$ for all $S \in \mathcal{R}$ having $|S| > |T|/r$.

The problem $\mathcal{Q}(P, \epsilon)$ can be formulated as a set cover problem as follows: For each point $q \in P$ and $\epsilon > 0$, define $S_{q,\epsilon} = \{x \in \mathbb{R}^d \mid q \leq (1+\epsilon) \cdot x\}$. $S_{q,\epsilon}$ is the subset of $\mathbb{R}^d$ that is $(1+\epsilon)$-covered by

q; it is a closed convex cone in $\mathbb{R}^d$ (a translation of the nonnegative orthant by the vector $q/(1+\epsilon)$). For each point $r \in P$, r is $(1+\epsilon)$-covered by q if and only if $r \in S_{q,\epsilon}$. Now consider the set system $\mathcal{F}(P,\epsilon) = (P, \mathcal{S}(P,\epsilon))$, where $\mathcal{S}(P,\epsilon) = \{P_{q,\epsilon} \equiv P \cap S_{q,\epsilon} \mid q \in P\}$. Clearly, there is a bijection between set covers of $\mathcal{F}(P,\epsilon)$ and ϵ-Pareto sets of P. We now establish the following:

Lemma 3.3.6. *a. For any finite set of points $P \subseteq \mathbb{R}^d$ and $\epsilon > 0$, it holds VC-dim$(\mathcal{F}(P,\epsilon)) \leq d$.*
b. There exists a set of points P such that VC-dim$(\mathcal{F}(P,\epsilon)) = d$.

Proof. a. Let P be a set of points in $\mathbb{R}^d$ and $\epsilon > 0$. We must argue that *no* subset $P' \subseteq P$ of cardinality $d+1$ can be shattered by $\mathcal{S}(P,\epsilon)$. Note that any such set $P' \subseteq P$ (of cardinality $d+1$) contains a point r none of whose coordinates is minimal, that is, a point r such that for all $i \in [d]$ there exists some point $q^i \in P'$ (different from r) with the property $(q^i)_i \leq r_i$. We claim that we cannot "separate" r from the remaining points of P' by any convex cone (as defined above). Indeed, a point that $(1+\epsilon)$-covers the q^i's is guaranteed to $(1+\epsilon)$-cover r (or equivalently, the "dichotomy" $\{q^i, i \in [d]\}$ cannot be realized).

b. Consider a set $P = A \cup C$, where $|A| = d$ and $|C| = 2^d$. Let $A = \{a_1, \ldots, a_d\}$. We select the a_i' s in A as follows: For each $i \in [d]$, the ith coordinate of a_i is equal to 1 and all the rest are equal to $1+2\epsilon$. The set A has two properties: (i) no two of its points $(1+\epsilon)$-cover each other and (ii) for any two points $p, q \in A$, we have $\text{argmin}_i p_i \neq \text{argmin}_i q_i$. The set C is selected such that each subset of A is $(1+\epsilon)$-covered by some point in C. In particular, let $X = \bigcup_{i \in \mathcal{I}(X)} \{a_i\}$ be a subset of A. We add the point c_X in C having each coordinate indexed by $\mathcal{I}(X)$ equal to $1+\epsilon$ and all the rest equal to $1+2\epsilon$. Clearly, the point c_X $(1+\epsilon)$-covers *exactly* the elements of X. ■

For $q \in P$ and $\epsilon > 0$, define $S^D_{q,\epsilon} = \{x \in \mathbb{R}^d \mid x \leq (1+\epsilon) \cdot q\}$; the cone $S^D_{q,\epsilon}$ is the subset of $\mathbb{R}^d$ that $(1+\epsilon)$-covers q. A point r $(1+\epsilon)$-covers q if and only if $r \in S^D_{q,\epsilon}$. The "dual" set system of $\mathcal{F}(P,\epsilon)$ is defined as $\mathcal{F}^D(P,\epsilon) = (P, \mathcal{S}^D(P,\epsilon))$, where $\mathcal{S}^D(P,\epsilon) = \{P^D_{q,\epsilon} \equiv P \cap S^D_{q,\epsilon} \mid q \in P\}$. In words, the elements are the points of P and for each point $q \in P$ we have a set consisting of the points $r \in P$ that $(1+\epsilon)$-cover q. An ϵ-Pareto set of P is equivalent to a hitting set of $\mathcal{F}^D$ (i.e. a subset $H \subseteq P$ that has non-empty intersection with every element of $\mathcal{S}^D(P,\epsilon)$).

It is well-known [As] that, if a set system has VC-dimension at most d, the VC-dimension of the dual set system is upper bounded by $2^{d+1}-1$. However, in our setting, essentially the same proof as in the previous lemma establishes the following:

Lemma 3.3.7. *For any finite set of points $P \subseteq \mathbb{R}^d$ and $\epsilon > 0$, it holds VC-dim($\mathcal{F}^D(P, \epsilon)$) $\leq d$. This bound is tight.*

Proof. Let P be a set of points in $\mathbb{R}^d$ and $\epsilon > 0$. We must argue that *no* subset $P' \subseteq P$ of cardinality $d+1$ can be shattered by $\mathcal{S}^D(P, \epsilon)$. Similarly to the previous lemma, any set $P' \subset P$ of cardinality $d+1$ contains a point r such that for all $i \in [d]$ there exists some point $q^i \in P'$ ($q^i \neq r$) satisfying $(q^i)_i \geq r_i$. We claim that we cannot "separate" r from the remaining points. Indeed, if some point is $(1+\epsilon)$-covered by all the q^i's, then is also $(1+\epsilon)$-covered by r. The tightness is similar. ■

It is well-known that, for a set system of VC-dimension at most d, we can efficiently construct an $1/r$-net of size $s(r) = O(dr \log r)$ [KPW]; this bound is tight in general [PW, KPW]. As shown in [BG, ERS], for such a set system, there exists a polynomial time $s(\mathrm{OPT})/\mathrm{OPT}$-factor approximation algorithm for the minimum *hitting set* problem, where OPT is the cost of the optimal solution. If we apply this result to the dual set system $\mathcal{F}^D(P, \epsilon)$ we conclude:

Proposition 3.3.8. *Problem $\mathcal{Q}$ can be approximated within a factor of $O(d \log \mathrm{OPT}_\epsilon)$.*

Part 1 of Theorem 3.3.1 follows by combining Lemma 3.3.3 and Proposition 3.3.8.

Remark 3.3.9. If $s(r) = O(r)$, the reduction in [BG, ERS] implies a polynomial time constant factor approximation algorithm for the corresponding hitting set problem. This is exactly the approach in [KP]: they show that, for $d = 3$, $\mathcal{F}^D(P, \epsilon)$ admits an $1/r$-net of size $s(r) = O(r)$ and that such a net can be efficiently constructed. Note that the constant approximation ratio c implied for set cover using this approach is identified with the constant hidden in the big-Oh of the net-size $s(r)$. The corresponding constant in the construction of [KP], itself based on a result of [CV], is quite large and no good bounds have been calculated for it. A recent result [PR] implies that the dual set system induced by a finite set of points and translates of an orthant in $\mathbb{R}^3$ (a generalization of $\mathcal{F}^D(P, \epsilon)$) admits a $1/r$-net of size at most $25r$ (that is efficiently constructible). Hence, for $d = 3$, problem $\mathcal{Q}$ can be efficiently approximated within a factor of 25 and the constant c in (the second statement of) Theorem 3.3.1 is at most 25. Improving the value of this constant is an interesting open problem.

3.3.2 The Dual Problem

For a d-objective problem Π with an associated GAP routine, given a parameter k, we want to find k solution points that provide the best approximation to the Pareto curve, i.e. such that every Pareto point is ρ^*-covered by one of the k selected points for the minimum possible ratio $\rho^* = 1 + \epsilon^*$. It was shown in [VY] that for $d = 2$ the problem is NP-hard but has a PTAS. We show below (Section 3.3.2.1) that for $d = 3$ any multiplicative factor for the dual problem is impossible, even for explicitly given points; we can only hope for a constant power, and only above a certain constant.

In [VY] the dual problem was related to the asymmetric k-center problem, and this was used to show that (i) for any d, a set of k points can be computed that approximates the Pareto curve with ratio $(\rho^*)^{O(\log^* k)}$, and (ii) for unbounded d and explicitly given points, it is hard to do much better. Since the metric ρ for the dual problem is a ratio (multiplicative coverage) versus distance (additive coverage) in the k-center problem, in some sense the analogue of constant factor approximation for the Dual problem is constant power. Can we achieve a constant power $(\rho^*)^c$ for all problems in MPTAS with a fixed number d of objectives? We show (Section 3.3.2.2) that the answer is Yes for $d = 3$ and provide a conjecture that implies it for general d.

3.3.2.1 Lower Bound

We start by formally defining the dual problem with explicitly given points:

Definition 3.3.10. Consider the problem $\mathcal{D}(P, k)$: We are given *explicitly* a set P of n points in $\mathbb{R}^d_+$ and a positive integer k and we want to compute a subset of P of cardinality (at most) k that ρ-covers P with minimum ratio ρ.

Let $\rho^* = 1 + \epsilon^*$ denote the optimal value of the ratio. Note that problems $\mathcal{Q}$ and $\mathcal{D}$ are polynomially equivalent with respect to exact optimization – as opposed to approximation. As shown in [KP], $\mathcal{Q}$ is NP-hard for $d \geq 3$; hence, for $d \geq 3$, problem $\mathcal{D}$ is also NP-hard.

By further exploiting the properties of the aforementioned reduction in [KP], we can show that problem $\mathcal{D}$ is NP-hard to approximate. Before we proceed with the formal statement and proof of this fact, it will be helpful to give some remarks regarding the notion of "approximate coverage" in the definition of the approximate Pareto set. Throughout this chapter, our notion of coverage is *multiplicative*: for $\rho \geq 1$, a point $u \in \mathbb{R}^d_+$ ρ-covers a point $v \in \mathbb{R}^d_+$ iff $u \leq \rho \cdot v$ (coordinate-wise). Alternatively, one could define the notion of coverage additively: for $c \geq 0$, the point $u \in \mathbb{R}^d_+$ *additively* c-covers $v \in \mathbb{R}^d_+$ iff $u_i \leq v_i + c$ for all i. A notion of *additive* c-Pareto set can be naturally defined using the additive coverage. (Note that with the additive definition of coverage

Pareto sets and approximate Pareto sets are invariant under translation of the input set, while with our multiplicative definition they are invariant under scaling.)

On the one hand, the selection of multiplicative metric is standard and more natural in the context of approximation algorithms. On the other hand, it is essential in our setting in the following sense: For a (implicitly represented) multiobjective combinatorial optimization problem, the basic existence theorem of [PY1] (i.e. the fact that there always exists an ϵ-Pareto set of polynomial size) is based crucially on the multiplicative coverage. (In fact, it clearly does not hold under the additive coverage. This, of course, rules out the possibility of efficient algorithms for computing (any) approximate Pareto set in this context.) However, for the case that the set of points is given explicitly in the input (i.e. for problems $\mathcal{Q}$ and $\mathcal{D}$) the aforementioned obstacle does not occur and one can select the definition of coverage that is more appropriate for the specific application.

We will denote by $\log \mathcal{Q}$ and $\log \mathcal{D}$ the primal and dual problems respectively under additive coverage. We now try to relate the problem pairs $(\mathcal{Q}, \log \mathcal{Q})$ and $(\mathcal{D}, \log \mathcal{D})$ with respect to their approximability. To this end, we need a couple of more definitions. For two points $p, q \in \mathbb{R}^d_+$ the *ratio distance* between p and q is defined by: $\mathcal{RD}(p,q) = \max\{\max_i(p_i/q_i), 1\}$. (The ratio distance between p and q is the minimum value $\rho^* = 1 + \epsilon^*$ of the ratio ρ such that p ρ-covers q.) The *additive distance* between p and q is defined by: $\mathcal{AD}(p,q) = \max\{\max_i(p_i - q_i), 0\}$. (Analogously, the additive distance between p and q is the minimum value c^* of the distance c such that p additively c-covers q.) It is easy to see that $\mathcal{AD}(\cdot,\cdot)$ is a directed pseudo-metric.

We claim that the problems $\mathcal{Q}$ and $\log \mathcal{Q}$ are in some sense "equivalent" with respect to approximability. Indeed, it is easy to see that an r-approximation algorithm for problem $\mathcal{Q}$ implies an r-approximation for problem $\log \mathcal{Q}$ and vice-versa (by taking logarithms and exponentials of the coordinates respectively). Suppose for example that there exists a factor r approximation for $\log \mathcal{Q}$. We argue that it can be used as a black box to obtain an r-approximation for $\mathcal{Q}$. Given an instance (P, ϵ) of $\mathcal{Q}$, we construct the following instance of $\log \mathcal{Q}$: We take the set of points P', where P' contains a point p' for every point $p \in P$ whose coordinates are the logarithms of the corresponding coordinates of p. We also take $c = \log(1+\epsilon)$. That is, we ask for the smallest *additive* c-Pareto set of P'. If $p', q' \in P'$ are the images of $p, q \in P$ respectively, we have that $\mathcal{RD}(p,q) = 2^{\mathcal{AD}(p',q')}$. Hence, there exists a bijection between ϵ-Pareto sets of P and additive c-Pareto sets of P', i.e. this simple transformation is an approximation factor preserving reduction of $\mathcal{Q}$ to $\log \mathcal{Q}$. There is however a subtle point regarding the bit complexity of the produced instance: the coordinates of the points in P' (and the desired additive coverage c) may be irrational, thus not computable exactly. We argue that this is not a significant problem below.

Consider an instance (P', c) of $\log \mathcal{Q}$. (The following remarks also hold for $\mathcal{Q}$ and the dual problems.) Clearly, the feasible solutions to the problem, i.e. the (additive) c-Pareto sets of P', do not depend on the actual coordinates of the points in P', but only on the additive distance between every pair of points. Hence, the only information an (exact or approximate) algorithm for $\log \mathcal{Q}$ needs to know about the input instance is the set of pairwise distances. In fact, such an algorithm does not need an explicit representation of these distances as rational numbers. It is sufficient to have a succinct representation that allows: (i) efficiently computing a succinct representation of the sum of two (or more) distances (ii) efficiently comparing any two (sums of) distances and (iii) efficiently comparing (sums of) distances with c. Now the aforementioned transformation produces instances (P', c) of problem $\log \mathcal{Q}$ that clearly satisfy these properties (since we have an explicit representation of the starting instance (P, ϵ) of $\mathcal{Q}$ and we take logarithms). Hence, an r-approximation algorithm for $\log \mathcal{Q}$ can be used as a black box to obtain an r-approximation for $\mathcal{Q}$. Similar arguments may be used for the other direction.

For the dual problem, the choice of coverage (multiplicative versus additive) changes the objective function, which affects the approximability. Roughly speaking, a factor r-approximation algorithm for $\log \mathcal{D}$ is "equivalent" to a $(\rho^*)^r$-approximation algorithm for $\mathcal{D}$, where ρ^* is the value of the optimal ratio for the latter problem. For example, it is easy to see (by taking logarithms as above) that a factor r approximation for $\log \mathcal{D}$ implies a $(\rho^*)^r$-approximation for $\mathcal{D}$.
We have the following:

Theorem 3.3.11. *Consider the problem $\mathcal{D}(P, k)$ for $d = 3$ objectives.*
1. It is NP-hard to approximate the minimum ratio ρ^* within any polynomial multiplicative factor.
2. It is NP-hard to compute k points that approximate the Pareto curve with ratio better than $(\rho^*)^{3/2}$.

Proof. To prove both parts we take advantage of the properties in the NP-hardness reduction of [KP]. It is shown there that problem $\log \mathcal{Q}$ is NP-hard for $d = 3$ via a reduction from 3-SAT. Given an instance of 3-SAT, the reduction produces an instance (P, c) of $\log \mathcal{Q}$ such that the smallest additive c-Pareto set of P reveals whether the 3-SAT formula is satisfiable. We will not repeat the reduction here, but we will just give the properties of the construction below needed for our purposes. We prove each part separately.

1. The crucial property we need here is that the reduction in [KP] is strongly polynomial: Given an instance (formula) φ of 3-SAT with n clauses, the reduction constructs an instance of $\log \mathcal{Q}$ (or $\log \mathcal{D}$), consisting of a set P of points in 3 dimensions and an additive error bound c such that, if the formula φ is satisfiable then P has a (additive) c-cover with g points (for some parameter g of the construction), whereas if φ is not satisfiable then every c-cover must contain at least $g + 1$ points. The construction has the property that all the points of P have rational coordinates with $O(\log n)$ bits and the error bound $c \sim 1/n^2$ (to be precise, $c = 1/4n^2$). This property implies that in the (additive) dual problem $\log \mathcal{D}$ with a bound $k = g$ for the number of points in the cover, the additive "gap" in the value of the optimal covering distance between the Yes case (satisfiable 3-SAT instance φ) and the No case (non-satisfiable 3-SAT instance) is at least inverse polynomial in n, i.e. at least $\delta = 1/n^r$, for a (small) constant r: If the 3-SAT instance φ is satisfiable, the optimal value of the covering distance for the $\log \mathcal{D}$ instance P with $k = g$ is c; if φ is not satisfiable, the optimal distance is at least $c' = c + \delta$. By multiplying all the coordinates of the constructed instance by a factor of $2n^{r+l}$, where $l > 0$ is a constant, and rounding to the nearest integer, we get a new instance of $\log \mathcal{D}$ where all the points have integer coordinates and the value of the additive gap between the satisfiable and the unsatisfiable case is at least n^l. We then exponentiate each coordinate ($x \to 2^x$). The number of bits remains polynomial in the size of the original 3-SAT instance (thus the overall reduction takes polynomial time) and the value of the *multiplicative* gap is now 2^{n^l}.

2. To prove this part, it suffices to show that problem $\log \mathcal{D}$ does not have an approximation ratio better than $3/2$. The reduction in [KP] uses a number g of gadgets. The construction has gadgets for the variables and for the clauses, which are connected by paths of flip-flop gadgets that cross using crossover gadgets. If the formula is satisfiable, then we can cover the points with additive distance c with g points, one from each gadget. Otherwise, this is not possible. We thus select $k = g$ and ask for the "best k points" and the corresponding optimal covering distance c^*. As previously mentioned, if the formula is satisfiable, we have $c^* = c$. Now, if the formula is not satisfiable, we argue below that the optimal covering distance is $c^* \geq 3c/2$. The proof follows directly from this.

Suppose that the 3-SAT formula is not satisfiable and we want to select the best g points. First, we note that we still need one point from each gadget because otherwise all the points of a gadget must be covered by points in other gadgets that are "far away" (much further than c), since the gadgets are well-separated; that is, if some gadget contains no point of the solution then the covering distance is much larger than c. Since the formula is not satisfiable, after selecting g points, at least one gadget will remain "badly covered", i.e. the point we selected must cover more points of its gadget than its c-neighborhood. An examination of the three types of gadgets used in the construction shows that this gives covering distance $2c$ for both the flip-flop and clause gadgets and at least $3c/2$ for the crossover gadgets. Hence, if the formula is not satisfiable, the optimal covering distance is $c^* \geq 3c/2$. ■

3.3.2.2 Upper Bound

Consider the following *generalization* $\mathcal{Q}'(A, P, 1+\epsilon)$ of problem $\mathcal{Q}$: Given a set of n points $P \subseteq \mathbb{R}^d_+$, a subset $A \subseteq P$ and $\epsilon > 0$, compute the smallest subset $P^*_\epsilon(A) \subseteq P$ that $(1+\epsilon)$-covers A. It is easy to see that for $d = 3$ the arguments of [KP, PR] for $\mathcal{Q}$ can be applied to $\mathcal{Q}'$ as well showing that it admits a constant factor approximation (see Remark 3.3.9). We believe that in fact for all fixed d there may well be a constant factor approximation. Proving (or disproving) this for $d > 3$ seems quite challenging. The following weaker statement seems more manageable:

Conjecture 3.3.12. *For any fixed d, there exists a polynomial time* $((1+\epsilon)^{\alpha(d)}, \beta(d))$-bicriterion *approximation algorithm for* $\mathcal{Q}'(A, P, 1+\epsilon)$*, i.e. an algorithm that outputs an* $(1+\epsilon)^{\alpha(d)}$*-cover* $C \subseteq P$ *of A, satisfying* $|C| \leq \beta(d) \cdot |P^*_\epsilon(A)|$*, for some functions* $\alpha, \beta : \mathbb{N} \to \mathbb{N}$.

For $d = 3$, Conjecture 3.3.12 holds with $\alpha(3) \leq 2$, and $\beta(3) \leq 4$. This can be shown by a technical adaptation of the 3-objectives algorithm in [VY].

For general implicitly represented multiobjective problems with a polynomial GAP_δ routine, we formulate the following conjecture:

Conjecture 3.3.13. *For any fixed d, there exists a polynomial time generic algorithm, that outputs an* $(1+\epsilon)^{\alpha(d)}$*-cover C, whose cardinality is* $|C| \leq \beta(d) \cdot \text{OPT}_\epsilon$*, for some functions* $\alpha, \beta : \mathbb{N} \to \mathbb{N}$.

The case of $d = 3$ is proved in [VY] with $\alpha(3) =$ any constant greater than 2 and $\beta(3) = 4$. Note that, by (a variant of) Lemma 3.3.3, Conjecture 3.3.12 implies Conjecture 3.3.13. The converse is also partially true: Conjecture 3.3.13 implies Conjecture 3.3.12, if in the statement of the latter, problem $\mathcal{Q}'$ is substituted with problem $\mathcal{Q}$.

In the following theorem, we show that a constant factor bicriterion approximation for $\mathcal{Q}'$ implies a constant power approximation for the dual problem, given the GAP routine.

Theorem 3.3.14. *Consider a (implicitly represented) d-objective problem in MPTAS and suppose that the minimum achievable ratio with k points is ρ^*.*
1. For $d = 3$ objectives we can compute k points which approximate the Pareto set with ratio $O((\rho^*)^9)$, using $O((m/\delta)^d)$ GAP_δ calls, where $1/\delta = O(1/(\epsilon' - \epsilon))$.
2. If Conjecture 3.3.12 holds, then for any fixed d we can compute k points which approximate the Pareto set with ratio $O((\rho^*)^c)$, using $O((m/\delta)^d)$ GAP_δ calls, where $1/\delta = O(1/(\epsilon' - \epsilon))$ and $c = c(d)$.

Proof. Part 1 follows from 2 since Conjecture 3.3.12 holds for $d = 3$. (It will follow from the proof that $c(3) \leq 9$.) To show Part 2, we exploit the relation of problem $\mathcal{D}(P, k)$ with the asymmetric k-center problem. As observed in [VY], the problem $\log \mathcal{D}$ is an instance of the asymmetric k-center problem, which we now define for the sake of completeness. In the asymmetric k-center problem we are given a set of n vertices V with distances, $\mathrm{dist}(u, v)$ that must satisfy the triangle inequality, but may be asymmetric, i.e. $\mathrm{dist}(u, v) \neq \mathrm{dist}(v, u)$. We are asked to find a subset $U \subseteq V$, $|U| = k$, that minimizes $\mathrm{dist}^* = \max_{v \in V} \min_{u \in U} \mathrm{dist}(u, v)$. (Note that $\log \mathcal{D}(P, k)$ is an instance of this problem, where there exists a bijection between vertices of V and points of P and the distance between points (vertices) $p, q \in P$ is defined as $d(p, q) = \mathcal{AD}(p, q)$.)

We claim that, if problem $\mathcal{Q}'(A, P, 1 + \epsilon)$ admits a $((1 + \epsilon)^{\alpha(d)}, \beta(d))$-bicriterion approximation, then problem $\mathcal{D}(P, k)$ admits a $(\rho^*)^{c(d)}$ approximation for some function c (that depends on α and β). This is implied by the aforementioned reduction and the following more general fact: If we have an instance of the asymmetric k-center problem (problem $\log \mathcal{D}(P, k)$ in our setting) such that a certain collection of associated set cover subproblems (which are instances of problem $\log \mathcal{Q}'(A, P, 1+\epsilon)$ here) admits a constant factor bicriterion approximation (an algorithm that blows up both criteria by a constant factor), then this instance admits a constant factor *unicriterion* approximation (an algorithm that outputs a set of no more than k centers). This implication is not stated in [PV, Ar1], but is implicit in their work. One way to prove it is to apply Lemma 5 of [PV] in a recursive manner. We will describe an alternative method [Ar2] that yields better constants. We prove this implication, appropriately translated to our setting, in Lemma 3.3.16.

For a general multiobjective problem where the solution points are not given explicitly, we impose a geometric $\sqrt{1+\delta}$ grid for a suitable δ, call GAP_δ 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 provides a $(1+\epsilon')^{c(d)}$-cover of the Pareto curve, where $1+\epsilon' = (1+\epsilon)(1+\delta)^2$. ■

Remark 3.3.15. Even though the $O(\log^* k)$-approximation ratio is best possible for the (general) asymmetric k-center problem [CG+], the corresponding hardness result does not apply for $\log \mathcal{D}$ as long as the dimension d is fixed.

Let $H(\alpha)$ denote the harmonic number extended to fractional arguments by linear interpolation (i.e. $H(\alpha) = \sum_{i=1}^{\lfloor\alpha\rfloor} 1/i + (\alpha - \lfloor\alpha\rfloor)/\lceil\alpha\rceil$). For a function g, let $g^{(i)}$ denote the function iterated i times. Finally, for $b > 1$ define $H_b^*(\alpha) = \min\{i : H^{(i)}(\alpha) \leq b\}$. The following lemma completes the proof of Theorem 3.3.14.

Lemma 3.3.16. *Suppose that there exists an* $((1+\epsilon)^\alpha, \beta)$*-bicriterion approximation for* $\mathcal{Q}'(A, P, 1+\epsilon)$*. Then, problem* $\mathcal{D}(P, k)$ *admits a* $(\rho^*)^c$ *approximation, where* $c = H_{4/3}^*(\beta) + \alpha + 4$*. In particular, for* $\alpha = 2$ *and* $\beta = 4$*, we can get* $c = 9$*.*

Proof. The desired result can be shown by a careful application of the techniques introduced in [PV, Ar1]. We describe an algorithm – that we denote $\mathcal{D}(P, k)$, as the corresponding problem – which, given a (ρ^α, β)-bicriterion approximation algorithm, denoted $\mathcal{B}(A, P, \rho)$, for problem $\mathcal{Q}'(A, P, \rho)$ as a black box, computes a set $Q \subseteq P$ of (at most) k points that $(\rho^*)^c$-cover the set P, where ρ^* is the minimum ratio achievable with k points. We will denote by $\mathcal{B}(A, P, \rho)$ the set of points output by the algorithm $\mathcal{B}$ on input (A, P, ρ).

We first note the simple (and well-known) fact that it is no loss of generality to assume that the algorithm $\mathcal{D}(P, k)$ "knows" the optimal ratio ρ^*; this is because ρ^* will be one of the $O(|P|^2)$ pairwise ratio distances, hence we can try the algorithm for all of them and pick the best solution (or do an appropriate binary search, see e.g. [Ar1]).

To describe the algorithm, we appropriately translate the notions from [PV, Ar1] to the current setting. In tandem, we also provide a proof of correctness. We begin with a basic definition.

Definition 3.3.17. For a point $q \in P$ and a parameter $\rho > 1$, we denote $\Gamma^+(q, \rho) = \{p \in P \mid q \leq \rho \cdot p\}$ the set of points in P ρ-covered by q and $\Gamma^-(q, \rho) = \{p \in P \mid p \leq \rho \cdot q\}$ the set of points in P that ρ-cover q. We naturally extend this notation to sets $S \subseteq P$: $\Gamma^{\pm}(S, \rho) = \{p \in P \mid p \in \Gamma^{\pm}(s, \rho)$ for some $s \in S\}$. We say that the point $q \in P$ is a *ρ-center capturing vertex* (denoted ρ-CCV) if it satisfies $\Gamma^{-}(q, \rho) \subseteq \Gamma^{+}(q, \rho)$.

Consider an instance of the problem $\mathcal{D}(P, k)$ as defined above. Suppose that $\rho \geq \rho^*$. In this case, if the point q is a ρ-CCV, it ρ-covers at least one point of the optimal solution – in particular, the point q^* that ρ^*-covers q. Indeed, $q^* \in \Gamma^{-}(q, \rho^*) \subseteq \Gamma^{-}(q, \rho) \subseteq \Gamma^{+}(q, \rho)$. Hence, q ρ^2-covers every point in P ρ-covered by the point q^*. This simple property is crucial for the algorithm.

The algorithm in [PV] has two phases. In the first phase, roughly, it preprocesses the input set by iteratively finding CCV's and in the second phase it uses a recursive set cover procedure to cover the points not covered in the first stage. (The algorithm in [Ar1] replaces the second phase by an LP-based method.)

The algorithm $\mathcal{D}(P, k)$ works in three phases. The first phase is identical to the first phase in [PV, Ar1]: We preprocess the input set P by iteratively finding ρ^*-CCVs. In the second phase, $\mathcal{D}$ calls the bicriterion approximation algorithm $\mathcal{B}$ (with appropriately selected values of its parameters) to cover the subset of P that is not covered in the first phase. The remaining phase involves a careful application of the recursive greedy set cover procedure of [PV] followed by an application of the greedy set cover algorithm. To show correctness of the last step, we use the structural lemma of [Ar1] (itself a variant of a similar lemma in [PV], albeit with improved constants). The algorithm is presented in detail below.

We now proceed with an intuitive explanation of the different steps in tandem with a proof of correctness. We explain first what happens during the first phase. We have as input the set P, the parameter k and the optimal ratio ρ^*. (Recall that the algorithm can "guess" the optimal ratio.) We iteratively select ρ^*-CCV's as follows: For each ρ^*-CCV we find, we remove from the "active" set A (initialized to P) all the points $(\rho^*)^2$-covered by it, until no more CCV's exist in A. Let C be the set of CCV's thus discovered ($|C| \leq k$) and $A = P \setminus \Gamma^{+}(C, (\rho^*)^2)$ be the set of points in P not $(\rho^*)^2$-covered by any point in C. At this point, we note the following simple fact:

Fact 3.3.18. *The set $A := P \setminus \Gamma^{+}(C, (\rho^*)^2)$ can be ρ^*-covered by $k' = k - |C|$ points in $P \setminus \Gamma^{+}(C, \rho^*)$.*

If $|C| = k$ ($k' = 0$, $A = \emptyset$), we have selected a set of k points that $(\rho^*)^2$-cover the set P and

```
**Algorithm** $\mathcal{D}(P,k)$
(The optimal radius $\rho^*$ is known
 to the algorithm.)
(*Phase 1*)
$A = P$; $k' = k$; $C = \emptyset$;
**While** $\exists\ \rho^*$-CCV $q \in A$ **and** $k' > 0$ **do**
{ $C = C \cup \{q\}$;
    $A = A \setminus \Gamma^+(q, (\rho^*)^2)$;
    $k' = k' - 1$; }
(*Phase 2*)
$S_0 = \mathcal{B}(A, P \setminus \Gamma^+(C, \rho^*), \rho^*)$;
(*Phase 3*)
$\widehat{S}_0 = S_0 \setminus \Gamma^+(C, (\rho^*)^2)$;
$S_1$ = Rec-Cover $(\widehat{S}_0, A, P, \rho^*, k')$;
$\widehat{S}_1 = S_1 \setminus \Gamma^+(C, (\rho^*)^4)$;
$S_2$ = Greedy-Set-Cover $(\widehat{S}_1, P, (\rho^*)^3)$;
**Return** $Q := C \cup S_2$.
```

```
**Routine** Rec-Cover (*Input:* $S, A, P, \rho, l$)
(There exist $l$ vertices in $P$ that
 $\rho$-cover $S$,  where $S \subseteq A \subseteq P$.)
$S^0 = S$; $i = 0$;
**While** $|S^i| > 4l/3$ **do**
{
    Run Greedy Set Cover to $\rho$-cover $S^i$
    using points of $P$ and let
    $\widetilde{S}^{i+1} \subseteq P$ be the produced set.
    $S^{i+1} = \widetilde{S}^{i+1} \cap A$;
    $i = i + 1$;
}
**Return** $S^i$.
```

Table 3.3: Algorithm for the Dual Problem.

we can just terminate the algorithm. Otherwise, we proceed with the next phase. In the second phase, we call the algorithm $\mathcal{B}$ to ρ^*-cover the set A. By Fact 3.3.18, there exists a ρ^*-cover of A with k' points. Moreover, it is clear that such a cover lies in $P \setminus \Gamma^+(C, \rho^*)$. Hence, we get a set $S_0 \subseteq P \setminus \Gamma^+(C, \rho^*)$ of cardinality $|S_0| \leq \beta \cdot k'$ that $(\rho^*)^\alpha$-covers A. To motivate the next step, we note the following immediate implication of Fact 3.3.18:

Fact 3.3.19. *Let $S \subseteq A$. Then S can be ρ^*-covered by k' points in $P \setminus \Gamma^+(C, \rho^*)$.*

We also recall the following well-known fact [Chv, Joh, Lov] about the performance guarantee of the greedy set cover algorithm:

Fact 3.3.20. *For a set system $(U, \mathcal{R})$ suppose that there exists a set cover of cardinality p. Then the greedy algorithm outputs a cover of size at most $p \cdot H(|U|/p)$.*

At this point, we apply the recursive greedy set cover procedure from [PV] to cover $\widehat{S}_0 = S_0 \cap A$ using points from A. (The points in $S_0 \setminus \widehat{S}_0$ are $(\rho^*)^2$-covered by C.) Note that in each round of the recursive cover, we attempt to cover only those points from the last round that do not lie in $\Gamma^+(C, (\rho^*)^2)$, since C will cover those ones. We thus get a set $S_1 \subseteq P$ of cardinality $|S_1| \leq 4k'/3$ with the property that S_1 covers $S_0 \setminus \Gamma^+(C, (\rho^*)^{1+H^*_{4/3}(\beta)})$ with ratio $(\rho^*)^{H^*_{4/3}(\beta)}$. The latter statement can be shown by induction, using Fact 3.3.19 as an invariant. Since this essentially appears in [PV, Ar1] (see e.g. Lemma 13 in [Ar1]), we do not repeat it here. To motivate the next step, we need the following combinatorial lemma from [Ar1]:

Lemma 3.3.21 (Theorem 17 in [Ar1], rephrased)**.** *Let $C \subseteq P$ and $A = P \setminus \Gamma^+(C, (\rho^*)^2)$. Suppose A has no ρ^*-CCV's and that there exist k' centers (points in P) that ρ^*-cover A. Then there exists a set of $2k'/3$ centers in $P \setminus \Gamma^+(C, \rho^*)$ that $(\rho^*)^3$-covers $A' = P \setminus \Gamma^+(C, (\rho^*)^4)$.*

As a final step of the algorithm, we apply the greedy set cover algorithm – that may be viewed as one iteration of the recursive procedure – with parameter $(\rho^*)^3$ to cover $\widehat{S}_1 = S_1 \setminus \Gamma^+(C, (\rho^*)^4)$ using points from P (so that the optimum has cardinality at most $2k'/3$, according to Lemma 3.3.21). (Note that the points in $S_1 \setminus \widehat{S}_1$ are $(\rho^*)^4$-covered by C.) We thus get a set $S_2 \subseteq P$ of cardinality at most $(2k'/3) \cdot H\left((4k'/3)/(2k'/3)\right) = (2k'/3) \cdot H(2) = k'$ with the property that S_2 covers $\widehat{S}_1$ within $(\rho^*)^3$. We output the set $Q := C \cup S_2$; this set has cardinality at most k and it remains to argue that it covers P with ratio $(\rho^*)^{4+\alpha+H^*_{4/3}(\beta)}$.

Indeed, every point $p \in P$ falls in one of the following categories:

- The point p is $(\rho^*)^2$-covered by a point in C, i.e. $p \in \Gamma^+(C, (\rho^*)^2)$. (Note that if this is *not* the case, i.e. if $p \in A$, then it is $(\rho^*)^\alpha$-covered by S_0.)

- The point p is $(\rho^*)^\alpha$-covered by a point $p_0 \in S_0$ that is *not* $(\rho^*)^{H^*_{4/3}(\beta)}$ - covered by S_1. In this case, $p_0 \in \Gamma^+(C, (\rho^*)^{1+H^*_{4/3}(\beta)})$, so C covers p within ratio $(\rho^*)^{1+H^*_{4/3}(\beta)+\alpha}$.

- The point p is $(\rho^*)^\alpha$-covered by a point $p_0 \in S_0$ that is $(\rho^*)^{H^*_{4/3}(\beta)}$ - covered by a point $p_1 \in S_1$ that is *not* $(\rho^*)^3$-covered by S_2. In this case, $p_1 \in \Gamma^+(C, (\rho^*)^4)$, so C covers p within ratio $(\rho^*)^{4+H^*_{4/3}(\beta)+\alpha}$.

- The point p is $(\rho^*)^{\alpha}$-covered by a point $p_0 \in S_0$ that is $(\rho^*)^{H^*_{4/3}(\beta)}$ - covered by a point $p_1 \in S_1$ that is in turn $(\rho^*)^3$-covered by $p_2 \in S_2$. In this case, the point p_2 covers p within ratio $(\rho^*)^{3+H^*_{4/3}(\beta)+\alpha}$.

Hence the overall covering ratio is $(\rho^*)^{4+H^*_{4/3}(\beta)+\alpha}$, which completes the proof. ■

Remark 3.3.22. We note here that the recursive set cover procedure (used in the above lemma) was useful merely to improve the constants in the reduction. One can alternatively prove a (quantitatively inferior) version of the lemma by the following two-phase algorithm: In the first phase, preprocess the input set P by iteratively finding ρ-CCVs for appropriately chosen values of the parameter ρ. In the second phase, call the algorithm $\mathcal{B}$ to "cover" the subset of P that is not covered in the first phase. The analysis of this alternative algorithm is based on repeated applications of Lemma 3.3.21.

Remark 3.3.23. We should remark that the algorithms of this section are less satisfactory than the bi-objective algorithm of the previous section (and the 2-d and 3-d algorithms of [VY]) in several respects. One weakness is that the constants c obtained (for $d = 3$) are quite large: in the case of Theorem 3.3.1, the best constant c we can get follows from the net construction of [PR] (and is about 25). In the case of Theorem 3.3.14 there is still a large gap between the upper bound (of 9) and the lower bound (of $3/2$) in the exponent.

A second weakness of the algorithms is that they start by applying the general method of [PY1] calling the GAP routine on a grid, and thus incur always the worst-case time complexity even if there is a very small ϵ-Pareto set. Thus, we view our algorithms in this section mainly as theoretical proofs of principle, i.e. that certain (constant) approximations can be computed in polynomial time, but it would be very desirable and important to improve both the constants and the time.

3.4 Conclusion

We investigated in this chapter the problem of computing a minimum set of solutions for a multiobjective optimization problem that represents approximately the whole Pareto curve within a desired accuracy ϵ. We developed tight approximation algorithms for the bi-objective shortest path problem, spanning tree, and a host of other bi-objective problems. Our algorithms compute efficiently an approximate Pareto set that contains at most twice as many solutions as the minimum one; furthermore improving on the factor 2 for these specific problems is NP-Hard. The algorithm works in general for all bi-objective problems for which we have a routine for the Restricted problem of approximating one objective subject to a (hard) bound on the other. The algorithm calls this Restricted routine and a dual one as black boxes and makes quite effective use of them: for every instance, the number of calls is linear (at most 4 times) in the number of points in the optimal solution for that instance.

We presented also results for three and more objectives, both for the problem of computing an optimal ϵ-Pareto set and for the dual problem of selecting a specified number k of points that provide the best approximation of the full Pareto curve. As we indicated at the end of the last section, there is still a lot of room for improvement both in the time complexity and the constants of the approximations achieved. We would like especially to resolve Conjecture 3.3.13, hopefully positively. It would be great to have a general efficient method for any (small) fixed number d of objectives that computes for every instance a succinct approximate Pareto set with small constant loss in accuracy and in the number of points, and do it in time proportional to the number of computed points, i.e., the optimal approximate Pareto set for the instance in hand.

Chapter 4

Approximate Convex Pareto Sets

In this Chapter we provide a simple necessary and sufficient condition for the polynomial-time constructibility of an ϵ-convex Pareto set (anyone, not necessarily a small one), in terms of the approximate optimization of monotone linear combining functions of the objectives.

4.1 Efficient Computability: The Comb Problem

In [PY1] it was shown that every multiobjective optimization problem, possesses an ϵ-Pareto set (thus, also an ϵ-convex Pareto set) of size polynomial in the size of the instance and $1/\epsilon$. Recall that there is a simple necessary and sufficient condition [PY1], for the efficient computability of an ϵ-Pareto set for a multiobjective problem Π, with a fixed number of objectives d. As shown in [PY1], there exists PTAS (resp. FPTAS) for the computation of an ϵ-Pareto set if and only if there is a subroutine $\mathsf{GAP}_\delta(b)$ that solves the GAP problem for Π in time polynomial in $|I|$ and $|b|$ (resp. in $|I|$, $|b|$, $|\delta|$ and $1/\delta$). Thus, a (fully) polynomial time algorithm for the GAP problem is a sufficient condition for the (fully) polynomial constructibility of an ϵ-convex Pareto set. However, as shown in this chapter, it is not a necessary condition.

We give a useful and natural condition which characterizes the approximability of ϵ-convex Pareto sets. Our main result (Theorem 4.1.1) is very intuitive in the following sense: An approximate convex Pareto set contains an approximate optimum for *any* monotone linear combining function of the objectives. Rather surprisingly, we show that the converse is also true: If we can efficiently approximate *any* monotone linear combining function of the objectives, then we can

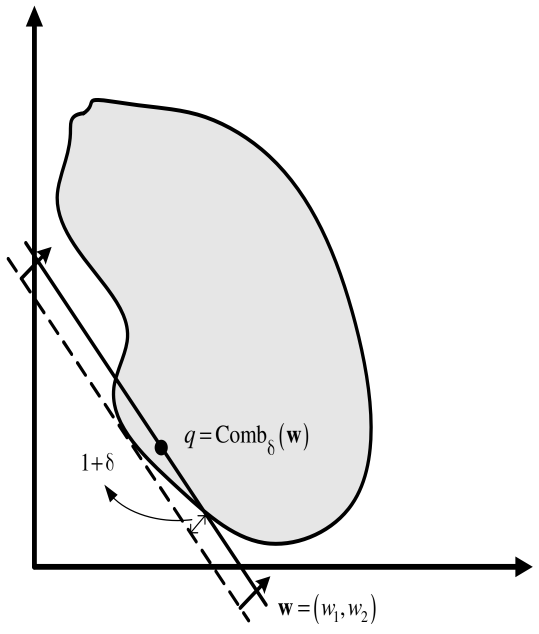
$q = \mathrm{Comb}_\delta(\mathbf{w})$
$1+\delta$
$\mathbf{w} = (w_1, w_2)$

Figure 4.1: Illustration of $\mathrm{Comb}_\delta(\mathbf{w})$ routine for two minimization objectives. The shaded region represents the (set of solution points in the) objective space. There exist no solution points below the dotted line.

efficiently compute an approximate convex Pareto set.

Let Π be an optimization problem with d minimization objectives $f_1, f_2, \dots, f_d$. We define the following associated single objective optimization problem:

Problem $\mathrm{Comb}_\Pi(I, \mathbf{w})$.
Input: d-objective problem Π with objective functions $\mathbf{f} = [f_1, \dots, f_d]$ (to be minimized), instance $I \in \mathcal{I}_\Pi$ and vector $\mathbf{w} \in \mathbb{R}^d_+$.
Goal: Compute a solution $s^* \in \mathcal{S}(I)$ minimizing the combined objective $v = \mathbf{w} \cdot \mathbf{f} = \sum_i w_i f_i$;

If the f_j's are all maximization objectives, then the problem $\mathrm{Comb}_\Pi(I, \mathbf{w})$ is defined in a similar way, the only difference being that the combined objective v is to be maximized. The case of mixed objectives raises some difficulties and is discussed after the proof of Theorem 4.1.1.

We say that the problem $\mathrm{Comb}_\Pi(I, \mathbf{w})$ has a PTAS (resp. FPTAS) if there exists an algorithm that, for every $I \in \mathcal{I}_\Pi$ and for all $\delta > 0$, computes a $(1 + \delta)$-approximate optimum and runs in time polynomial in $|I|$ and $|\mathbf{w}|$ (resp. $|I|$, $|\mathbf{w}|$, $|\delta|$ and $1/\delta$). For simplicity, we will usually drop the problem Π and the instance I from the notation and use $\mathrm{Comb}(\mathbf{w})$ (resp. $\mathrm{Comb}_\delta(\mathbf{w})$) to denote the corresponding problem. We say that the problem of constructing an ϵ-convex Pareto set has a PTAS (resp. FPTAS) if there is an algorithm that for every instance I and $\epsilon > 0$ constructs an ϵ-convex Pareto set $CP_\epsilon(I)$ and runs in time polynomial in $|I|$ (resp. polynomial in $|I|$, $|\epsilon|$ and $1/\epsilon$).

We are now ready to state our main theorem for this chapter:

Theorem 4.1.1. *Let the number of objectives d be fixed and of the same type. There is a (F)PTAS for constructing an ϵ-convex Pareto set* iff *the problem* Comb *admits a (F)PTAS.*

We postpone the proof of the theorem for the following section.

For the two dimensional case we will use the following equivalent notation for the Comb routine: For $\lambda \in \mathbb{R}^+$, we will denote by $\mathrm{Comb}_\delta(\lambda)$ a routine that returns a point $q \in \mathfrak{X}(\mathcal{I})$ with the following property: Consider the line $\ell(q, \lambda)$ through q with slope $-\lambda$, i.e. $\ell(q, \lambda) = \{(x, y) \in \mathbb{R}^2 \mid y + \lambda x = y(q) + \lambda x(q)\}$. Then there exists no solution point (in $\mathcal{I}$) below the line $(1 + \delta)^{-1} \cdot \ell(q, \lambda) \stackrel{\text{def}}{=} \{(x, y) \in \mathbb{R}^2 \mid y + \lambda \cdot x = (y(q) + \lambda x(q))/(1 + \delta)\}$. Geometrically we "sweep" the space with a line of absolute slope λ until the line "hits" $\mathfrak{X}(\mathcal{I})$. See Figure 4.2. We use the convention that, for $\lambda = +\infty$, we minimize the x objective. Let δ be the accuracy of the Comb oracle. We assume that either $\delta = 0$ (i.e. we have an exact routine), or we have a PTAS, i.e. can efficiently compute Comb_δ for all $\delta > 0$. For $\delta = 0$, i.e. when the optimization is exact, we omit the subscript and denote the Comb routine by $\mathrm{Comb}(\lambda)$.

4.2 Proof of Theorem 4.1.1

We prove the theorem for the case of minimization objectives. The proof is similar for the case of maximization objectives.

($\Rightarrow$) The reverse direction of the equivalence is quite simple. Suppose that there exists a (F)PTAS for constructing an ϵ-convex Pareto set. For an instance I, we are given a weight vector $\mathbf{w} \in \mathbb{R}^d_+$ and we want to compute a $(1 + \epsilon)$-approximate solution to $\mathrm{Comb}(\mathbf{w})$. To do so, we construct an ϵ-convex Pareto set $CP_\epsilon(I)$ and output the best solution point of $CP_\epsilon(I)$ under the combined objective function $v = \mathbf{w} \cdot \mathbf{f}$.

Since $|CP_\epsilon(I)| = O((4m/\epsilon)^{d-1})$, it is clear that this procedure takes (fully) polynomial time. We need to argue that the solution point output by this algorithm is an $(1 + \epsilon)$-approximate op-

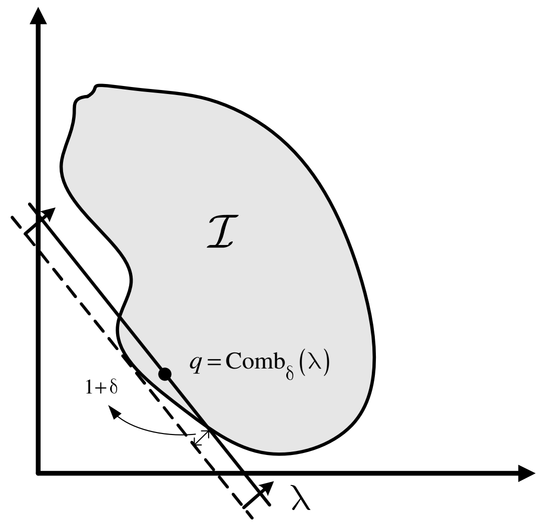

Figure 4.2: Illustration of $\text{Comb}_{\delta}(\lambda)$ routine. The shaded region represents the (set of solution points in the) objective space $\mathcal{I}$. There exist no solution points below the dotted line.

timum for v. Indeed, let q^* be an optimal *solution point* for the combined objective, i.e. $q^* = \text{argmin}\{v(q) \mid q \in \mathfrak{X}(I)\}$. By the definition of $CP_\epsilon(I)$, there exists a convex combination $cc = \sum_i \lambda_i p_i$ ($\lambda_i \geq 0$, $\sum_i \lambda_i = 1$) with $p_i \in CP_\epsilon(I)$, that $(1+\epsilon)$-covers q^*, i.e. $cc \leq (1+\epsilon)q^*$. By linearity and monotonicity of v, it follows that $v(cc) \leq (1+\epsilon)v(q^*)$ and $v(cc) \geq \min_i v(p_i)$. Hence, there exists some $p_i \in CP_\epsilon(I)$ such that $v(p_i) \leq (1+\epsilon)v(q^*)$ and in particular the solution point output by the procedure described above satisfies this property.

($\Leftarrow$) For the other direction, suppose that we have a (fully) polynomial time routine $\text{Comb}_{\delta}(\mathbf{w})$ and we want to efficiently construct an ϵ-convex Pareto set. We will describe an algorithm that makes a polynomial number of calls to the routine – for a fixed (appropriate) value of the parameter δ and (different) judiciously chosen values of the weight-tuple $\mathbf{w}$ – and uses the returned results (solutions) to compute an ϵ-convex Pareto set. We note that the algorithm is essentially the same as an algorithm given in [PY1], albeit in a more restricted context there, where all the objective functions are linear. We show however that the algorithm works in general, optimizing various parameters (e.g. number of calls to Comb) and giving a different (more general) proof of correctness, which is also much simpler.

We will need the following definition:

Definition 4.2.1. We say that a point $s \in \mathbb{R}_+^d$ is *approximately balanced* if all its coordinates are within a factor of 2 of each other.

The basic idea for the algorithm is the following: There is a way to $(1+\epsilon)$-cover all approximately balanced solution points using $O_d((1/\epsilon)^{d-1})$ many calls to the Comb routine (we use the notation $O_d()$ to indicate that the hidden constant of the big-Oh depends on d).

More precisely, let $\mathfrak{X}_{\text{bal}}$ denote the set of approximately balanced solution points in the objective space. It is not hard to see that there exists an ϵ-convex Pareto set for $\mathfrak{X}_{\text{bal}}$ of cardinality $O((1/\epsilon)^{d-1})$. We claim that we can compute such a set using $O_d((1/\epsilon)^{d-1})$ many calls to Comb. In particular, there exists a non-adaptively selected set of weight-vectors $\mathcal{W}_{\text{bal}} \subseteq \mathbb{R}_+^d$ of cardinality $|\mathcal{W}_{\text{bal}}| = O_d((1/\epsilon)^{d-1})$ with the property that the corresponding set of solution points $Q_{\text{bal}} = \{\mathsf{Comb}_\delta(\mathbf{w})\}_{\mathbf{w}\in\mathcal{W}_{\text{bal}}}$ – for an appropriate $\delta = \delta(\epsilon)$ – is an ϵ-convex Pareto set for $\mathfrak{X}_{\text{bal}}$. We stress the fact that the set $\mathcal{W}_{\text{bal}}$ will be selected in a non-adaptive way (a priori), which will make the overall algorithm non-adaptive.

Formally, to achieve this we proceed as follows: We start by picking two (rational) parameters $0 < \delta_1, \delta_2 < \epsilon$, with bit length representation $|\delta_i| = O(|\epsilon|)$ $(i = 1, 2)$, satisfying $(1+\delta_1)(1+\delta_2) \leq (1+\epsilon)$. (For simplicity, we could for example select $\delta_1 = \delta_2 = \delta = \sqrt{1+\epsilon} - 1$ ($\approx \epsilon/2$, for small ϵ). If $\sqrt{1+\epsilon} - 1$ is not rational, then we pick δ to be a rational satisfying $(1+\delta)^2 \leq (1+\epsilon)$ with bit length representation $O(|\epsilon|)$.) (We also note that, if the Comb routine is exact, we can set $\delta_1 = 0$ and $\delta_2 = \epsilon$.)

We continue by calling $\mathrm{Comb}_{\delta_1}(\mathbf{w})$, for all $\mathbf{w} \in \mathcal{W}_{\text{bal}}$, and outputting the convex Pareto set Q'_{bal} of the corresponding set of computed points Q_{bal}. To complete the description of the scheme, it remains to define the set of weight-vectors $\mathcal{W}_{\text{bal}}$. To wit; let $M = \lceil 2(d-1)/\delta_2 \rceil$. The set $\mathcal{W}_{\text{bal}}$ will be expressed as the union of d sets, i.e. $\mathcal{W}_{\text{bal}} = \bigcup_{j=1}^d \mathcal{W}_{\text{bal}}^j$. For $j \in [d]$, the set $\mathcal{W}_{\text{bal}}^j$ contains the vectors $\mathbf{w}$ whose j-th coordinate is equal to 1 ($w_j = 1$) and whose i-th coordinate w_i, $i \in [d] \setminus \{j\}$, is in the set $\mathcal{G} = \{l/M, l \in [M]\}$ – an additive "grid" between 0 and 1 with step $1/M$. That is, $\mathcal{W}_{\text{bal}}^j = \mathcal{G}^{j-1} \times 1 \times \mathcal{G}^{n-j}$.

It is clear that $|\mathcal{W}_{\text{bal}}^j| = O((4d/\epsilon)^{d-1})$, which in turn implies $|\mathcal{W}_{\text{bal}}| = O(4^{d-1}d^d \cdot (1/\epsilon)^{d-1})$. The correctness of this scheme is shown in Lemma 4.2.2. In particular, this lemma shows that the set Q_{bal} is an ϵ-convex Pareto set for the set of approximately balanced solution points.

Of course, the aforementioned scheme is not sufficient, since not all solution points are approximately balanced. However, we can exploit the fact that the convex Pareto set and ϵ-convex Pareto sets are invariant under scaling of the different objectives; this is a consequence of the multiplicative approximation. To efficiently construct an ϵ-convex Pareto set for the entire space, it would thus suffice to scale the different objectives f_j, $j \in [d]$, using a *polynomial* number of different scalings, so that for *every* solution point s in the objective space, there exists *some* scaling for which the *scaled version* of s is approximately balanced; at this point, one can apply the above scheme for *each* such scaling. Indeed, this can be achieved, as described below.

Recall that, by assumption, there exists a polynomial $p_\Pi(\cdot)$, such that for each instance $I \in \mathcal{I}_\Pi$, the values of the different objectives are between 2^{-m} and 2^m, where $m = p_\Pi(|I|)$. We will argue that there exists a set $\mathcal{R} \subseteq \mathbb{R}^d_+$ of $d(2m)^{d-1}$ scalings of the objectives with the desired property. To see this, divide the objective space into d subsets, according to the maximum valued coordinate (objective), i.e. $\mathfrak{X} = \bigcup_{j=1}^d \mathfrak{X}_j$, where $\mathfrak{X}_j = \{s = (s_1, \ldots, s_d) \in \mathfrak{X} \mid s_j = \max_{i \in [d]} s_i\}$. The claim is that we can make every point in $\mathfrak{X}_j$ approximately balanced using $(2m)^{d-1}$ scalings. To do this, we consider the set $\mathcal{R}_{-j}$ of all the scalings obtained by multiplying each of the objectives, *except* for the j-th one, by all the powers of 2 between 1 and 2^{2m-1}. Clearly, there exist $|\mathcal{R}_{-j}| = (2m)^{d-1}$ such scalings and it is easy to see that for each solution point $s \in \mathfrak{X}_j$ there exists some scaling $r \in \mathcal{R}_{-j}$ for which the scaled version of s, i.e. $s' = r \cdot s$, is approximately balanced. Naturally, we set $\mathcal{R} = \bigcup_{j=1}^d \mathcal{R}_{-j}$.

For $r = (r_1, \ldots, r_d) \in \mathcal{R}$, denote $r \circ \mathcal{W}_{\text{bal}} = \{(r_1 w_1, \ldots, r_d w_d) \mid (w_1, \ldots, w_d) \in \mathcal{W}_{\text{bal}}\}$. To summarize, the algorithm works as follows: For each scaling $r \in \mathcal{R}$, we call $\text{Comb}_{\delta_1}(\mathbf{w})$, for all $\mathbf{w} \in r \circ \mathcal{W}_{\text{bal}}$, and output the convex Pareto set of the corresponding set Q of solutions. The algorithm is given in Table 4.1:

The overall algorithm involves $O_d((m/\epsilon)^{d-1})$ (i.e. polynomially many for fixed d) calls to $\text{Comb}_{\delta_1}(\mathbf{w})$. Hence, it runs in polynomial time. For correctness, one needs to show that the computed set of solution points Q is an ϵ-convex Pareto set (if this is the case, then so is Q'). That is, we want to show that for any solution point $s \in \mathfrak{X}$ there exists a convex combination of points in Q that $(1+\epsilon)$-covers s. Note that, if a solution point s is approximately balanced with respect to the f_j's (i.e. without re-scaling the objectives), then, by Lemma 4.2.2 below, it is $(1+\epsilon)$-covered by a convex combination of points in Q_{bal}. Otherwise, the lemma applies for some scaled version of the point s and the correctness follows by invariance under scaling. Therefore, the proof is complete by

```
**Generic Algorithm** $\mathcal{A}(\Pi, I, \epsilon)$.
*Input:* Problem $\Pi$, instance $I \in \mathcal{I}_\Pi$ and $\epsilon > 0$.
*Output:* Set of solution points $Q'$ ($\epsilon$-convex Pareto set for $\mathcal{S}(I)$).

Pick $\delta_1, \delta_2 > 0$ satisfying $(1+\delta_1)(1+\delta_2) \leq (1+\epsilon)$ and $|\delta_1|, |\delta_2| = O(|\epsilon|)$;
$Q = \emptyset$;
$M = \lceil 2(d-1)/\delta_2 \rceil$ ; $m = p_\Pi(|I|)$;
For each scaling $r \in \mathcal{R}$ of the objectives
  For each weight vector $\mathbf{w} \in r \circ \mathcal{W}_{\text{bal}}$ do
  {   $q = \text{Comb}_{\delta_1}(\mathbf{w})$;
      $Q = Q \cup \{q\}$; }
Return $Q'$ = convex Pareto set of $Q$.
```

Table 4.1: Generic oblivious algorithm for the construction of a polynomial size ϵ-convex Pareto set.

the following:

Lemma 4.2.2. *For any approximately balanced solution point $s \in \mathfrak{X}_{\text{bal}}$, there exists a convex combination of points in Q_{bal} that $(1+\epsilon)$-covers s.*

Proof. Let $s \in \mathfrak{X}_{\text{bal}}$ be an approximately balanced solution point. We will show that:

There exist $\{\lambda_i\}_{i=1}^k$ – satisfying $\lambda_i \geq 0$, $\sum_{i=1}^k \lambda_i = 1$ – and $\{q_i\}_{i=1}^k \subseteq Q_{\text{bal}}$ such that $\sum_{i=1}^k \lambda_i q_i \leq (1+\epsilon) \cdot s$.

Claim 4.2.3. *The previous statement is equivalent to the following: For* any *weight vector $\mathbf{w} \in \mathbb{R}_+^d$, there exists a point $q \in Q_{\text{bal}}$ such that $\mathbf{w} \cdot q \leq \mathbf{w} \cdot (1+\epsilon)s$.*

Proof. ($\Rightarrow$) Suppose that $\{\lambda_i, q_i\}_{i=1}^k$ defines a convex combination that $(1+\epsilon)$-covers s, i.e. $\sum_{i=1}^k \lambda_i q_i \leq (1+\epsilon)s$. Let $\mathbf{w} \in \mathbb{R}_+^d$. By taking the inner product with $\mathbf{w}$, the latter coordinate-wise inequality yields $\sum_{i=1}^k \lambda_i (\mathbf{w} \cdot q_i) \leq \mathbf{w} \cdot (1+\epsilon)s$. Since $\lambda_i \geq 0$ and $\sum_{i=1}^k \lambda_i = 1$, it follows that there exists some $i \in [k]$ such that $\mathbf{w} \cdot q_i \leq \mathbf{w} \cdot (1+\epsilon)s$.

($\Leftarrow$) Suppose that for *any* direction $\mathbf{w} \in \mathbb{R}^d_+$, there exists a point $q \in Q_{\text{bal}}$ such that $\mathbf{w} \cdot q \leq \mathbf{w} \cdot (1+\epsilon)s$. This means that the point $(1+\epsilon)s$ is dominated by the lower envelope of Q_{bal}. Hence, there exists a convex combination of points in Q_{bal} that $(1+\epsilon)$-covers s. ■

We now prove the equivalent statement of Claim 4.2.3. Let $\mathbf{w} \in \mathbb{R}^d_+$ be an arbitrary weight-vector. It is no loss of generality (by scaling) to assume $\max_{i \in [d]} w_i = 1$ and suppose that $w_j = 1$, for some $j \in [d]$. Since $s = (s_1, \ldots, s_d)$ is approximately balanced it follows that

$$s_j \leq \mathbf{w} \cdot s \leq (2d-1)s_j \tag{4.1}$$

The left inequality is the contribution of the j-th coordinate to the corresponding inner product and the right inequality uses the fact that s is approximately balanced.

Let $\mathbf{w}^*$ be the weight-vector obtained from $\mathbf{w}$ by rounding *up* each coordinate w_i to the grid $\mathcal{G}$ - i.e the closest integer multiple of $1/M$. Therefore, $w^*_j = 1$ and, for $i \neq j$, $w_i \leq w^*_i \leq w_i + 1/M$. Note that $\mathbf{w}^* \in \mathcal{W}^j_{\text{bal}}$, hence it was considered by the algorithm $\mathcal{A}$. Let q^* be the solution returned by the algorithm for $\mathbf{w}^*$, i.e. $q^* = \text{Comb}_{\delta_1}(\mathbf{w}^*) \in Q_{\text{bal}}$. We will show that q^* satisfies the statement in Claim 4.2.3, i.e., that $\mathbf{w} \cdot q^* \leq \mathbf{w} \cdot (1+\epsilon)s$. We now turn to proving the latter inequality, which will complete the proof.

The crux of the argument lies in the following claim. The claim states that $\mathbf{w}^* \cdot s$ – the value of the inner product of the "rounded" vector $\mathbf{w}^*$ with s – is "close" to $\mathbf{w} \cdot s$ – the value of the inner product of the "original" vector $\mathbf{w}$ with s. We remark that the proof is crucially based on the fact that the point s is approximately balanced.

Claim 4.2.4. $\mathbf{w} \cdot s \leq \mathbf{w}^* \cdot s \leq (1+\delta_2)\mathbf{w} \cdot s$

Proof. The left inequality is clear, since $\mathbf{0} \leq \mathbf{w} \leq \mathbf{w}^*$ and $s \geq \mathbf{0}$. The right inequality follows from the following chain of calculations:

$$\begin{aligned}
(\mathbf{w}^* - \mathbf{w}) \cdot s &\leq (1/M) \sum_{i \in [d] \setminus \{j\}} s_i && \text{(because } 0 \leq w^*_i - w_i \leq 1/M,\ i \in [d] \setminus \{j\} \text{ and } w^*_j = w_j = 1) \\
&\leq (1/M) 2(d-1) s_j && (s \text{ is approximately balanced}) \\
&\leq \delta_2 s_j && \text{(from our choice of } M) \\
&\leq \delta_2 (\mathbf{w} \cdot s) && \text{(from (4.1))}
\end{aligned}$$

■

Also note that, by definition of the Comb routine, we have:

$$\mathbf{w}^* \cdot q^* \leq (1+\delta_1) \min_{q \in CP(I)} \mathbf{w}^* \cdot q \ \leq (1+\delta_1)\mathbf{w}^* \cdot s \tag{4.2}$$

By using the fact that $\mathbf{0} \leq \mathbf{w} \leq \mathbf{w}^*$, (4.2) and Claim 4.2.4 above, we get:

$$\mathbf{w} \cdot q^* \leq \mathbf{w}^* \cdot q^* \leq (1+\delta_1)\mathbf{w}^* \cdot s \leq (1+\delta_1)(1+\delta_2)\mathbf{w} \cdot s \leq (1+\epsilon)\mathbf{w} \cdot s$$

which is the desired result. ■

This completes the proof of Theorem 4.1.1.

4.3 Discussion

Remark 4.3.1. We comment on the case of mixed objectives: In this case the weights in the Comb problem must be positive for the one type of objectives and negative for the other. If we have an exact algorithm for Comb then we can construct again in polynomial time an ϵ-convex Pareto set; also, all the results in the following chapters that use an exact Comb routine hold. However, as far as approximate Comb is concerned, note that the weighted linear combination for mixed objectives may take negative values, and technically speaking, approximation ratios are defined only for positive functions. We can circumvent this by using absolute values, and requiring that the absolute difference between the value of the computed solution and OPT be bounded by $\delta|OPT|$. Such an approximate $Comb$ routine is also sufficient for the polynomial time construction of an ϵ-CP, and for the relevant algorithms of the following chapters.

Remark 4.3.2. We would like to emphasize that Theorem 4.1.1 does *not* depend on the nature of the objective space. The objective space can be either a convex set (not necessarily a polytope), a discrete set or even any continuous non-convex set. The only non-trivial assumption we need is that the values of the objectives are between 2^{-m} and 2^m, where m has bit representation polynomial in the size of the input. The following special cases are of particular interest:

1. The objective functions are linear and the feasible space is convex (not necessarily a convex polytope). In this case, the convexity is transferred to the objective space. An interesting subcase is when the decision space is polyhedral (i.e. multiobjective linear programming).

2. The decision space (thus, the objective space also) is discrete (finite). This includes all discrete combinatorial optimization problems of interest.

Let $\mathcal{A}$ be a *linear* multiobjective problem; a problem for which all $f_j(I, s)$, $j \in [d]$, are linear, that is, each solution $s \in \mathcal{S}(I)$ is a nonnegative n-dimensional vector, where $n = \text{poly}(|I|)$, and $f_j(I, s) = v_j \cdot s$, where the v_j's are nonnegative n-vectors given in the instance I. The single-objective version of $\mathcal{A}$ involves optimizing one linear function (with nonnegative coefficients) over the same solution space. For linear multiobjective problems, Theorem 4.1.1 reduces to the following:

Theorem 4.3.3 ([PY1]). *If $\mathcal{A}$ is a linear d-objective optimization problem, then there is a PTAS (resp. FPTAS) for constructing a $CP_\epsilon(I)$* iff *the single-objective version of $\mathcal{A}$ can be approximated within $1 + \epsilon$, in time polynomial in $|I|$ (resp. in time polynomial in $|I|$, $|\epsilon|$ and $1/\epsilon$).*

An important corollary of Theorem 4.1.1 is that the class of problems for which an *ϵ-convex Pareto set* is efficiently constructible is *much broader* than the corresponding class for the *ϵ-Pareto set*. Consider for example the multi-objective $s - t$ min-cut problem: We are given an undirected graph $G = (V, E)$ with a d-vector of nonnegative weights on each edge (i.e. a function $\mathbf{w} : E \to \mathbb{R}_+^d$) and a pair of nodes (s, t). The goal is to find an $s - t$ cut such that the sum of the weights of the edges crossing the cut (for each of the d objectives) is minimized. This is a linear multiobjective problem whose single-objective version is tractable (exactly solvable in polynomial time). Therefore, by Theorem 4.1.1, for any fixed d there is an FPTAS for constructing an *ϵ-convex Pareto set* for the problem. However, as shown in [PY1], even for $d = 2$, there is no FPTAS for constructing an *ϵ-Pareto set* for this problem (unless $\mathbf{P} = \mathbf{NP}$). (We note that the proof in [PY1] is a reduction from the minimum bisection problem that is approximation preserving for $d \geq 3$. This suggests that, for $d \geq 3$, there is no PTAS for constructing an ϵ-Pareto set for the considered problem.)

Chapter 5

Succinct Approximate Convex Pareto Sets

5.1 Chapter Organization

This chapter concerns the computation/approximation of the smallest ϵ-convex Pareto set CP^*_ϵ. There are two variants of this problem, depending on whether the (objective) space is continuous (convex) or discrete. We analyze each case separately. It turns out that the corresponding problems are related, but not identical. The structure of this Chapter is as follows: In Section 5.2 we address the bi-objective problem when the Pareto set is given explicitly in the input. We show that, in both the continuous and the discrete case, a minimum ϵ-convex Pareto set can be computed efficiently.

In Section 5.3 we consider the bi-objective problem when the Pareto set is not given explicitly, but is only accessible through a routine Comb that optimizes (exactly or approximately) monotone linear combinations of the objectives; we seek general-purpose algorithms that use Comb as a black box and achieve guaranteed performance in terms of the approximation error of the convex Pareto set and its size. Note that the Comb-based model is the one used typically in the multicriteria literature, it is consistent with the characterization of Chapter 4, and applies to a broad class of continuous and discrete multiobjective problems: Linear Programming, Markov Decision Processes, Shortest Paths, Spanning Trees, etc.; all of these have an exact Comb routine.

If the Comb routine is exact, we show that in the continuous case we can compute a minimum ϵ-convex Pareto set (with a polynomial number of calls to the Comb routine). In the important special case of Bi-objective Linear Programming, our algorithm uses roughly 2 LP calls per generated point of the minimum ϵ-convex Pareto set. In the discrete case, we can compute a factor-2 approximation, and the factor 2 is intrinsic, in the sense that no general-purpose algorithm can improve it; we show also NP-hardness specifically for the bi-objective shortest path problem. If we have an approximate Comb routine, we present an approximation algorithm for the construction of the minimum ϵ-convex Pareto set that achieves factor 3 in the continuous case and factor 6 in the discrete case; for both cases we show a lower bound of 2.

In Section 5.4 we discuss the problem for $d \geq 3$ objectives. We present upper and lower bounds, both for explicitly given point sets, and for implicitly specified instances. For explicitly given points we provide a constant factor approximation for $d = 3$, and an approximation with logarithmic factor for fixed $d \geq 4$; for arbitrary (unbounded) number d of objectives, the approximation problem is at least as hard as Set Cover (thus is not approximable better than $\Omega(\log n)$). For implicitly given instances we show that no bounded factor can be achieved for the same ϵ; but if we relax the allowed error to any $\epsilon' > \epsilon$, we can compute an ϵ'-convex Pareto set which achieves the same approximation factors (with respect to the minimum ϵ-CP) as the explicit point case.

5.2 Two Objectives – Explicitly Given Points

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, $p = \big(x(p), y(p)\big)$.

Problem Definition We consider the problem of computing the smallest ϵ-convex Pareto set of an *explicitly given* set of points $A \subseteq \mathbb{R}^2_+$. We assume that the objectives are to be minimized. The results can be easily extended to the case of maximization or mixed objectives. As previously mentioned, we will consider the following two natural "variants" of this problem:

- Problem $\mathcal{Q}_{\mathrm{D}}(A, \epsilon)$: The input set of points is the *discrete* set A and we are only allowed to use points of A for the approximation. Denote the optimal solution to this problem by $CP^*_{\mathrm{D}}(A, \epsilon)$. This variant corresponds to the case of a discrete objective space.

- Problem $\mathcal{Q}_{\mathrm{C}}(A, \epsilon)$: The input set of points is *convex*, in particular a *convex polytope*, represented by its set of vertices A. In this case we are allowed to use all points in $\mathcal{CH}(A)$ for the approximation. Denote the optimal solution to this problem by $CP^*_{\mathrm{C}}(A, \epsilon)$. This variant corresponds to the case of a continuous (convex) objective space.

In this section we show the following theorem, that applies for both versions of the problem:

Theorem 5.2.1 (informal statement)**.** *Given a set of points in the plane and a rational parameter $\epsilon > 0$, we can compute its smallest ϵ-convex Pareto set in polynomial time. The same holds for the dual problem of computing a set of k points that form an ϵ-convex Pareto set with minimum ϵ.*

The two variants of the problem are very related and the corresponding algorithms are similar. Even though a unified exposition is possible (in part), for the sake of clarity, we analyze each case separately. In subsection 5.2.1 we analyze the case of a convex objective space (problem $\mathcal{Q}_{\mathrm{C}}(A, \epsilon)$). The algorithm in this case is very simple and intuitive. In subsection 5.2.2, we point out that a naive adaptation of the aforementioned algorithm is suboptimal and we give a different algorithm that works in the discrete setting.

The definitions given in this section are mostly dimension-specific. We remark that these definitions will be naturally generalized to higher dimensions in later sections.

5.2.1 Convex (Objective) Space – problem $\mathcal{Q}_{\mathrm{C}}$:

Given a discrete set of points $A \subseteq \mathbb{R}^2_+$ and a rational error tolerance $\epsilon > 0$, we want to compute a minimum cardinality ϵ-convex Pareto set of A; the approximate set is allowed to contain any point of $\mathcal{CH}(A)$.

We begin with some notation and basic definitions. If a and b are points in the plane, we denote by $\overline{ab}$ the line segment they define. To simplify the exposition in some proofs, we will say that a lies to the left of (resp. below) b if $x(a) \leq x(b)$ (resp. $y(a) \leq y(b)$). If $x(a) < x(b)$, we will say that a lies strictly to the left of b. Similarly, if S is a set of points in the plane, the *leftmost* point in S is the point of S with minimum x-coordinate, etc. When we write $S = \{s_1, s_2, \ldots, s_n\}$, by convention it holds $x(s_i) < x(s_{i+1})$, $i \in [n-1]$, i.e. the points are ordered in increasing order of their x-coordinates.

Let $C = \langle c_1, \ldots, c_n \rangle$ denote a *polygonal chain* whose vertices are $c_1, \ldots, c_n$ and whose edges are $\overline{c_1c_2}, \overline{c_2c_3}, \ldots, \overline{c_{n-1}c_n}$. We assume that $x(c_i) < x(c_{i+1})$ and $y(c_i) > y(c_{i+1})$. It will be convenient to view C both as a set of points in the plane and as a function of x, $y = C(x)$.

Let $CP(A) = \{p_1, \ldots, p_n\}$ be the convex Pareto set of A. (By the above-mentioned convention, we have $x(p_i) < x(p_{i+1})$, $i \in [n-1]$.) The following notion will be useful:

Definition 5.2.2. The *lower envelope* of A, denoted by $\mathrm{LE}(A)$, is the polygonal chain defined by the vertices in $CP(A)$, i.e. $\mathrm{LE}(A) = \langle p_1, \ldots, p_n \rangle$.

For notational simplicity, we will write LE instead of $\mathrm{LE}(A)$ when there is no confusion on the underlying set A. It should be noted that the curve defined by LE has a special structure. Viewed as a function of x, $y = \mathrm{LE}(x)$, it is strictly decreasing and *convex*.

We also consider the (strictly decreasing convex) curve LE'_ϵ obtained from LE by scaling its points by a factor of $(1+\epsilon)$ in each coordinate, i.e. $\mathrm{LE}'_\epsilon \doteq (1+\epsilon) \cdot \mathrm{LE} = \{p' \mid p'/(1+\epsilon) \in \mathrm{LE}\}$. The following definition will be very useful in the description of the algorithm:

Definition 5.2.3. We say that a point q lies *between* the curves LE and LE'_ϵ if q is dominated by some point of LE and is not strictly dominated by any point of LE'_ϵ.

The above definition is straightforwardly generalized to sets of points. We now define the distance function used to measure the error in the approximation:

Definition 5.2.4. For two points $p, q \in \mathbb{R}^2_+$ we define the *ratio distance* between p and q by $\mathcal{RD}(p,q) = \max\{x(p)/x(q), y(p)/y(q), 1\}$.

Intuitively, the ratio distance between p and q is the minimum value $\rho^* = 1 + \epsilon^*$ of the ratio ρ such that p ρ-covers q. Note also that the ratio distance does not satisfy the "additive" triangle inequality. However, its logarithm does (i.e. $\log \mathcal{RD}(\cdot,\cdot)$ is a "pseudo-metric"). If S is a set of points, we define (as is standard) $\mathcal{RD}(p,S) = \min_{q \in S} \mathcal{RD}(p,q)$. Let p, q be points in LE. Denote by $\widehat{pq}$ the subset of LE with endpoints p and q. The ratio distance between $\overline{pq}$ and $\widehat{pq}$ is defined by: $\mathcal{RD}(\overline{pq}, \widehat{pq}) = \max\{\mathcal{RD}(s, \widehat{pq}) \mid s \in \overline{pq}\}$.

We now proceed to describe a restatement of the problem $\mathcal{Q}_{\mathrm{C}}(A, \epsilon)$ that forms the basis for the algorithm. As a first observation, we remark that there always exists an optimal solution to the problem that uses points *only* from LE. This follows directly from the fact that for any point $q \in \mathcal{CH}(A)$ there exists a point in LE that dominates q. Since points in $A \setminus CP(A)$ do not lie in the lower envelope, we can (and will) assume that $A = CP(A)$; otherwise, we can simply (and efficiently) pre-process the set A. We also assume that the points of A are sorted in increasing order of their x-coordinate.

For simplicity, we omit the subscript C and the input set A from some expressions. For example, we denote CP_ϵ instead of $CP_{\mathrm{C}}(A, \epsilon)$ for a solution to $Q_{\mathrm{C}}(A, \epsilon)$.

Lemma 5.2.5. *The problem $\mathcal{Q}_{\mathrm{C}}(A, \epsilon)$ is equivalent to the following: Compute a convex polygonal chain C with minimum number of vertices in* LE *having the following properties: (i) its leftmost (resp. rightmost) vertex $(1+\epsilon)$-covers the leftmost (resp. rightmost) point of $P(A)$ (ii) the curve C lies between* LE *and* LE'_ϵ.

Proof. ($\Rightarrow$) First, we show necessity, i.e. that any feasible solution to $\mathcal{Q}_{\mathrm{C}}(A, \epsilon)$ having *no redundant points* (i.e. points dominated by convex combinations of others) defines a polygonal chain satisfying the properties of the lemma.

To wit, consider an ϵ-convex Pareto set $CP_\epsilon = \{q_1, q_2, \ldots, q_r\}$, with $x(q_i) < x(q_{i+1})$, $i \in [r-1]$, having no redundant points. As observed above, we can assume w.l.o.g. that $q_i \in \mathrm{LE}$ for all $i \in [r]$. Since LE is strictly decreasing and convex, CP_ϵ defines a polygonal chain $C = \langle q_1, \ldots, q_r \rangle$ in $\mathbb{R}^2$ such that the function $y = C(x)$ is strictly decreasing and convex. By definition, for any point in $P(A)$ there exists a point in C (i.e., a convex combination of at most two points in CP_ϵ) that $(1+\epsilon)$-covers it.

We first argue about the necessity of property (i). The leftmost point of $P(A)$ (call it p_{l}) must be $(1+\epsilon)$-covered by a convex combination of points in CP_ϵ. Hence, we must have $x(q_1) \leq (1+\epsilon)x(p_{\mathrm{l}})$. Otherwise, since q_1 is the leftmost point of CP_ϵ, it is clear that no convex combination of points in CP_ϵ can $(1+\epsilon)$-cover p_{l} (because of the x-coordinate). By a symmetric argument, we get that $y(q_r) \leq (1+\epsilon)y(p_{\mathrm{r}})$. Thus, the first property must be satisfied.

Now we argue about the necessity of property (ii). Note that, by construction, every point of C is dominated by some point in LE. Indeed, this holds because its vertices q_i are points of LE and both C and LE are strictly decreasing and convex. We need to show that no point of C can be strictly dominated by any point in LE'_ϵ. Suppose, for the sake of contradiction, that this is the case, i.e. there exists a point $c \in C$ that is strictly dominated by some point in LE'_ϵ. (Such a point c must be an interior point of some edge $e \in C$.) Then, the following claim provides the desired contradiction:

Claim 5.2.6. *Suppose there exists a point $c \in C$ that is strictly dominated by some point in LE'_ϵ. Then, there exists a point in $CP(A)$ that is not $(1+\epsilon)$-covered by* any *point of C.*

Proof. To see why, observe that, if there exists a point $c \in C$ strictly dominated by a point in LE'_ϵ, then the curves C and LE'_ϵ *must* intersect in two points s_1 and s_2 (assume w.l.o.g. that $x(s_1) < x(s_2)$) so that C lies "above and to the right" of LE'_ϵ between these points (i.e. for all $x \in (x(s_1), x(s_2))$ it holds $C(x) > \mathrm{LE}'_\epsilon(x)$). This implies that there exists a point $s' \in \mathrm{LE}'_\epsilon$ "between" s_1 and s_2 (that is, whose x-coordinate satisfies $x(s_1) < x(s') < x(s_2)$) such that $s = s'/(1+\epsilon) \in CP(A)$ (i.e. s' is the scaled version of a vertex in $CP(A)$). The implication follows easily from the the fact that LE'_ϵ and C are strictly decreasing, piecewise linear and convex. Now, the line segment $\overline{Os}$, where O denotes the origin, intersects LE'_ϵ at the point s' and C at the point p_s. By construction, it holds $s' = (1+\epsilon)s$ and $p_s = (1+\delta)s'$, for some $\delta > 0$. Therefore, s is *not* $(1+\epsilon)$-covered by the point $p_s \in C$.

To complete the proof we must argue that no *other* point of C can $(1+\epsilon)$-cover s. Since C is strictly decreasing (viewed as a function of x), it is easy to verify that p_s is the *unique* point of C that minimizes its "error ratio" from s, i.e. that $\mathcal{RD}(C, s) = \mathcal{RD}(p_s, s) = (1+\epsilon)(1+\delta) > 1+\epsilon$ and that the minimizer is actually unique. ■

($\Leftarrow$) For sufficiency, we need to show that a polygonal chain C satisfying the two properties forms an ϵ-convex Pareto set. It is indeed easy to show that the set of vertices of C, $V(C) = \{c_1, c_2, \dots, c_r\}$ (by assumption $c_i \in \mathrm{LE}$ and this is no loss of generality) is an ϵ-convex Pareto set.

To wit, property (i) guarantees that all the points of $P(A)$ to the left of c_1 and below c_r are $(1+\epsilon)$-covered by these points. Now consider a point $q \in P(A)$ strictly to the right of c_1 and strictly above c_r. The line segment $\overline{Oq}$ intersects C at a point c', LE'_ϵ at a point $q' = (1+\epsilon')q$, for some $\epsilon' \leq \epsilon$, and by property (ii) c' dominates q'. Otherwise, q' would strictly dominate c'; this holds true because the two points lie in the same line through the origin. Thus, the point c' $(1+\epsilon)$-covers q. To finish the argument, notice that c' can be expressed as a convex combination

```
Algorithm Explicit–Convex–2D.
Input: $CP(A) = \{p_1, p_2, \dots, p_n\}$ and $\epsilon > 0$.
Output: $Q = \{q_1, \dots, q_r\}$ (optimal $\epsilon$-convex Pareto set).

Construct the polygonal chain $\mathrm{LE}'_\epsilon$;
If $\big(x(p_n) \leq (1+\epsilon)x(p_1)\big)$ then { $Q = \{p_n\}$; halt; }
Compute the point $q_1 \in$ LE with $x(q_1) = (1+\epsilon)x(p_1)$;
$Q = \{q_1\}$ ; $i = 1$;

While $\big(y(q_i) > (1+\epsilon)y(p_n)\big)$ do
{ Compute the rightmost point $q_{i+1}$ of LE $\epsilon$-visible from $q_i$;
  $Q = Q \cup \{q_{i+1}\}$;
  $i = i + 1$; }
```

Table 5.1: Optimal algorithm for explicit two dimensional convex case.

of two points of $V(C)$, namely the endpoints of the edge of C that contains it. This completes the proof. ■

Before we state the algorithm, we give the following definition:

Definition 5.2.7. We say that a point p *ϵ-sees* a point q (or q is *ϵ-visible* from p) if no point of the line segment $\overline{pq}$ is strictly dominated by a point of LE'_ϵ.

(In other words, this means that $\overline{pq}$ does not intersect the region in $\mathbb{R}^2$ that is strictly "above and to the right" of LE'_ϵ.) Motivated by Lemma 5.2.5, we proceed as follows: Given $CP(A)$ and ϵ, we construct the polygonal chain LE'_ϵ defined by $CP'(A) = \{p'_i \doteq (1+\epsilon) \cdot p_i \mid i \in [n]\}$. We then select a set of points $Q = \{q_1, q_2, \dots, q_r\}$ as follows: If p_n has $x(p_n) \leq (1+\epsilon)x(p_1)$, we select p_n and halt. Otherwise, the *leftmost* point $q_1 \in Q$ is the point of LE having x-coordinate $x(q_1) = (1+\epsilon)x(p_1)$. The remaining points are computed by the following iterative procedure: Point q_{i+1} is the *rightmost* point of LE that is ϵ-visible from q_i. The algorithm terminates when the point $p_n \in$ LE is $(1+\epsilon)$-covered by the current point $q_r \in Q$, i.e. $y(q_r) \leq (1+\epsilon)y(p_n)$.

We give pseudo-code for the algorithm in Table 5.2.1.

A schematic representation of the i-th iteration of the algorithm is given in Figure 5.1.

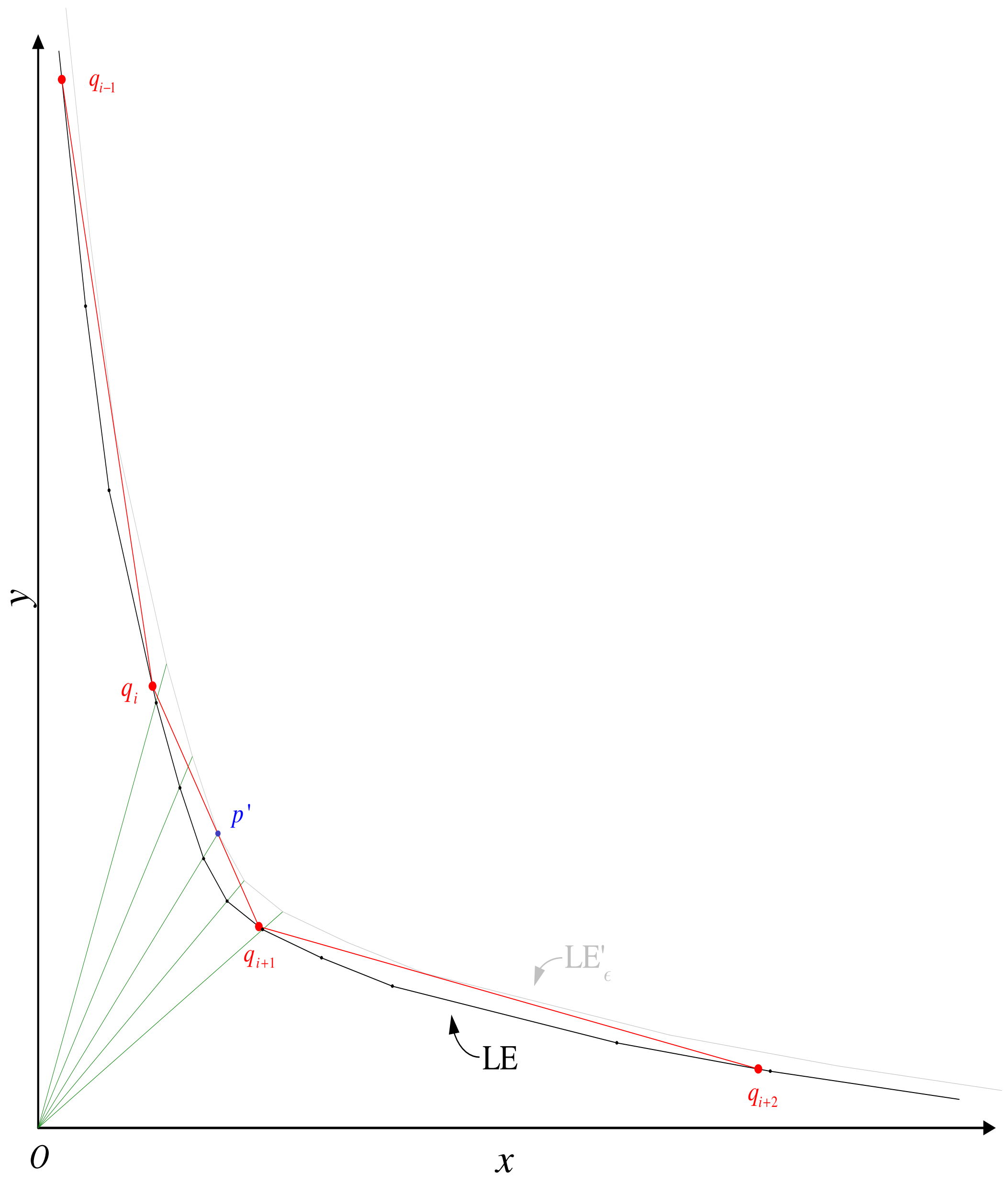

Figure 5.1: Schematic performance of the i-th iteration of algorithm CONVEX-2D.

Observe that the point $p \in \mathrm{LE}$ is $(1+\epsilon)$-covered by the point p_c; the intersection point of the segments $\overline{q_i q_{i+1}}$ and $\overline{p p_{1+\epsilon}}$.
We now proceed to analyze the algorithm. The following lemma establishes its correctness.

Lemma 5.2.8. *The set Q output by the described algorithm is an ϵ-convex Pareto set of minimum cardinality.*

Proof. First, we show that the set of points selected by the algorithm forms an ϵ-convex Pareto set. To do that, it suffices to verify that the two properties of lemma 5.2.5 are satisfied. By construction, the point $q_1 \in Q$ (resp. $q_r \in Q$) $(1+\epsilon)$-covers the point $p_1 \in CP(A)$ (resp. $p_n \in CP(A)$). So, the first property is satisfied. To see that the second property also holds, consider two consecutive points $q_i, q_{i+1} \in Q$. By the definition of visibility, there does not exist *any* point of the line segment $\overline{q_i q_{i+1}}$ that is *strictly* dominated by any point of LE'_ϵ.

The optimality of the algorithm can be shown by induction. Let $CP^*_\epsilon = \{p^*_1, p^*_2, \ldots, p^*_k\}$ denote the smallest ϵ-convex Pareto set. (The p^*_i' s are ordered "left to right" by convention.) Then, it is easy to show the following claim: For all $i \in [k]$, $x(p^*_i) \leq x(q_i)$. This implies that $r = k$, which is the desired result. The base case is straightforward; no point to the right of q_1 can $(1+\epsilon)$-cover p_1. The induction step follows from the following *monotonicity* property of visibility (itself a consequence of convexity):

Fact 5.2.9. *For all $i \in [r-1]$, consider the closed interval $[x(q_i), x(q_{i+1})]$. Any point in* LE *that lies strictly to the left of q_i does* not *ϵ-see q_{i+1}.*

It is now easy to see that the induction hypothesis – in particular, $x(p^*_i) \leq x(q_i)$, $i \in [r-1]$ – and the above property imply that $x(p^*_{i+1}) \leq x(q_{i+1})$. For the sake of contradiction, assume that this is not the case, i.e. $x(p^*_{i+1}) > x(q_{i+1})$. Then, by definition, the line segment $\overline{p^*_i p^*_{i+1}}$ contains a point *strictly* dominated by a point of LE'_ϵ, which is a contradiction due to Lemma 5.2.5. This finishes the induction and the proof. ■

We now analyze the time complexity of the algorithm. It is not hard to see that the algorithm uses a linear number of arithmetic operations (i.e. has time complexity $O(n)$ in the real RAM model). First, it is clear that LE'_ϵ can be constructed in linear time. Also note that each edge $\overline{q_i q_{i+1}}$

selected by the algorithm "supports" LE'_ϵ. Thus, it can be found in time linear in the number of vertices of LE "between" q_i and q_{i+1}. Therefore, the overall time complexity (in the real RAM model) is $O(n)$. This is not enough however: we need to bound also the bit precision of the computed points because n operations can generate in principle numbers with an exponential number of bits. We show in Lemma 5.2.10 below that this does not happen here: the computed points have rational coordinates with polynomially bounded bit complexity. We elaborate on this issue below.

We remark that it is not a priori clear that each arithmetic operation performed by the algorithm runs in time polynomial in the size of the input. This is due to the fact that we have to deal with the *bit precision needed to represent the vertices in the approximation*. By assumption, the points of the input set A are rational vectors, i.e. each coordinate is a rational number. (As is standard, rational numbers are represented in the form n_1/n_2, where n_1, n_2 are integers – $n_2 \neq 0$ – and the bit complexity of such a number is the sum of the bit representations of its numerator and denominator.) Now recall that the points q_i selected by the aforementioned algorithm are in general *not* points of the input set A, but rather convex combinations of such points. Hence, (some of) the q_i's may – in principle – even be irrational. On the other hand, even if the q_i's are guaranteed to be rational, their bit complexity may scale at each step of the algorithm, thus leading to a super-polynomial bit complexity for the selected set of points Q. To ensure a polynomial time algorithm in the *bit model*, we must rule out such "scenarios". It turns out that these possibilities can be indeed ruled out, as shown in the following lemma. It is in fact easy to argue that the q_i's are guaranteed to have rational coefficients, but proving a polynomial upper bound on the bit complexity is non-trivial.

Lemma 5.2.10. *The solution set $Q = \{q_1, \dots, q_k\}$ computed by the above algorithm has* total *bit complexity $O(k^2 m)$, where $k = |Q|$ and m is the maximum number of bits required to describe* any *vertex $p \in CP(A)$ and the error tolerance ϵ.*

Proof. The idea of the proof is to relate the bit complexities of any two consecutive points selected by the algorithm. Consider two such points $q_i, q_{i+1} \in Q$. If x is a vector with rational coefficients, we denote by $|x|$ its bit complexity. We will argue that the bit complexities of two consecutive points are "linearly related". More precisely, we show that $|q_{i+1}| = |q_i| + O(m)$, where m is the *maximum* number of bits required to represent *any* vertex $p \in CP(A)$ and the error tolerance ϵ. Since the point q_1 is easily seen to have bit complexity $O(m)$, the recurrence relation above implies that the bit complexity of the set Q is $O(k^2 m)$.

As previously mentioned, the edge $\overline{q_i q_{i+1}}$ "supports" the curve LE'_ϵ. Thus, the point q_{i+1} is the intersection of two lines: the line (defined by the segment) $\overline{q_i p'}$, where $p' \in CP'(A)$ and the line (defined by the segment) $\overline{p_l p_{l+1}}$, where $p_l, p_{l+1} \in CP(A)$ (see Figure 5.1 for an illustration). It follows easily from this that the q_i's have rational coefficients. Indeed, assuming that the point q_i is rational, the coordinates $x(q_{i+1})$ and $y(q_{i+1})$ are the solution of a linear system with rational coefficients (the one defined by the equations of the intersecting lines), thus rational (which follows for example from Cramer's rule). As far as bit complexity is concerned, the following claims hold true:

Claim 5.2.11. *For all $i \in [k]$ we have $y(q_i) = \alpha_i \cdot x(q_i) + \beta_i$, where α_i and β_i are rational numbers with bit complexity $O(m)$.*

Proof. Given that each vertex of $CP(A)$ has bit representation $O(m)$, the condition $q_{i+1} \in \overline{p_l p_{l+1}}$ implies that, for all $i \in [k]$, $y(q_i) = \alpha_i \cdot x(q_i) + \beta_i$, where α_i and β_i are rational numbers with bit complexity $O(m)$. (Indeed, $y = \alpha_i \cdot x + \beta_i$ is the equation of the line defined by the vertices p_l and p_{l+1}.) ■

The above claim implies that $|y(q_i)| = |x(q_i)| + O(m)$. Combining this fact with the following claim completes the proof.

Claim 5.2.12. *For all $i \in [k-1]$ we have: $|x(q_{i+1})| = |x(q_i)| + O(m)$.*

Proof. The condition $q_{i+1} \in \overline{q_i p'}$ implies:

$$\frac{y(q_{i+1}) - y(p')}{x(q_{i+1}) - x(p')} = \frac{y(q_i) - y(p')}{x(q_i) - x(p')}.$$

Combining with the equation from the previous claim, we get:

$$\frac{\alpha_{i+1} \cdot x(q_{i+1}) + \beta_{i+1} - y(p')}{x(q_{i+1}) - x(p')} = \frac{\alpha_i \cdot x(q_i) + \beta_i - y(p')}{x(q_i) - x(p')}.$$

Given that $|x(p')|, |y(p')| = O(m)$ (since p' is a vertex of LE'_ϵ), it can be shown (by elementary manipulations) that we can rewrite the above equality in the form:

$$c_1 + \frac{c_2}{\alpha \cdot x(q_{i+1}) + \beta} = c'_1 + \frac{c'_2}{\alpha' \cdot x(q_i) + \beta'}$$

where all the constants are integers with $O(m)$ bits each. From the latter relation, the claim follows easily. ■

This completes the proof of the lemma. ■

In other words, the described algorithm runs in polynomial time in the *bit model*. We remark that this property of the algorithm is also crucial for the generic algorithms described in later sections.

5.2.2 Discrete (Objective) Space – problem $\mathcal{Q}_{\mathrm{D}}$

Given a discrete set of points $A \subseteq \mathbb{R}^2_+$ and a rational error tolerance $\epsilon > 0$, we want to compute a minimum cardinality ϵ-convex Pareto set of A; the approximate set is allowed to contain only points from A.

We essentially use the same notation here as in the previous subsection. Let $P(A) = \{p_1, \ldots, p_n\}$ denote the Pareto set of A and $CP(A)$ its convex Pareto set. For simplicity, we omit the subscript D and the input set A from some expressions. For example, we denote CP_ϵ instead of $CP_{\mathrm{D}}(A, \epsilon)$ for a solution to $Q_{\mathrm{D}}(A, \epsilon)$.

Clearly, it suffices to consider the points in $P(A)$ for the approximation, in the sense that there always exists an optimal solution contained in $P(A)$. So, we can assume that $A = P(A)$. We also observe that points in $P(A) \setminus CP(A)$ (i.e. points dominated by convex combinations of others) - even points of $P(A)$ that do *not* lie on the lower envelope LE - are actually necessary for the approximation. It is easy to see that if we ignore such points, we lose at most a factor of 2; and as shown in the next section, this is tight.

We first show that a structural lemma very similar to lemma 5.2.5 holds in this setting also. The proof is almost identical, but we sketch it here for completeness.

Lemma 5.2.13. *The problem $\mathcal{Q}_{\mathrm{D}}(A, \epsilon)$ is equivalent to the following: Compute a convex polygonal chain C with minimum number of vertices in $P(A)$ having the following properties: (i) its leftmost (resp. rightmost) vertex $(1 + \epsilon)$-covers the leftmost (resp. rightmost) point of $P(A)$ (ii) the curve C lies between* LE *and* LE'_ϵ.

Proof. For necessity, consider an ϵ-convex Pareto set $CP_\epsilon = \{q_1, q_2, \ldots, q_r\}$, with $x(q_i) < x(q_{i+1})$, $i \in [r-1]$, having no redundant points. As observed above, we can assume w.l.o.g. that $q_i \in P(A)$

for all $i \in [r]$. By the definition of $P(A)$, CP_ϵ defines a polygonal chain $C = \langle q_1, \ldots, q_r \rangle$ in $\mathbb{R}^2$ such that the function $y = C(x)$ is strictly decreasing. The chain is convex, since otherwise there would exist a redundant point in CP_ϵ. By definition, for any point in $P(A)$ there exists a point in C that $(1+\epsilon)$-covers it. Similarly to the convex case, we can argue about the necessity of property (i). Note that, by construction, it suffices to argue that no point of C can be strictly dominated by any point in LE'_ϵ. For contradiction, suppose that there exists a point $c \in C$ strictly dominated by some point in LE'_ϵ. Then, we claim that there exists a point in $CP(A)$ not $(1+\epsilon)$-covered by *any* point of C. Sufficiency is also based on arguments parallel to those in Lemma 5.2.5. ■

In view of this similarity between the problem $\mathcal{Q}_{\mathrm{D}}$ and its convex counterpart $\mathcal{Q}_{\mathrm{C}}$, a naive approach to compute $CP^*_{\mathrm{D}}(A, \epsilon)$ would be the following: Given $P(A)$ and ϵ, compute its lower envelope LE (the convex polygonal curve having $CP(A)$ as its vertex set) and its scaled counterpart LE'_ϵ. Select as the leftmost point q_1 in the approximation the *rightmost* point of $P(A)$ that $(1+\epsilon)$-covers p_1 and at each iteration select as q_{i+1} the *rightmost* point of $P(A)$ ϵ-visible from q_i. It is not hard to construct examples for which this approach is suboptimal. It thus turns out that the *greedy criterion* "pick as next the *rightmost* point visible from the current one" fails for this variant of the problem.

We describe next a modified criterion that works. It can be assumed that $P(A)$ does not contain any points that are strictly dominated by LE'_ϵ; such points are redundant and cannot be part of an optimal solution. Let us now define a total order on the points of $P(A)$.

Definition 5.2.14. For $p, q \in P(A)$ we say that p is *ϵ-better* than q (we denote this by $p \succeq_\epsilon q$) if either (i) the *rightmost* vertex $v_p \in \mathrm{LE}'_\epsilon$ ϵ-visible from p, lies to the right of the corresponding vertex v_q for q, or (ii) $v_p = v_q = v$ and the line (defined by the segment) $\overline{pv}$ lies above the line $\overline{qv}$ to the *right* of v.

To simplify the exposition, we extend the definition to sets. Given a set of points S, we say that the point $b \in S$ is *an ϵ-best* point in the set if for any other point $y \in S$ it holds $b \succeq_\epsilon y$. We note that there may exist more than one points with this property; in such a case we can arbitrarily pick one of them. The modified algorithm selects a set of points $Q \subseteq P(A)$ as follows: For the computation of the leftmost point $q_1 \in Q$, consider the set of eligible points $E_1 = \{\sigma \in P(A) \mid x(\sigma) \leq (1+\epsilon)x(p_1)\}$. If there exists a point in E_1 that $(1+\epsilon)$-covers p_n, select it and halt.

Otherwise, select an ϵ-best point in E_1. For each $i \geq 2$, select q_i from the set (of eligible points) $E_i = \{\sigma \in P(A) \mid x(\sigma) > x(q_{i-1})$ and σ is $\epsilon -$ visible from $q_{i-1}\}$. If one of the points in E_i $(1+\epsilon)$-covers p_n, select it and halt. Otherwise, select an ϵ-best point in E_i and iterate.

The intuitive justification of the modified criterion is the following: From the analysis of the convex case, we know that a point of the lower envelope is a better choice than all the points to its left (for any $\epsilon > 0$). However, as opposed to the convex case, not all such points can be selected. In any case, a point of $CP(A)$ is a better choice than all the points to its left. So, suppose that we have selected the i-th point q_i and consider all the points to its right ϵ-visible from it (call this set E_{i+1}); the next point q_{i+1} must be one of them. Let v denote the rightmost vertex in $E_{i+1} \cap CP(A)$. By the analysis of the convex case, we can ignore all the points of E_{i+1} that lie to the left of v. But how can we "compare" v to the remaining points of E_{i+1}? These are undominated points that lie to its right, but none of them is in $CP(A)$. As mentioned, it is not always the case that the rightmost point is the correct choice. It turns out that this can be done by comparing the corresponding visibility regions.

Lemma 5.2.15. *The set Q output by the above algorithm is an ϵ-convex Pareto set of minimum cardinality.*

Proof. As in the convex case, it is straightforward to verify that the two properties of Lemma 5.2.13 are satisfied, thus the set Q forms an ϵ-convex Pareto set. Let $CP^*_\epsilon = \{p^*_1, \dots, p^*_k\}$ denote the smallest ϵ-convex Pareto set. For optimality, we will argue that for all $i \in [k]$ it holds $p^*_i \in E_i$; by construction, this implies the desired result. This is clearly true for $i = 1$; the leftmost point of any ϵ-convex Pareto set must $(1+\epsilon)$-cover p_1, so $p^*_1 \in E_1$. The proof follows from the next claim:

Claim 5.2.16. *For all $i \in [k-1]$, if $p^*_i \in E_i$ then $p^*_{i+1} \in E_{i+1}$.*

Proof. The claim holds in vacuum for $k = 1$. (If $k = 1$, then a point in E_1 $(1+\epsilon)$-covers p_n; such a point is selected as q_1 and forms the minimum cover.) We need to consider the case $k \geq 2$. Suppose that for some $i \in [k-1]$ it holds $p^*_i \in E_i$; either $q_i \equiv p^*_i$ or $q_i \succeq_\epsilon p^*_i$. This means that the rightmost vertex $v' \in CP(A)$ ϵ-visible from q_i lies to the right of the rightmost vertex $v \in CP(A)$ ϵ-visible from p^*_i. Notice that we can assume w.l.o.g. that the point p^*_{i+1} (is equal to or) lies to the right of v. This follows from the analysis of the convex case. The following fact is straightforward from the definition of the relation "$\succeq_\epsilon$" and finishes the proof of the claim:

```
Algorithm Explicit–Discrete–2D.
*Input:* $P(A) = \{p_1, p_2, \dots, p_n\}$ and $\epsilon > 0$.
*Output:* $Q = \{q_1, \dots, q_r\}$ (optimal $\epsilon$-convex Pareto set).

Construct the polygonal chain $\mathrm{LE}'_\epsilon$;
$E_1 = \{\sigma \in P(A) \mid x(\sigma) \le (1+\epsilon)x(p_1)\}$;
If $\big(\exists q \in E_1 : y(q) \le (1+\epsilon)y(p_n)\big)$ then { $Q = \{q\}$; halt; }
Compute the point $q_1 = \epsilon - \mathrm{best}(E_1)$;
$Q = \{q_1\}$ ; $i = 1$;

While $\big(y(q_i) > (1+\epsilon)y(p_n)\big)$ do
{ $E_{i+1} = \{\sigma \in P(A) \mid x(\sigma) > x(q_i)$ and $\sigma$ is $\epsilon -$ visible from $q_i\}$;
  If $\big(\exists q \in E_{i+1} : y(q) \le (1+\epsilon)y(p_n)\big)$ then { $Q = Q \cup \{q\}$; halt; }
  Compute the point $q_{i+1} = \epsilon - \mathrm{best}(E_{i+1})$;
  $Q = Q \cup \{q_{i+1}\}$;
  $i = i + 1$; }
```

Table 5.2: Optimal algorithm for explicit two-dimensional discrete case.

Fact 5.2.17. *Let $x, y \in E_i$ with $x \succeq_\epsilon y$. Let v be the rightmost vertex $v \in CP(A)$ ϵ-visible from y. Then x ϵ-sees v and any point to the right of v ϵ-visible from y.*

By this fact, p^*_{i+1} is ϵ-visible from $q_i \in E_i$ (since it is visible from $p^*_i \in E_i$ and $q_i \succeq_\epsilon p^*_i$). Thus, $p^*_{i+1} \in E_{i+1}$. ■

■

The algorithm is easily seen to run in time $O(n^2)$, where $n = |P(A)|$.

5.2.3 Best k solutions

In this subsection, we consider the "dual" versions of the problems $\mathcal{Q}_\mathrm{C}$ and $\mathcal{Q}_\mathrm{D}$. Namely, we are given as input a discrete set of points $A \subseteq \mathbb{R}^2_+$ and a positive integer k and we want to compute a set of (at most) k points that form an ϵ-convex Pareto set with minimum ϵ. Similarly, we consider the following two variants of the dual problem:

- Problem $\mathcal{D}_\mathrm{D}(A, k)$: Given as input a discrete set of points $A \subseteq \mathbb{R}^2_+$ and a positive integer k, compute a set $S \subseteq A$ of cardinality at most k that forms an ϵ-convex Pareto set with minimum ϵ.
- Problem $\mathcal{D}_\mathrm{C}(A, k)$: Given as input a discrete set of points $A \subseteq \mathbb{R}^2_+$ and a positive integer k, compute a set $S \subseteq \mathcal{CH}(A)$ of cardinality at most k that forms an ϵ-convex Pareto set with minimum ϵ.

We analyze each problem separately.

Problem $\mathcal{D}_\mathrm{D}(A, k)$: Denote by $P(A) = \{p_1, \dots, p_n\}$ the Pareto set of A. As in problem $\mathcal{Q}_\mathrm{D}(A, \epsilon)$, it suffices to select points of $P(A)$ for the approximation. Let ϵ^* be the optimal value of the error tolerance ϵ and $\rho^* = 1 + \epsilon^*$ the corresponding value of the ratio.

A first observation is that there exist only polynomially many possible values for ρ^*; in particular, $O(n^2)$ values, where $n = |P(A)|$. This holds for the following reason: Consider a feasible solution to the problem, i.e. a set $S = \{s_1, \dots, s_k\} \subseteq A$. The set S forms an ϵ_S-convex Pareto curve of A where $\rho_S \equiv 1 + \epsilon_S = \max\{\mathcal{RD}(s_1, p_1), \mathcal{RD}(s_k, p_n), \max\{\mathcal{RD}(\overline{s_i s_{i+1}}, q) \mid i \in [n-1], q \in A, x(s_i) \leq x(q) \leq x(s_{i+1})\}\}$. It is easy to see that there are $O(n^2)$ such values for all possible combinations of sets S. Let $\epsilon_1 < \epsilon_2 < \ldots <$ be the candidate values of the error.

We can compute the optimal ϵ value by a binary search procedure on the ϵ_i's using the algorithm for the problem $\mathcal{Q}_{\mathrm{D}}(A, \epsilon)$ as the decision procedure. More specifically, for a given ϵ_i we call $\mathcal{Q}_{\mathrm{D}}(A, \epsilon_i)$. If the optimal solution has more than k points, then we conclude that $\epsilon^* > \epsilon_i$, otherwise, $\epsilon^* \leq \epsilon_i$ and so on. The algorithm terminates when we find an i^* such that the solution returned by $\mathcal{Q}_{\mathrm{D}}(A, \epsilon_{i^*})$ has at most k points while $\mathcal{Q}_{\mathrm{D}}(A, \epsilon_{i^*+1})$ returns a solution with more than k points.

The enumeration of the ϵ_i's takes $O(n^2)$ time and the binary search procedure can be done in $O(n^2 \log n)$ time. Thus, the overall time complexity is $O(n^2 \log n)$.

Problem $\mathcal{D}_{\mathrm{C}}(A, k)$: Denote by $CP(A) = \{p_1, \ldots, p_n\}$ the convex Pareto set of A. As in problem $\mathcal{Q}_{\mathrm{C}}(A, \epsilon)$, it suffices to select points of $\mathrm{LE}(A)$ for the approximation.

This version of the problem is a bit trickier. Now a feasible solution $S = \{s_1, \ldots, s_k\}$ can select points from the continuous set $\mathrm{LE}(A)$. Thus, the above approach does not immediately apply. We can first compute a lower bound ϵ_{LB} for ϵ^* by solving $\mathcal{D}_{\mathrm{D}}(CP(A), 2k)$ and an upper bound ϵ_{UB} by solving $\mathcal{D}_{\mathrm{D}}(CP(A), k)$. We can then do a similar binary search procedure between these values to compute ϵ^*, using the algorithm for the problem $\mathcal{Q}_{\mathrm{C}}(A, \epsilon)$ as the decision procedure. The search terminates when we restrict the candidate values of ϵ in a “small enough” interval such that we “know that nothing can change” in this interval. This procedure will run in polynomial time because of Lemma 5.2.10. This essentially means that we do not need to consider a continuum of points, but only a finite subset with bit complexity polynomial in the size of the input.

5.3 Two Objectives – General Results

In this section we consider implicitly represented instances and give generic algorithms and lower bounds for the problem of computing a minimum cardinality ϵ-convex Pareto set. In Section 5.3.1, we consider the special case that an exact Comb routine is available and Section 5.3.2 considers the general case of an approximate routine.

5.3.1 Exact Comb routine

In this section we analyze the case that a polynomial exact Comb routine is available. This special case is particularly interesting, because it includes many problems of interest; for example, bi-objective linear programming and all linear discrete bi-objective problems whose single objective version is tractable (e.g. bi-objective shortest path, spanning tree, matching, $s-t$ min-cut, etc.) belong to this class.

We first give a lower bound showing that no generic algorithm can guarantee a ratio better than 2. We then give a matching upper bound; we exhibit a generic algorithm that guarantees ratio 2 and applies to all such problems with a discrete objective space. For continuous problems with a polyhedral objective space we can compute an ϵ-convex Pareto set of minimum cardinality. In particular, for the case of bi-objective linear programming we give an optimal algorithm that is in addition very efficient.

5.3.1.1 Lower Bound for Discrete Case

Computing the smallest ϵ-convex Pareto set is typically NP-hard for common problems even for two objectives. The following proposition is an illustration of this fact.

Consider the bi-objective Shortest Path (*BSP*) problem: we are given an undirected graph $G = (V, E)$, rational "costs" $c(e)$ and "delays" $d(e)$ for each edge $e \in E$ and two specified nodes s and t. The set of feasible solutions is the set of $s-t$ paths; the goal is to minimize cost and delay.

Proposition 5.3.1. *For the BSP problem, for any k from $k = 1$ to a polynomial, it is NP-hard to distinguish the case that the minimum size of the optimal ϵ-convex Pareto set is k from the case that it is $k+1$. In particular, no polynomial time algorithm can approximate the size of the smallest ϵ-convex Pareto set to a factor better than 2 (unless P = NP).*

Proof. The proof is by an adaptation of Theorem 3.1.2. showing that it is NP-hard to distinguish the case that the size of the smallest ϵ-*Pareto set* for this problem is k from the case it is $2k-1$.

The reduction is from the Partition problem [GJ]; given a set A of n positive integers $A = \{a_1, a_2, \ldots, a_n\}$, we wish to determine whether it is possible to partition A into two subsets with equal sum. Given an instance of Partition, we construct an instance of the *BSP* problem as follows: Let G be a graph with $n+1$ nodes v_i and $2n$ edges $\{e_i, e'_i\}$; the pair of (parallel) edges $\{e_i, e'_i\}$ has endpoints v_i and v_{i+1}. Consider k disjoint copies of the graph G, $G^j = (V^j, E^j)$, $j \in [k]$, with $V^j = \{v_i^j\}_{i=1}^{n+1}$ and $E^j = \{e_i^j, e'^j_i\}_{i=1}^{n}$. Add a (source) node s and a (sink) node t; for each j add an edge from s to v_1^j and one from v_{n+1}^j to t. Assign zero cost and delay to each edge incident to s or t and set: $(1+2\epsilon)^{2(j-1)}c(e_i^j) = d(e'^j_i)/(1+2\epsilon)^{2(j-1)} = S + 2\epsilon a_i n$ and $(1+2\epsilon)^{2(j-1)}c(e'^j_i) = d(e_i^j)/(1+2\epsilon)^{2(j-1)} = S$. Let H denote the resulting edge weighted graph.

It is easy to see that the Pareto set for this instance is the union of k disjoint "clusters"; the points in each cluster lie on the line segment $l_j r_j$, where $l_j = \big(Sn/(1+2\epsilon)^{2(j-1)}, Sn(1+2\epsilon)^{2(j-1)+1}\big)$ and $r_j = \big(Sn/(1+2\epsilon)^{2(j-1)-1}, Sn(1+2\epsilon)^{2(j-1)}\big)$. Notice that no $s-t$ path using graph G^j is $(1+\epsilon)$-covered by any $s-t$ path using graph G^l for $j \neq l$. Thus, any ϵ-Pareto set for this instance must contain *at least* k points. Also observe that the points l_j and r_j $(1+\epsilon)$-cover the jth cluster. Thus, $2k$ points suffice to $(1+\epsilon)$-cover the feasible set. Now the jth cluster is $(1+\epsilon)$-covered by one point iff there exists an $s-t$ path with coordinates $m_j = \big((1+\epsilon)Sn/(1+2\epsilon)^{2(j-1)}, (1+\epsilon)Sn(1+2\epsilon)^{2(j-1)}\big)$ which in turn holds iff the original Partition instance is a Yes instance. (This also implies that there exists an ϵ-Pareto set with at most $2k-1$ points iff there exists an ϵ-Pareto set with exactly k points.) So, if there exists a partition of the set A, the smallest ϵ-Pareto set contains exactly k points. Otherwise, the smallest such set must contain $2k$ points.

Notice that the case of $k=1$ of Proposition 5.3.1 directly follows from the aforementioned. To show that the result holds also for more general k, we need to modify the above construction as follows. Consider the graph H as defined above. The idea of the proof is the following: Add one solution point ($s-t$ path) between any two adjacent clusters satisfying the following two properties: (i) Its *ratio distance* from the line segment connecting the "midpoints" of the clusters is equal to $(1+\epsilon)$. (ii) The Pareto set of the new instance contains all the previous points plus the added ones. It is not difficult to verify that the above conditions can be simultaneously achieved, if ϵ is small enough. For the augmented instance, if all the midpoints correspond to solution points, they constitute an ϵ-convex Pareto set. If this is not the case, we can inductively argue that any ϵ-convex Pareto set must contain at least $k+1$ points. Therefore, it is NP-hard to decide whether the smallest ϵ-convex Pareto set has k or $k+1$ points. ■

The NP-hardness also holds for bi-objective spanning tree and many other common problems. We note that the above proof does not rule out the possibility of an asymptotic approximation scheme for this specific problem. Identifying the exact limit of approximation is an interesting question.

In general however, no generic algorithm can distinguish whether the optimal set has k points or $2k-1$ points (for any k), because of the following:

Theorem 5.3.2. *For the bi-objective discrete setting, no generic algorithm having oracle access to* Comb *for constructing a small ϵ-convex Pareto set can be better than 2-competitive.*

Proof. The proof is based on the fact that, by using the Comb primitive as a black box, we cannot access solution points that do not lie in the convex Pareto set.

The idea of the proof is the following: Given an error $\epsilon > 0$ and a positive integer k, we will exhibit the existence of a set $S = \{p_i, q_i, r_i \mid i \in [k]\} \subseteq \mathbb{R}^2_+$ (of $3k$ points) such that the following three conditions are simultaneously satisfied: (i) $CP(S) = \{p_i, r_i \mid i \in [k]\}$ (ii) $CP^*_{\mathrm{C},R}(S, \epsilon) = CP(S)$ and (iii) $CP^*_{\mathrm{D}}(S, \epsilon) = \{q_i \mid i \in [k]\}$. These conditions say that the smallest ϵ-convex Pareto set that uses points *only* from $CP(S)$ (i.e. solution points accessible with our primitive) contains exactly $2k$ points (in fact all the points in $CP(S)$), while the (globally) smallest ϵ-convex Pareto set contains exactly k points (none of which is in $CP(S)$).

We now translate the aforementioned into a set of geometric constraints. First, the points of $CP(S)$ are ordered as follows: For $i \in [k-1]$, p_i lies to the left and above r_i which lies to the left and above p_{i+1}. Recall that the points of $CP(S)$ are the vertices of a strictly decreasing convex polygonal chain. For $i \in [k]$, q_i lies in the interior of the line segment $\overline{p_i r_i}$. The point q_1 has $x(q_1) = (1+\epsilon)x(p_1)$. Similarly, $y(q_k) = (1+\epsilon)y(r_k)$. Now the line $\overline{q_i q_{i+1}}$ is parallel to $\overline{r_i p_{i+1}}$ and the ratio distance of both r_i and p_{i+1} from the former line is exactly $(1+\epsilon)$. On the other hand, the ratio distance of r_i from the line $\overline{p_i p_{i+1}}$ is (strictly) more than $(1+\epsilon)$. Similarly, the ratio distance of p_{i+1} from $\overline{r_i r_{i+1}}$ is (strictly) more than $(1+\epsilon)$. It is easy to see that, if the above geometric constraints are satisfied, the three conditions of the previous paragraph are met. Thus, it suffices to show that there exists a set of points in the plane satisfying these geometric constraints.

It does not seem easy to give general closed forms for the coordinates of the points in S, even for $k = 2$. However, it is sufficient to show that a set of points with the aforementioned properties exists for any given $\epsilon > 0$ and k. We next argue about the existence of an instance with the desired properties for $k = 2$. The case of general k can be shown by induction.

By a geometric argument, we show that for $k = 2$, for any given $\epsilon > 0$, there exists a family of instances in the plane satisfying the desired constraints. Given ϵ, we first select the points r_1 and p_2 such that $y(r_1) > (1+\epsilon)y(p_2)$ and $x(p_2) > (1+\epsilon)x(r_1)$. Now we consider the line l_0 parallel to $\overline{r_1 p_2}$ and scaled by $(1+\epsilon)$; q_1 and q_2 are points of l_0. It remains to explain how we select the slopes λ of the line $\overline{p_1 q_1 r_1}$ and λ' of $\overline{p_2 q_2 r_2}$. We give the argument for λ. The case of λ' is symmetric.

Intuitively, if $|\lambda|$ is large enough, then the desired conditions should hold. Indeed, consider the following constraints: (i) $x(p_1) = x(q_1)/(1+\epsilon)$ (ii) q_1 is the intersection of $\mathbf{l}_0$ and $\overline{p_1 q_1 r_1}$ and (iii) r_1's ratio distance from $\overline{p_1 p_2}$ is larger than $(1+\epsilon)$. It can be shown algebraically that these constraints are satisfied for $\lambda < \lambda_0$, where $\lambda_0 < 0$ is the negative root of a second order equation. Choosing an appropriate value for λ determines the coordinates of q_1 and p_1.

A useful illustration of the described construction is provided in Figure 5.2. ■

It is interesting to identify natural bi-objective problems for which it is NP-hard to distinguish whether the optimal set has k points or $2k-1$ points (for general k). We conjecture that the bi-objective $s-t$ min-cut problem has this property (for small enough ϵ).

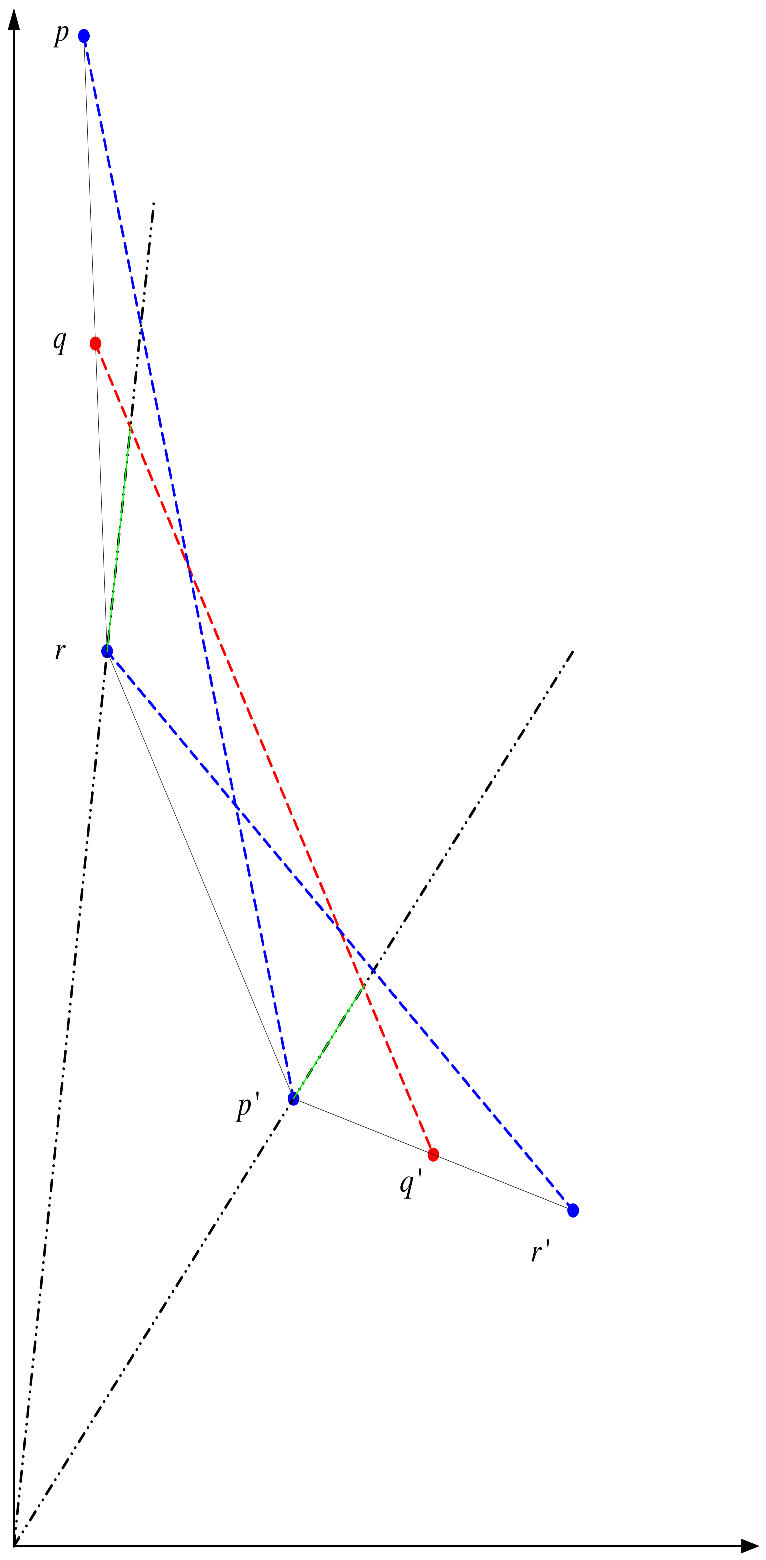

Figure 5.2: Illustration of factor 2 lower bound for exact Comb ($k = 2$).

5.3.1.2 Upper Bounds

In this section, we give an efficient algorithm that computes an ϵ-convex Pareto set of size at most twice the optimal. The algorithm applies to all *discrete* bi-objective problems that possess an exact Comb routine. We moreover show that, if the objective space is polyhedral and the corresponding vertices have polynomial bit complexity, we can efficiently compute an ϵ-convex Pareto set of minimum cardinality. For the important special case of bi-objective linear programming, we describe an implementation of our generic scheme that is more efficient in a precise way.

Discrete Problems: The idea of the generic algorithm is to appropriately use the routine so as to simulate the algorithm for problem $\mathcal{Q}_{\mathrm{D}}$ (i.e. with explicitly given points). As noted in the previous section, this is not exactly possible, since, using the given routine as a black box, we cannot access the solution points that do not lie in the convex Pareto set of the objective space. However, as noted before, if we ignore such points we do not lose more than a factor of 2 in the approximation. With that in mind, the generic algorithm outputs a set of points $Q = \{q_1, q_2, \dots, q_r\}$ as follows: We first compute the leftmost and rightmost points of the convex Pareto set. Let us denote these points by p_{l} and p_{r} respectively. This can be easily done, since our routine is exact. If these two points do not constitute an ϵ-convex Pareto set, we select as q_1 the rightmost solution point in *the convex Pareto set* that lies at most $(1 + \epsilon)$ to the right of p_{l}. The remaining points are selected by the following iterative procedure: The point q_{i+1} is the rightmost point of the convex Pareto set that is ϵ-visible from q_i. To achieve the simulation of visibility we use a "binary search procedure on the slopes" with an application of the given routine at each step of the search. In particular, the algorithm uses an operation Next(q_i) that computes the next point to be selected, i.e. the rightmost point of the convex Pareto set that is ϵ-visible from q_i.; for uniformity, we write $q_1 = \mathrm{Next}()$ for the first point selected by the algorithm. We elaborate on this procedure and prove its polynomial running time in Lemma 5.3.4.

The algorithm is described in pseudo-code in Table 5.3:

Theorem 5.3.3. *For any discrete bi-objective optimization problem possessing an exact* Comb *routine, for any $\epsilon > 0$, the algorithm of Table 5.3 computes a 2-approximation to the smallest ϵ - convex Pareto set in polynomial time.*

Proof. It suffices to argue that the algorithm faithfully simulates the "relaxed" algorithm (i.e. the

```
Algorithm Exact Comb–Discrete–2D($\Pi, I, \epsilon$).
Input: (Discrete) Problem $\Pi$, instance $I \in \mathcal{I}_\Pi$ and $\epsilon > 0$.
Output: Set of solution points $Q = \{q_1, \dots, q_r\}$ ($\epsilon$-convex Pareto for $\mathcal{S}(I)$).

$m = p_\Pi(|I|)$; $\lambda_{\max} = 2^{2m}$; $\lambda_{\min} = 2^{-2m}$;
$p_{\rm l} = \mathrm{Comb}(\lambda_{\max})$; $p_{\rm r} = \mathrm{Comb}(\lambda_{\min})$;
If $\big(y(p_{\rm l}) \le (1+\epsilon) y(p_{\rm r})\big)$ then { $Q = \{p_{\rm l}\}$; halt }.
If $\big(x(p_{\rm r}) \le (1+\epsilon) x(p_{\rm l})\big)$ then { $Q = \{p_{\rm r}\}$; halt }.
$\lambda_0 =$ absolute slope of $\overline{p_{\rm l} p_{\rm r}}$; $r = \mathrm{Comb}(\lambda_0)$;
If $\big(\mathcal{RD}(\overline{p_{\rm l} p_{\rm r}}, r) \le (1+\epsilon)\big)$ then { $Q = \{p_{\rm l}, p_{\rm r}\}$; halt. }
$q_1 =$ Next();
$Q = \{q_1\}$; $i = 1$;
While $\big(y(q_i) > (1+\epsilon) y(p_{\rm r})\big)$ do
{ $q_{i+1} =$ Next($q_i$);
  $Q = Q \cup \{q_{i+1}\}$;
  $i = i + 1$; }
```

Table 5.3: Algorithm for two dimensional discrete convex Pareto approximation (exact Comb).

one that ignores the points that do not lie in $CP(I)$) and that it runs in polynomial time. Having shown this, it follows from the analysis of the previous section that it achieves a factor 2 approximation to the smallest ϵ-convex Pareto set.

First, it is clear that the solution points p_{l} and p_{r} are the leftmost and rightmost points of the convex Pareto set $CP(I)$, for the given instance I. We first check whether either or both these points constitute an ϵ-convex Pareto set, in which case we output the appropriate set and halt. Otherwise, we call the procedure Next() to find the rightmost point $q_1 \in CP(I)$ that $(1+\epsilon)$-covers p_{l}. If q_1 $(1+\epsilon)$-covers p_{r}, we output q_1 and halt. Otherwise, for $i \geq 1$, we call Next(q_i) to find the rightmost vertex $q_{i+1} \in CP(I)$ that is ϵ-visible from q_i. If q_{i+1} $(1+\epsilon)$-covers p_{r}, we terminate the algorithm. It is thus easy to see that the algorithm is a 2-approximation.

The following lemma describes the operation Next(), establishes the fact that it can be implemented to run in polynomial time and thus completes the proof. We remark here that the lemma crucially uses the fact that the underlying problem Π is a *discrete combinatorial* problem.

Lemma 5.3.4. *The operation Next(q_i) can be implemented to run in time polynomial in $|I|$ and $1/\epsilon$.*

Proof. The operation is implemented by using a binary search procedure on the slopes (in the objective space) with an application of Comb at each step of the search. Since the problem Π is discrete, it follows that the (absolute value of the) slope of each edge of the lower envelope is a rational number with at most $2m$ bits, where $m = p_{\Pi}(|I|)$; hence, it is in the interval $(2^{-2m}, 2^{2m})$. Also, the absolute value of the difference between any two such slopes is lower bounded by 2^{-2m}. These facts are crucial because they guarantee that the binary search performed on the value of the slope will terminate after a polynomial number of steps.

Let us first describe how to implement Next() (i.e. the selection of the leftmost point $q_1 \in Q$); this operation computes the rightmost *vertex* q_1 of the convex Pareto set $CP(I)$ that satisfies $x(q_1) \leq (1+\epsilon)x(p_{\mathrm{l}})$. To find q_1 we do a binary search on (the absolute value of the slope) $\lambda \in [2^{-2m}, 2^{2m}]$. At each step of the search, we call $\mathrm{Comb}(\lambda)$ to compute a solution point p_λ that minimizes the linear objective function $(\lambda, 1)$. Note that p_λ necessarily lies on the lower envelope, but is not necessarily a vertex in $CP(I)$. The search terminates the first time we find two points $q_{\mathrm{YES}} = \mathrm{Comb}(\lambda_{\mathrm{YES}})$ with $x(q_{\mathrm{YES}}) \leq (1+\epsilon)x(p_{\mathrm{l}})$ and $q_{\mathrm{NO}} = \mathrm{Comb}(\lambda_{\mathrm{NO}})$ with $x(q_{\mathrm{NO}}) > (1+\epsilon)x(p_{\mathrm{l}})$ such that $|\lambda_{\mathrm{YES}} - \lambda_{\mathrm{NO}}| < 2^{-2m}$. This is clearly achieved after a polynomial number of calls to the routine. By the discreteness of the space, the solution points $q_{\text{YES}}, q_{\text{NO}}$ will be adjacent vertices in $CP(I)$; we therefore set $q_1 = q_{\text{YES}}$.

The implementation of Next(q_i) is similar. The main difference is in the termination criterion of the binary search procedure. To wit, suppose we have computed the solution points $q_1, \ldots, q_i$ and we call Next(q_i) to compute the point q_{i+1}. We do a binary search on $\lambda \in (\lambda_i, 2^{2m}]$, where λ_i is the terminating value of the absolute slope from the previous iteration. Similarly, at every step of the search, we call $\mathrm{Comb}(\lambda)$ to compute the solution point p_λ that minimizes the linear objective $(\lambda, 1)$. Consider the line segment $\overline{q_i p_\lambda}$ and the absolute value λ' of its slope. By definition, the point $r = \mathrm{Comb}(\lambda')$ is a solution whose *ratio distance* from $\overline{q_i p_\lambda}$ is *maximum* among all solution points with x-objective value between q_i and p_λ. That is, the solution point p_λ is ϵ-visible from q_i if and only if $\mathcal{RD}(\overline{q_i p_\lambda}, r) \leq 1 + \epsilon$. This criterion guides the binary search which terminates the first time that we find solution points q_{YES} and q_{NO} such that q_{YES} is ϵ-visible from q_i, q_{NO} is not and $|\lambda_{\text{YES}} - \lambda_{\text{NO}}| < 2^{-2m}$. Similarly, by the discreteness of the space, the aforementioned is achieved after a polynomial number of calls to the Comb routine. Moreover, the solution points $q_{\text{YES}}, q_{\text{NO}}$ can be assumed without loss of generality to be adjacent vertices in $CP(I)$; hence, we set $q_{i+1} = q_{\text{YES}}$. ■

■

We remark that, even though the above generic scheme runs in polynomial time, it is not very efficient. An interesting research problem is to obtain more efficient algorithms for important combinatorial problems such as shortest paths, spanning trees, etc.

Convex Problems: If the objective space is convex, in particular a *convex polytope* (polygon) whose vertices have polynomial bit complexity, then we can compute in polynomial time an ϵ-convex Pareto set of minimum cardinality by an adaptation of the aforementioned scheme. We note that this class of problems includes bi-objective linear programming; however, our generic algorithm does not assume linearity of the objective functions.

The algorithm is similar in spirit to the previous case: It faithfully simulates the exact algorithm for problem $\mathcal{Q}_{\text{C}}$. For the simulation we use a similar binary search procedure as the one described in Lemma 5.3.4. The main difference is that after we compute the adjacent vertices q_{YES} and q_{NO} such that q_{YES} is ϵ-visible from q_i and q_{NO} is not, we compute the rightmost convex combination

q^* of q_{YES} and q_{NO} that is ϵ-visible from q_i. By convexity of the objective space, we know that q^* corresponds to a solution point and we set $q_{i+1} = q^*$.

Lemma 5.2.10 guarantees that the points q^* selected in this manner have polynomial bit complexity (in $|I|$ and $1/\epsilon$) – assuming that the extreme points in the objective space have polynomial bit complexity – given that the size of the smallest ϵ-convex Pareto set is polynomial in $|I|$ and $1/\epsilon$. Hence, q^* can be computed in polynomial time. We thus have the following:

Theorem 5.3.5. *For any bi-objective optimization problem with polyhedral objective space possessing an exact* Comb *routine, for any $\epsilon > 0$, we can compute in polynomial time a set of* solution points *that form an ϵ - convex Pareto set of minimum cardinality.*

Remark 5.3.6. As stated in the theorem above, in this setting, we can compute a set of *solution points* that form an ϵ - convex Pareto set of minimum cardinality. However, since the objective functions are black-boxes, there is no general way to efficiently compute a solution $s_i \in \mathcal{S}(I)$ such that $\mathbf{f}(s_i) = q_i$. In this sense, the lower bound of 2 shown in the previous section also holds for the class of convex problems considered here. However, if the *solution space* is a convex polytope and the objectives are *linear*, the convex combination of solutions corresponding to the adjacent vertices q_{YES} and q_{NO} is a feasible solution corresponding to q_i. This immediately yields a polynomial time algorithm to compute an optimal ϵ-convex Pareto set of minimum cardinality for *bi-objective linear programming* (and all problems reducible to it such as flow problems for example). Since this problem is of particular interest, we give a more efficient construction of an optimal ϵ-convex Pareto set in the following subsection.

Bi-objective Linear Programming: An important special case of the continuous space setting is Bi-objective Linear Programming. Even though there has been extensive work on this problem, we are not aware of any optimal approximation algorithm. Many existing algorithms use variations of an intuitive approach which starts with the segment connecting the leftmost and rightmost points of the Pareto curve (which can be found by LP), and then iteratively refines the current polygonal line by introducing additional points that optimize judiciously chosen linear combinations of the objectives, until the desired approximation error ϵ is achieved. Various versions have been introduced under various names (ES, sandwich method, chord rule). Such a method has been studied analytically in [RF] in the context of the bi-objective min cost flow problem (a special case of LP).

They use the same multiplicative metric for the approximation error ϵ as we do, and show that the number of points generated is at most pseudopolynomial in the size of the instance and $1/\epsilon$; they do not compare it with the size of the minimum ϵ-CP. With a slightly more careful analysis one can show actually that the number of points is polynomial, however it is suboptimal and, as shown in Chapter 6, it is not within any constant factor of OPT.

Theorem 5.3.5 implies that we can compute the optimal ϵ-convex Pareto set for bi-objective LP in polynomial time. We show now that we can in fact do it using essentially 2 LP calls per generated point.

Theorem 5.3.7. *For the case of bi-objective linear programs, we can efficiently compute the optimal ϵ-convex Pareto set by solving a linear number of LP's. In particular, if k is the size of the optimal ϵ-convex Pareto set, our algorithm solves $2k+3$ LP's whose size is of the same order as the size of the initial LP.*

Proof. We give the proof for the case of minimization objectives, the extension to maximization or mixed objectives being straightforward.

Given a bi-objective linear program with two minimization objectives, we describe an algorithm to compute a set of solutions that form an optimal ϵ-convex Pareto set. The described algorithm is efficient in a precise way: Let k be the cardinality of the optimal ϵ-convex Pareto set. The algorithm involves the solution of $O(k)$ (single-objective) linear programs whose size is roughly the same as the size of the linear program under consideration.

To simplify the exposition, let us assume that we have a compact decision space defined by:

$$\mathcal{Z} = \{z \in \mathbb{R}^{n\times 1} \mid A \cdot z \geq b, z \geq \mathbf{0}_{n\times 1}\}$$

where $A \in \mathbb{Q}^{m\times n}$ and $b \in \mathbb{Q}^{1\times m}$. We have two minimization objectives defined by the vectors $c, d \in \mathbb{R}^{1\times n}$. Hence, the objective space is given by:

$$\mathfrak{X} = \{(x, y) \in \mathbb{R}^2_+ \mid x = c \cdot z, y = d \cdot z, z \in \mathcal{Z}\}$$

We need to assume that the objective values are constrained to be positive so that approximation makes sense. Note that the objective space $\mathfrak{X}$ is a convex polygon.

The overall framework of the algorithm is the one described above for convex problems. We select a set of solution points $Q = \{q_1, \ldots, q_k\}$ in the objective space "left to right" such that q_{i+1} is the rightmost solution point that is ϵ-visible from q_i. However, the special structure of the considered problem allows for a *much more efficient* implementation of the Next operation, as compared to the generic binary search procedure described above. In particular, we do not use binary search at all, but we directly simulate the explicit optimal algorithm. We next describe how this can be achieved.

A first ingredient is the computation of the points p_{l} and p_{r}, the leftmost and rightmost points of the (convex) Pareto set respectively. We describe how to compute p_{l}, the computation of p_{r} being symmetric. This can be done by solving two linear programs; the first one to compute $x(p_{\mathrm{l}})$ and the second one to compute $y(p_{\mathrm{l}})$. We first solve the LP: $\min x$ s.t. $\{x = c \cdot z, z \in \mathcal{Z}\}$. The solution of this LP gives the x-value of p_{l}, i.e. $x_{\min} = x(p_{\mathrm{l}})$. Having computed $x_{\min}$, the second LP is: $\min y$ s.t. $\{c \cdot z = x_{\min}, z \in \mathcal{Z}\}$ and gives $y(p_{\mathrm{l}})$. Similarly, the computation of the first point q_1 in the optimal approximate set, i.e. the point of LE having x-coordinate $x(q_1) = (1+\epsilon)x(p_{\mathrm{l}})$ can be easily done by solving one linear program; namely, $y(q_1)$ is the solution of the LP: $\min y$ s.t. $\{y = d \cdot z, c \cdot z = (1+\epsilon)x_{\min}, z \in \mathcal{Z}\}$.

The main ingredient of the algorithm is the efficient implementation of the Next operation: Given a point p on $P(\mathfrak{X})$, the Pareto curve of $\mathfrak{X}$, and $\epsilon > 0$, compute the rightmost point $q \in P(\mathfrak{X})$ that is ϵ-visible from p. We now show how to implement this operation by solving two linear programs of size roughly the same as the original one.

Recall that our ultimate goal is to compute the point q, given p and ϵ. Let $r^*x + y = t^*$, $r^*, t^* > 0$ be the equation of the line $\overline{pq}$. We will first identify this line, i.e. compute the values of the parameters r^* and t^*. Since the point p belongs to this line, it follows that $r^*x(p) + y(p) = t^*$ – note that $x(p)$, $y(p)$ are constants and r^*, t^* variables in this equation. By construction of the explicit optimal algorithm, r^* is the minimum absolute value of the slope such that for all solution points $(x, y) \in \mathfrak{X}$ it holds $r^*x + y \geq t^*/(1+\epsilon)$. In other words, we want the following implication to hold:

$$z \in \mathcal{Z} \Rightarrow r^*(c \cdot z) + (d \cdot z) \geq t^*/(1+\epsilon).$$

From duality theory, this implication holds iff there exists a vector $w \in \mathbb{R}_+^{1\times m}$ (the dual variables corresponding to the rows of A) such that $r^*c + d \geq wA$ and $w \cdot b \geq t^*/(1+\epsilon)$.

Therefore, we first solve the LP:

$$\begin{aligned}
&\min r^* \text{ s.t.} \\
r^* x(p) + y(p) &= t^* \\
r^* c + d &\geq wA \\
w \cdot b &\geq t^*/(1+\epsilon) \\
w &\geq \mathbf{0}_{1\times m} \\
r^* &\geq 0
\end{aligned}$$

The solution of this LP gives the equation of the line $\overline{pq}$. The point q is the solution point on this line with minimum y-value. To compute it, we thus solve the following LP:

$$\begin{aligned}
&\min y \text{ s.t.} \\
r^* x + y &= t^* \\
x = c \cdot z, y &= d \cdot z \\
z &\in \mathcal{Z}
\end{aligned}$$

Note that the x and y are now variables and r^*, t^* are the parameters of the line computed by the previous LP. The solution of this LP gives the coordinates of the point q. ■

Remark 5.3.8. We briefly mention that, in this case, we can solve the dual problem exactly by using the same binary search procedure that gives a PTAS in the next section.

5.3.2 Approximate Comb routine

In this section we give algorithms and lower bounds for the problem of computing a small ϵ-convex Pareto set that apply to all bi-objective problems (in our general framework) that have a polynomial approximate Comb_δ routine. Our results apply a fortiori if a GAP_δ routine is available. If this is the case, we point out that we can usually save a factor of 2 in the approximation.

We first point out that no generic algorithm can be better than 2-competitive in this setting, even if it has access to the (stronger) GAP_δ routine. Our main result is a generic algorithm that efficiently computes a 6-approximation to the optimal ϵ-convex Pareto set.

5.3.2.1 Lower Bounds

By adapting the argument in Theorem 5.3.2, we can show the following:

Theorem 5.3.9. *For two objectives, no generic algorithm having oracle access to* GAP_δ *for constructing a small ϵ-convex Pareto set can be better than 2-competitive.*

Proof. The proof of the lower bound is along the lines of Theorem 5.3.2. See Figure 5.3 for an illustration. Consider the instance $S = \{p_i, q_i, r_i \mid i \in [k]\}$ as described in the proof of Theorem 5.3.2 and its convex Pareto set $CP(S) = \{p_i, r_i \mid i \in [k]\}$. Now add the set of points $E = \{p'_i, r'_i \mid i \in [k]\}$, where each p'_i (resp. r'_i) is "slightly better" than p_i (resp. r_i) in both objectives, so that the GAP_δ routine cannot distinguish between these two points, unless δ becomes exponentially small in the size of the instance. If our input instance is the set S, then its smallest ϵ-convex Pareto set contains exactly k points (namely the q_i's). But, if our instance is the set $S \cup E$ then we claim that the smallest ϵ-convex Pareto set contains exactly $2k$ points. This is not difficult to see. It follows by inductively arguing that we must pick at least two points from each "cluster" to guarantee an ϵ-approximation. ∎

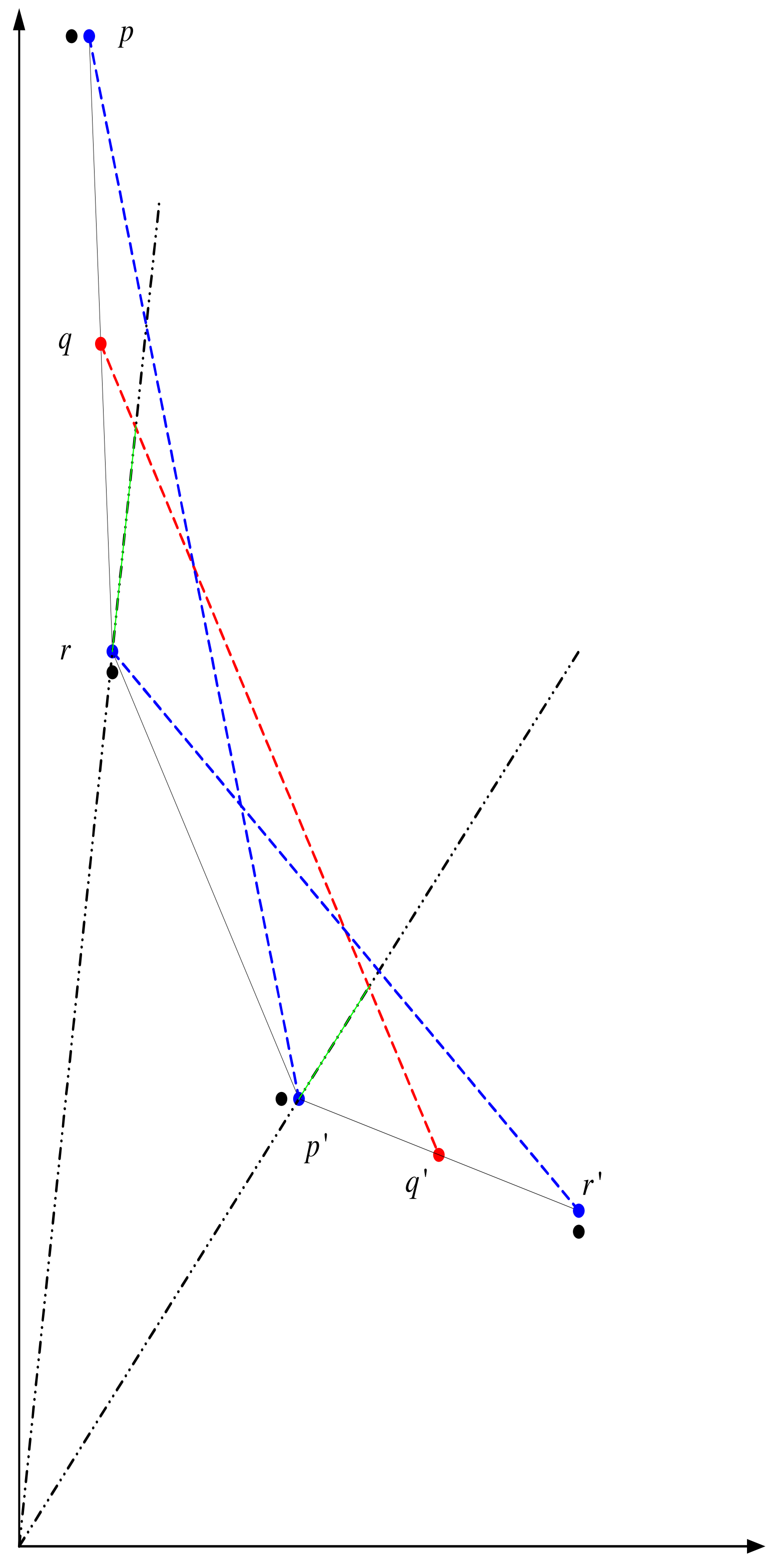

Figure 5.3: Illustration of factor 2 lower bound for generic algorithms with *approximate* GAP_δ.

5.3.2.2 Upper Bound

In this section we give our main algorithmic result for this chapter: a factor-6 approximation algorithm (factor-3 for the convex setting) for the problem of computing a minimum size ϵ-convex Pareto set, that is applicable to all discrete bi-objective problems that possess a polynomial Comb_δ routine. (We remark that if the GAP_δ routine is available, we can get a 3-approximation for discrete problems too.)

In this case – as opposed to the previous section, where the Comb routine is available exactly – we can access a scaled version of the lower envelope, rather than the lower envelope itself. So, in a sense, we need to somehow compare the size of the optimal ϵ'-CP to that of the optimal ϵ-CP, where $\epsilon' < \epsilon$.

The algorithm is based on the following geometric lemmas. The first one is for the convex setting and the second for the discrete.

Lemma 5.3.10 (convex objective space)**.** *Let $A \subseteq \mathbb{R}^2_+$ be a discrete set of points. For any $\epsilon > 0$ and any $\epsilon' > 0$ satisfying $1 + \epsilon' \geq \sqrt{1+\epsilon}$, we have: $|CP^*_{\mathrm{C}}(A, \epsilon')| \leq 3 \cdot |CP^*_{\mathrm{C}}(A, \epsilon)|$.*

Proof. Fix an optimal ϵ-CP set $CP^*_{\mathrm{C}}(A, \epsilon) = \{q^*_1, \dots, q^*_k\}$ constructed using the algorithm Explicit-Convex-2D (described in Section 5.2.1). We will establish the existence of an ϵ'-CP set $CP_{\mathrm{C}}(A, \epsilon') = \{p_1, \dots, p_r\}$ such that $r \leq 3k - 1$. To achieve this, we make essential use of the way the greedy algorithm Explicit-Convex-2D works. First, recall that $q^*_i \in \mathrm{LE}$, $i \in [k]$. To compute $CP^*_{\mathrm{C}}(A, \epsilon)$, we select (the *leftmost* point) q^*_1 as the point of LE having x-coordinate equal to $(1+\epsilon)x_{\min}$. Moreover, point q^*_{i+1} is the *rightmost* point of LE that is ϵ-visible from q^*_i, i.e. the line segment $\overline{q^*_i q^*_{i+1}}$ supports LE'_ϵ.

Consider the lower envelope $\mathrm{LE} = \mathrm{LE}(A)$ and its scaled versions – by factors of $(1 + \epsilon')$ and $(1 + \epsilon)$ respectively – $\mathrm{LE}'_{\epsilon'}$ and LE'_ϵ. We will prove the following:

Claim 5.3.11. *For all $i \in [k]$, there exists a set of solution points $CP_i = \{p_1, \dots, p_{3i-1}\}$ such that (i) p_1 $(1 + \epsilon')$-covers the leftmost point of A, (ii) any two consecutive points in CP_i are ϵ'-visible from each other and (iii) we have $p_{3i-1} \equiv q^*_i$.*

Lemma 5.3.10 follows from the above claim for $i = k$, i.e. if we set $CP_{\mathrm{C}}(A, \epsilon') = CP_k$. Indeed, CP_k is an ϵ'-CP set for A, due to Lemma 5.2.5, and it has cardinality (at most) $3k - 1$ by the claim above. To prove the claim, we proceed by induction on i.

Basis ($i = 1$): For the base case we need to select points p_1, p_2 that are ϵ'-visible from each other such that $x(p_1) \leq (1+\epsilon')x_{\min}$ and $x(p_2) \geq x(q_1^*)$, where $x_{\min} \doteq \min\{x(p) \mid p \in A\}$. Recall that q_1^* is the point of LE having $x(q_1^*) = (1+\epsilon)x_{\min}$. We select p_1 to be the point of LE having $x(p_1) = (1+\epsilon')x_{\min}$ and $p_2 = q_1^*$. We need to argue that q_1^* is ϵ'-visible from p_1. To wit: From the properties of visibility, it is clear that any point $q \in$ LE ϵ'-sees the point $q' \in$ LE that satisfies $x(q') = (1+\epsilon')x(q)$. From this we get the desired result because, by our choice of ϵ', $x(q_1^*) \leq (1+\epsilon')x(p_1)$.

Inductive step: Suppose the claim is true for some $i \in [k-1]$. We will show that it holds for $i+1$. In particular, we will select three points $p_{3i+j} \in$ LE, $j = 0, 1, 2$, – ordered by increasing x-coordinate – so that consecutive pairs are ϵ'-visible and such that $x(p_{3i+2}) \geq x(q_{i+1}^*)$. It is then clear that the set $CP_{i+1} = CP_i \cup \{p_{3i}, p_{3i+1}, p_{3i+2}\}$ satisfies the properties of Claim 5.3.11.

Let $\alpha \equiv q_i^*$ and $\beta \equiv q_{i+1}^*$. Consider the line defined by the segment $\overline{\alpha\beta}$. This line supports LE'_ϵ, i.e. it lies below it and there exists some vertex $v \in CP(A)$ (not necessarily unique) such that $\overline{\alpha\beta}$ intersects LE'_ϵ at the point $(1+\epsilon)v$. Also consider the line parallel to $\overline{\alpha\beta}$ point-wise scaled (down) by a factor of $(1+\epsilon')/(1+\epsilon)$. It is clear that this line supports $\text{LE}'_{\epsilon'}$ and that it goes through the point $(1+\epsilon')v$. Let α' and β' denote the intersection points of this scaled line with LE, so that $x(\alpha') < x(\beta')$. (See Figure 5.4 for an illustration.) We set $p_{3i} = \alpha'$, $p_{3i+1} = \beta'$ and $p_{3i+2} = q_{i+1}^*$. By construction, α' ϵ'-sees β'. The main part of the proof involves showing that $\alpha \equiv p_{3i-1}$ ϵ'-sees α' and β ϵ'-sees β'.

We now show, via a geometric argument, that α ϵ'-sees α', the argument for β and β' being symmetric. Consider the line segment $\overline{\alpha\alpha'}$. We claim that no point of $\text{LE}'_{\epsilon'}$ intersects this segment. (This clearly implies the desired result, since both α and α' belong to LE.) To see this, we observe that the points of $\text{LE}'_{\epsilon'}$ *eligible* to intersect $\overline{\alpha\alpha'}$ must have x-coordinate in the interval $[(1+\epsilon')x(\alpha), (1+\epsilon')x(\alpha')]$. (These points are drawn in gray color in Figure 5.4.) Note that these points of $\text{LE}'_{\epsilon'}$ are exactly the $(1+\epsilon')$-scaled versions of $\text{LE}(\alpha\alpha')$ – the subset of LE with endpoints α and α'. We denote this set by $(1+\epsilon') \cdot \text{LE}(\alpha\alpha')$. Clearly, the set of points $\text{LE}(\alpha\alpha')$ lies *above* the line $\overline{\alpha'\beta'}$. That is, if $\overline{\alpha'\beta'} = \{(x,y) \in \mathbb{R}^2 \mid c_1x + c_2y = 1, c_1, c_2 > 0\}$, then for each $s = (x(s), y(s)) \in \text{LE}(\alpha\alpha')$ we have $c_1x(s) + c_2y(s) \geq 1$. This in turn implies that its $(1+\epsilon')$-scaled version $s' = (1+\epsilon') \cdot s \in (1+\epsilon') \cdot \text{LE}(\alpha\alpha')$ satisfies $c_1x(s') + c_2y(s') \geq 1+\epsilon'$. Now, by

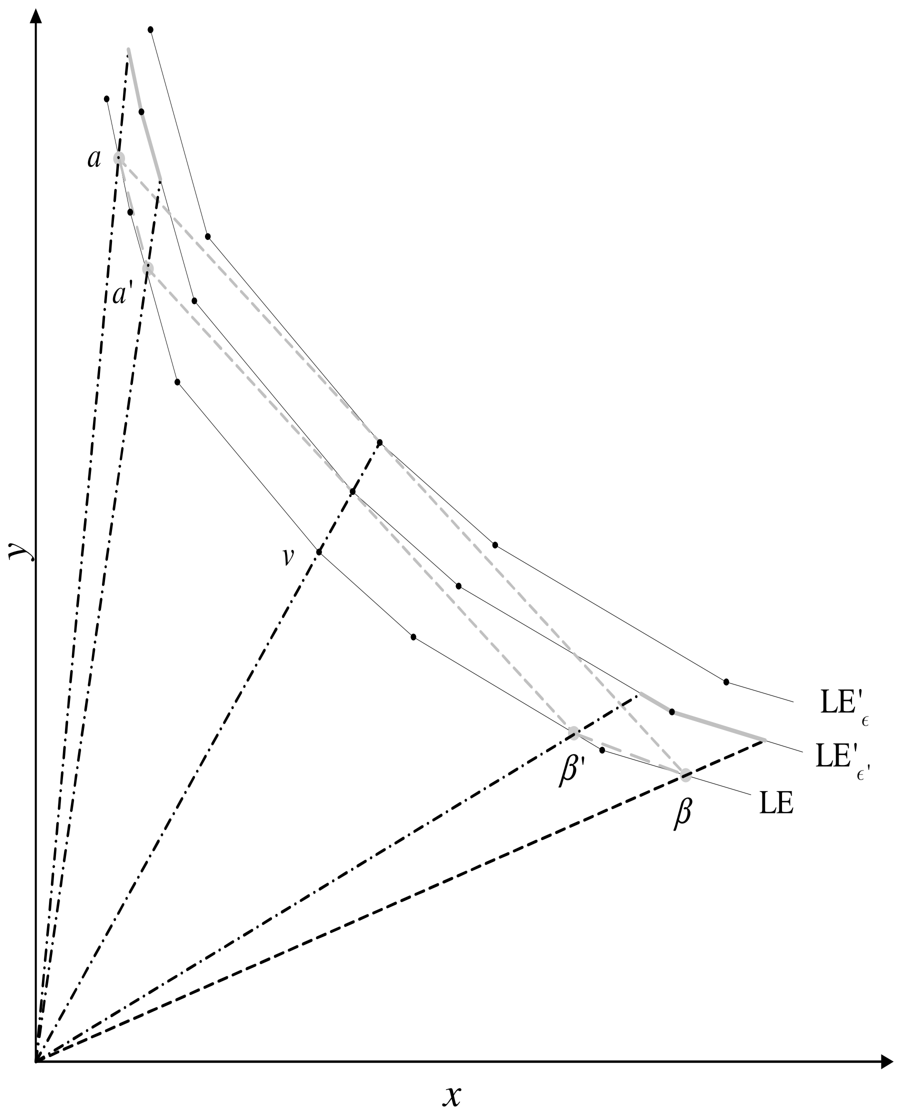

Figure 5.4: Illustration of (the proof of) Lemma 5.3.10. The figure clearly indicates that the pairs of points (α, α') and (β, β') are ϵ'-visible from each other with respect to $\mathrm{LE}'_{\epsilon'}$. The bold gray segments correspond to $(1+\epsilon') \cdot \mathrm{LE}(\alpha\alpha')$ and $(1+\epsilon') \cdot \mathrm{LE}(\beta\beta')$.

definition we have $\overline{\alpha\beta} = \{(x, y) \in \mathbb{R}^2 \mid c_1x + c_2y = (1+\epsilon)/(1+\epsilon')\}$ and $(1+\epsilon)/(1+\epsilon') \leq 1+\epsilon'$ by our choice of ϵ'. Therefore, the segment $(1+\epsilon') \cdot \mathrm{LE}(\alpha\alpha')$ lies above the line $\overline{\alpha\beta}$, which clearly implies these points do not intersect the line segment $\overline{\alpha\alpha'}$. This completes the proof of Claim 5.3.11 and hence also of Lemma 5.3.10. ■

Since for any $\epsilon > 0$ and any finite set $A \in \mathbb{R}^2_+$, it holds $|CP^*_{\mathrm{D}}(A,\epsilon)| \leq 2 \cdot |CP^*_{\mathrm{C}}(A,\epsilon)|$, the previous lemma directly implies that $|CP^*_{\mathrm{D}}(A,\epsilon')| \leq 6 \cdot |CP^*_{\mathrm{D}}(A,\epsilon)|$ as long as $\epsilon' \geq \sqrt{1+\epsilon} - 1$. However, we can directly prove a better bound by a discrete analogue of the geometric lemma.

Lemma 5.3.12 (discrete objective space)**.** *Let $A \subseteq \mathbb{R}^2_+$ be a discrete set of points. For any $\epsilon > 0$ and any $\epsilon' > 0$ satisfying $1 + \epsilon' \geq \sqrt{1+\epsilon}$, we have: $|CP^*_{\mathrm{D}}(A,\epsilon')| \leq 3 \cdot |CP^*_{\mathrm{D}}(A,\epsilon)|$.*

Proof. The proof is in the same spirit as the one for the convex case above. However, the discrete setting requires a more careful analysis. This is due to the fact that there is more freedom in the structure of the optimal solution. Recall for example that, in this case, the points of an optimal ϵ-CP set cannot be assumed to belong to the lower envelope.

We fix an optimal ϵ-CP set, denoted by $CP^*_{\mathrm{D}}(A,\epsilon) = \{q^*_1, \dots, q^*_k\}$, constructed using the algorithm Explicit-Discrete-2D (described in Section 5.2.2). We will establish the existence of an ϵ'-CP set, $CP_{\mathrm{D}}(A,\epsilon') = \{p_1, \dots, p_r\}$ such that $r \leq 3k$. We similarly proceed by an inductive argument that exploits the optimal algorithm Explicit-Discrete-2D. The main difference here, as compared to the convex case above, is the fact that the points q^*_i do not necessarily belong to LE. Moreover, it may be the case that these points lie above $\mathrm{LE}'_{\epsilon'}$. As a consequence, the inductive argument becomes more intricate. We prove the following (by induction on i):

Claim 5.3.13. *For all $i \in [k]$, there exists a set of solution points $CP_i = \{p_1, \dots, p_{3i}\}$ such that (i) p_1 $(1+\epsilon')$-covers the leftmost solution point, (ii) any two consecutive points in CP_i are ϵ'-visible from each other and (iii) we have either $p_{3i} = q^*_i$, in which case q^*_i lies below $\mathrm{LE}'_{\epsilon'}$, or p_{3i} is a vertex of* LE *to the right of q^*_i.*

Similarly, Lemma 5.3.12 follows from the above claim for $i = k$, i.e. if we set $CP_{\mathrm{D}}(A,\epsilon') = CP_k$. Indeed, CP_k is an ϵ'-CP set for A, due to Lemma 5.2.13, and it has cardinality (at most) $3k$ by the above claim.

We now proceed with the proof of Claim 5.3.13. The proof is by induction on i.

Basis ($i = 1$): Recall that q^*_1 satisfies $x(q^*_1) \leq (1+\epsilon)x_{\min}$, where $x_{\min} \doteq \min\{x(p) \mid p \in A\}$. Similarly, p_1 must satisfy $x(p_1) \leq (1+\epsilon')x_{\min}$; we pick p_1 to be the rightmost vertex of LE that satisfies this constraint. Clearly, p_1 ϵ'-sees the leftmost vertex v that lies to the right of the line $x = (1+\epsilon')x_{\min}$; we set $p_2 = v$. If $x(v) > (1+\epsilon)x_{\min}$, we are done. Otherwise, note that p_2 ϵ'-sees the rightmost vertex v' that lies to the left of the line $x = (1+\epsilon)x_{\min}$ (since $x(v') \leq (1+\epsilon')x(v)$).

Now, if $x(q_1^*) \leq x(v')$, we set $p_3 = v'$. (In fact, in such a case, we can assume that $q_1^* \equiv v'$, since $v' \in E_1(\epsilon)$ and $v' \succeq_\epsilon q_1^*$.) Now, if $x(v') < x(q_1^*) \leq (1+\epsilon)x_{\min}$, we set $p_3 = q_1^*$ and claim that q_1^* lies below $\mathrm{LE}'_{\epsilon'}$. For the sake of contradiction assume that this is not the case. Then, given that v' is the rightmost vertex in the interval $[(1+\epsilon')x_{\min}, (1+\epsilon)x_{\min}]$ we get that $y(q_1^*) > y(v')$, i.e. q_1^* is a dominated point, a contradiction. This completes the analysis of the base case.

Inductive step: Suppose the claim is true for $i \in [k-1]$. We will show it is also true for $i+1$. To do this, we will select three solution points p_{3i+j}, $j = 1, 2, 3$, – ordered by increasing x-coordinate – such that any two consecutive pairs are ϵ'-visible and the set $CP_{i+1} = CP_i \cup \{p_{3i+1}, p_{3i+2}, p_{3i+3}\}$ satisfies Claim 5.3.13.

Consider two consecutive points $q_i^*, q_{i+1}^* \in CP_{\mathrm{D}}^*(A, \epsilon)$. Recall that q_{i+1}^* is a best point ϵ-visible from q_i^*. Let us extend the line $\overline{q_i^* q_{i+1}^*}$ so that it intersects the lower envelope LE at the points α and β. Since q_i^* and q_{i+1}^* are ϵ-visible from each other, the line $\overline{\alpha\beta} \equiv \overline{q_i^* q_{i+1}^*}$ lies *below* LE'_ϵ. Also consider the line $\overline{\alpha'\beta'}$ constructed by scaling $\overline{\alpha\beta}$ point-wise by a factor $(1+\epsilon')/(1+\epsilon))$, where α' and β' are the intersection points with LE. By definition, α' ϵ'-sees β' and, by the argument of Lemma 5.3.10, we immediately get:

Fact 5.3.14. *The subsets* $(1+\epsilon') \cdot \mathrm{LE}(\alpha\alpha')$, $(1+\epsilon') \cdot \mathrm{LE}(\beta\beta')$ *of* $\mathrm{LE}'_{\epsilon'}$ *lie above the line* $\overline{\alpha\beta}$.

However, as already noted, the points α' and β' do not necessarily correspond to solution points; in general, we expect them to be interior points of some edge of LE. Now consider the adjacent vertices of the points α, α', β and β'. We use subscripts l and r to denote "left" and "right" respectively. Let O be the origin.

We select the three additional solution points as follows: We set $p_{3i+1} = \alpha'_r$ and $p_{3i+2} = \beta'_l$, while p_{3i+3} depends on the position of q_{i+1}^* with respect to the line $O\beta'$. If q_{i+1}^* lies to the right of this line, we set $p_{3i+3} = q_{i+1}^*$. Otherwise, we set $p_{3i+3} = \beta'_r$.

We now proceed with the analysis. It is first clear that p_{3i+1} and p_{3i+2} are ϵ'-visible from each other. We first need to show that α'_r is ϵ'-visible from p_{3i}. By the induction hypothesis, we have two subcases, depending on the position of p_{3i}, which we analyze in tandem. First, we observe that, by Fact 5.3.14 above, α ϵ'-sees α'_r. Indeed, the eligible points of $\mathrm{LE}'_{\epsilon'}$ to intersect the segment $\overline{\alpha\alpha'_r}$ are precisely the points of $(1+\epsilon') \cdot \mathrm{LE}(\alpha\alpha')$. (Note that we now use a stronger corollary of Fact 5.3.14, compared with the convex case.) We claim that if q_i^* is below $\mathrm{LE}'_{\epsilon'}$, q_i^* ϵ'-sees α'_r. This holds , essentially because q_i^* lies on the line $\overline{\alpha\beta}$. Formally, consider the segment $\overline{q_i^*\alpha_r'}$ and its intersection t with the line $O\alpha'$. It is clear that $\overline{q_i^*\alpha_r'}$ lies below $\overline{\alpha\beta}$ and thus below $(1+\epsilon')\cdot \mathrm{LE}(\alpha\alpha')$. So, $\overline{q_i^* t}$ lies below $\mathrm{LE}'_{\epsilon'}$ and so does $\overline{t\alpha_r'}$, since there are no vertices of $\mathrm{LE}'_{\epsilon'}$ in the corresponding interval. And in any case, α_r' is ϵ'-visible from any vertex to the right of q_i^* (hence, also to the right of α). Thus, in both subcases, we get that p_{3i} ϵ'-sees α_r'.

To complete the proof, it remains to argue that, if q_{i+1}^* lies to the right of $O\beta'$ then it is below $\mathrm{LE}'_{\epsilon'}$ and it ϵ'-sees β_l'. The first statement is implied by Fact 5.3.14, since q_{i+1}^* belongs to $\overline{\alpha\beta}$. The argument for the second statement is symmetric to the argument of the previous paragraph (showing that, as long as q_i^* lies below $\mathrm{LE}'_{\epsilon'}$, it ϵ'-sees α_r'). Finally, if q_{i+1}^* lies to the right of $O\beta'$, it is easy to see that $x(q_{i+1}^*) \leq x(\beta_r')$. Indeed, in this case, $y(\beta_r') < y(\beta) < y(q_{i+1}^*)$, so assuming otherwise would imply that q_{i+1}^* is a dominated point, a contradiction. This completes the proof of Claim 5.3.13 and hence of Lemma 5.3.12. ■

By exploiting the aforementioned lemmas, we can derive the following algorithmic result.

Theorem 5.3.15. *For any bi-objective problem possessing a polynomial (resp. fully polynomial)* Comb_δ *routine, for any* $\epsilon > 0$*, we can compute in polynomial (resp. fully polynomial) time a* 6*-approximation (*3 *in the convex case) to the smallest* ϵ *- convex Pareto set.*

Proof. We first describe the algorithm and analysis for the discrete case. The algorithm and analysis for the convex case is very similar.

We will describe an algorithm that uses a linear $O(m/\epsilon)$ calls to Comb, hence runs in polynomial time. Consider the following procedure:

1. Compute a δ-convex Pareto set R_δ of $\mathbf{f}(\mathcal{S}(I))$ by using the generic algorithm of Chapter 4 for $\delta = \sqrt[4]{1+\epsilon} - 1$ ($\approx \epsilon/4$ for small ϵ). If $\sqrt[4]{1+\epsilon}$ is not rational, then we pick δ to be a rational number satisfying $(1+\delta)^4 \leq 1+\epsilon$ and has bit representation $O(|\epsilon|)$.

2. Define ϵ', ϵ'' such that: $1+\epsilon = (1+\epsilon')(1+\delta)^2$ and $1+\epsilon'' = (1+\epsilon')(1+\delta)$. Use the algorithm for problem $\mathcal{Q}_{\mathrm{D}}$ to compute the smallest ϵ''-convex Pareto set Q of R_δ.

3. Output Q.

By construction of the generic algorithm of Chapter 4, the set R_δ is a δ-convex Pareto set (having no redundant points) for the (exponentially large or infinite) set of solution points (in the objective space) $\mathbf{f}(\mathcal{S}(I))$ and has size polynomial in the size of the instance $|I|$ and $1/\delta$. In the second step, we apply the algorithm EXPLICIT DISCRETE-2D to optimally solve the problem $\mathcal{Q}_{\mathrm{D}}(R_\delta, \epsilon'')$, i.e. to compute the smallest ϵ''-convex Pareto set $Q \subseteq R_\delta$ of R_δ. By using the notation of the previous section, $Q = CP^*_{\mathrm{D}}(R, \epsilon'')$. It is straightforward to verify that the set of solution points Q is an ϵ-convex Pareto set for the original instance $\mathcal{S}(I)$. It is also clear that the overall algorithm runs in polynomial time.

Now we proceed to analyze its performance. We will argue that $|Q| \leq 6|CP^*_\epsilon(I)|$. This follows by observing that $|Q| \leq 2 \cdot |CP^*_{\epsilon'}(I)|$ and applying the (appropriate version of the) geometric lemma shown above. To wit, let $CP_{\epsilon'}(I)$ be an ϵ'-convex Pareto set for $\mathcal{S}(I)$. (We remark that such a set may use solution points not in R_δ.) For each solution point $s \in CP_{\epsilon'}(I)$ pick two solution points $r_1, r_2 \in R_\delta$ whose convex combinations $(1+\delta)$-cover s; this can be done, since R is a δ-convex Pareto set for $\mathcal{S}(I)$. By this procedure, we get a set R' with the following properties: (i) $R' \subseteq R_\delta$ (ii) R' is an ϵ''-convex Pareto set for $\mathbf{f}(\mathcal{S}(I))$ (thus also for R_δ) and (iii) $|R'| \leq 2|CP_{\epsilon'}(I)|$. Therefore, we conclude that $|Q| \leq 2 \cdot |CP^*_{\epsilon'}(I)|$. By the geometric lemma, we have that $|CP^*_{\epsilon'}(I)| \leq 3|CP^*_\epsilon(I)|$. Therefore, $|Q| \leq 6 \cdot |CP^*_\epsilon(I)|$ and the proof is complete for the discrete case.

For convex problems, the algorithm is the same except that, in the postprocessing step, we use the explicit algorithm for the convex setting, i.e. EXPLICIT CONVEX-2D. The analysis is also identical the only difference being that $|Q| \leq |CP^*_{\epsilon'}(I)|$, which saves the factor of 2 in the approximation ratio.

■

We remark that if the objective space is convex, we can alternatively use the algorithm Convex-2D for problem $\mathcal{Q}_{\mathrm{C}}$ in the second step and save a factor of 2 in the approximation. However, as noted in the previous subsection, if the objectives are non-linear, there is no general way to find a set of solutions (as opposed to a set of solution *points*) with this cardinality. Moreover, if an approximate GAP routine is available, we can save a factor of 2 in the approximation.

Theorem 5.3.16. *For any bi-objective problem possessing a (fully) polynomial* GAP_δ *routine, for any* $\epsilon > 0$*, we can compute a* 3*-approximation to the smallest* ϵ *- convex Pareto set in (fully)* *polynomial time.*

Proof. If GAP_δ is available we save a factor of 2 by computing a δ-Pareto set in the first step. The algorithm (and the analysis) are essentially the same. We compute a δ-Pareto set for the given instance for $\delta \leq \sqrt[4]{1+\epsilon} - 1$ using the original algorithm of [PY1] and then use the algorithm for problem Q_D to find its smallest ϵ''-convex Pareto set. The analysis is essentially the same. The only difference now is that $|Q| \leq |CP^*_{\epsilon'}(I)|$. ■

Relaxed ϵ: Suppose that we are allowed to output an ϵ'-convex Pareto set for some $\epsilon' > \epsilon$. Then, by using a similar approach we can compute an (essentially optimal) factor 2 approximation to the smallest ϵ-convex Pareto set, if we spend time proportional to $1/(\epsilon' - \epsilon)$.

Theorem 5.3.17. *For any bi-objective optimization problem possessing a (fully) polynomial* Comb_δ *routine, for any* $\epsilon' > \epsilon > 0$*, we can compute an* ϵ'*-convex Pareto set* $CP_{\epsilon'}$ *such that* $|CP_{\epsilon'}| \leq 2|CP^*_\epsilon|$ *in (fully) polynomial time.*

Proof. Consider a suitable small rational $\delta > 0$ such that $1 + \epsilon' \geq (1+\epsilon)(1+\delta)^2$. For small ϵ, we can pick δ so that $1/\delta = O(1/(\epsilon' - \epsilon))$. Consider the following algorithm: Given an instance, ϵ' and ϵ, construct a δ-convex Pareto set CP_δ for this instance by using the algorithm of Section 3. (CP_δ has polynomial size and does not contains any redundant points.) Then, compute the smallest $(1+\epsilon)(1+\delta)$-convex Pareto set of CP_δ (by using the algorithm for $\mathcal{Q}_{\mathrm{D}}$). It is not hard to see that the produced set is an ϵ'-Pareto set with the desired cardinality bound. ■

Remark 5.3.18. If the GAP_δ routine is available, we can compute an ϵ'-convex Pareto set $CP_{\epsilon'}$ such that $|CP_{\epsilon'}| \leq |CP^*_\epsilon|$. The same holds if the objective space is convex. The only difference is that, if the GAP_δ routine is available, in the first phase we construct a δ-Pareto set, while for the convex case, we use the algorithm that is allowed to selects convex combinations of solution points in CP_δ (algorithm for $\mathcal{Q}_{\mathrm{C}}$).

Computing the best k solutions: We now address 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 want to find a set S of k solutions that minimizes the value of the ratio $\rho = 1+\epsilon$ such that S is an ϵ-convex Pareto set for the whole set of solutions. The above algorithm (for the relaxed ϵ) can be used as an *approximate decision procedure* to show the following.

Theorem 5.3.19. *For any bi-objective optimization with a polynomial* Comb_δ *routine, and for any* $\theta > 0$*, we can efficiently compute* $2k$ *points that approximate the smallest ratio* $\rho^* = 1 + \epsilon^*$ *for which there exists an* ϵ^**-convex Pareto set with at most* k *points to a factor of* $1 + \theta$*.*

Proof. Let $\delta > 0$ be a suitable small rational such that $(1 + \delta)^2$ approximates from below $1 + \theta$. Consider the following set of candidate ratios: $\rho_i = (1 + \delta)^i$, for $i = 0, 1, \dots, 2m/\log(1 + \delta) + 1$. It is clear that when i takes its maximum value $2m/\log(1 + \delta) + 1$, then any single solution point suffices to ρ_i-cover the feasible space.

The algorithm works as follows: We perform a binary search procedure among the candidate ratios ρ_i by calling the relaxed ϵ algorithm described above for $1+\epsilon' = \rho_i$ and $1+\epsilon = (1+\epsilon')/(1+\delta)$ at each step of the search until we find the minimum value of i^* for which the relaxed algorithm returns $2k$ points.

Since the relaxed algorithm returns more than $2k$ points when called with $1 + \epsilon' = (1 + \delta)^{i^*-1}$, it follows that $\rho^* \geq (1 + \delta)^{i^*-2}$. Therefore, the ratio achieved by the algorithm is ρ_{i^*}/ρ^* which is at most $1 + \theta$ by the definition of δ.

It is also easy to see that the overall procedure uses a polynomial number of calls to the relaxed algorithm; hence, it runs in polynomial time. ■

It is easy to see that the above result is best possible with respect to its dependence on the number of points it outputs; indeed, with $2k - 1$ points we can have no guarantee on the ratio, given the Comb_δ routine only.

5.4 d Objectives

In this section we consider the problem of computing near-minimum cardinality ϵ-convex Pareto sets for problems with d objectives, where $d > 2$ is assumed to be a constant. In order to design generic algorithms, for problems possessing a polynomial Comb_δ (or GAP_δ) routine, we analyze first (Section 5.4.1) the (very special) case that the set of points is given explicitly in the input. In Section 5.4.2, we give our generic algorithms. Analogously to the bi-objective case, there are two versions of the considered problem, depending on the nature of the objective space (convex or discrete).

5.4.1 Explicitly Given Points

Problem Definition We consider the problem of computing the smallest ϵ-convex Pareto set of an *explicitly given* set of points $A \subseteq \mathbb{R}^d_+$. We can assume that the objectives are to be minimized. The results can be easily extended to the case of maximization or mixed objectives. We consider the following versions of the problem:

- Problem $\mathcal{Q}_{\mathrm{D}}(A, \epsilon)$: The input set of points is the *discrete* set A and we are only allowed to use points of A for the approximation. Denote the optimal solution to this problem by $CP^*_{\mathrm{D}}(A, \epsilon)$. This variant corresponds to the case of a discrete objective space.

- Problem $\mathcal{Q}_{\mathrm{C}}(A, \epsilon)$: The input set of points is *convex*, in particular the *convex polytope* identified with the convex hull of the discrete set A. In this case we are allowed to use all points in $\mathcal{CH}(A)$ for the approximation. Denote the optimal solution to this problem by $CP^*_{\mathrm{C}}(A, \epsilon)$. This variant corresponds to the case of a continuous (convex) objective space.

Our main result in this section is the following:

Theorem 5.4.1. *a. For any* fixed d*, we can efficiently approximate the size* OPT_ϵ *of the optimal* ϵ*-convex Pareto set within a factor of* $O_d(\log \mathrm{OPT}_\epsilon)$.
b. In particular, for $d = 3$*, we can efficiently approximate* OPT_ϵ *within a constant factor.*

Remark 5.4.2. The notation $O_d(\cdot)$ means that the constant inside the $O(\cdot)$ depends on d. More specifically, the approximation guarantee is $O(d \log(d\mathrm{OPT}_\epsilon))$ for problem $\mathcal{Q}_{\mathrm{D}}$ and $O(d^2 \log(d\mathrm{OPT}_\epsilon))$

for problem $\mathcal{Q}_{\mathrm{C}}$. Moreover, for $d = 3$, the constant factor achieved for $\mathcal{Q}_{\mathrm{C}}$ is three times the corresponding factor for $\mathcal{Q}_{\mathrm{D}}$.

To formally present our results, we need to generalize the definitions and notation from Section 5.2 to higher dimensions. Throughout this section, A will denote a discrete set of points in $\mathbb{R}^d_+$. If S is a (non-necessarily discrete) subset of $\mathbb{R}^d_+$, we denote by $P(S)$ (resp. $CP(S)$) its Pareto set (resp. convex Pareto set).

Definition 5.4.3. We define the *lower envelope* of the set A, denoted by $\mathrm{LE}(A)$, as the Pareto set of the convex hull of A, i.e. $\mathrm{LE}(A) = P(\mathcal{CH}(A))$.

We remark that the lower envelope of A geometrically corresponds to the union of all facets of its convex hull whose inward normal vector has positive coefficients. Also notice that the set of extreme points of $\mathrm{LE}(A)$ is identified with $CP(A)$. We similarly define its "scaled" counterpart $\mathrm{LE}'_\epsilon = (1+\epsilon) \cdot \mathrm{LE}(A)$.

Definition 5.4.4. For two points $p, q \in \mathbb{R}^d_+$ the *ratio distance* between p and q is defined by: $\mathcal{RD}(p,q) = \max\{\max_i(p_i/q_i), 1\}$.

Analogously to the bi-objective case, the ratio distance between p and q is the minimum value $\rho^* = 1+\epsilon^*$ of the ratio ρ such that p ρ-covers q. It is easy to see that $\log \mathcal{RD}(\cdot,\cdot)$ satisfies the (directed) triangle inequality. Indeed, consider a triple of points $u, v, w \in \mathbb{R}^d_+$. We will show that $\log \mathcal{RD}(u,w) \leq \log \mathcal{RD}(u,v) + \log \mathcal{RD}(v,w)$ or equivalently $\mathcal{RD}(u,w) \leq \mathcal{RD}(u,v) \cdot \mathcal{RD}(v,w)$. By definition, for every coordinate $i \in [d]$ we have: $u_i \leq \mathcal{RD}(u,v) \cdot v_i$ and $v_i \leq \mathcal{RD}(v,w) \cdot w_i$. Hence, $u \leq \mathcal{RD}(u,v) \cdot \mathcal{RD}(v,w) \cdot w$, which implies that $\mathcal{RD}(u,w) \leq \mathcal{RD}(u,v) \cdot \mathcal{RD}(v,w)$ as desired.

Before we present the proof of Theorem 5.4.1, we briefly discuss the complexity of the problems $\mathcal{Q}_{\mathrm{D}}$ and $\mathcal{Q}_{\mathrm{C}}$. Recall that for $d = 2$ we showed in Section 5.2 that both problems can be solved (exactly) in polynomial time. For $d \geq 3$, we conjecture that both these problems are NP-hard.

A first complication arises from the fact that $\mathcal{Q}_{\mathrm{C}}$ is a *continuous* problem. Since there are no a priori guarantees on the bit complexity of the points selected in an optimal solution, it is not clear that, for $d > 2$, the problem is even in NP. (We showed that this is the case for $d = 2$.) However, if we only consider points in $CP(A)$ for the computation of a near optimum ϵ-convex Pareto set, we do not lose more than a factor of d in the approximation. This is established by a simple but crucial observation. To phrase it formally, we define an intermediate problem:

Definition 5.4.5. Define the problem $\mathcal{Q}_{\mathrm{C},R}(A,\epsilon)$: Given $A \subseteq \mathbb{R}_+^d$ and $\epsilon > 0$, compute the smallest ϵ-convex Pareto set of A that is allowed to use points *only* from its convex Pareto set $CP(A)$. Denote the optimal solution to this problem by $CP^*_{\mathrm{C},R}(A,\epsilon)$.

This problem is a restriction to both $\mathcal{Q}_{\mathrm{C}}(A,\epsilon)$ and $\mathcal{Q}_{\mathrm{D}}(A,\epsilon)$ (i.e. any feasible solution to $\mathcal{Q}_{\mathrm{C},R}(A,\epsilon)$ is feasible for the other problems). It therefore follows that

$$|CP^*_{\mathrm{C},R}(A,\epsilon)| \geq \max\{|CP^*_{\mathrm{C}}(A,\epsilon)|, |CP^*_{\mathrm{D}}(A,\epsilon)|\}.$$

Also note that $|CP^*_{\mathrm{C}}(A,\epsilon)| \leq |CP^*_{\mathrm{D}}(A,\epsilon)|$. This holds because any point of A is dominated by some point of $\mathrm{LE}(A)$ which implies that any feasible solution to $\mathcal{Q}_{\mathrm{D}}(A,\epsilon)$ can be directly translated to a feasible solution to $\mathcal{Q}_{\mathrm{C}}(A,\epsilon)$ of the same cardinality. Hence, $|CP^*_{\mathrm{C}}(A,\epsilon)| \leq |CP^*_{\mathrm{D}}(A,\epsilon)| \leq |CP^*_{\mathrm{C},R}(A,\epsilon)|$.

The next simple observation establishes the fact that the optimal solution to $\mathcal{Q}_{\mathrm{C},R}(A,\epsilon)$ is of cardinality at most d times that of an optimal solution to $\mathcal{Q}_{\mathrm{C}}(A,\epsilon)$.

Claim 5.4.6. $|CP^*_{\mathrm{C},R}(A,\epsilon)| \leq d \cdot |CP^*_{\mathrm{C}}(A,\epsilon)|$

Proof. Let $CP^*_{\mathrm{C}}(A,\epsilon) = \{p_i^* \mid i \in [k]\}$. We can assume without loss of generality that for all $i \in [k]$, $p_i^* \in \mathrm{LE}(A)$. This holds for the following reason: Suppose that there exists a point $p_r^* \in CP^*_{\mathrm{C}}(A,\epsilon)$ that does not lie on the lower envelope of A. Then, p_r^* is dominated by a point q of the lower envelope. The set $(CP^*_{\mathrm{C}}(A,\epsilon) \setminus \{p_r^*\}) \cup \{q\}$, is clearly an ϵ-convex Pareto set of the same cardinality satisfying the assumed property.

Then, p_i^* can be expressed as a convex combination of (at most) d vertices of $CP(A)$. Indeed, p_i^* belongs to a facet $\mathfrak{F}$ of the lower envelope. Since $\mathfrak{F}$ is a $(d-1)$ dimensional convex polytope, the claim follows from Carathéodory's lemma. The union of these d-sets of vertices is a solution to $CP^*_{\mathrm{C},R}(A,\epsilon)$ having the desired cardinality. ■

Corollary 5.4.7. $|CP^*_{\mathrm{C},R}(A,\epsilon)| \leq d \cdot |CP^*_{\mathrm{C}}(A,\epsilon)| \leq d \cdot |CP^*_{\mathrm{D}}(A,\epsilon)|$

We are now ready to prove Theorem 5.4.1.

Proof. (Theorem 5.4.1) To prove the theorem we phrase the problem $\mathcal{Q}_{\mathrm{D}}$ (and as a consequence $\mathcal{Q}_{\mathrm{C},R}$) as a hitting set problem on a set system of bounded VC–dimension. For $d = 3$, an additional property of the set system implies a constant factor approximation. We remark here that the corresponding set system is of size roughly $O(n^d)$, where $n = |A|$, and is not given explicitly; our input is the discrete set A and the error ϵ (i.e. size of the input $O(dn + |\epsilon|)$). It is therefore crucial for the dimension d to be fixed so that the set system is constructible in polynomial time.

We give the reduction for the problem $\mathcal{Q}_{\mathrm{D}}$ and then argue that, by a trivial modification, it holds for $\mathcal{Q}_{\mathrm{C},R}$ also.

Lemma 5.4.8. *For any* fixed *d, there exists a polynomial time constructible set system $\mathcal{F} = \mathcal{F}(A, \epsilon)$ with the property that there exists a bijection between hitting sets for $\mathcal{F}$ and feasible solutions to $\mathcal{Q}_{\mathrm{D}}(A, \epsilon)$.*

Proof. Recall that our goal in problem $\mathcal{Q}_{\mathrm{D}}(A, \epsilon)$ is to compute a minimum cardinality subset of $P(A)$ such that any point of $CP(A)$ is $(1 + \epsilon)$-covered by a convex combination of selected points.

We start with some notation. As is standard, we express a hyperplane h in $\mathbb{R}^d$ (that does not go through the origin) in the form $h = \{x \in \mathbb{R}^d \mid \alpha_h \cdot x = 1\}$. Let $\mathcal{H}$ denote the set of *supporting hyperplanes* of $\mathrm{LE}(A)$ with *positive coefficients*. In particular, this means that each $h \in \mathcal{H}$ can be expressed as $h = \{x \in \mathbb{R}^d \mid \alpha_h \cdot x = 1\}$, such that the following conditions are satisfied: (i) $\alpha_h \in \mathbb{R}^d_+$, (ii) $\alpha_h \cdot v \geq 1$ for all $v \in CP(A)$ and (iii) $\alpha_h \cdot p_h = 1$ for some $p_h \in CP(A)$. Now for $h \in \mathcal{H}$ denote by $h_{1+\epsilon} = \{x \in \mathbb{R}^d \mid \alpha_h \cdot x = 1 + \epsilon\}$ the "boosted" hyperplane obtained from h by scaling each of its points by a $(1 + \epsilon)$ factor in each coordinate. Also denote by $\mathcal{H}_{1+\epsilon} = \{h_{1+\epsilon} \mid h \in \mathcal{H}\}$ the "boosted" set of hyperplanes corresponding to $\mathcal{H}$. We will say that a point is *below* the hyperplane $h_{1+\epsilon} \in \mathcal{H}_{1+\epsilon}$ if it lies in the *closed* negative halfspace $h^-_{1+\epsilon} = \{x \in \mathbb{R}^d \mid \alpha_h \cdot x \leq 1 + \epsilon\}$. If the inequality is strict, we say that the point is strictly below the hyperplane, etc.

Consider the following partition of $\mathcal{H}_{1+\epsilon}$ into equivalence classes, each class $\mathbf{h}_i$ having the following property: Any two hyperplanes in $\mathbf{h}_i$ cannot *distinguish* any two points of $P(A)$; that is, all points of $P(A)$ lie on the "same side" of the hyperplanes in the class (either below or *strictly* above). We denote by $\mathbf{H}_{1+\epsilon} = \{\mathbf{h}_i \mid i \in [m_\epsilon]\}$ the collection of equivalence classes, by $\mathbf{h}_i^- = \bigcap_{h \in \mathbf{h}_i} h^-$ the set of points in $\mathbb{R}^d$ that lie below all the hyperplanes in the class $\mathbf{h}_i$ and by $\mathbf{H}^-_{1+\epsilon} =$

$\{\mathbf{h}_i^- \mid i \in [m_\epsilon]\}$ the corresponding collection.

The set system $\mathcal{F}$ contains an "element" for each point of $P(A)$ and a subset of $P(A)$ for each $\mathbf{h}_i$; in particular, the subset that lies below (all the hyperplanes in the class) $\mathbf{h}_i$. That is, $\mathcal{F} = (P(A), \mathcal{S}(A,\epsilon))$, where $\mathcal{S}(A,\epsilon) = \{P(A) \cap \mathbf{h}_i^- \mid i \in [m_\epsilon]\}$. By slight abuse of notation, we will denote $\mathcal{S}(A,\epsilon) = P(A) \cap \mathbf{H}_{1+\epsilon}^-$. As is standard, we say that a point $p \in P(A)$ "hits" a hyperplane class $\mathbf{h}_i$ if p belongs to $\mathbf{h}_i^-$.

Clearly, the partition (thus, its cardinality m_ϵ) depends on the parameter ϵ. An upper bound of $m_\epsilon = O(n^d)$ follows by geometric duality and the fact that an arrangement of n hyperplanes in $\mathbb{R}^d$ defines $O(n^d)$ cells. It is also not hard to show that the family $P(A) \cap \mathbf{H}_{1+\epsilon}^-$ can be constructed in polynomial time; this can be done by standard techniques. (We have a simple efficient construction in time $O(n^2 m_\epsilon)$). The following claim proves the desired equivalence:

Claim 5.4.9. *There is a bijection between hitting sets of $\mathcal{F}$ and solutions to $\mathcal{Q}_{\mathrm{D}}$.*

Proof. ($\Rightarrow$) Let $V_{\mathcal{F}} = \{v_i \mid i \in [k], v_i \in P(A)\}$ be a hitting set of $\mathcal{F}$. We want to show that $V_{\mathcal{F}}$ is an ϵ-convex Pareto set. For the sake of contradiction, suppose that no convex combination of the v_i's $(1+\epsilon)$-covers some point $v \in CP(A)$. Consider the lower envelope LE$'$ of the set $V_{\mathcal{F}} \cup \{(1+\epsilon)v\}$; the aforementioned assumption means that the point $(1+\epsilon)v$ is a *vertex* of LE$'$. Thus, there exists a direction $c \in \mathbb{R}_+^d$ such that $(1+\epsilon)v$ is the *unique* minimizer of (the linear function) $c \cdot x$ over LE$'$, i.e. for all $i \in [k]$ it holds $c \cdot (1+\epsilon)v < c \cdot v_i$. Now let $v^* \in CP(A)$ denote *some* minimizer of $c \cdot x$ over LE(A); hence, for all $p \in CP(A)$ we have $c \cdot v^* \leq c \cdot p$. We claim that the hyperplane $h' = \{x \in \mathbb{R}^d \mid \alpha_c \cdot x = 1+\epsilon\}$, where $\alpha_c = c/(c \cdot v^*)$ is in $\mathcal{H}_{1+\epsilon}$ and is not "hit" by $V_{\mathcal{F}}$.

For the first part of the claim, it is easy to verify that the hyperplane $h'/(1+\epsilon) = \{x \in \mathbb{R}^d \mid \alpha_c \cdot x = 1\}$ is in $\mathcal{H}$. Indeed, since $c, v^* \in \mathbb{R}_+^d$ it follows that $\alpha_c \in \mathbb{R}_+^d$ (property (i)); by the definition of v^* we get $\alpha_c \cdot p = (c \cdot p)/(c \cdot v^*) \geq 1$ (property (ii)) and the definition of α_c implies $\alpha_c \cdot v^* = 1$ (property (iii)). For the second part of the claim we need to show that for all $i \in [k]$ it holds $\alpha_c \cdot v_i \in (h')^+$ or equivalently $\alpha_c \cdot v_i > 1+\epsilon$. This follows from the sequence of inequalities: $\alpha_c \cdot v_i = (c \cdot v_i)/(c \cdot v^*) \geq (c \cdot v_i)/(c \cdot v) > 1+\epsilon$, where the first inequality is a consequence of the definition of v^* and the second of the definition of v. This completes the argument and provides the desired contradiction.

($\Leftarrow$) Let $CP_\epsilon = \{v_i \mid i \in [k], v_i \in P(A)\}$ be an ϵ-convex Pareto set of A. We will show that each $h' \in \mathcal{H}_{1+\epsilon}$ is "hit" by some point in CP_ϵ, i.e. that for each $h' \in \mathcal{H}_{1+\epsilon}$ there exists some $v_{h'} \in CP_\epsilon$ satisfying $v_{h'} \in (h')^-$. Recall that h' is of the form $\alpha_{h'} \cdot x = 1 + \epsilon$ with $\alpha_{h'} \in \mathbb{R}^d_+$ and $\alpha_{h'} \cdot v = 1$ for some $v \in CP(A)$. By definition, the point v is $(1+\epsilon)$-covered by a convex combination of elements in CP_ϵ, i.e. there exist $\{\lambda_i\}_{i=1}^k$ with $\lambda_i \geq 0$ and $\sum_{i=1}^k \lambda_i = 1$ such that $\sum_{i=1}^k \lambda_i \cdot v_i \leq (1+\epsilon)v$. Since $\alpha_{h'}$ has only positive coefficients it follows that: $\sum_{i=1}^k \lambda_i(\alpha_{h'} \cdot v_i) \leq (1+\epsilon)(\alpha_{h'} \cdot v) = 1+\epsilon$. Thus, we *must* have that $\alpha_{h'} \cdot v_i \leq 1+\epsilon$ for some $v_i \in CP_\epsilon$. This completes the proof of the claim. ■

The proof of Lemma 5.4.8 is now complete. ■

Since $\mathcal{Q}_{\mathrm{C},R}$ is a special case of $\mathcal{Q}_{\mathrm{D}}$ the same reduction essentially works for this problem also. The only difference is that the set of "elements" for the corresponding set system (call it $\mathcal{F}'$) is the convex Pareto set $CP(A)$ and the hyperplane classes are defined *with respect to* $CP(A)$ (as opposed to $P(A)$). For the rest of the proof, we focus on $\mathcal{F}$, but all subsequent properties trivially hold for $\mathcal{F}'$ too.

Lemma 5.4.8 immediately implies an $O(d \log n)$ factor approximation algorithm for both problems in hand. However, we can do better by exploiting additional properties of the set systems. In particular, we point out the following:

Claim 5.4.10. *For any (discrete) set $A \subseteq \mathbb{R}^d_+$ and $\epsilon > 0$ it holds: VC-dim($\mathcal{F}(A,\epsilon)$) $\leq d + 1$. Moreover, for $d = 3$, $\mathcal{F}(A,\epsilon)$ admits an $1/r$-net of cardinality $O(r)$ that is constructible in polynomial time.*

Proof. Both statements of the claim follow by noting that $\mathcal{F}(A,\epsilon)$ is a "special case" of the set system of (closed) halfspaces in $\mathbb{R}^d$. This set system has a *finite* set X of points in $\mathbb{R}^d$ as its set of "elements" and its collection of sets consists of all subsets of X that can be obtained by intersecting X with a halfspace.

The latter set system is well-known to have VC–dimension equal to $d+1$. Moreover, $1/r$-nets of size $O(dr \cdot \log(dr))$ can be constructed for it in poly$(r, |X|)$ time, for any *fixed* d [BCM]. In particular, for $d = 3$, the construction of [MSW] yields a deterministic polynomial time algorithm to construct an $1/r$-net of size $O(r)$. ■

The theorem now follows from the aforementioned by using the approximation algorithms of [BG, ERS] to compute an approximate solution to the *hitting set* problem for $\mathcal{F}$. If the net-finding algorithm produces an $1/r$-net of size $s(r)$, the approximation guarantee of these algorithms is $s(|\mathrm{OPT}_\epsilon|)/|\mathrm{OPT}_\epsilon|$. The case of general (fixed) d follows since $s(r) = O(dr \cdot \log(dr))$. For $d = 3$, we have that $s(r) = cr$, where c is a constant, which implies a factor c approximation.

Therefore, problems $\mathcal{Q}_{\mathrm{D}}$ and $\mathcal{Q}_{\mathrm{C},R}$ can be approximated within a factor of $O(d \log(d\mathrm{OPT}_\epsilon))$ for general d and within a factor of c for $d = 3$. Finally, recall that in the case of problem $\mathcal{Q}_{\mathrm{C}}$, we lose an additional factor of (at most) d by ignoring the non-extreme points. Thus, $\mathcal{Q}_{\mathrm{C}}$ can be approximated within a factor of $O(d^2 \log(d\mathrm{OPT}_\epsilon))$ for general d and within a factor of $3c$ for $d = 3$. ■

An illustration of (part of) the set system $\mathcal{F}$ for a small two dimensional instance is given in Fig. 5.5.

Theorem 5.4.11. *If d is unbounded, even if all the solution points are given explicitly in the input, for any $\epsilon > 0$, we cannot approximate the smallest ϵ-*convex Pareto set *on d objectives in polynomial time to a factor better than $\Omega(\log d)$, unless $P = NP$.*

Proof. The proof is via a gap-preserving reduction from the Set Cover problem (which is known to have this property [LY, Fei, RS]): Given a universe of elements $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ and a family $\mathcal{S} = \{S_1, S_2, \ldots, S_l\}$ of *proper* subsets of $\mathcal{U}$ (such that $\bigcup_{S_i \in \mathcal{S}} S_i = \mathcal{U}$), select a minimum number of sets from $\mathcal{S}$ such that their union is $\mathcal{U}$. The result holds for both variants of the problem (i.e. problems $\mathcal{Q}_{\mathrm{D}}$, Q_{C}).

The reduction is by a modification of the corresponding reduction in [VY]. It is as follows: We define a set A of $n + l + 1$ points in n-dimensions (i.e. $d = n$). For each element $u_i \in \mathcal{U}$ ($i \in [n]$), add a point p_i whose ith coordinate is $1/(1+\epsilon)$ and all other coordinates are at ∞ (a very large finite number will also do). For each set $S_j \in \mathcal{S}$ ($j \in [l]$), add a point q_j such that the ith coordinate of q_j is 1 if $u_i \in S_j$ and $n(1+\epsilon)^3$ otherwise. We also add a point r with value $(1+\epsilon)$ in all coordinates. It is easy to see that all points of A are vertices of its lower envelope.

We mention the basic (straightforward) properties of the reduction. First, the point r is not $(1+\epsilon)$-covered by any other point. Second, r $(1+\epsilon)$-covers all the q_j's and does not $(1+\epsilon)$-cover any p_i. Third, each q_j $(1+\epsilon)$-covers *exactly* those p_i's such that $u_i \in S_j$. Last, no p_i $(1+\epsilon)$-covers

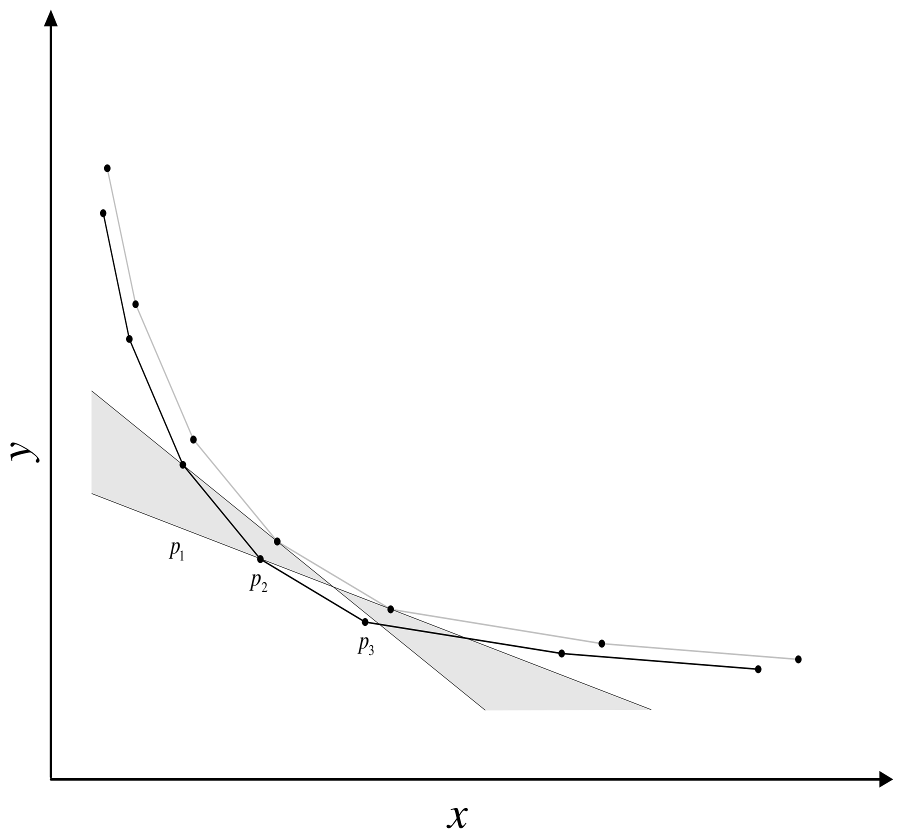

Figure 5.5: An illustration of the set system $\mathcal{F}$. It is assumed for simplicity that $A = CP(A)$. The shaded region represents the hyperplane class that lies above the set of points $\{p_2, p_3\}$.

any other point.

By using the above properties, it is easy to show the following: Any ϵ-convex Pareto set of A must be of cardinality at least equal to the cardinality of the minimum Set Cover of $(\mathcal{U}, \mathcal{S})$. Indeed, in order to $(1+\epsilon)$-cover p_i we need to select either p_i itself or some q_j such that $u_i \in S_j$. Selecting convex combinations of the q_j's does not help; a convex combination will $(1+\epsilon)$-cover p_i iff for each q_j in the combination it holds $u_i \in S_j$. The reduction is easily see to be gap-preserving, which finishes the proof. ■

5.4.2 Approximate Comb Routine

So far we have considered the problem of computing the smallest ϵ-convex Pareto set if the points are explicitly given in the input. In this section we consider the same problem for a multi-objective problem Π possessing a black box routine to access its solutions. We assume that either the Comb_δ routine or the GAP_δ routine are available.

5.4.2.1 Lower Bound

Theorem 5.4.12. *For $d \geq 3$, any polynomial generic algorithm having oracle access to* GAP_δ *cannot be c-competitive for any constant c.*

Proof. The lower bound holds *a fortiori* if only the Comb_δ routine is available. However, it is not clear that it holds when the primitives are exact.

We exploit the fact that $\mathrm{GAP}_\delta(b)$ is not uniquely defined for some points b. We construct two sets of points S and S' in $\mathbb{R}^3$ such that GAP_δ cannot distinguish between them unless the error parameter δ becomes inverse exponential in the size of the input.

Given an $\epsilon > 0$ construct a set of points $Q = \{q_1, \ldots, q_k\}$ (points ordered left to right) in the same plane (having $z = z_0$) such that the smallest ϵ-convex Pareto set of Q contains all its points $CP(Q) = Q$. This can clearly be done. Also consider the set $Q' = \{q'_1, \ldots, q'_k\}$ where each q'_i has the same x and y coordinates as q_i and whose z coordinate is by 1 smaller $z' = z_0 - 1$. Now choose a point p that lies an $(1+\epsilon)$ factor above the $z = z_0$ plane and $(1+\epsilon)$-covers all the points of Q. Moreover, we need $x(q_1) = \min_{q \in Q} x(q) > (1+\epsilon)x(p)$ and $y(q_k) = \min_{q \in Q} y(q) > (1+\epsilon)y(p)$. Clearly, this last condition implies that no convex combination of points in Q (or Q') $(1+\epsilon)$-covers p (because of the x and y coordinates). So, p must belong to any ϵ-convex Pareto set for the instances

$S \doteq \{p\} \cup Q$ and $S' \doteq \{p\} \cup Q \cup Q'$. The smallest ϵ-convex Pareto set for the former set is $\{p\}$, while for the latter $\{p\} \cup Q'$.

It is easy to see that if $z_0 = M$ is exponential in the size of the input and $1/\epsilon$, the primitive cannot distinguish between the two cases. ■

5.4.2.2 Upper Bound

As a consequence of the lower bound, for $d \geq 3$ objectives we are forced to compute an ϵ'-convex Pareto set, where $\epsilon' > \epsilon$, if we are to have a guarantee on its size. We show in this section that for any $\epsilon' > \epsilon$, we can get a constant factor approximation for $d = 3$ and a logarithmic approximation for any fixed d if we spend time proportional to $1/(\epsilon' - \epsilon)$. This positive result applies to all d-objective problems that possess a polynomial Comb_δ routine. We note that if the (stronger) GAP_δ routine is available, the corresponding guarantees can be improved by a factor of d.

Our main positive result for the case of general (fixed) d is the following theorem that applies to all d-objective problems that possess a polynomial Comb_δ routine:

Theorem 5.4.13. *1. For any $\epsilon' > \epsilon$ there exists a polynomial time generic algorithm that computes an ϵ'-convex Pareto set Q such that $|Q| \leq O(d^2 \log(d\mathrm{OPT}_\epsilon)) \cdot \mathrm{OPT}_\epsilon$.*
2. For $d = 3$, we can efficiently compute a constant factor approximation to OPT_ϵ.

We first prove the following lemma that relates the approximability of problem $\mathcal{Q}_{\mathrm{C},R}$ with the problem in hand. Let $\epsilon > 0$ be a given rational number. For any $\epsilon' > \epsilon$, we can find a $\delta > 0$ such that $1/\ \delta = O(1/(\epsilon' - \epsilon))$ satisfying $1 + \epsilon' \geq (1 + \epsilon)(1 + \delta)^2$.

Lemma 5.4.14. *Suppose that there exists an r-factor approximation algorithm for $\mathcal{Q}_{\mathrm{C},R}$. Then, for any $\epsilon' > \epsilon$, we can compute an ϵ'-convex Pareto set Q, such that $|Q| \leq dr\mathrm{OPT}_\epsilon$ using $O((m/\delta)^d)$ Comb_δ calls.*

Proof. The generic algorithm proceeds in two phases; in the first phase, we compute a δ-convex Pareto set, by using the oblivious algorithm of Chapter 4 and in the second phase we post-process the points produced by the latter algorithm by using the r-approximation algorithm for $\mathcal{Q}_{\mathrm{C},R}$ as a black box.

So, suppose that R_δ is the output of the aforementioned algorithm; R_δ is a δ-convex Pareto set (without redundant points) for the (exponentially large or infinite) feasible space S and has size polynomial in the size of the input and $1/\delta$. We apply the r-approximation algorithm for $\mathcal{Q}_{\mathrm{C},R}$ on input R_δ to produce a set $Q \subseteq R_\delta$ of solution points whose convex combinations $(1+\epsilon)(1+\delta)$-cover R_δ. It is easy to see that Q is an ϵ'-convex Pareto set for the feasible space S. Indeed, for any solution point $p \in S$ there exists a convex combination of points $r_1, r_2, \ldots, r_l$ in R_δ that $(1+\delta)$-covers p. Also, each r_i is $(1+\epsilon)(1+\delta)$-covered by some convex combination of points in Q. Therefore, there exists a convex combination of points in Q that $(1+\epsilon)(1+\delta)^2$-covers p.

We will now argue that $|Q| \leq dr\mathrm{OPT}_\epsilon$. Let R^* denote a minimum cardinality subset of R_δ whose convex combinations $(1+\epsilon)(1+\delta)$-cover R_δ. By assumption we have that $|Q| \leq r|R^*|$. The proof is complete by the following claim:

Claim 5.4.15. $|R^*| \leq d \cdot \mathrm{OPT}_\epsilon$

Proof. Let CP_ϵ be any ϵ-convex Pareto set for S. It suffices to show that there exists an $(1+\epsilon)(1+\delta)$-convex cover C for R_δ of cardinality at most $d \cdot |CP_\epsilon|$. Recall that R_δ is a δ-convex Pareto set for S (having no redundant points). Thus, for any solution point $s \in CP_\epsilon \subseteq S$, there exists a convex combination of points in R_δ that $(1+\delta)$-covers s. C is constructed as follows: For each $s \in CP_\epsilon$ pick a set of at most d points in R_δ whose convex combinations $(1+\delta)$-cover s. It is clear that $|C| \leq d|CP_\epsilon|$ and that C is an $(1+\epsilon)(1+\delta)$-convex cover for S (and thus also for R_δ) - using points only from R_δ. ■

■

Recall from Section 5.4.1 that problem $\mathcal{Q}_{\mathrm{C},R}$ can be approximated within a factor of $O(d\log(d\mathrm{OPT}_\epsilon))$ for general d and within a constant factor for $d = 3$. Theorem 5.4.13 follows by combining this fact with Lemma 5.4.14.

We remark that there is a direct analogue of the above lemma (without the extra factor of d) for the case that the GAP_δ routine is available. In this case, we can save a factor of d by computing a δ-Pareto set in the first step followed by the approximation algorithm for problem $\mathcal{Q}_{\mathrm{D}}$.

Chapter 6

The Chord Algorithm

6.1 Introduction

The Chord algorithm is a popular, simple method for the succinct approximation of planar curves, which is widely used, under different names, in a variety of areas, such as, bi-objective and parametric optimization, computational geometry, and graphics. We analyze the performance of the chord algorithm, as compared to the optimal approximation that achieves a desired accuracy with the minimum number of points. We prove sharp upper and lower bounds, both in the worst case and average case setting.

We now briefly describe the algorithm. Let f_1, f_2 be the two objectives (say minimization objectives for concreteness), and let P be the (unknown) convex Pareto curve. First optimize f_1, and f_2 separately (i.e. call Comb for the weight tuples $(1, 0)$ and $(0, 1)$) to compute the leftmost and rightmost points a, b of the curve P. The segment (a, b) is a first approximation to P; its quality is determined by a point $q \in P$ that is least well covered by the segment. It is easy to see that this worst point q is a point of the Pareto curve P that minimizes the linear combination $f_2 + \lambda f_1$, where λ is the absolute value of the slope of (a, b), i.e. it is a point of P with a supporting line parallel to the 'chord' (a, b). Compute such a worst point q; if the error is $\leq \epsilon$, then terminate, otherwise add q to the set S to form an approximate set $\{a, q, b\}$ and recurse on the two intervals (a, q) and (q, b). In Section 6.2 we give a more detailed formal description (for example, in some cases we can determine from previous information that the maximum possible error in an interval is $\leq \epsilon$ and do not need to call Comb).

In this chapter we analyze the performance of the Chord algorithm, both in the worst case and in the average case setting. We define the performance ratio to be the ratio between the number of Comb calls performed by the Chord algorithm and the minimum number of points required to obtain an ϵ-convex Pareto set for the given instance (see Section 6.2)[1]. We provide sharp upper and lower bounds on the performance (competitive) ratio of the Chord algorithm, both in the worst case and in the average case setting. Consider a bi-objective problem where the objective functions take values in $[2^{-m}, 2^m]$. We prove that the worst-case performance ratio of the Chord algorithm for computing an ϵ-convex Pareto set is $\Theta(\frac{m+\log(1/\epsilon)}{\log m+\log\log(1/\epsilon)})$. The upper bound implies in particular that for problems with polynomially computable objective functions and a polynomial-time (exact or approximate) Comb routine, the Chord algorithm runs in polynomial time in the input size and $1/\epsilon$. We show furthermore that there is no algorithm with constant performance ratio. In particular, every algorithm (even randomized) has performance ratio at least $\Omega(\log m + \log\log(1/\epsilon))$.

Similar results hold for the approximation of convex curves with respect to the Hausdorff distance. That is, the performance ratio of the Chord algorithm for approximating a convex curve of length L within Hausdorff distance ϵ, is $\Theta(\frac{\log(L/\epsilon)}{\log\log(L/\epsilon)})$. Furthermore, every algorithm has worst-case performance ratio at least $\Omega(\log\log(L/\epsilon))$.

We also analyze the expected performance of the Chord algorithm for some natural probability distributions. Given that the algorithm is used in practice in various contexts with good performance, and since worst-case instances are often pathological and extreme, it is interesting to analyze the average case performance of the algorithm. Indeed, we show that the performance on the average is exponentially better. Note that Chord is a simple natural greedy algorithm, and is not tuned to any particular distribution. We consider instances generated by a class of product distributions that are "approximately" uniform and prove that the expected performance ratio of the Chord algorithm is $\Theta(\log m + \log\log(1/\epsilon))$ (upper and lower bound). Again similar results hold for the Hausdorff distance.

Related Work. There is extensive work on multiobjective optimization, as well as on approximation of curves in various contexts. We have discussed already the main related references. The problem addressed by the Chord algorithm fits within the general framework of determining the shape by probing [CY]. Most of the work in this area concerns the exact reconstruction, and the

[1]We note that this notion of performance is different than the ones used in the previous chapters.

analytical works on approximation (e.g., [LB, Ro, YG]) compute only the worst-case cost of the algorithm in terms of ϵ (showing bounds of the form $O(\sqrt{L/\epsilon})$). There does not seem to be any prior work comparing the cost of the algorithm to the optimal cost for the instance at hand, i.e., the approximation ratio, which is the usual way of measuring the performance of approximation algorithms.

Organization. The rest of the chapter is organized as follows. Section 6.2 describes the model and states our main results, Section 6.3 concerns the worst-case analysis, and Section 6.4 the average-case analysis.

6.2 Model and Statement of Results

6.2.1 Preliminaries.

Let Π be a bi-objective optimization problem in the aforementioned framework. We access the objective space $\mathcal{I}$ of Π via an oracle Comb that (exactly or approximately) minimizes non-negative linear combinations $y + \lambda x$ of the objectives. We assume that either $\delta = 0$ (i.e. we have an exact routine), or we have a PTAS, i.e. can efficiently compute Comb_δ for all $\delta > 0$. Recall that the existence of a PTAS for Comb_δ is necessary and sufficient for the efficient computability of an ϵ-CP set.

Note that, obviously every algorithm that constructs an ϵ-CP, must certainly make at the very least $\mathrm{OPT}_\epsilon(\mathcal{I})$ calls to Comb, just to get $\mathrm{OPT}_\epsilon(\mathcal{I})$ points – which are needed at a minimum to form an ϵ-CP ; this holds even if the algorithm somehow manages to always be lucky and call Comb with the right values of λ that identify the points of an optimal ϵ-CP. (Having obtained the points of an optimal ϵ-CP, another $\mathrm{OPT}_\epsilon(\mathcal{I})$ many calls to Comb with the slopes of the edges of the polygonal line defined by the points, suffice to verify that the points form an ϵ-CP. Hence, the "offline" optimum number of calls is at most $2 \cdot \mathrm{OPT}_\epsilon(\mathcal{I})$.) Let $\mathrm{CHORD}_\epsilon(\mathcal{I})$ be the number of Comb calls required by the chord algorithm. The worst-case performance ratio of the algorithm is $\sup_{\mathcal{I}} \mathrm{CHORD}_\epsilon(\mathcal{I})/\mathrm{OPT}_\epsilon(\mathcal{I})$. If the inputs are drawn from some probability distribution $\mathcal{D}$, then we will use the expected performance ratio $\mathbb{E}_{\mathcal{I}\in\mathcal{D}}[\mathrm{CHORD}_\epsilon(\mathcal{I})/\mathrm{OPT}_\epsilon(\mathcal{I})]$ as a measure (note that we shall omit the subscript "$\mathcal{I} \in \mathcal{D}$" when the underlying distribution $\mathcal{D}$ over instances will be clear from the context).

We denote by pq the line segment with endpoints p and q, (pq) denotes its length and $\triangle(pqr)$ is the triangle defined by p, q, r. Also $S(A)$ denotes the area of A. We now define the horizontal distance. (We use this distance as an intermediate tool for our lower bound construction in Section 6.3.1.) The horizontal distance from p to q is defined $\Delta x(p,q) = \max\{x(q) - x(p), 0\}$. The horizontal distance from p to the line segment ℓ is $\Delta x(p,\ell) = \Delta x(p, p_\ell)$, where p_ℓ is the y-projection of p on ℓ. Also $[n] := \{1, 2, \ldots, n\}$ and $[i,j] := \{i, i+1, \ldots, j\}$.

Remark 6.2.1. All the upper bounds of this chapter on the performance of the Chord algorithm hold under the assumption that we have a PTAS for the Comb routine. On the other hand, our lower bounds apply even for the special case that an exact routine is available. For the simplicity of the exposition, we first describe the Chord algorithm and prove our upper bounds for the case of an exact Comb routine. We then describe the differences for the case of an approximate routine.

6.2.2 The Chord Algorithm.

We have set the stage to formally describe the algorithm. Let Π be a bi-objective problem with an efficient exact Comb routine. Given $\epsilon > 0$ and an instance $\mathcal{I}$ of Π (implicitly given via Comb), we would like to construct an ϵ-convex Pareto set for $\mathcal{I}$ using as few calls to Comb as possible. As mentioned in the introduction, a popular algorithm for this purpose is the chord algorithm that is the main object of study in this chapter.

In Table 6.1 we describe the algorithm in detailed pseudo-code. The pseudo-code corresponds exactly to the description of the algorithm in the introduction.

The basic routine Chord is called recursively from the main algorithm. This recursive description will be useful in the analysis that follows.

An illustration of one iteration of the algorithm (i.e. recursive call of the Chord routine) is given in Figure 6.1 below. The points a_i, b_i are solution points (in $\mathcal{I}$) and are the results of previous recursive calls. The point $q_i = \mathrm{Comb}(\lambda_{a_i b_i})$ is the solution point computed in the current call. The algorithm will recurse on the triangles $\triangle(a_i a'_i q_i)$ and $\triangle(q_i b_i b'_i)$. Note that the line $a'_i b'_i$ is parallel to $a_i b_i$ and $\angle a_i c_i b_i \in [\pi/2, \pi)$.

During the execution of the algorithm, we "learn" the objective space in an "online" fashion. After a number of iterations, we have obtained information that imposes an upper and a lower approximation to $\mathrm{CP}(\mathcal{I})$. In particular, the computed points define a polygonal chain that is an "upper"

```
**Chord Algorithm** (*Input:* $\mathcal{I}$, $\epsilon$)

$a = \mathrm{Comb}(+\infty)$; $b = \mathrm{Comb}(0)$;
$c = (x(a), y(b))$;
**Return** Chord $(\{a, b, c\}, \epsilon)$.

**Routine** Chord (*Input:* $\{l, r, s\}, \epsilon$)
**If** $\mathcal{RD}(s, lr) \leq \epsilon$ **return** $\{l, r\}$;
$\lambda_{lr} =$ absolute slope of $lr$; $q = \mathrm{Comb}(\lambda_{lr})$;
**If** $\mathcal{RD}(q, lr) \leq \epsilon$ **return** $\{l, r\}$;
$\ell(q) :=$ line parallel to $lr$ through $q$;
$s_l = ls \cap \ell(q)$; $s_r = rs \cap \ell(q)$;
$Q_l =$ Chord$(\{l, q, s_l\}, \epsilon)$;
$Q_r =$ Chord$(\{q, r, s_r\}, \epsilon)$;
**Return** $Q_l \cup Q_r$.
```

Table 6.1: Pseudo-code for Chord algorithm.

approximation to $\mathrm{CP}(\mathcal{I})$ and the supporting lines at these points define a "lower" approximation.

The pseudo-code above is specialized for the ratio distance, but one may use other metrics based on the application. In the context of convex curve simplification, our upper and lower bounds for the chord algorithm also apply for the Hausdorff distance (i.e. the maximum euclidean distance of a point in the actual curve from the approximate curve).

Consider the recursion tree built by the Chord algorithm. In the analysis we use the following convention: There is no node in the tree if, at the corresponding step, the routine terminates without calling the Comb routine (i.e. if $\mathcal{RD}(s, lr) \leq \epsilon$ in the above).

The following lemma is straightforward:

Lemma 6.2.2. *The set of points Q computed by the algorithm is an ϵ-convex Pareto set.*

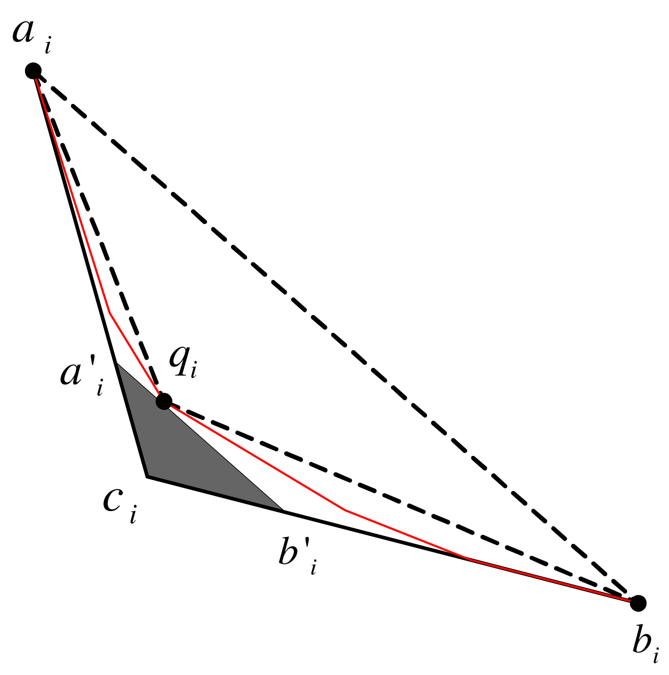

Figure 6.1: Illustration of the Chord algorithm.

6.2.3 Our Results.

We are now ready to state our main results.

Our first main result is an analysis of the chord algorithm on worst-case instances that is tight up to constant factors. In particular, for the ratio distance we prove

Theorem 6.2.3. *The worst-case performance ratio of the chord algorithm (wrt the ratio distance) is* $\Theta\big(\frac{m+\log(1/\epsilon)}{\log m+\log\log(1/\epsilon)}\big)$. *Furthermore, no algorithm can have performance ratio better than* $\Omega(\log m+\log\log(1/\epsilon))$.

The lower bound is proved in Section 6.3.1 (Theorem 6.3.1). In Section 6.3.2 we show our upper bound. We start by presenting a slightly weaker upper bound of $O(m+\log(1/\epsilon))$; this has the advantage that its proof is simple and intuitive. The proof of the asymptotically tight upper bound requires a more careful analysis and is presented next. The lower bound against general algorithms is given in Section 6.3.1.3 (Theorem 6.3.6).

Remark 6.2.4. It turns out that the Hausdorff distance behaves very similarly to the ratio distance.

In particular, by essentially identical proofs, it follows that the performance ratio of the Chord algorithm for approximating a convex curve of length L within Hausdorff distance ϵ, is $\Theta(\frac{\log(L/\epsilon)}{\log\log(L/\epsilon)})$. Furthermore, every algorithm has worst-case performance ratio at least $\Omega(\log\log(L/\epsilon))$.

We also analyze the Chord algorithm with respect to the horizontal distance metric (or by symmetry the vertical distance). We prove that in this setting the performance ratio of the algorithm is unbounded. In fact, we can prove a strong lower bound in this case: *Any* algorithm has an unbounded performance ratio (see Theorem 6.3.7).

Our second main result is a tight analysis of the Chord algorithm in an average case setting (wrt the ratio distance). Our random instances are drawn from two classes of standard distributions that have been widely used for the average case analysis of geometric algorithms in a variety of settings. In particular, we consider (i) a Poisson Point Process on the plane and (ii) n points drawn from "un-concentrated" product distribution. We now formally define these distributions.

Definition 6.2.5. *A (spatial, homogeneous) Poisson Point Process (PPP) of intensity λ on a bounded subset $\mathcal{S} \subseteq \mathbb{R}^d$ is a collection of random variables $\{N(A) \mid A \subseteq \mathcal{S}$ is Lebesgue measurable$\}$ (representing the number of points occurring in every subset of $\mathcal{S}$), such that: (i) for any Lebesgue measurable A, $N(A)$ is a Poisson random variable with parameter $\lambda \cdot S(A)$; (ii) for any collection of* disjoint *subsets $A_1, \dots, A_k$ the random variables $\{N(A_i), i \in [k]\}$ are mutually independent.*

Definition 6.2.6. *Let S be some bounded Lebesgue-measurable subset of $\mathbb{R}^2$, and let $\mathcal{D}$ be a distribution over S. The distribution $\mathcal{D}$ is called γ-*balanced*, $\gamma \in [0, 1)$, if for all Lebesgue measurable subsets $S' \subseteq S$, $\mathcal{D}(S') \in \left[(1-\gamma) \cdot \mathcal{U}(S'), \frac{\mathcal{U}(S')}{(1-\gamma)}\right]$, where $\mathcal{U}$ is the uniform distribution over S.*

We assume that γ is an absolute constant and we omit the dependence on γ in the performance ratio below. We prove:

Theorem 6.2.7. *For the aforementioned classes of random instances, the expected performance ratio of the Chord algorithm is $\Theta\big(\log m + \log\log(1/\epsilon)\big)$.*

The upper bound can be found in Section 6.4.1 and the lower bound in Section 6.4.2. We first present detailed proofs for the case of PPP and then present the (more involved) case of Product distributions.

We note that similar results apply also for the Hausdorff distance.

6.3 Worst–Case Analysis

6.3.1 Lower Bounds.

In this section we prove the aforementioned lower bounds. In Section 6.3.1.1 we prove a tight lower bound for the chord algorithm. In Section 6.3.1.3 we give the proof of our general lower bound.

6.3.1.1 Lower Bound for Chord Algorithm.

In fact, we prove a stronger result that also rules out the possibility of constant factor bi-criteria approximations, i.e. the lower bound applies even if the algorithm is allowed error $O(\epsilon)$ and compare against the optimal ϵ-approximation.

Theorem 6.3.1. *Let $\mu \geq 1$ be an absolute constant. Let $\epsilon > 0$ be smaller than a sufficiently small constant and $m > 0$ be large enough. There exists an instance $\mathcal{I}_{LB} = \mathcal{I}_{LB}(\epsilon, m, \mu)$ such that $\mathrm{OPT}_{\epsilon}(\mathcal{I}_{LB}) = O(1)$ and $\mathrm{CHORD}_{\mu\cdot\epsilon}(\mathcal{I}_{LB}) = \Omega\Big((1/\mu)\cdot\frac{m+\log(1/\epsilon)}{\log m+\log\log(1/\epsilon)}\Big)$.*

Proof. The lower bound applies even if an exact Comb routine is available, hence we restrict ourselves to this case for the proof. Before we proceed with the formal proof, we give an explanation of our construction for the case $\mu = 1$ and $m = 1$. The (rough) intuition is that the algorithm can perform poorly when the input instance is "skewed", i.e. we have a triangle $\triangle(abc)$ where $(ac) >> (bc)$. For such instances one can force the algorithm to select many "redundant" points (hence, perform many calls to Comb) to guarantee an ϵ-approximation, even when few points (calls) suffice.

For the special case under consideration, the hard instance has endpoints $a = (1, 2)$ and $b = (1+2\epsilon, 1)$, where ϵ is sufficiently small. Initially, the only available information is that the instance lies in the right triangle $\triangle(acb)$, where $c = (1, 1)$. Observe that, for an instance with these endpoints, the original error (i.e. $\mathcal{RD}(c, ab)$) is roughly 2ϵ and one intermediate point q^* *always* suffices to ϵ-cover, i.e. the optimal size is (at most) 3. We want to define a sequence of points $\{q_1, \ldots, q_j\}$ – all of which will be vertices of the convex pareto set for the corresponding instance – that force the algorithm to select *all* the q_i's (in order of increasing i), until it finds $q^* = q_j$. That is, we want the algorithm to monotonically converge to the optimal point by visiting *all* the vertices of the instance in order.

Let $\lambda_{ab} = 1/(2\epsilon)$ be the slope of ab. In the first step, the algorithm calls $\mathrm{Comb}(\lambda_{ab})$ to find a solution point at maximum ratio distance from ab. We want this call to return q_1. The idea is to subdivide the (length of the) edge ac *geometrically* with ratio k – for an appropriately selected k – and place q_1 on ac so that $(q_1c) = (ac)/k$. Let $\ell(q_1)$ be the line parallel to ab through q_1. By definition, this line *supports* the objective space. Denote $q_1^* = \ell(q_1) \cap bc$. Then the error of the approximation $\{a, q_1, b\}$ equals $\mathcal{RD}(q_1^*, q_1b)$ (the error to the left of q_1 is 0). If $k \leq 2$, we are already done, since $x(q_1^*) \geq 1 + \epsilon$, which implies $\mathcal{RD}(q_1^*, q_1b) \leq \epsilon$. On the other hand, if $k \geq 1/\epsilon$, we are also done since $y(q_1) \leq 1 + \epsilon$, hence $\mathcal{RD}(q_1^*, q_1b) \leq \epsilon$. If $\omega(1) \leq k \leq o(1/\epsilon)$, it can be argued that $\mathcal{RD}(q_1^*, q_1b) \approx (q_1^*b) = 2\epsilon \cdot (1 - 1/k)$. (Observe that the RHS is actually the horizontal distance of q_1^* from q_1b.) Hence, for this regime the error decreased by only an additive $2\epsilon/k << \epsilon$ and the algorithm needs to recurse on the triangle $\triangle(q_1q_1^*b)$. Note that $\lambda_{q_1b} = \lambda_{ab}/k = (2\epsilon)^{-1}/k$. The algorithm now calls $\mathrm{Comb}(\lambda_{q_1b})$ and this call will return q_2. To select this point we repeat our "geometric subdivision trick". Recall that there are no points below the line $q_1q_1^*$. Let q_2' be the projection of q_2 on ac. We select q_2 on the segment $q_1q_1^*$ so that $(q_2'c) = (q_1c)/k$. Let $\ell(q_2)$ be the (supporting) line parallel to q_1b through q_2. Similarly, the error of the approximation $\{a, q_1, q_2, b\}$ equals $\mathcal{RD}(q_2^*, q_2b)$ where $q_2^* = \ell(q_2) \cap bc$. Now, if $(q_2'c) = (ac)/k^2 >> \epsilon$, we can approximate $\mathcal{RD}(q_2^*, q_2b) \approx (q_2^*b) \approx 2\epsilon \cdot (1 - 2/k)$, i.e. the error has decreased by another additive $2\epsilon/k$. We can repeat this process iteratively, where (roughly) in step i we select q_i on the line $q_{i-1}q_{i-1}^*$, so that the length of the projection satisfies $(q_i'c) = (q_{i-1}'c)/k$. The iterative process can continue, as long as $(q_i'c) >> \epsilon$. Also note that the number j of iterations cannot be more than $\approx k/2$ because $x(q_i) \approx 1 + i \cdot (2\epsilon/k)$ and $x(q^*) \leq 1 + \epsilon$. Since, $(q_i'c) = 1/k^i$ it turns out that the *optimal* choice of parameters is $2j \approx k \approx \frac{\log(1/\epsilon)}{\log\log 1/\epsilon}$.

We stress that the actual construction is more elaborate than the one presented in the intuitive explanation above. Also, to show a bi-criterion lower bound, we need to add one more point q_{j+1} so that the chord algorithm selects $\{q_1, \ldots, q_j\}$ until it covers by $\mu \cdot \epsilon$, while the last point q_{j+1} (along with the endpoints) suffice to ϵ-cover.

The formal proof comes in two steps. We first analyze the chord algorithm wrt to the horizontal distance metric. We show that the performance ratio of the chord algorithm is unbounded in this setting (this also holds for the vertical distance by symmetry). In particular, for every $k \in \mathbb{N}$, there exists an instance $\mathcal{I}_G$ (actually in the unit square) so that the chord algorithm has ratio k for additive error $1/2$. We then show that, for an appropriate setting of the parameters in $\mathcal{I}_G$, we obtain the instance $\mathcal{I}_{LB}$ that yields the desired lower bound for the ratio distance.

Step 1: The instance $\mathcal{I}_G(H, L, k, j)$ lies in the triangle $\triangle(abc)$, where $a = (1, 1+H)$, $b = (1+L, 1)$ and $c = (1, 1)$. The points a and b are (the extreme) vertices of the convex Pareto set. We introduce two additional parameters. The first one, $k \in \mathbb{N}$, is the ratio used in the construction to geometrically subdivide the length of the line ac in every iteration. The second one, $j \in \mathbb{N}$ with $j \in [1, k-1]$, is the number of iterations and equals the number of vertices in the instance.

We define a set of points $Q = \{q_i\}_{i=0}^{j+2}$ ordered in increasing x–coordinate and decreasing y–coordinate. Our instance will be the convex polygonal line with vertices the points in Q. We set $q_0 = a$ and $q_{j+2} = b$. The set of points $\{q_1, \ldots, q_{j+1}\}$ is defined recursively as follows:

1. The point q_1 has $x(q_1) = x(a)$ and $y(q_1) = y(c) + \big(y(a) - y(c)\big)/k$.

2. For $i \in [2, j+1]$ the point q_i is defined as follows: Let $\ell(q_{i-1})$ denote the line parallel to $q_{i-2}b$ through q_{i-1}. The point q_i is the point of this line having $y(q_i) = y(c) + \big(y(q_{i-1}) - y(c)\big)/(k+i-1)$.

We want to compute an ϵ_L-approximation – recall that the error is measured wrt the horizontal distance – with $\epsilon_L(L, k, j) \overset{\text{def}}{=} L \cdot \frac{k-1}{k+j-1}$. Also denote $\epsilon'_L(L, k, j) \overset{\text{def}}{=} L \cdot \frac{k-1}{k} \cdot \frac{j}{k+j-1}$. Note that $\epsilon'_L/\epsilon_L = j/k$.

See Figures 6.2 and 6.3 for a graphic illustration of the worst-case instances for the chord algorithm. We would like to stress that the figures are not drawn to scale. In particular, in the figures below we have $H = L$, while the actual lower bound for the ratio distance applies for $H >> L$; in particular, for $H = 2^m$ and $L = O(\epsilon)$.

We show the following:

Lemma 6.3.2. *The chord algorithm applied to the instance $\mathcal{I}_G$ (wrt horizontal distance) selects the sequence of points $\langle q_1, q_2, \ldots, q_j \rangle$ to get error ϵ_L, while the set $\{a, q_{j+1}, b\}$ attains error ϵ'_L.*

Proof of Lemma 6.3.2. We first provide a brief overview of the proof. For $i \in [j-1]$, let q_i^* be the intersection of the line $\ell(q_i)$ – the line parallel to $q_{i-1}b$ through q_i–with bc. The error of $\{a, q_{j+1}, b\}$ is exactly $\Delta x(q_1, aq_{j+1})$. Observe that $\Delta x(q_1, aq_{j+1}) < \Delta x(q_1, aq_j^*)$. By a triangle similarity argument we obtain $\Delta x(q_1, aq_j^*) = \epsilon'_L$, which yields the second statement. For the first statement,

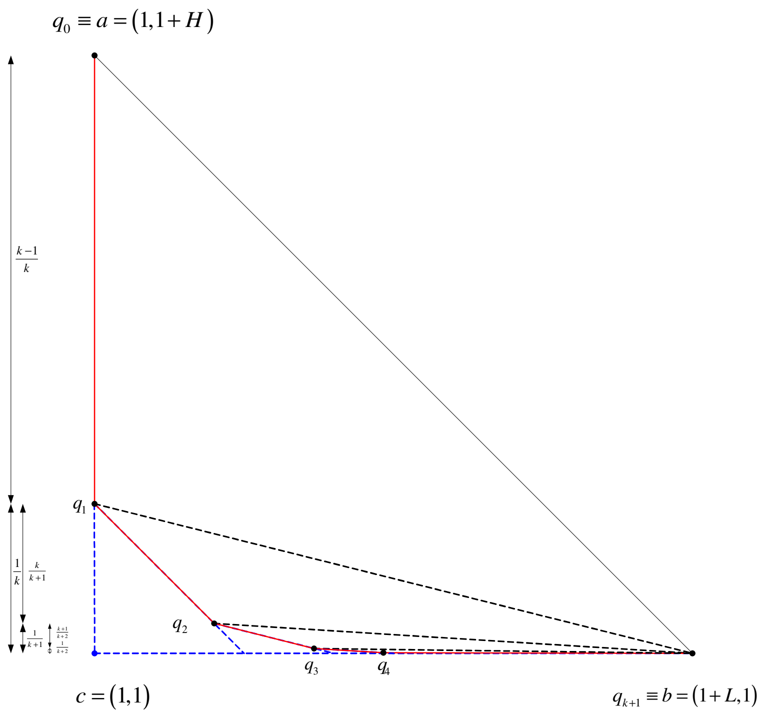

Figure 6.2: Lower bound for Chord. The figure depicts the case $j = k = 4$.

we show inductively that the recursion tree built by the algorithm for $\mathcal{I}_G$ is a path of length $j-1$ and at depth $i-1$, for $i \in [j]$, the chord subroutine selects point q_i. The proof amounts to noting that the error of the approximation $\{q_1, \dots, q_i\}$ is $\Delta x(q_i^*, q_i b)$, which is $> \epsilon_L$ for $i < j$ and $= \epsilon_L$ for $i = j$.

We now provide the details. We need some notation: For $i \in [j]$, we denote $\Delta x_i \overset{\text{def}}{=} \Delta x(q_i, q_{i-1}b)$. Let p_i be the y-projection of q_i on $q_{i-1}b$, so that $\Delta x_i = (q_i p_i)$. Also, for $i \in [j-1]$, let q_i^* be the intersection of the line $\ell(q_i)$ – the line parallel to $q_{i-1}b$ through q_i–with bc. If p_i^* is the y-projection of q_1 on aq_i^*, we have $\Delta x(q_1, aq_i^*) = (q_1 p_i^*)$. (See Figure 6.3 for an illustration of these definitions.) We start with the following claim:

Claim 6.3.3. *For all $i \in [j]$, we have $\Delta x_i = L \cdot (k-1)/(k+i-1)$.*

Proof of Claim 6.3.3. By induction on i. For the induction basis ($i = 1$), we observe that the

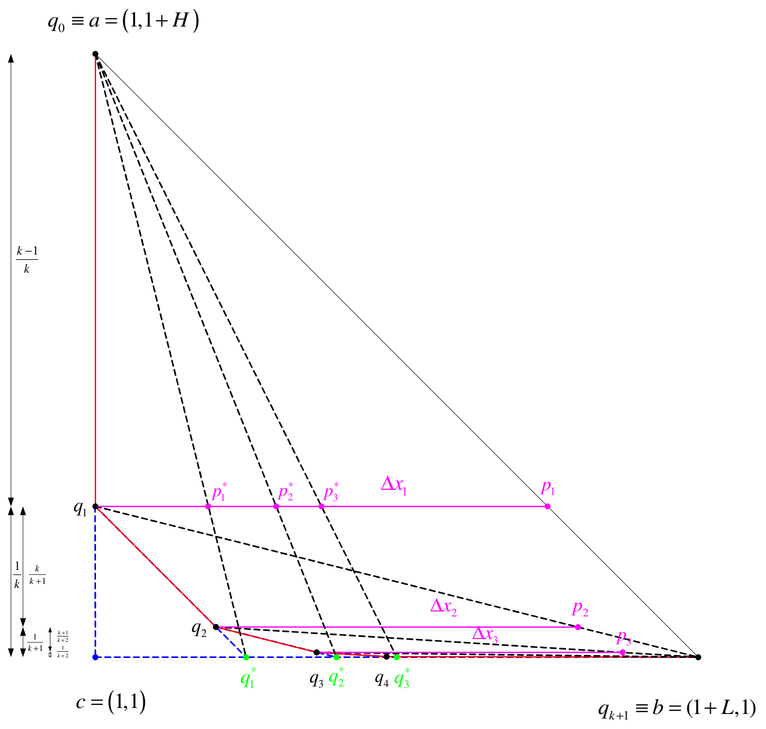

Figure 6.3: Illustration of definitions for counterexample in Figure 6.2.

triangles $\triangle(aq_1p_1)$ and $\triangle(acb)$ are similar, hence

$$\frac{\Delta x_1}{(bc)} = \frac{(aq_1)}{(ac)}$$

which yields $\Delta x_1 = (bc) \cdot (1 - 1/k) = L \cdot (k-1)/k$ as desired.

Suppose that the claim is true for $i \in [j-1]$. We will prove it for $i+1$. We similarly exploit the similarity of the triangles $\triangle(q_i q_i^* b)$ and $\triangle(q_i q_{i+1} p_{i+1})$, from which we get

$$\frac{\Delta x_{i+1}}{(q_i^* b)} = \frac{(q_i q_{i+1})}{(q_i q_i^*)} = \frac{y(q_i) - y(q_{i+1})}{y(q_i)} \tag{6.1}$$

where the second equality follows from the collinearity of q_i, q_{i+1}, q_i^*. Observe that the third term in the relation (6.1) above is equal to $(k+i-1)/(k+i)$ by construction. Now note that, because of the parallelogram $(q_i p_i b q_i^*)$, we have $(q_i^* b) = \Delta x_i$. Hence, (6.1) and the induction hypothesis now imply

$$\Delta x_{i+1} = \Delta x_i \cdot \frac{k+i-1}{k+i} = L \cdot \frac{k-1}{k+i-1} \cdot \frac{k+i-1}{k+i} = L \cdot \frac{k-1}{k+i}$$

which completes the proof. ∎

Since $(q_i^* b) = \Delta x_i$ (as noted in the proof of the above claim), it follows that for all $i \in [j-1]$ we have

$$x(q_i^*) = 1 + i/(k+i-1). \tag{6.2}$$

We will start by showing that the set $\{a, q_{j+1}, b\}$ is an ϵ'_L-convex Pareto (wrt to horizontal distance). (This implies that 3 calls to Comb suffice for the optimal algorithm, i.e. $\mathrm{OPT}_{\epsilon'_L} = 3$.) First, note that the error to the right of q_{j+1}, i.e. the distance of the lower envelope from $q_{j+1}b$ is actually zero ($q_{j+1}b$ is the rightmost edge of the lower envelope). It suffices to bound from above the error to its left. Since aq_{j+1} has absolute slope larger than ab, the unique point (of the lower envelope) at maximum distance from aq_{j+1} is q_1. We have that $\Delta x(q_1, aq_{j+1}) < \Delta x(q_1, aq_j^*)$. From the similarity of the triangles $\triangle(caq_j^*)$ and $\triangle(q_1 ap_j^*)$ we get $\Delta x(q_1, aq_j^*) = (1-1/k)\cdot(cq_j^*) = (1-1/k)\cdot(x(q_j^*) - x(c)) = L\cdot(1-1/k)\cdot\frac{j}{k+j-1} = \epsilon'_L$. Hence, $\Delta x(q_1, aq_{j+1}) < \epsilon'_L$ as desired.

We now proceed to analyze the behavior of the chord algorithm. We will show that the algorithm selects the points $q_1, q_2, \ldots, q_j$ (in this order) till it terminates. To this end, we prove: Consider the recursion tree built by the algorithm for the instance $\mathcal{I}_G$. The tree is a path of length $j-1$. In particular, for all $i \in [j]$, at depth $i-1$, the chord subroutine selects point q_i.

We prove the aforementioned statement by induction on the depth d of the tree. (Note that the chord algorithm initially finds the extreme points a and b.) For $d = 0$ (first recursive call), the algorithm selects a point of the lower envelope with maximum horizontal distance from ab. By construction, all the points (of the lower envelope) in the line segment q_1q_2 have the same (maximum) distance from ab (since q_1q_2 is parallel to ab). Hence, any of those points may be potentially selected. Since the Comb routine is a black-box oracle, we may indeed assume that and indeed q_1 is selected[†]. The maximum error after q_1 is selected equals $\Delta x(q_1^*, q_1 b) = x(b) - x(q_1^*) = L\cdot(1-1/k) > \epsilon_L$. Hence, the algorithm will not terminate after it has selected q_1.

For the inductive step, we assume that the recursion tree is a path up to depth $d \in [j-2]$ and the algorithm selected the points $\{q_1, q_2, \ldots, q_{d+1}\}$ up to this depth. We analyze the algorithm at depth $d+1$. At depth $d+1$ the algorithm knows that the error to the left of q_{d+1} is 0 and the error to its right is $\Delta x(q_{d+1}^*, q_{d+1}b) = L\cdot(k-1)/(k+d) > \epsilon_L$ (since $d \le j-2$). Hence, the algorithm does

[†]This simplifying assumption is not necessary. We can slightly perturb the instance so that the absolute slope of q_1q_2 is "slightly" smaller than λ_{ab}, so that the effect on the actual distances is negligible. By doing that, the point q_1 will be the "unique minimizer" for $\mathrm{Comb}(\lambda_{ab})$.

not terminate, but it calls Comb to find a point between q_{d+1} and b at maximum distance from $q_{d+1}b$. By construction, the points of maximum distance are those belonging to the line $q_{d+2}q_{d+3}$ (which is parallel to $q_{d+1}b$); hence, we can assume q_{d+2} is selected. At this point the algorithm knows that the error to the left of q_{d+2} is 0. The error to its right is is $\Delta x(q^*_{d+2}, q_{d+2}b) = L\cdot(k-1)/(k+d+1) > \epsilon_L$, unless $d = j - 2$. This completes the proof of the lemma. ■

Step 2: We now define the instance $\mathcal{I}_{LB}$. Fix $H^* := 2^m - 1$, $L^* := (\mu + 1)\cdot\epsilon$, $j^* := \Theta\big((1/\mu)\cdot\frac{\log(H^*/\epsilon)}{\log\log(H^*/\epsilon)}\big)$ and $k^* := \mu \cdot j^* + 1$. We set $\mathcal{I}_{LB}(\epsilon, m, \mu) := \mathcal{I}_G(H^*, L^*, k^*, j^*)$. Also, let ${\epsilon_L}^* := \epsilon_L(L^*, k^*, j^*)$ and $\epsilon'^*_L := \epsilon'_L(L^*, k^*, j^*)$. Observe that, under this choice of parameters, we have ${\epsilon_L}^* \geq \mu\cdot\epsilon$ and ${\epsilon'_L}^* < \epsilon$. We show:

Lemma 6.3.4. *The chord algorithm applied to $\mathcal{I}_{LB}$ selects (a superset of) the points $\{q_1, \dots, q_{j^*/8}\}$ to attain ratio distance ${\epsilon_L}^*$, while the set $\{a, q_{j^*+1}, b\}$ is an ${\epsilon'_L}^*$-convex Pareto set.*

Proof. The proof amounts to proving that *for the particular instance*, the horizontal distance is a very good approximation to the ratio distance. Hence, the behavior of the algorithm in these metrics is similar. The reason we lose a constant factor in the number of points (i.e. $j^*/8$ as opposed to j^*) is due to the error term in the approximation between the metrics. We now proceed with the details. Consider the set $Q = \{q_i\}_{i=0}^{j^*+2}$ defining the instance $\mathcal{I}_{LB}$. The proof makes essential use of the following lemma that quantifies the closeness of the two metrics in the current setting. Its proof is deferred to the end of this section.

Lemma 6.3.5. *Let s_1 be a point in $\triangle(abc)$ and denote by λ the absolute slope of s_1b. Let c' be the x-projection of s_1 on bc. For any point s_2 in $\triangle(s_1c'b)$, we have*

$$\mathcal{RD}(s_2, s_1b) \leq \Delta x(s_2, s_1b) \leq \mathcal{RD}(s_2, s_1b) + O((L^*)^2 + L^*/\lambda).$$

By Lemma 6.3.2, the set $\{a, q_{j^*+1}, b\}$ attains horizontal error at most ${\epsilon'_L}^*$. By the first inequality of Lemma 6.3.5, it follows that the multiplicative error of $\{a, q_{j^*+1}, b\}$ is at most as big, hence we directly obtain the second statement.

By Lemma 6.3.2, we also know that the chord algorithm under the horizontal distance selects all the points $\{q_1, \dots, q_{j^*}\}$ to find an ϵ^*_L-convex cover. As explained in its proof, after the algorithm has selected the subset $\{q_1, \dots, q_i\}$, for $i \in [j^*]$, the horizontal approximation error is $\Delta x(q^*_i, q_ib) = L^*\cdot(k^* - 1)/(k^* + i - 1)$.

We remark that the error term in the Lemma 6.3.5 leads to the "constant factor loss", i.e. the fact that the chord algorithm under the ratio distance picks $j^*/8$ (as opposed to j^*) points. Note that, since the ratio distance is a lower bound, the chord algorithm will select at most j^* points.

Suppose we invoked the chord algorithm with error 0, i.e. we wanted to find the lower envelope exactly. Then the algorithm would select the points q_i in order of increasing i. It is also clear that the error of the approximation decreases monotonically with the number of steps. It thus suffices to show that after the algorithm has selected $\{q_1, \ldots, q_{j^*/8}\}$, the multiplicative error will be bigger than ϵ^*_L. To do this, we appropriately apply Lemma 6.3.5.

Indeed, the multiplicative approximation error of the set $\{q_1, \ldots, q_{j^*/8}\}$ is $\mathcal{RD}(q^*_{j^*/8}, q_{j^*/8}b)$. Applying (the right inequality of) Lemma 6.3.5 for $s_1 = q_{j^*/8}$, $s_2 = q^*_{j^*/8}$ we get

$$\mathcal{RD}(q^*_{j^*/8}, q_{j^*/8}b) \geq \Delta x(q^*_{j^*/8}, q_{j^*/8}b) - O\big((L^*)^2 + L^*/\lambda_{q_{j^*/8}b}\big).$$

For the first term of the RHS we have

$$\Delta x(q^*_{j^*/8}, q_{j^*/8}b) > \epsilon^*_L + L^*/4$$

We now bound from above the error term. The first summand of the error is $(L^*)^2 = \mu^2 \cdot \epsilon^2$ that is negligible compared to $L^*/4$, as long as $\epsilon << 1/\mu$. For the second summand we need a lower bound on the slope $\lambda_{q_{j^*/8}b}$.

Recall Equation (6.2) in Lemma 6.3.3: $x(q^*_i) = 1 + L^* \cdot i/(k^* + i - 1)$. It is not hard to verify that $|x(q_i) - x(q^*_{i-1})| = O(L^*/(k^*)^i)$. Also recall that $y(q_i) = 1 + H^*/\prod_{j=1}^{i}(k^* + j - 1) > 1 + H^*/(2k^*)^{-i}$. Hence, it follows that $\lambda_{q_ib} > (H^*/L^*)/(2k^*)^{-i} = (2^m/\epsilon)/(2k^*)^{-i}$. It is thus easy to see that for the chosen value of j^* the corresponding slope $\lambda_{q_{j^*/8}b}$ will be much larger than μ, hence the second term of the error will be also outweighed by $L^*/4$.

By combining the aforementioned, we get that the actual ratio distance $\mathcal{RD}(q^*_{j^*/8-1}, q_{j^*/8-1}b)$ will be strictly bigger than ϵ^*_L and Lemma 6.3.4 follows. ■

This completes the proof of Theorem 6.3.1. ■

6.3.1.2 Proof of Lemma 6.3.5.

For convenience we restate the lemma:

Lemma 6.3.5. *Let s_1 be a point in $\triangle(abc)$ and denote by λ the absolute slope of $s_1 b$. Let c' be the x-projection of s_1 on bc. For any point s_2 in $\triangle(s_1 c' b)$, we have*

$$\mathcal{RD}(s_2, s_1 b) \leq \Delta x(s_2, s_1 b) \leq \mathcal{RD}(s_2, s_1 b) + O((L^*)^2 + L^*/\lambda).$$

Proof. Let $c' = (1+\delta, 1)$ be the x-projection of s_1 on the segment cb. If λ is the absolute slope of $s_1 b$, we have that $y(s_1) = 1 + \lambda \cdot (L^* - \delta)$. Now fix a point $s_2 = (1+\delta_x, 1+\delta_y)$ in $\triangle(s_1 c' b)$. We want to show that the horizontal distance of s_2 from $s_1 b$ is a good approximation to the corresponding ratio distance when λ is large. (See Figure 6.4 for an illustration.)

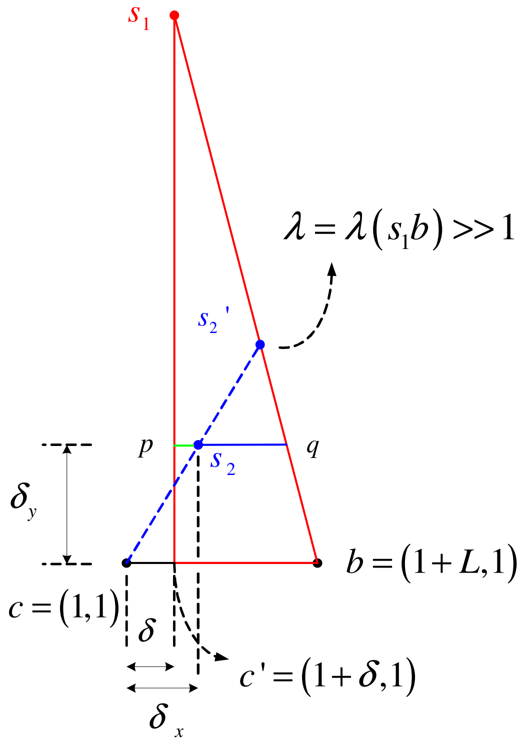

Figure 6.4: Illustration of the relation between the horizontal and the ratio distance.

We first calculate the horizontal distance $\Delta x(s_2, s_1 b)$. Observe that

$$\Delta x(s_2, s_1 b) = (s_2 q) = (pq) - (p s_2).$$

It is clear that $(ps_2) = x(s_2) - x(p) = \delta_x - \delta$. From the similarity of the triangles $\triangle(s_1pq)$ and $\triangle(s_1c'b)$ we get

$$\frac{(pq)}{(c'b)} = \frac{(s_1p)}{(s_1c')}.$$

Since $(c'b) = L^* - \delta$, $(s_1c') = \lambda \cdot (L^* - \delta)$ and $(s_1p) = (s_1c') - (pc') = (s_1c') - \delta_y$ we get

$$(pq) = (L^* - \delta) \cdot \Big(1 - \frac{\delta_y}{\lambda \cdot (L^* - \delta)}\Big).$$

Therefore,

$$\Delta x(s_2, s_1b) = (L^* - \delta) \cdot \Big(1 - \frac{\delta_y}{\lambda \cdot (L^* - \delta)}\Big) - (\delta_x - \delta) = L^* - \delta_x - \frac{\delta_y}{\lambda}.$$

The ratio distance $\mathbf{r} \overset{\text{def}}{=} \mathcal{RD}(s_2, s_1b)$ by definition is such that $s_2' = (1 + \mathbf{r}) \cdot s_2 \in s_1b$. We thus get

$$(1 + \mathbf{r}) \cdot y(s_2) + \lambda(1 + \mathbf{r}) \cdot x(s_2) = y(b) + \lambda x(b)$$

or equivalently

$$\mathbf{r} = \frac{\lambda\big(x(b) - x(s_2)\big) - \big(y(s_2) - y(b)\big)}{y(s_2) + \lambda x(s_2)}.$$

Substitution yields that

$$\mathbf{r} = \frac{\lambda(L^* - \delta_x) - \delta_y}{1 + \delta_y + \lambda(1 + \delta_x)} = \frac{L^* - \delta_x - \frac{\delta_y}{\lambda}}{1 + \delta_x + \frac{1}{\lambda} + \frac{\delta_y}{\lambda}}.$$

Observe that the numerator of the above fraction equals $\Delta x(s_2, s_1b)$ and the denominator is at least 1. Hence, $\mathcal{RD}(s_2, s_1b) \leq \Delta x(s_2, s_1b)$. For the other inequality we have:

$$\begin{aligned}
\Delta x(s_2, s_1b) - \mathcal{RD}(s_2, s_1b) &= \Big(L^* - \delta_x - \frac{\delta_y}{\lambda}\Big) \cdot \Big(1 - \frac{1}{1 + \delta_x + \frac{1+\delta_y}{\lambda}}\Big) \\
&= \Big(L^* - \delta_x - \frac{\delta_y}{\lambda}\Big) \cdot \Big(\frac{\delta_x + \frac{1}{\lambda} + \frac{\delta_y}{\lambda}}{1 + \delta_x + \frac{1}{\lambda} + \frac{\delta_y}{\lambda}}\Big) && (6.3) \\
&\leq L^* \cdot (2L^* + 1/\lambda) && (6.4) \\
&= O((L^*)^2 + L^*/\lambda)
\end{aligned}$$

as desired. To obtain the inequality (6.4), we bound the product (6.3) term by term. The first term is clearly at most L. For the second term, the denominator is at least 1. For the numerator, we use the fact that $\delta_x \leq L^*$ and $\frac{\delta_y}{\lambda} \leq L^* - \delta_x \leq L^*$ (which holds because s_2 was assumed to lie in $\triangle(s_1c'b)$). This completes the proof. ■

6.3.1.3 General Lower Bounds.

We can prove a (weaker) lower bound against *any* algorithm that rules out the possibility of a constant factor approximation for minimizing the number of Comb Calls. In particular, we have

Theorem 6.3.6. *For any algorithm (even randomized), there exists an instance such that the performance ratio of the algorithm is* $\Omega(\log m + \log\log(1/\epsilon))$.

Proof. The proof uses the construction of the previous subsection as a black box. It works by essentially reducing the computation of an ϵ-CP (given access to Comb) to a comparison-based binary search on a set of cardinality j^*.

Recall that we consider algorithms that have access to Comb and measure the number of calls made by the algorithm as compared to the minimum possible for the given instance. The proof uses the lower bound $\mathcal{I}_{LB}$ from the previous theorem essentially as a black box. For simplicity, let $\mu = 1$ and $Q = \{q_1, \ldots, q_{j^*}\}$ be the corresponding set of points. Suppose the error we care about is $\epsilon > \mathcal{RD}(q^*_{j^*}, q_{j^*}b)$. Define the family of "prefixes" $\mathcal{Q} = \{Q_i\}_{i=1}^{j^*}$ of Q, where $Q_i = \{q_1, \ldots, q_i\}$, $i \in [j^*]$. It follows from the properties of the set Q that the convex polygonal instance corresponding to each Q_i has the following property: its last point (i.e. q_i) suffices to ϵ-cover (together with the endpoints), while the set $Q_i \setminus q_i$ does not.

Consider a general algorithm $\mathcal{A}$. In order to find an ϵ-convex Pareto, it must be able to "distinguish" among Q_i's. Since otherwise, it cannot detect the rightmost point of the given instance, hence cannot guarantee an ϵ-approximation. This essentially means that the algorithm must perform a binary search on the slopes $\{\lambda_{q_i b}\}_{i=1}^{j^*}$. Having reduced the problem to a comparison-based binary search on a set of cardinality j^* a lower bound of $\Omega(\log j^*)$ follows (for randomized algorithms as well). ■

■

Our next theorem says that, if our metric is the horizontal/vertical distance, then every algorithm with access to a Comb routine has an unbounded performance ratio. This holds even for instances that lie in the unit square and for additive error $1/2$. The proof of the following theorem builds on the lower bound construction for the chord algorithm (Section 6.3.1.1, Step 1).

Theorem 6.3.7. *For any $k \in \mathbb{N}$ and any algorithm $\mathcal{A}$, there exists an instance for which the performance ratio of $\mathcal{A}$ wrt horizontal/vertical distance is $\Omega(k)$. This is true even for instances that lie in the unit square and for error (horizontal/vertical distance) $1/2$.*

Proof. We start from an instance in the unit square with endpoints $a = (0, 1)$ and $b = (1, 0)$. The algorithm $\mathcal{A}$ has the following properties: It "knows" the value of the desired approximation ϵ. We set $\epsilon = 1/2$. It adaptively selects a sequence of absolute slopes (inputs to calls to the Comb routine) $\lambda_1, \lambda_2, \dots, \lambda_k$ and terminates when it has a "certificate" that the error of the approximation it has computed is at most ϵ. In the argument below, the adversary *forces* the algorithm to select a *decreasing* sequence of absolute slopes $\lambda_1, \lambda_2, \dots, \lambda_k$, i.e. $\lambda_i < \lambda_{i+1}$. It is very similar to the chord counterexample. The main reason that it works is that for the horizontal distance there is no finite "ϵ-net" (i.e. obliviously selected finite set of slopes that suffices). This is also why we get *infinite* ratio in the lower bound.

The algorithm starts by making its first call λ_1 to Comb (knowing the endpoints only). The adversary "sees" λ_1 and "restricts" the instance accordingly. In particular, in our context, it suffices for the adversary to add one vertex q_1 appropriately – so that $q_1 = \mathrm{Comb}(\lambda_1)$. Then the algorithm makes its second call λ_2, based on the current information it has about the instance: q_1 and the lower bound (supporting line with slope λ_1 through q_1). The adversary selects q_2, based on λ_1, λ_2. The algorithm selects λ_3 based on the current instance; and so forth.

The whole point that makes the proof simple is that we can use an inductive argument in the current triangle – similar to the one for the chord rule – while maintaining a basic invariant. The only invariant we need is this: the point q_i^* has x-coordinate less than $1/2$. If this is the case, then it is possible that one point $1/2$-covers q_1 (hence the entire instance). The adversary argument is this: If the slope λ_i selected by $\mathcal{A}$ in the $(i + 1)$-th step of the computation is greater or equal to the slope that the chord would have used, select the point q_{i+1} in the same way as the chord counterexample. Otherwise, select q_{i+1} (on the line $q_i q_i^*$ as before) such that $y(q_{i+1})$ is "scaled down" appropriately, so that q_{i+1}^* has "not too large" x-coordinate. This immediately guarantees that the distance of q_{i+1} from $q_i b$ is larger than ϵ.

The following figure illustrates the i-th step of the lower bound.

From this figure we immediately get that $\Delta x_{i+1} \le 1/(2k)$, hence $x(q_{i+1}^*) - x(q_i^*) \le 1/(2k)$. So, in every step we move at most a $1/(2k)$ additive to the right.

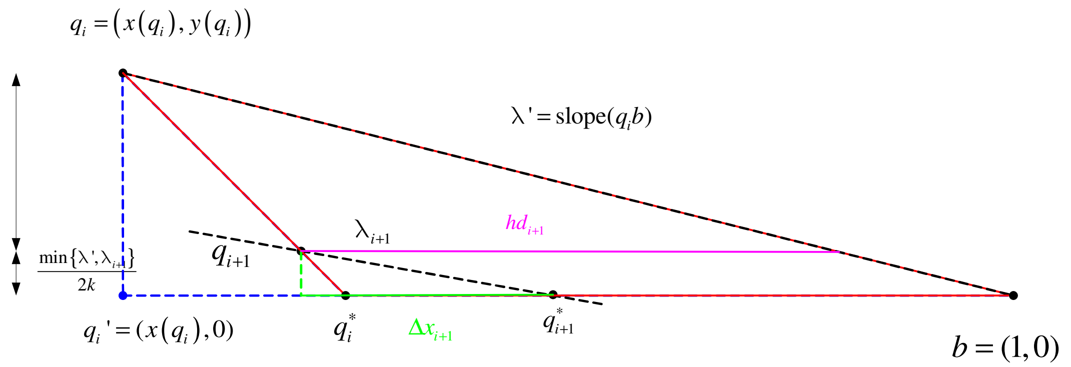

Figure 6.5: General lower bound for horizontal distance.

We also need to show that the distance decreases very slowly. For this, we have $\Delta x_{i+1} \geq (q_{i+1}^* b)$. This it true because the line $q_{i+1} q_{i+1}^*$ has slope either λ', when $\lambda_{i+1} > \lambda'$, or λ_{i+1} otherwise. So, in every step the distance deceases by at most $1/(2k)$.

Thus, in k steps the error of the algorithm remains more than $1/2$, while the distance of the optimal solution (e.g. q_k^*) is at most $1/2$. This completes the proof. ■

6.3.2 Upper Bound.

In this section we prove that the performance ratio of the chord algorithm wrt the ratio distance is $O\left(\frac{m+\log(1/\epsilon)}{\log m+\log\log(1/\epsilon)}\right)$. For the sake of the exposition, we start by showing the slightly weaker upper bound of $O(m + \log(1/\epsilon))$. The proof of the tight upper bound is more involved and builds on the understanding obtained from the simpler argument establishing the weaker upper bound.

6.3.2.1 An $O(m + \log(1/\epsilon))$ Upper Bound.

Before we proceed with the argument, some comments are in order. Perhaps the most natural approach to prove an upper bound would be to argue that the error of the approximation constructed by the chord algorithm decreases substantially (say by a constant factor) in every subdivision. This, if true, would yield the desired result – since the initial error cannot be more than $2^{O(m)}$. Unfortunately, such an approach badly fails, as implied by the construction of Theorem 6.3.1. Recall that, in the simplest setting of that construction, the initial error is 2ϵ and decreases by an additive $2\epsilon/k$ in every iteration, where $k \approx \log(1/\epsilon)/\log\log(1/\epsilon)$. Hence, the error decreases by a sub-constant factor in every iteration. In fact, we note that such an argument cannot hold for *any* algorithm (given access to Comb), as follows from our general lower bound (Theorem 6.3.6)[†].

Our approach is somewhat indirect. We prove that the *area* between the "upper and lower approximation" decreases by a constant factor in every step of the algorithm. This can be interpreted as a "potential function type argument." It is not hard to argue that, when the area has become "small enough" (roughly at most $\epsilon^2/2^{2m}$), the error of the approximation (ratio distance of the lower approximation from the upper approximation) is at most ϵ. We then use a simple charging argument to show that this suffices to yield the desired performance guarantee. Formally, we prove:

Theorem 6.3.8. *Let T_1 be the triangle at the root of the chord algorithm's recursion tree and denote $\alpha \overset{def}{=} \min\{x(q), y(q) \mid q \in T_1\}$. The chord algorithm finds an ϵ-convex Pareto set after at most $O\Big(\log\big(S(T_1)/S_0\big)\Big) \cdot \mathrm{OPT}_\epsilon$ calls to* Comb*, where $S_0 \overset{def}{=} \epsilon^2 \cdot \alpha^2/2$.*

The desired result follows from the above, since $T_1 \subseteq [2^{-m}, 2^m]^2$, which implies $S(T_1) \le 2^{2m}$ and $\alpha \ge 2^{-m}$.

Proof. It follows from Lemma 6.2.2 that, upon termination, the algorithm has found an ϵ-CP set.

To upper bound the performance ratio we need a few lemmas. Our first lemma quantifies the area shrinkage property. We remark that this is a statement independent of ϵ.

Lemma 6.3.9. *Let $T_i = \triangle(a_i b_i c_i)$ be the triangle processed by the chord algorithm at some recursive step. Denote $q_i = \mathrm{Comb}(\lambda_{a_i b_i})$. Let $T_{i,l} = \triangle(a_i a_i' q_i)$, $T_{i,r} = \triangle(b_i b_i' q_i)$ be the triangles corresponding to the two new subproblems. Then, we have*

$$S(T_{i,l}) + S(T_{i,r}) \le S(T_i)/4.$$

Proof. Let d_i', e_i' be the projections of q_i on $b_i c_i$ and $a_i c_i$ respectively; see Figure 6.6 for an illustration. Let

$$y = (a_i' c_i)/(a_i c_i) = (b_i' c_i)/(b_i c_i) \in (0,1).$$

[†]In [RF] the authors – in essentially the same model as ours – propose a variant of the Chord algorithm (that appropriately subdivides the current triangle into three sub-problems). They claim (Lemma 3 in [RF]) that the error reduces by a factor of 2 in every such subdivision. However, their proof is incorrect. In fact, our counterexample from Section 6.3.1 implies the same lower bound for their proposed heuristic as for the chord algorithm.

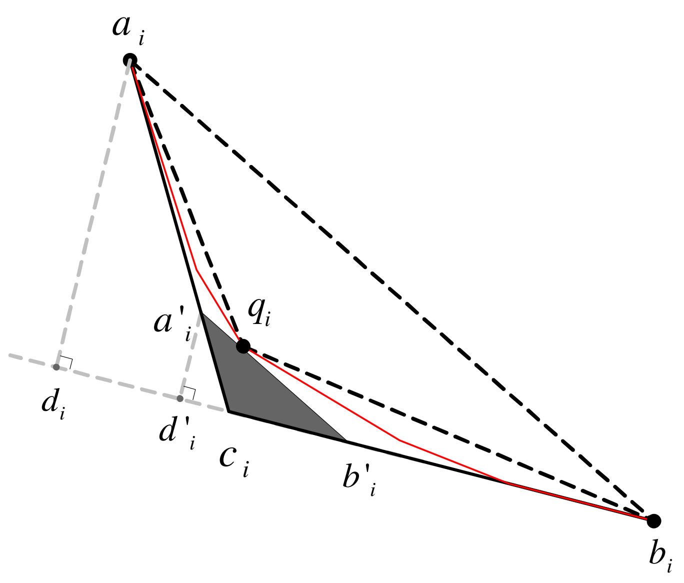

Figure 6.6: Area shrinkage property of the Chord algorithm.

Then,

$$S(\triangle(a'_i b'_i c_i)) = y^2 S(T_i).$$

We have

$$\begin{aligned} S(T_{i,l}) \quad + \quad S(T_{i,r}) &= \big((a_i a'_i)\cdot(q_i e'_i) + (b_i b'_i)\cdot(q_i d'_i)\big)/2 \\ &= (1-y)\cdot\big((a_i c_i)\cdot(q_i e'_i) + (b_i c_i)\cdot(q_i d'_i)\big)/2 \\ &= (1-y)\cdot\big(S(\triangle(a_i c_i q_i)) + S(\triangle(b_i c_i q_i))\big) \\ &= (1-y)\cdot\big(S(T_{i,l}) + S(T_{i,r}) + S(\triangle(a'_i b'_i c_i))\big) \end{aligned}$$

which gives

$$S(T_{i,l}) + S(T_{i,r}) = y\cdot(1-y)\cdot S(T_i) \le S(T_i)/4$$

as desired. ■

Our second lemma gives a convenient lower bound on the value of the optimum.

Lemma 6.3.10. *Consider the recursion tree $\mathcal{T}$ [‡] built by the algorithm and let $\mathcal{L}'$ be the number of*

[‡]We use the following convention: There is no node in the tree if, for a triangle, the routine terminates without calling the Comb routine.

lowest non-leaf nodes. Then $\mathrm{OPT}_\epsilon \geq |\mathcal{L}'|$.

Proof. Each such node in the tree corresponds to a triangle $T_i = \triangle(a_i b_i c_i)$ with the property that the ratio distance of the convex pareto set from the segment $a_i b_i$ is strictly greater than ϵ (o/w the node would be a leaf). Hence, each such triangle must contain a point of the optimal ϵ-convex Pareto set. Since all these triangles are disjoint, the result follows. ∎

Finally, we need a lemma that relates the ratio distance within a triangle to its area:

Lemma 6.3.11. *Consider a triangle* $T_i = \triangle(a_i b_i c_i)$ *such that* $T_i \subseteq T_1$. *If* $S(T_i) \leq \epsilon^2 \cdot \alpha^2/2$, *then* $\mathcal{RD}(c_i, a_i b_i) \leq \epsilon$.

Proof of Lemma 6.3.11. The above lemma follows essentially by observing that the worst-case for the area-error trade-off is when $c_i = (\alpha, \alpha)$, and the triangle is right and isosceles (i.e. $(a_i c_i) = (a_i b_i)$). ∎

At this point we have all the tools we need to complete the proof of the theorem. First, by Lemma 6.3.9, when a node of the tree is at depth $\log(S(T_1)/S_0)$, it will have area at most S_0. By Lemma 6.3.11, this implies that the depth of the tree $\mathcal{T}$ is $O(\log(S(T_1)/S_0))$. Lemma 6.3.10 implies that $\mathrm{CHORD}_\epsilon \leq O(\log(S(T_1)/S_0)) \cdot \mathrm{OPT}_\epsilon$, which concludes the proof. ∎

6.3.2.2 Tight Upper Bound.

In this subsection, we prove the asymptotically tight upper bound of $O\big(\frac{m+\log(1/\epsilon)}{\log m+\log\log(1/\epsilon)}\big)$ on the worst-case performance ratio of the Chord algorithm. The analysis is more involved in this case and builds on the intuition obtained from the simple analysis of the previous subsection. The crucial observation used to improve upon the aforementioned bound is this: In each iteration of the algorithm, there is a trade-off between the following two quantities: (i) the area between the upper and lower approximations and (ii) the approximation error (ratio distance). In particular, the intuition is that if, in some iteration, the area reduces moderately (say, only by a constant factor), then the ratio error must reduce substantially. The argument below formalizes this intuition.

We now proceed with the formal proof. We begin by analyzing the case $\mathrm{OPT}_\epsilon = 3$ (i.e. the special case that one intermediate point suffices – and is required – for an ϵ-approximation) and then handle the general case. It turns out that this special case captures most of the difficulty in the analysis.

Let a (leftmost) and b (rightmost) be the extreme points of the convex Pareto curve as computed by the algorithm. We consider the case $\mathrm{OPT}_\epsilon = 3$, i.e. (i) the set $\{a, b\}$ is *not* an ϵ-CP and (ii) there exists a solution point q^* such that $\{a, q^*, b\}$ is an ϵ-CP.

Fix $k \in \mathbb{N}$ with $k = \Omega\big(\frac{m+\log(1/\epsilon)}{\log m+\log\log(1/\epsilon)}\big)$. We will prove that, for an appropriate choice of the constant in the big-Omega, the Chord algorithm introduces at most k points in either of the intervals $[a, q^*]$, $[q^*, b]$. Suppose, for the sake of contradiction, that the Chord algorithm adds more points than that in the segment aq^* (the proof for q^*b being symmetric.)

We say that, in some iteration of the Chord algorithm, a triangle is *active*, if it contains the optimal point q^*. In each iteration, the Chord algorithm has an active triangle which contains the optimal point q^*. Outside that triangle, the algorithm has constructed an ϵ-approximation. We note that the Chord algorithm may in principle go back and forth between the two sides of q^*; i.e., in some iterations the line parallel to the chord touches the lower envelope to the left of q^* and in other iterations to the right.

Let $\triangle(abc)$ be the initial triangle. We focus our attention on the (not necessarily consecutive) iterations of the Chord algorithm that add points to the left of q^*. We index these iterations in increasing order with $i \in [1, k+1]$. Consider the "active" triangle $\triangle(a_ic_ib_i)$ generated in each such iteration, where a_i is the new point of the curve added in the iteration. It is clear that (i) a_{i+1} lies to the right of a_i, (ii) b_{i+1} lies to the left of (or is identified with) b_i and (iii) $\triangle(a_{i+1}c_{i+1}b_{i+1}) \subseteq \triangle(a_ic_ib_i)$. Also, let us denote $b' := b_{k+1}$, that is b' is the b-vertex (i.e. the vertex that lies to the right of q^*) of the active triangle in the last iteration $k + 1$. Note that b' could be identified with the point b (which would happen if the Chord algorithm introduces points only to the left of q^*, i.e. converges to q^* monotonically), but in general this will not be the case.

Let e_i be the intersection point of the line a_ic_i with the line $b'q^*$. Note that, to the left of q^* the convex Pareto curve has no points below the line $b'q^*$. All the point of the convex Pareto curve that are left of q^* and lie in the active triangle $\triangle(a_ic_ib_i)$ (these are the potentially not ϵ-covered points) are actually in the triangle $\triangle(a_ie_iq^*)$. Consider the line ℓ that goes through a_i and is parallel to $a_{i-1}b'$ and let d_i be its intersection with $b'q^*$. Note that the line a_ic_i is parallel to $a_{i-1}b_i$ (by construction of the algorithm, this is the line that added the point a_i and formed the active triangle

$\triangle(a_ic_ib_i)$) and b_i lies to the right of (or is identified with) b', so the line a_id_i is to the left of a_ic_i, hence d_i lies to the left of e_i. Now, let f_i be the intersection point of $a_{i-1}a_i$ with the line $b'q^*$. Clearly, f_i lies to the left of d_i. Furthermore, f_i lies to the right of (or is identified with) d_{i-1}. The reason is that, below a_{i-1} the curve has no points (strictly) to the left of the line $a_{i-1}d_{i-1}$, so a_i is to the right of the line (or on the line).

Consider the sequence of triangles $\triangle(a_id_ib')$. Let H_i be the ratio distance of a_i from the line $b'q^*$ for $i \geq 0$. Let G_i be the ratio distance of b' from the line a_id_i for $i \geq 1$. Let $P_i = H_i/H_{i-1}$ and let $R_i = 1 - P_i$.

By definition, we have $H_i = P_i \cdot H_{i-1}$ for all i, and since $H_0 \leq 2^{O(m)}$ we get

$$H_k \leq 2^{O(m)} \cdot \prod_{i=1}^{k} P_i.$$

Also, $(d_ib') = R_i \cdot (f_ib')$ and since d_{i-1} lies left of f_i, we have $(d_ib') \leq R_i \cdot (d_{i-1}b')$, and therefore

$$(d_kb') \leq (d_1b') \cdot \prod_{i=2}^{k} R_i.$$

Similarly, G_i is upper bounded by R_i times the ratio distance from b' to the line $a_{i-1}f_i$, which is in turn upper bounded by the ratio distance from b' to the line $a_{i-1}d_{i-1}$ (i.e. G_{i-1}). Thus, $G_i \leq R_i \cdot G_{-1}$ and

$$G_k \leq G_1 \cdot \prod_{i=2}^{k} R_i.$$

Since the last iteration of the Chord algorithm adds a new point a_{k+1}, we must have $H_k > \epsilon$ and $G_k > \epsilon$ (since otherwise, the segment a_kb' ϵ-covers all the Pareto points in the active triangle). Thus, we get

$$\prod_{i=1}^{k} P_i > \epsilon/2^{O(m)}$$

and

$$G_1 \cdot \prod_{i=2}^{k} R_i > \epsilon.$$

We now consider two subcases.

Case I: $G_1 \leq 2\epsilon$.

In this case, the second inequality above gives $\prod_{i=2}^{k} R_i > 1/2$. For a fixed product of the P_i's, the product of the R_i's is maximized if all factors are equal. That is, $P_i = 1/t$ for all i and $R_i = 1 - 1/t$, which gives $k = O(t)$ and $t = O\big(\frac{m+\log(1/\epsilon)}{\log m + \log\log(1/\epsilon)}\big)$.

Case II: $G_1 > 2\epsilon$.

In this case, the point a_1 is at ratio distance at most ϵ from the line aq^*, which implies that q^* is at most ratio distance ϵ from the line a_1d_1 (because a_1d_1 is parallel to ab' and b' is to the right of q^*). Since $G_1 > 2\epsilon$, it follows that $(d_1q^*) < (q^*b')$ and hence $(d_1b') < 2(q^*b')$. Therefore,

$$(d_kb') \leq (d_1b') \cdot \prod_{i=2}^{k} R_i < 2(q^*b') \cdot \prod_{i=2}^{k} R_i.$$

Since $(d_kb') > (q^*b')$ (as d_k is left of q^*), we conclude that

$$\prod_{i=2}^{k} R_i > 1/2.$$

It follows again as in Case I that $k = O\big(\frac{m+\log(1/\epsilon)}{\log m + \log\log(1/\epsilon)}\big)$.

We now proceed to analyze the general case by reducing it to the aforementioned special case. Suppose that the optimal solution has an arbitrary number of points, i.e. has the form $Q^* = \langle a, q_1, q_2, \ldots, q_t, b\rangle$. Charge the points computed by the Chord algorithm to the edges of the optimal solution as follows: Suppose that an iteration of the Chord algorithm refines a triangle $\triangle(a_ic_ib_i)$ by bringing the parallel to the chord a_ib_i which touches the curve, say, at point e_i, forming two new triangles $\triangle(a_ia_i'e_i)$ and $\triangle(b_ib_i'e_i)$. Charge this iteration to the edge q_iq_{i+1} of Q^* if and only if one of the two segments q_ie_i or e_ib_i of the curve has no point of Q^* and is contained in the segment q_iq_{i+1} (note that it will be contained in a unique segment of Q^*).

This leaves out the iterations of the algorithm where both segments contain points of Q^*. The total number of such iterations is at most linear in t, because every such iteration splits a subset of the optimal set into two nonempty sets (i.e. these are the vertices of the recursion tree with two children) and there can be at most OPT such splits. At this point we have essentially reduced the general case to the $\text{OPT}_\epsilon = 3$ case. By arguments parallel to the ones for this special case it follows that every edge of the optimal solution is charged at most $O\big(\frac{m+\log(1/\epsilon)}{\log m + \log\log(1/\epsilon)}\big)$ points of Chord. This completes the proof.

Remark 6.3.12. We briefly sketch the differences for the case of an approximate Comb routine. First we note that, in this case, the description of the Chord algorithm (Table 6.1) has to be slightly modified to guarantee that the set of computed points is an ϵ-CP. In particular, in the Chord routine, we need to check whether $\mathcal{RD}(q, lr) \leq \epsilon'$ for some $\epsilon' < \epsilon$. As a consequence, to prove the desired upper bound, in addition to the above lemmas we need a way to relate $\mathrm{OPT}_{\epsilon'}$ and OPT_{ϵ}. This is provided to us by the planar geometric lemma from Chapter 5 (Lemmas 5.3.10, 5.3.12) that roughly say that as ϵ decreases, the size of the optimal ϵ-CP (for the same instance) does not increase too fast.

6.4 Average Case Analysis

In this section we prove our average case results.

6.4.1 Upper Bounds.

We start by presenting the analysis of the Poisson point process. The proofs for unconcentrated product distributions are somewhat more involved and are presented next. The analysis for both cases has the same overall structure, however each case has its difficulties, as elaborated below.

Overview of the Proofs. We now give a brief overview of the proof. For the sake of simplicity, in the following intuitive explanation, let n denote: (i) the expected number of points in the instance for a PPP and (ii) the actual number of points for product distributions. As in the worst-case analysis, we resort to an indirect measure of the algorithm's progress, namely the area of the triangles maintained in the algorithm's recursive tree. We think that this feature of our analysis is quite interesting and indicates that this measure is quite robust.

We first show (see Lemma 6.4.3 for the case of PPP) that every subdivision performed by the algorithm decreases the area between the upper and lower approximations by a significant amount (roughly an exponential decrease) with high probability. It follows then that at depth $\log \log n$ of the recursion tree, each "surviving triangle" contains an expected number of at most $\log \log n$ points with high probability. We use this fact, together with a charging argument in the same spirit as in the worst-case, to argue that the expected competitive ratio is $\log \log n$ in this case.

To analyze the expected competitive ratio in the complementary event, we break it into a "good" event, under which the competitive ratio is $\log n$ with high probability, and a "bad" event, where the competitive ratio is potentially unbounded (in the Poisson case) or at most n (for the case of product distributions). The potential unboundedness of the competitive ratio in the Poisson case creates complications in bounding the expected competitive ratio of the algorithm over the full space. We overcome this difficulty by bounding the upper tail of the Poisson distribution (see Lemma 6.4.8).

In the case of product distributions, the worst case bound of n on the competitive ratio is sufficient to conclude the proof, but the technical challenges present themselves in a different form. Here, the "contents" of a triangle being processed by the algorithm depend on the information coming from the previous recursive calls making the analysis harder. We overcome this by understanding the nature of the information provided from the conditioning.

On the choice of parameters. A simple but crucial observation concerns the "interesting range" for the parameters of the distributions. Consider for example the uniform distribution: we select n independent random points, each uniformly distributed in $[2^{-m}, 2^m]^2$. We are given an $\epsilon > 0$ and we run the chord algorithm on this random instance. It is not hard to see that, if n is larger than some $\mathrm{poly}(2^m/\epsilon)$, then the algorithm makes a constant number of calls in expectation. Hence, we can assume wlog that $n = O(\mathrm{poly}(2^m/\epsilon))$. A similar bound also holds for the intensity λ of the PPP.

Let us elaborate for the case of the uniform distribution. Suppose we are given a uniform random instance $\mathcal{I}$ – i.e. we have n independent random points, each uniformly distributed in some region $S \subseteq [2^{-m}, 2^m]^2$– and we apply the chord algorithm to find an ϵ-CP set for $\mathcal{I}$. We claim that, if n is very large, larger than some $N = \mathrm{poly}(2^m/\epsilon)$, then the expected number of calls performed by the chord algorithm is trivially $O(1)$. Hence, the interesting case for the distribution is when $n \leq N$. Formally, we have:

Fact 6.4.1. *Let $T = \triangle(abc)$ be the triangle at the root of the Chord algorithm's recursion tree and suppose that there are n points independently and uniformly distributed in T. Also denote by $\alpha \overset{def}{=} \min\{x(a), y(b)\}$, let λ be the absolute slope of ab and $\beta \overset{def}{=} \frac{\epsilon^2\alpha^2}{2S(T)} \cdot \min\{\lambda, 1/\lambda\}$ ($S(T)$ is the area of T). If $n \geq \frac{3}{\beta} \cdot \log(1/\beta)$, then $\mathbb{E}[\mathrm{CHORD}_\epsilon(T)] = O(1)$.*

Proof. We can of course assume that $y(a) > (1+\epsilon) \cdot y(b)$ and $x(b) > (1+\epsilon) \cdot x(a)$, otherwise there is nothing to prove. Let $p_1 = (x(a), (1+\epsilon) \cdot y(b)) \in ac$ and $p_2 = ((1+\epsilon) \cdot x(a), y(b)) \in bc$. Let $T^* \subseteq \triangle(cp_1p_2)$ be the right triangle of *maximum area* whose hypotenuse is parallel to ab. (This is the shaded triangle in Figure 6.7.)

Let us lower bound the area of T^*. First, it is clear that c is a vertex of T^*. It is also clear that either p_1 or p_2 (or both) are vertices. Hence, one of the edges of T^* has length at least $\epsilon \cdot \alpha$. Since λ is the slope of the hypotenuse, the other edge has length at least $\min\{\lambda, 1/\lambda\} \cdot (\epsilon \cdot \alpha)$. Hence, $S(T^*) \geq \frac{\epsilon^2\alpha^2}{2} \cdot \min\{\lambda, 1/\lambda\}$. Therefore, the probability that a given uniform point falls into T^* is at least β, which means that the probability than *none* of the n points falls into T^* is at most $(1-\beta)^n$. Now if there is a point in T^*, the chord algorithm will find it in one step (by calling $\mathrm{Comb}(\lambda)$) and terminate. Otherwise, the algorithm will terminate after at most $2n$ calls to Comb. Hence,

$\mathbb{E}[\text{CHORD}_\epsilon(T)] \leq 2+1+2n\cdot(1-\beta)^n$. Note that the last summand is at most $2n\cdot e^{-\beta\cdot n} = O(1)$ by our choice of n. ■

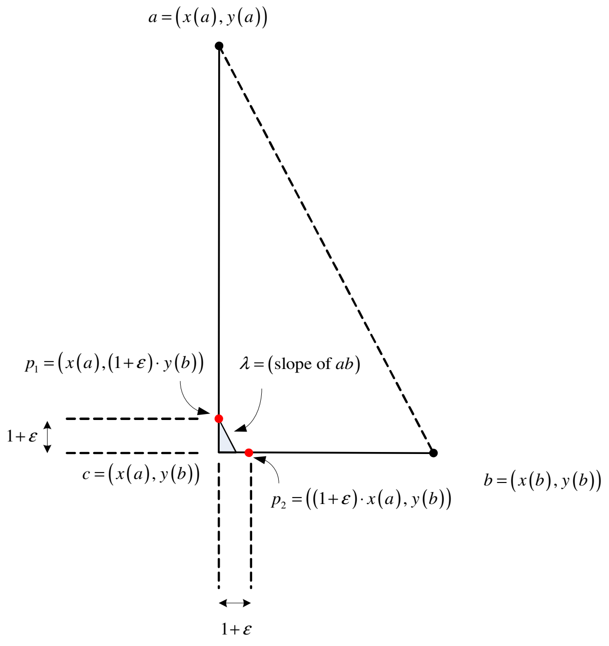
$a=(x(a),y(a))$
$p_1=(x(a),(1+\varepsilon)\cdot y(b))$
$\lambda=(\text{slope of } ab)$
$1+\varepsilon$
$c=(x(a),y(b))$
$p_2=((1+\varepsilon)\cdot x(a),y(b))$
$b=(x(b),y(b))$
$1+\varepsilon$

Figure 6.7: On the Choice of Parameters.

Note that by assumption $T \subseteq [2^{-m}, 2^m]^2$. Hence, $\alpha \geq 2^{-m}$. We also have that $\epsilon/2^{2m} \leq \lambda \leq 2^{2m}/\epsilon$ (since otherwise the set $\{a, b\}$ is an ϵ-CP) and $S(T) \leq 2^{2m}$, hence $\beta \geq \epsilon^3/2^{6m+1}$. Thus, we get an upper bound on n of $\text{poly}(2^m/\epsilon)$. We note that essentially the same argument applies for γ-balanced distributions (and PPP) and also when we have an approximate Comb_δ routine (in which case we replace ϵ by an appropriate $\epsilon' = O(\epsilon)$ in the argument).

A similar claim also applies for the case of the Hausdorff distance. Hence, our average case upper bounds also apply for this metric. However, the argument fails for the horizontal (and vertical) distance.

6.4.1.1 Poisson Point Process.

We prove the following theorem – which combined with the aforementioned discussion yields the desired upper bound of $O(\log m + \log\log(1/\epsilon))$.

Theorem 6.4.2. *Let T_1 be the triangle at the root of the Chord algorithm's recursion tree, and suppose that points are inserted into T_1 according to a Poisson Point Process with intensity λ, $\lambda S(T_1) > c > e$, where c is an absolute constant. The expected performance ratio of the Chord algorithm on this instance is $O\Big(\log\log\big(\lambda S(T_1)\big)\Big)$.*

Proof. Let us denote $S_1 = S(T_1)$. We can assume that $\lambda S_1 > 10$, since otherwise the expected total number of points inside T_1 is $O(1)$ and the bound trivially holds. We show next that the area of the triangles maintained by the algorithm decreases rapidly at every recursive step. Namely,

Lemma 6.4.3. *Let $T_i = \triangle(a_i b_i c_i)$ be the triangle processed by the chord algorithm at some recursive step. Denote $q_i = \mathrm{Comb}(\lambda_{a_i b_i})$. Let $T_{i,l} = \triangle(a_i a_i' q_i)$ and $T_{i,r} = \triangle(b_i b_i' q_i)$. For all $c > 0$, with probability at least $1 - \frac{1}{(\ln \lambda S_1)^c}$ conditioning on the information available to the algorithm,*

$$S(T_{i,l}), S(T_{i,r}) \leq \sqrt{S(T_i)} \cdot \sqrt{\frac{c \cdot \ln\ln \lambda S_1}{\lambda}}.$$

Proof of Lemma 6.4.3. It follows from the properties of the chord algorithm that, before the Comb routine on input T_i is invoked, the following information is known to the algorithm, conditioning on the history:

- there exist solution points at the locations q_j, for all $j = 1, 2, \ldots, i-1$;
- there is no point to the left of (below) the line defined by a_j and c_j, for all $j = 1, 2, \ldots, i$;
- there is no point below the line defined by c_j and b_j, for all $j = 1, 2, \ldots, i$;

See Figure 6.8 for an illustration. It follows from the properties of the Poisson Point Process that, conditioned on the above information, the number of points in a triangle of area S inside T_i follows a Poisson distribution with parameter $\lambda \cdot S$. Hence, letting $\zeta_i \eta_i$ being parallel to $a_i b_i$ so that the triangle $T^* \overset{\text{def}}{=} \triangle(c_i \zeta_i \eta_i)$ has area $S^* \overset{\text{def}}{=} \frac{c \cdot \ln\ln \lambda S_1}{\lambda}$, it follows that, with probability at least $1 - \frac{1}{(\ln \lambda S_1)^c}$, the point q_i is contained in the triangle T^*. We bound from above the area of $T_{i,l}$ by the area of $T'_{i,l} = \triangle(a_i c_i \eta_i)$ and similarly the area of $T_{i,r}$ by the area of $T'_{i,r} = \triangle(\zeta_i c_i b_i)$.

From the similarity of the triangles T_i and T^* we get

$$\frac{(\zeta_i \eta_i)}{(a_i b_i)} = \frac{(c_i \eta_i)}{(b_i c_i)} = \frac{(\zeta_i \theta_i)}{(a_i d_i)}$$

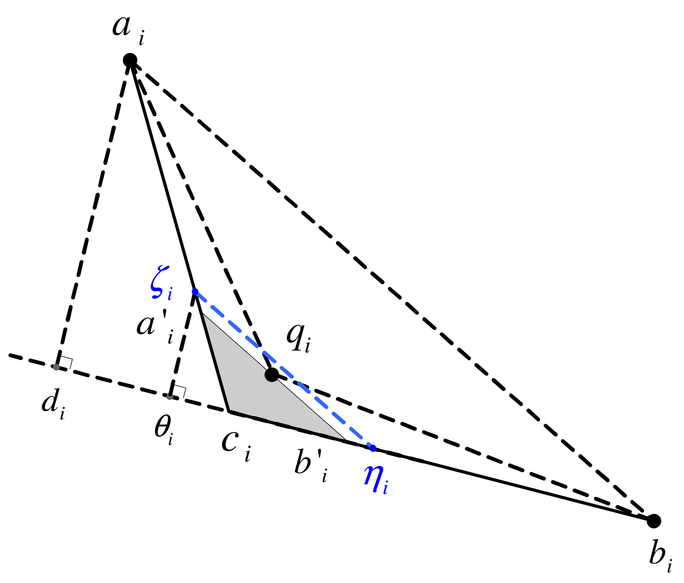

Figure 6.8: Average case area shrinkage property of the Chord algorithm.

Hence,

$$\frac{S^*}{S(T_i)} = \frac{\frac{1}{2}(c_i\eta_i)(\zeta_i\theta_i)}{\frac{1}{2}(b_ic_i)(a_id_i)} = \left(\frac{(c_i\eta_i)}{(b_ic_i)}\right)^2,$$

which gives $(c_i\eta_i) = (b_ic_i)\sqrt{\frac{S^*}{S(T_i)}}$. Therefore,

$$\begin{aligned} S(T'_{i,l}) &= \frac{1}{2}(a_id_i)(c_i\eta_i) = \frac{1}{2}(a_id_i)(b_ic_i)\sqrt{\frac{S^*}{S(T_i)}} \\ &= \sqrt{S(T_i)\cdot S^*} = \sqrt{S(T_i)}\cdot\sqrt{\frac{c\cdot\ln\ln\lambda S_1}{\lambda}}. \end{aligned}$$

Finally,

$$S(T'_{i,r}) = \frac{1}{2}(\zeta_i\theta_i)(b_ic_i) = \frac{1}{2}\frac{(c_i\eta_i)}{(b_ic_i)}(a_id_i)(b_ic_i) = S(T'_{i,l}).$$

This concludes the proof of the lemma. ■

Let us choose $c \in \left(\frac{1}{\ln\ln\lambda S_1}, \frac{\lambda S_1}{\ln\ln\lambda S_1}\right)$, and let $S(T)$ be the area of a triangle T maintained by the algorithm at depth d of the recursion. It follows from Lemma 6.4.3 that, with probability at least $1 - \frac{d}{(\ln\lambda S_1)^c}$,

$$S(T) \leq S_1^{\frac{1}{2^d}} \cdot \left(\frac{c\cdot\ln\ln\lambda S_1}{\lambda}\right)^{1-\frac{1}{2^d}},$$

where to bound the probability of the above event we have taken a union bound only over the events on the path of the recursion tree connecting T to the root of the recursion. Now consider the top $d^* := \lceil \log_2 \ln \lambda S_1 \rceil$ levels of the recursion tree of the algorithm. The tree has at most $2 \cdot \ln \lambda S_1$ internal nodes. Notice that the tree in general has many internal nodes with out-degree 1. Intuitively, for each such node the algorithm performs a redundant call to Comb. Using Lemma 6.4.3 and a union bound it follows that, with overall probability at least $1 - \frac{2 \cdot \ln \lambda S_1}{(\ln \lambda S_1)^c}$, the area of *every* triangle at depth d^* of the recursion tree is at most $S^{**} := (e \cdot c \cdot \ln \ln \lambda S_1)/\lambda$, where we used our assumption on the range of c.

Let $\mathcal{A}$ be the event that all the nodes (triangles) maintained by the algorithm at depth d^* of the recursion tree (if any) have area at most S^{**}. We just argued that the probability of the event $\mathcal{A}$ is at least $1 - \frac{2}{(\ln \lambda S_1)^{c-1}}$. We now show the following:

Lemma 6.4.4. *Conditioning on the event $\mathcal{A}$, the expected performance ratio of the algorithm is* $O(\log \log \lambda S_1)$.

Proof of Lemma 6.4.4. Let $\mathcal{T}$ be the recursion tree of the algorithm pruned at level d^*, let $\mathcal{V}$ be the set of nodes of $\mathcal{T}$[‡], and let $\mathcal{L}_{d^*}$ be the subset of nodes in $\mathcal{V}$ that lie at depth d^* from the root. (Note that the set $\mathcal{L}_{d^*}$ is a subset of the leaves of $\mathcal{T}$.) For a triangle (node) T maintained by the algorithm at depth d^* of the recursion, that is $T \in \mathcal{L}_{d^*}$, we let the random variable X_T denote the number of points inside T. We denote by $\mathcal{L}'$ the set of lowest non-leaf nodes of the tree $\mathcal{T}$. As in the worst-case analysis, we have $\mathrm{OPT}_\epsilon \geq |\mathcal{L}'|$. Also note that $|\mathcal{V}| \leq 3d^*|\mathcal{L}'|$ and that the Chord algorithm makes a call to Comb for every node in the tree.

We condition on the information $\mathcal{F}$ available to the algorithm in the first d^* levels of its recursion-tree (without the information obtained from processing – i.e. calling Comb for – any triangle at depth d^*). By assumption $\mathcal{F}$ satisfies the event $\mathcal{A}$. Moreover, conditioning on the information $\mathcal{F}$, for all $T \in \mathcal{L}_{d^*}$, X_T follows a Poisson distribution with parameter $\lambda \cdot S(T)$. So, given that the event $\mathcal{A}$ holds, we have $\mathbb{E}[X_T \,|\, \mathcal{F}] \leq \lambda \cdot S^{**}$.

Also recall that the number of Comb calls performed by the algorithm on a triangle T containing a total number of X_T points is at most $2X_T$. Hence, the expected total number of calls performed

[‡]We remind the reader that by convention we do not have a node in the tree for those triangles $T = \triangle(a_i b_i c_i)$ for which $\mathcal{RD}(c_i, a_i b_i) \leq \epsilon$.

by the algorithm is at most

$$\begin{aligned}\mathbb{E}[\mathrm{CHORD}_\epsilon \mid \mathcal{F}] &\leq |\mathcal{V}| + 2 \cdot \sum_{T \in \mathcal{L}_{d^*}} \mathbb{E}[X_T \mid \mathcal{F}] \\ &\leq |\mathcal{V}| + 2\,|\mathcal{L}_{d^*}| \cdot \lambda \cdot S^{**} \\ &\leq |\mathcal{V}| + 4|\mathcal{L}'| \cdot \lambda \cdot S^{**} \\ &\leq |\mathcal{L}'| \cdot (3d^* + 4\lambda \cdot S^{**}),\end{aligned}$$

where we made use of the fact that $|\mathcal{L}_{d^*}| \leq 2\,|\mathcal{L}'|$. So, conditioning on the information $\mathcal{F}$ available to the algorithm in the first d^* levels of its recursion tree (before the processing of any triangle in the d^*-th level), the expected performance ratio of the algorithm is

$$\begin{aligned}\mathbb{E}\left[\frac{\mathrm{CHORD}_\epsilon}{\mathrm{OPT}_\epsilon} \middle| \mathcal{F}\right] &\leq \mathbb{E}\left[\frac{\mathrm{CHORD}_\epsilon}{|\mathcal{L}'|} \middle| \mathcal{F}\right] \\ &\leq (3d^* + 4\lambda \cdot S^{**}) \\ &= O\left(\log\log \lambda S_1\right).\end{aligned}$$

Integrating over all possible $\mathcal{F}$ inside $\mathcal{A}$ concludes the proof of the lemma. ■

From Lemma 6.4.4 we have that

$$\mathbb{E}\left[\frac{\mathrm{CHORD}_\epsilon}{\mathrm{OPT}_\epsilon} \middle| \mathcal{A}\right] = O\left(\log\log \lambda S_1\right),$$

and from the discussion above we have that $\Pr\left[\bar{\mathcal{A}}\right] \leq \frac{2}{(\ln \lambda S_1)^{c-1}}$. Hence, we have established the following.

Lemma 6.4.5. *For $c \in \left(\frac{1}{\ln\ln \lambda S_1}, \frac{\lambda S_1}{\ln\ln \lambda S_1}\right)$, there exists an event $\mathcal{A}$, with $\Pr[\mathcal{A}] \geq 1 - \frac{2}{(\ln \lambda S_1)^{c-1}}$, such that the expected performance ratio of the algorithm conditioning on $\mathcal{A}$ is $O\left(\log\log \lambda S_1\right)$.*

Let $\mathcal{B}$ be the event that all the triangles at the level $\lceil \log_2 \lambda S_1 \rceil$ of the recursion tree of the algorithm (if any) have area at most $(e \cdot c' \cdot \ln \lambda S_1)/\lambda$. With the same technique, but using different parameters in the argument, we can also establish the following:

Lemma 6.4.6. *For $c' \in \left(\frac{1}{\ln \lambda S_1}, \frac{\lambda S_1}{\ln \lambda S_1}\right)$, there exists an event $\mathcal{B}$, with $\Pr[\mathcal{B}] \geq 1 - \frac{2}{(\lambda S_1)^{c'-1}}$, such that the expected performance ratio of the algorithm conditioning on $\mathcal{B}$ is $O\left(\log \lambda S_1\right)$.*

Finally, the performance ratio can be bounded by twice the total number of points in the triangle at the root of the recursion tree. Hence,

Lemma 6.4.7. *The expected performance ratio of the algorithm is $O(\lambda S_1)$.*

We want to use Lemmas 6.4.5, 6.4.6 and 6.4.7 to deduce that the expected performance ratio of the algorithm is $O(\log\log\lambda S_1)$. This may seem intuitive, but it is in fact not immediate. For technical purposes let us define the event $\mathcal{C} = \mathcal{B} \setminus \mathcal{A}$, where $\mathcal{A}$ is the event defined in the proof of Lemma 6.4.5. It is easy to see that conditioned on $\mathcal{C}$ the expected performance ratio of the algorithm can still be bounded by $O(\log\lambda S_1)$, since this expectation is affected only by whatever happens at level $\lceil\log_2\lambda S_1\rceil$ of the recursion tree and below. On the other hand, using the fact that $\Pr[\mathcal{A}] \geq 1 - \frac{2}{(\ln\lambda S_1)^{c-1}}$, it follows that

$$\Pr[\mathcal{C}] \leq \frac{2}{(\ln\lambda S_1)^{c-1}}.$$

Now we can bound the expectation of the performance ratio as follows:

$$\begin{aligned}
\mathbb{E}\left[\frac{\text{CHORD}_\epsilon}{\text{OPT}_\epsilon}\right] &\leq \mathbb{E}\left[\frac{\text{CHORD}_\epsilon}{\text{OPT}_\epsilon}\middle|\mathcal{A}\right]\cdot\Pr[\mathcal{A}] \\
+ \quad & \mathbb{E}\left[\frac{\text{CHORD}_\epsilon}{\text{OPT}_\epsilon}\middle|\mathcal{C}\right]\cdot\Pr[\mathcal{C}] \\
+ \quad & \mathbb{E}\left[\frac{\text{CHORD}_\epsilon}{\text{OPT}_\epsilon}\middle|\overline{\mathcal{A}\cup\mathcal{C}}\right]\cdot\Pr\left[\overline{\mathcal{A}\cup\mathcal{C}}\right] \\
\leq \quad & O(\log\log\lambda S_1)\cdot\left(1-\frac{2}{(\ln\lambda S_1)^{c-1}}\right) \\
& +O(\log\lambda S_1)\cdot\frac{2}{(\ln\lambda S_1)^{c-1}} \\
& +\mathbb{E}\left[\frac{\text{CHORD}_\epsilon}{\text{OPT}_\epsilon}\middle|\overline{\mathcal{A}\cup\mathcal{C}}\right]\cdot\Pr\left[\overline{\mathcal{A}\cup\mathcal{C}}\right].
\end{aligned}$$

To conclude, we need to bound the last term of the above expression. Note first that $\mathcal{B} \subseteq \mathcal{A}\cup\mathcal{C}$. Hence,

$$\Pr\left[\overline{\mathcal{A}\cup\mathcal{C}}\right] \leq \frac{2}{(\lambda S_1)^{c'-1}}.$$

We again use the fact that performance ratio is bounded by twice the total number of points in the triangle at the root of the recursion tree. This number X follows a Poisson random variable with parameter $\lambda\cdot S_1$. Hence, we have

$$\mathbb{E}\left[\frac{\text{CHORD}_\epsilon}{\text{OPT}_\epsilon}\middle|\overline{\mathcal{A}\cup\mathcal{C}}\right]\cdot\Pr\left[\overline{\mathcal{A}\cup\mathcal{C}}\right] \leq 2\cdot\mathbb{E}\left[X\,\middle|\,\overline{\mathcal{A}\cup\mathcal{C}}\right]\cdot\Pr\left[\overline{\mathcal{A}\cup\mathcal{C}}\right].$$

To bound the right hand side of the above we use the following technical lemma.

Lemma 6.4.8. *Let X be a* $\mathrm{Poisson}(\lambda)$ *random variable, with $\lambda \geq 1$, and let $\mathcal{E}$ be some event. Then*

$$\mathbb{E}[X \mid \mathcal{E}]\Pr[\mathcal{E}] \leq \max\left\{\frac{1}{\lambda}, O(\lambda^3)\Pr[\mathcal{E}]\right\}.$$

Proof of Lemma 6.4.8. Let k^* be such that $\Pr[X \geq k^*+1] < \Pr[\mathcal{E}] \leq \Pr[X \geq k^*]$. Clearly,

$$\mathbb{E}[X \mid \mathcal{E}]\Pr[\mathcal{E}] \leq \sum_{i=k^*}^{+\infty} i \cdot \Pr[X=i] = \sum_{i=k^*}^{+\infty} i \cdot \frac{e^{-\lambda}\lambda^i}{i!}$$
$$= \lambda \sum_{i=k^*-1}^{+\infty} \frac{e^{-\lambda}\lambda^i}{i!} = \lambda \Pr[X \geq k^*-1].$$

Now we distinguish two cases. If $k^*-1 \geq 2\lambda^2$, then from Chebyshev's inequality we get:

$$\Pr[X \geq k^*-1] \leq \frac{1}{\lambda^2}.$$

Hence,

$$\mathbb{E}[X \mid \mathcal{E}]\Pr[\mathcal{E}] \leq \frac{1}{\lambda}.$$

If $k^*-1 \leq 2\lambda^2$, then

$$\Pr[X=k^*] \leq \frac{k^*+1}{\lambda}\Pr[X=k^*+1] \leq O(\lambda)\Pr[\mathcal{E}]$$

and

$$\Pr[X=k^*-1] \leq \frac{(k^*+1)^2}{\lambda^2}\Pr[X=k^*+1] \leq O(\lambda^2)\Pr[\mathcal{E}].$$

Hence,

$$\begin{aligned}
\mathbb{E}[X \mid \mathcal{E}]\Pr[\mathcal{E}] &\leq \lambda \cdot \Pr[X \geq k^*-1] \\
&\leq \lambda \cdot \big(\Pr[\mathcal{E}] + \Pr[X=k^*-1] + \\
&\quad \Pr[X=k^*]\big) \\
&\leq O(\lambda^3)\cdot\Pr[\mathcal{E}].
\end{aligned}$$

This concludes the proof of the lemma. ■

From Lemma 6.4.8 we obtain

$$\begin{aligned}
&\mathbb{E}\left[\frac{\mathrm{CHORD}_\epsilon}{\mathrm{OPT}_\epsilon}\middle|\overline{\mathcal{A}\cup\mathcal{C}}\right]\cdot\Pr\left[\overline{\mathcal{A}\cup\mathcal{C}}\right] \leq \\
\leq\ & \max\left\{\frac{1}{\lambda S_1}, O((\lambda S_1)^3)\cdot\Pr\left[\overline{\mathcal{A}\cup\mathcal{C}}\right]\right\} \\
\leq\ & \max\left\{\frac{1}{\lambda S_1}, O((\lambda S_1)^3)\frac{2}{(\lambda S_1)^{c'-1}}\right\}.
\end{aligned}$$

Choosing $c' = 4$, the above RHS becomes $O(1)$. Plugging this into (6.5) with $c = 2$ gives

$$\mathbb{E}\left[\frac{\text{CHORD}_\epsilon}{\text{OPT}_\epsilon}\right] = O(\log\log(\lambda S_1)).$$

This concludes the proof of Theorem 6.4.2. ■

6.4.1.2 Product Distributions.

We now state our theorem for product distributions.

Theorem 6.4.9. *Let n points be chosen independently from a γ-balanced distribution on S_1, where S_1 is the triangle at the root of the chord algorithm's recursion tree, and $\gamma \in [0, 1)$ some constant. The expected performance ratio of the algorithm on this instance is $O_\gamma(\log\log n)$.*

Proof. The proof has the same overall structure as the proof of Theorem 6.4.2, but the details are more elaborate. We emphasize below the required modifications to the argument. In the arguments below we assume that $n \geq 12$. Otherwise, the bound on the performance ratio trivially holds.

We show first an area shrinkage lemma, similar to Lemma 6.4.3.

Lemma 6.4.10 (Area Shrinkage-Product Distributions). *Let T_i, $T_{i,l}$ and $T_{i,r}$ be as in the statement of Lemma 6.4.3. Let also $T_i' = \triangle(a_i'c_ib_i')$. See Figure 6.8. Finally, suppose that T_i is processed at recursion depth at most $\lceil\log_2 \ln n\rceil - 1$. For all $c > 0$, with probability at least $1 - \left(\frac{1}{\ln n}\right)^{\frac{c(1-\gamma)^2}{2}}$ conditioning on the information available to the algorithm,*

$$S(T_{i,l}), S(T_{i,r}) \leq \sqrt{S(T_i)S_1}\cdot\sqrt{\frac{c\cdot\ln\ln n}{n}};$$

$$\textit{and} \qquad S(T_i') \leq S_1\frac{c\cdot\ln\ln n}{n}.$$

Proof of Lemma 6.4.10. We follow the proof of Lemma 6.4.3 with the appropriate modifications. Let T_i be the triangle maintained by the algorithm at some node i of the recursion tree, and suppose that T_i is at depth at most $\lceil\log_2 \ln n\rceil - 1$ from the root. The information available to the algorithm when it processes T_i is

- the location of all points q_j, $j = 1, \ldots, i$; so, in particular, the location of at most $2\ln n$ points is known (given our assumption about the depth);
- moreover, there is no point to the left of the line defined by a_j and c_j, for all $j = 1, 2, \ldots, i$, or below the line defined by c_j and b_j, for all $j = 1, 2, \ldots, i$.

Given this information the probability that, among the remaining $n_i \geq n - 2\ln n$ points whose location is unknown, none falls inside a triangle T of area S inside T_i, is at most

$$\left(1 - (1-\gamma)^2 \frac{S}{S_1}\right)^{n_i}.$$

Indeed, let $T_i^* \subseteq T_1$ be the subset of the root triangle which is available at step i for the location of q_i; this is the convex set below the line $a_1 b_1$, to the right of all lines $a_j c_j$, for all $j = 1, 2, \ldots, i$, and above all lines $c_j b_j$, for all $j = 1, 2, \ldots, i$. The probability that a point whose location is unknown falls inside $T \subseteq T_i^*$ is

$$\frac{\mathcal{D}[T]}{\mathcal{D}[T_i^*]} \geq \frac{(1-\gamma)\mathcal{U}[T]}{\mathcal{U}[T_i^*]/(1-\gamma)} \geq \frac{(1-\gamma)\mathcal{U}[T]}{\mathcal{U}[T_1]/(1-\gamma)} = (1-\gamma)^2 \frac{S}{S_1}.$$

Choosing $S \overset{\text{def}}{=} S_1 \frac{c \cdot \ln\ln n}{n}$, the probability becomes

$$(1-\gamma)^2 \cdot \frac{c \cdot \ln\ln n}{n}.$$

Hence, the probability that T is empty is at most

$$\left(1 - c(1-\gamma)^2 \frac{\ln\ln n}{n}\right)^{n_i} \leq e^{-c(1-\gamma)^2 \frac{\ln\ln n}{n} n_i} = \left(\frac{1}{\ln n}\right)^{\frac{c(1-\gamma)^2}{2}}.$$

The proof of the lemma is concluded by similar arguments as in the proof of Lemma 6.4.3. ■

Using Lemma 6.4.10 and the union bound, we can show that, with probability at least

$$1 - \frac{2}{(\ln n)^{\frac{c(1-\gamma)^2}{2} - 1}},$$

the following are satisfied:

- all triangles maintained by the algorithm at depth $\lceil \log_2 \ln n \rceil$ of its recursion tree have area at most
$$S_1 \cdot \frac{e \cdot c \cdot \ln\ln n}{n}.$$
- for every node (triangle) i in the first $\lceil \log_2 \ln n \rceil - 1$ levels of the recursion tree
$$S(T_i') \leq S_1 \frac{c \cdot \ln\ln n}{n},$$
where T_i' is defined as in the statement of Lemma 6.4.10.

The proof of the second assertion above follows immediately from Lemma 6.4.10 and the union bound. The first assertion is shown similarly to the analogous assertion of Theorem 6.4.2. For the above we assumed that $c \in \left(\frac{1}{\ln\ln n}, \frac{n}{\ln\ln n}\right)$.

Now let us call $\mathcal{A}_c$ the event that the above assertions are satisfied. We can show the following.

Lemma 6.4.11. *Suppose $c \leq \frac{n}{4 \cdot \ln n \cdot \ln\ln n}$. Conditioning on the event $\mathcal{A}_c$, the expected performance ratio of the algorithm is $O_{c,\gamma}(\log\log n)$.*

Proof of Lemma 6.4.11. The proof is in the same spirit to the proof of Lemma 6.4.4, but more care is needed. We need to argue that, under $\mathcal{A}_c$, the expected number of points falling inside a triangle at depth $\lceil \log_2 \ln n \rceil$ of the recursion tree is $O_{c,\gamma}(\log\log n)$. Using rationale similar to that used in the proof of lemma 6.4.10 above, we have the following. Let T_i be the triangle maintained by the algorithm at a node i at depth $\lceil \log_2 \ln n \rceil$ of the recursion tree. Let also p be a point whose location is unknown to the algorithm (conditioning on the information known to the algorithm after processing the first $\lceil \log_2 \ln n \rceil - 1$ levels of the recursion tree). The probability that the point p falls inside T_i is

$$\frac{\mathcal{D}\left[T_i\right]}{\mathcal{D}[T_i^*]} \leq \frac{\mathcal{U}\left[T_i\right]/(1-\gamma)}{(1-\gamma)\mathcal{U}[T_i^*]},$$

where T_i^* is the region below the line $a_1 b_1$, above the lines $a_j c_j$, for all j inside the first $\lceil \log_2 \ln n \rceil$ levels of the recursion tree, and above the lines $c_j b_j$, for all j in the first $\lceil \log_2 \ln n \rceil$ levels of the recursion tree. To upper bound the probability that p falls inside T_i, we need a lower bound on the size of the area $S(T_i^*)$. Such bound can be obtained by noticing that

$$S(T_i^*) = S_1 - \sum_j S(T_j'),$$

where the summation ranges over all j in the first $\lceil \log_2 \ln n \rceil - 1$ levels of the recursion tree. Hence,

$$\begin{aligned} S(T_i^*) &\geq S_1 - 2\ln n \cdot S_1 \frac{c \cdot \ln\ln n}{n} \\ &= S_1 \cdot \frac{n - 2 \cdot c \cdot \ln n \cdot \ln\ln n}{n} \geq \frac{S_1}{2}, \end{aligned}$$

where we used that $c \leq \frac{n}{4 \cdot \ln n \cdot \ln\ln n}$. Hence, the probability that a point falls inside T_i is at most

$$\frac{\mathcal{U}\left[T_i\right]/(1-\gamma)}{(1-\gamma)\mathcal{U}[T_i^*]} \leq \frac{1}{(1-\gamma)^2} \frac{S_1 \cdot \frac{e \cdot c \cdot \ln\ln n}{n}}{S_1/2} \leq \frac{2 \cdot e \cdot c \cdot \ln\ln n}{(1-\gamma)^2 n}.$$

Therefore, the expected number of points falling in T_i is at most

$$\frac{2 \cdot e \cdot c \cdot \ln\ln n}{(1-\gamma)^2}.$$

■

We have established the following

Lemma 6.4.12. *For $c \in \left(\frac{1}{\ln\ln n}, \frac{n}{4 \cdot \ln n \cdot \ln\ln n}\right)$, there exists an event $\mathcal{A}_c$, with $\Pr[\mathcal{A}_c] \geq 1 - \frac{2}{(\ln n)^{0.5c(1-\gamma)^2-1}}$, such that the expected performance ratio of the algorithm conditioning on $\mathcal{A}_c$ is $O_{c,\gamma}(\log\log n)$.*

We can also show the equivalent of Lemma 6.4.6. Namely,

Lemma 6.4.13. *For $c' \in \left(\frac{1}{\ln n}, \frac{\ln n}{6}\right)$, there exists an event $\mathcal{B}_{c'}$, with $\Pr[\mathcal{B}_{c'}] \geq 1 - \frac{2}{n^{0.5c'(1-\gamma)^2-1}}$, such that the expected performance ratio of the algorithm conditioning on $\mathcal{B}$ is $O_{c',\gamma}\left((\log n)^3\right)$.*

Proof. Lemma(Lemma 6.4.13) The proof is similar to the proof of Lemma 6.4.12, except the bound is now a bit trickier. For $c' \in \left(\frac{1}{\ln n}, \frac{\ln n}{6}\right)$, let $\mathcal{B}_{c'}$ be the event that

- all the triangles maintained by the algorithm at depth $\lceil \log_2 n \rceil$ of its recursion tree have area at most $S_1 \cdot \frac{e \cdot c' \cdot \ln n}{n}$.
- for every node i inside the first $\lceil \log_2 n \rceil - 1$ levels of the recursion tree

$$S(T_i') \leq S_1 \frac{c' \cdot \ln n}{n},$$

  where $S(T_i')$ is defined as in the statement of Lemma 6.4.10.

Using arguments similar to those in the proof of Lemma 6.4.10 and the union bound, we obtain that

$$\Pr[\mathcal{B}_{c'}] \geq 1 - \frac{2}{n^{0.5c'(1-\gamma)^2-1}}.$$

Now let $\mathcal{T}$ be the recursion tree of the algorithm pruned at level $\lceil \log_2 n \rceil$. We define the set $\mathcal{L}'$ as in the proof of Lemma 6.4.4, but with d^* replaced by $\lceil \log_2 n \rceil$. In that proof we argued that any optimal ϵ-approximate convex pareto curve needs to use at least $|\mathcal{L}'|$ points. Moreover, we argued that the total number of nodes inside $\mathcal{T}$ is at most $3\lceil \ln n \rceil |\mathcal{L}'|$. Whenever the algorithm processes a triangle, a planar region of area at most $S_1 \cdot \frac{c' \cdot \ln n}{n}$ is removed from T_1 (the root triangle). Therefore, after finishing the processing of the first $\lceil \ln n \rceil - 1$ levels of the tree, a total area of at most

$$3\lceil \ln n \rceil S_1 \cdot \frac{c' \cdot \ln n}{n} |\mathcal{L}'|$$

is removed from T_1. We distinguish next two cases. If $|\mathcal{L}'| \geq \frac{n}{\lceil \ln n \rceil^3}$, then the size of the optimum is at least $\frac{n}{\lceil \ln n \rceil^3}$ points. Since there is a total of n points (and the algorithm never performs more than $2n$ Comb calls), it follows that in this case the performance ratio is $O(\lceil \ln n \rceil^3)$. On the other hand, if $|\mathcal{L}'| \leq \frac{n}{\lceil \ln n \rceil^3}$, then the total area that has been removed from T_1 is at most $3S_1 \cdot \frac{c'}{\ln n}$. Hence, the remaining area is at least $S_1/2$, assuming $c' \leq \ln n/6$. Given this bound it follows that the expected number of points inside a triangle at level $\lceil \ln n \rceil$ of the recursion tree is at most

$$\frac{2 \cdot e \cdot c' \cdot \ln n}{(1-\gamma)^2}.$$

The remaining of the argument is similar to that used in the proof of Lemma 6.4.11. Using the above and noting that the performance ratio "paid" within the first $\lceil \log_2 n \rceil - 1$ levels of the recursion tree is at most $O_{c',\gamma}(\ln n)$, we can conclude the proof via arguments parallel to the proof of Lemma 6.4.12.

■

Now let us choose $c = \frac{8}{(1-\gamma)^2}$ and $c' = \frac{4}{(1-\gamma)^2}$. From Lemmas 6.4.12 and 6.4.13 we have that

$$\Pr[A_c] \geq 1 - \frac{2}{(\ln n)^3} \quad \text{and} \quad \Pr[B_c] \geq 1 - \frac{2}{n}.$$

Given this, we can conclude the proof Theorem 6.4.9. The argument is the same as the end of the proof of Theorem 6.4.2, except that now we can trivially bound the performance ratio of the algorithm by $2n$ in the event $\overline{\mathcal{A} \cup \mathcal{C}}$. ■

6.4.2 Lower Bounds.

We prove lower bounds on the expected performance ratio of the algorithm that match our upper bounds. For the case of the PPP we prove

Theorem 6.4.14. *There exists a family of instances for which the expected performance ratio of the Chord algorithm is $\Omega(\log\log \lambda S_1)$, where S_1 is the area of the triangle at the root of the recursion tree of the algorithm and λ is the intensity of the Poisson Point Process. In particular, we can select the parameters so that $\lambda S_1 = 2^m/\epsilon$, which yields a lower bound of $\Omega(\log m + \log\log 1/\epsilon)$.*

Proof. The lower bound applies even if the Comb routine is exact. In analogy to the worst-case (Section 6.3.1), the hard instances for the average case lower bound are "skewed". In particular, the initial triangle $T_1 = \triangle(a_1 b_1 c_1)$ (at the root of the recursion tree) will be right and $(a_1 c_1) >> (b_1 c_1)$.

To avoid clutter in the expressions, we will write the proof for the case $m = 1$. The generalization for all values of m is similar. We select $a_1 = (1, 2)$, $b_1 = (1 + 2\epsilon, 1)$ and $c_1 = (1, 1)$. We also choose the intensity of the Poisson process to be $\lambda = 1/\epsilon^2$. Note that $\lambda S_1 = 1/\epsilon$, hence we get an $\Omega(\log \log 1/\epsilon)$ lower bound.

As in the worst-case lower bound, given the endpoints, it is clear that $\mathrm{OPT}_\epsilon = O(1)$ (with probability 1). We will show that the chord algorithm needs $\Omega(\log \log \frac{1}{\epsilon})$ calls to the Comb routine to find an ϵ-convex Pareto.

In particular, we are going to argue that for ϵ small enough, with constant probability, there exists a path of length $\Omega(\log \log \frac{1}{\epsilon})$ in the recursion tree of the algorithm. The path $\mathcal{P}$ is defined by the triangles with $b_i = b_1$, i.e. the triangles corresponding to the "right" subproblem in every subdivision, see Figure 6.1.

Recall from the worst-case arguments that for such "skewed" instances, the ratio distance is very well approximated by the horizontal distance. Hence, we will consider the latter for the rest of the proof without loss of generality. For notational convenience in the arguments below we shift the coordinate system to the point c_1, so that $c_1 = (0, 0)$, $a_1 = (0, 1)$ and $b_1 = (2\epsilon, 0)$. (Note that the horizontal distance is invariant under translation.) Also, for convenience we label the triangles in the path $\mathcal{P}$ by $T_1, T_2, \ldots$, and we let the vertices of triangle T_i be $a_i = (x_i, y_i)$, $c_i = (x'_i, 0)$, and $b_i = b_1$. Suppose that when the chord algorithm processes the triangle T_i, the Comb routine returns the point q_i on a line $a'_i b'_i$ parallel to $a_i b_i$ (as in Figure 6.1). Clearly, we have $b'_i = (x'_{i+1}, 0) = c_{i+1}$ and $q_i = (x_{i+1}, y_{i+1}) = a_{i+1}$. Let us call $\mathcal{X}_i$, $\mathcal{Y}_i$ the coordinates of the point a'_i.

It is easy to see that, as long as $x'_i < \epsilon$ the node of the recursion tree corresponding to triangle T_i is not a leaf, i.e. the algorithm will recurse on T_i. We show the following:

Lemma 6.4.15. *For ϵ small enough, with probability at least $1 - 4\frac{\ln \ln \frac{1}{\epsilon}}{(\ln \frac{1}{\epsilon})^{1/4}}$, for all $i \in \{1, \ldots, \ln \ln 1/\epsilon\}$,*

$$y_i \geq c'_i \cdot \epsilon^{1 - \frac{1}{2^{i-1}}} / \log(1/\epsilon);$$

$$x'_i \leq c''_i \cdot \epsilon^{1 + \frac{1}{2^{i-1}}} \cdot \log(1/\epsilon),$$

where $c'_i = 2^{1 - \frac{1}{2^i}}$ and $c''_i = \sqrt{2} \cdot \sum_{j=1}^{i} 2^{1/2^j}$.

Proof of Lemma 6.4.15. Inductively we are going to show that, if the assertion of the theorem holds for some i, then it also holds for $i + 1$ with probability at least $1 - 4\frac{1}{(\ln \frac{1}{\epsilon})^{1/4}}$. First, by the law of similar triangles we have

$$\frac{y_i}{\mathcal{Y}_i} = \frac{2\epsilon - x'_i}{x'_{i+1} - x'_i}. \tag{6.5}$$

From the properties of the Poisson Point Process we have the following: conditioning on the information available to the algorithm when it processes the triangle T_i, if a (measurable) region inside T_i has area $\epsilon^2 \ln 1/\epsilon$, then the number of points inside this region follows a Poisson distribution with parameter $\ln 1/\epsilon$. Hence, with probability at least $1-\epsilon$, the region contains at least one point. Using (6.5) and the induction hypothesis, this implies that, with probability at least $1-\epsilon$,

$$x'_{i+1} - x'_i \le \frac{2}{\sqrt{c'_i}} \cdot \log\frac{1}{\epsilon} \cdot \epsilon^{1+\frac{1}{2^i}};$$

hence

$$x'_{i+1} \le \left(\frac{2}{\sqrt{c'_i}} + c''_i\right) \cdot \log\frac{1}{\epsilon} \cdot \epsilon^{1+\frac{1}{2^i}}.$$

On the other hand, if the area of a region is less than $\epsilon^2/\sqrt{\ln(1/\epsilon)}$ the probability that a point is contained in that region is at most $1/\sqrt{\ln(1/\epsilon)}$. This implies that, with probability at least $1 - 1/\sqrt{\ln(1/\epsilon)}$,

$$\mathcal{Y}_i \ge \frac{\sqrt{c'_i} \cdot \epsilon^{1-\frac{1}{2^i}}}{(\ln 1/\epsilon)^{3/4}}.$$

By the properties of the Poisson Point Process it follows that the point q_i is uniformly distributed on the segment $a'_i b'_i$. Hence, with probability at least $1 - \frac{1}{(\ln\frac{1}{\epsilon})^{\frac{1}{4}}}$, it holds $y_{i+1} \ge \frac{\sqrt{2c'_i} \cdot \epsilon^{1-\frac{1}{2^i}}}{\ln 1/\epsilon}$. This concludes the proof. ■

Note first that, for all i:

$$\sqrt{2} \cdot \sum_{j=1}^{i} 2^{1/2^j} \le 2\sqrt{2} \cdot i.$$

Hence, by Lemma 6.4.15, for ϵ small enough, with probability at least $1 - 4\frac{\ln\ln\frac{1}{\epsilon}}{(\ln\frac{1}{\epsilon})^{1/4}}$,

$$x_i \le 2\sqrt{2} \cdot \ln\ln 1/\epsilon \cdot \epsilon^{1+\frac{2}{\ln 1/\epsilon}} \cdot \ln(1/\epsilon),$$

$$\text{and } y_i \ge \epsilon^{1-\frac{2}{\ln 1/\epsilon}}/\log(1/\epsilon),$$

for all $i \le \ln\ln 1/\epsilon$. Hence, all the triangles $T_1, \ldots, T_{\ln\ln 1/\epsilon}$ survive in the recursion tree of the algorithm, since the algorithm maintains no certificate that the points already computed are enough to ϵ-cover. This concludes the proof that the expected performance ratio of the algorithm is $O(\log\log \lambda S_1)$. ■

6.5 Conclusions and Open Problems

We studied the Chord algorithm, a simple popular greedy algorithm that is used (under different names) for the approximation of convex curves in various areas. We analyzed the performance ratio of the algorithm, i.e. the ratio of the cost of the algorithm over the minimum possible cost required to achieve a desired accuracy for an instance, with respect to the Hausdorff and the ratio distance. We showed sharp upper and lower bounds, both in a worst case and in an average setting. In the worst case the Chord algorithm is roughly at most a logarithmic factor away from optimal, while in the average case it is at most a doubly logarithmic factor away.

We showed also that no algorithm can achieve a constant ratio in the number of Calls performance metric, in particular, at least a doubly logarithmic factor is unavoidable. We leave as an interesting open problem to determine if there is an algorithm with a better performance than the Chord algorithm, and to determine what is the best ratio that can be achieved. Another interesting direction of further research is to analyze the performance of the Chord algorithm in three and higher dimensions, and to characterize what is the best performance ratios that can be achieved by any algorithm.

Chapter 7

Conclusions and Open Problems

This work brings a new way of treating systematically optimization problems with multiple objectives genuinely as such, rather than forcing them into a single objective. The main theme of the thesis is the design and analysis of efficient approximation algorithms that are applicable to wide-classes of multi-objective optimization problems in a well-defined natural framework. Our goal has been to develop general methods for addressing the issues of the *size* of the approximate Pareto set, the *time* to construct it, and the *approximation error* that is achievable. All our algorithms are guaranteed to run in polynomial time and output a succinct approximate representation of the set of undominated solutions (Pareto set) for the instance of the problem at hand.

Depending on the setting, different notions of approximation to the Pareto set may be appropriate. In Chapter 3, we consider the notion of the ϵ-Pareto set, while in Chapters 4-6 we define and study the notion of the ϵ-convex Pareto set, as the appropriate one for convex problems. Moreover, different applications dictate for different notions for the performance of an algorithm. In Chapters 3-5, our goal was to construct an approximation to the Pareto set using as few solutions as possible (with the requirement that our algorithms run in polynomial time, of course). Hence, our metric in this setting is the ratio between the number of points output by our algorithm divided by the minimum number of points. In Chapter 6, our goal was to minimize the number of queries our algorithm performs, as compared to the minimum number of queries required on the given instance. Hence, our metric in this setting is the ratio between these two quantities[1]

[1]We note that, in Chapter 6, the performance ratio is defined as the ratio between the number of queries of our algorithm and the minimum number of *points* required for an ϵ-approximation. However, as argued there, these two

In Chapter 3, we study the problem of computing an ϵ-Pareto set with as few points as possible. We obtained a tight factor-2 approximation for bicriteria problems, a constant factor pseudo[2]-approximation for 3 criteria and a logarithmic pseudo-approximation for $d > 3$ criteria. The most important remaining open problem is the resolution of Conjecture 3.3.13: Is there a constant factor pseudo-approximation for $d > 3$ criteria? We believe this is an important open problem and hope to answer it in the affirmative. Note that, by the results of Chapter 3, this would also resolve the approximability of the dual problem.

In Chapters 4-5 we define and study the notion of the ϵ-convex Pareto set as the appropriate notion of approximation to the Pareto set for convex problems. We obtained a necessary and sufficient condition for its efficient computability, in terms of an efficient routine Comb for optimizing (exactly or approximately) monotone linear combinations of the objectives. We then considered the problem of constructing an ϵ-convex Pareto set using as few solutions as possible, and obtained efficient algorithms with tight or nearly tight performance guarantees for the case of 2 objectives. Our results for 3 (and more) objectives are not tight (as far as we know). Obtaining algorithms with improved guarantees is a most important open problem. We remark that almost nothing is known about the approximability of the dual problem in the convex setting, for $d > 3$. We believe that progress in this direction is attainable.

In Chapter 6, we consider the problem of computing an ϵ-convex Pareto set using as few queries to Comb as possible. We focused on the case of two criteria, and analyzed the performance of the Chord algorithm – a natural Greedy algorithm for this purpose. We gave a detailed analysis for the primal problem (given an error $\epsilon > 0$, minimize number of queries k). A related question concerns the performance of the Chord algorithm for the dual problem (given k queries, minimize ϵ). The main remaining open problem for the bi-objective case is to obtain an algorithm with an asymptotically optimal performance guarantee. We would like to stress that the case of 3 (or more) objectives is entirely open. Is there a generalization of the Chord algorithm with a poly-logarithmic performance guarantee? And what is the optimal ratio attainable by any algorithm with access to a Comb routine?

---

quantities are always within a factor of 2 for the case of two criteria.

[2]Recall that for 3 or more objectives, the algorithms obtain an ϵ'-Pareto set for some $\epsilon' > \epsilon$.

Bibliography

[AAR] V. S. Anil Kumar, Sunil Arya, H. Ramesh. Hardness of Set Cover with Intersection 1. *ICALP*, pp. 624-635, 2000.

[ABK1] E. Angel, E. Bampis, A. Kononov. An FPTAS for Approximating the Unrelated Parallel Machines Scheduling Problem with Costs. *In Proc. ESA*, pp. 194–205, 2001.

[ABG] E. Angel, E. Bampis and L. Gourves. Approximating the Pareto curve with local search for the bicriteria $TSP(1,2)$ problem. *Theor. Comput. Sci.* 310(13), 135146 (2004).

[ABK2] E. Angel, E. Bampis, A. Kononov. On the approximate trade-off for bicriteria batching and parallel machine scheduling problems. *Theoretical Computer Science*, 306(1-3), pp. 319–338 (2003).

[ABRSY] A. Aggarwal, H. Booth, J. O'Rourke, S. Suri, C.K. Yap. Finding minimal convex nested polygons. *Inform. and Control*, 83, pp. 98-110, 1989.

[ABV1] H. Aissi, C. Bazgan, D. Vanderpooten. Approximation Complexity of Min-Max (Regret) Versions of Shortest Path, Spanning Tree and Knapsack. In *Proc. ESA*, pp. 862–873, 2005.

[ABV2] H. Aissi, C. Bazgan, D. Vanderpooten. Complexity of Min-Max (Regret) Versions of Cut problems. In *Proc. ISAAC*, pp. 789–798, 2005.

[AHK] S. Arora, E. Hazan, S. Kayle. The multiplicative weights update method: a meta-algorithm and applications. *Manuscript*, 2006.

[AN] Y. P. Aneja, K. P. K. Nair. Bicriteria transportation problems. *Management Science*, pp. 73-78, 1979.

[AN+] H. Ackermann, A. Newman, H. Röglin, and B. Vöcking. Decision Making Based on Approximate and Smoothed Pareto Curves. *Theoretical Computer Science*, 378(3), p. 253–270 (2007).

[Ar] Archimedes. Quadratura parabolae. *Archimedis opera omnia*, vol II, J. L. Heiberg (ed.), B. G. Teubner, Leibzig, pp. 261-315, 1913.

[Ar1] A. F. Archer. Two $O(\log^* k)$-approximation algorithms for the asymmetric k-center problem. In *Proc. IPCO*, pp. 1–14, 2001.

[Ar2] A. F. Archer. Personal communication, 2007.

[As] P. Assouad. Densité et Dimension. *Ann. Institut Fourier, Grenoble*, 3:232–282, 1983.

[AS1] P.K. Agarwal and M. Sharir. Efficient Algorithms for Geometric Optimization. *ACM Computing Surveys*, 30, 412-458, 1998.

[AS2] P.K. Agarwal and S. Sen. Randomized Algorithms for Geometric Optimization Problems. In *Handbook of Randomized Computation*, Kluwer Academic Press, The Netherlands, pp. 151-201, 2001.

[AZ] A. Armon, U. Zwick. Multicriteria Global Minimum Cuts. *Algorithmica*, 46, pp. 15–26, 2006.

[BMP] Markus Blser, Bodo Manthey and Oliver Putz. Approximating Multi-criteria Max-TSP. In *ESA*, 2008.

[BBGS] A. Berger, V. Bonifaci, F. Grandoni, G. Schäfer. Budgeted Matching and Budgeted Matroid Intersection Via the Gasoline Puzzle. In *IPCO.*, pp. 273–287, 2008.

[BCM] H. Brönnimann, B. Chazelle, J. Matoušek. Product range spaces, sensitive sampling, and derandomization. *SIAM J. Comput.*, 28 (1999), 1552-1575.

[BG] H. Brönnimann, M.T. Goodrich. Almost Optimal Set Covers in Finite VC-Dimension. *Discrete and Computational Geometry*, 14(4): 463–479 (1995).

[BHR] R. E. Burkard and H. W. Hamacher and G. Rote. Sandwich approximation of univariate convex functions with an application to separable convex programming. *Naval Res. Logistics*, 38, pp. 911-924, 1991.

[BV04] R. Beier and B. Vocking. Typical Properties of Winners and Losers in Discrete Optimization. In *SIAM J. Comput.*, 35(4), p. 855-881, 2006.

[Car] P. Carstensen. The complexity of some problems in parametric linear and combinatorial programming. *PhD Thesis, University of Michigan*, 1983.

[CCS] J. Cohon, R. Church, D. Sheer. Generating multiobjective tradeoffs: An algorithm for the bicriterion problem. *Water Resources Research* 15, pp. 1001-1010, 1979.

[Chan] R. Chandrasekaran. Minimal ratio spanning tree. *Networks* 7, pp. 335–342, 1977.

[Chv] V. Chvátal. A greedy heuristic for the set covering problem. *Math. Oper. Res.*, 4: 233–235 (1979).

[CG+] J. Chuzhoy, S. Guha, E. Halperin, S. Khanna, G. Kortsartz, R. Krauthgamer, S. Naor. Asymmetric k-center is $\log^* n$-hard to Approximate. *J. ACM*, 52(4): 538–551 (2005).

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

[CHSB] D. L. Craft, T. F. Halabi, H. A. Shih, T. R. Bortfeld. Approximating convex Pareto surfaces in multiobjective radiotherapy planning. *Med. Phys.*, 33, pp. 3399-3407, 2006.

[Cl] K. Clarkson. Algorithms for polytope covering and approximation. *WADS*, 246-252, 1993.

[Coh] J. Cohon. Multiobjective Programming and Planning. *Dover*, 2004.

[CX] G. Chen, G. Xue. A PTAS for weight constrained Steiner trees in series-parallel graphs. *Theoretical Computer Science*, 1-3(304), pp. 237–247, 2003.

[CX2] G. Chen, G. Xue. k-pair delay constrained minimum cost routing in undirected networks. In *Proc. SODA*, pp. 230–231, 2001.

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

[CMH] K. Chatterjee, R. Majumdar, T.A. Henzinger. Markov Decision Processes with Multiple Objectives. *STACS*, 325-336, 2006.

[CV] K. L. Clarkson, K. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43-58, 2007.

[CY] R. Cole, C. Yap. Shape from probing. *J. Algorithms* 8, pp. 19-38, 1987.

[Das] G. Das. Approximation schemes in computational geometry. *PhD Thesis, U. Wisconsin*, 1990.

[DasDen] Indraneel Das and J. E. Dennis Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. *SIAM J. Optim.*, 8, pp. 631-657.

[De] T. Dey. Improved bounds on Planar k-sets and k-levels. *FOCS*, pp. 165–171, 1997.

[DJSS] J. Dongarra, E. Jeannot, E. Saule, Z. Shi. Bi-objective Scheduling Algorithms for Optimizing Makespan and Reliability on Heterogeneous Systems. In *Proc. SPAA*, pp. 280–288, 2007.

[SOD] H. Safer, J.B. Orlin, and M. Dror. Fully polynomial approximation in multi-criteria combinatorial optimization. *Technical report*, MIT Sloan School of Management, 2004.

[DP] D. Douglas, T. Peucker. Algorithms for the reductions of the number of points required to represent a digitized line or its caricature. The Canadian Cartographer 10(2), pp. 112-122, 1973.

[DDY] C. Daskalakis, I. Diakonikolas and M. Yannakakis. How good is the Chord Algorithm ? In *SODA*, 2010.

[DY1] I. Diakonikolas and M. Yannakakis. Small Approximate Pareto Sets for Bi-objective Shortest Paths and Other Problems. In *SIAM J. Comput.*, 2009.

[DY2] I. Diakonikolas, M. Yannakakis. Succinct Approximate Convex pareto Curves. *Proc. ACM-SIAM Symp. on Discrete Algorithms*, pp. 74-83, 2008.

[Ehr] M. Ehrgott. *Multicriteria optimization*, 2nd edition, Springer–Verlag, 2005.

[EG] M. Ehrgott, X. Gandibleux. An annotated bibliography of multiobjective combinatorial optimization problems. *OR Spectrum* 42, pp. 425–460, 2000.

[EKVY] K. Etessami, M. Kwiatkowska, M. Y. Vardi, and M. Yannakakis. Multi-Objective Model Checking of Markov Decision Processes. *Proc. 13th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'07)*, 2007.

[ERS] G. Even, D. Rawitz, S. Shahar. Hitting sets when the VC-dimension is small. *Information Processing Letters*, 95(2):358–362, 2005.

[ES] M. J. Eisner, D. G. Severance. Mathematical techniques for efficient record segmentaton in large shared databases. *J. ACM* 23(4), pp. 619-635, 1976.

[ESZ] F. Ergun, R. Sinha, L. Zhang. An improved FPTAS for Restricted Shortest Path. *Information Processing Letters*, 83(5):237–239, 2002.

[EV] E. Erkut, V. Verter. Modeling of transport risk for hazardous materials. *Operations Research*, 46, pp. 625–642, 1998.

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

[FBR] B. Fruhwirth, R. E. Burkard, G. Rote. Approximation of convex curves with application to the bicriteria minimum cost flow problem. *European J. of Operational Research* 42, pp. 326-338, 1989.

[FGE] J. Figueira, S. Greco, M. Ehrgott, eds. *Multiple Criteria Decision Analysis: State of the Art Surveys*, Springer, 2005.

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

[GR] M.X. Goemans, R. Ravi. The Constrained Minimum Spanning Tree Problem. In *Proc. SWAT*, 1996, pp. 66–75.

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

[Gus] D. Gusfield. Sensitivity Analysis for Combinatorial Optimization. *PhD Thesis, UC Berkeley*, 1980.

[Han] P. Hansen. Bicriterion Path Problems. In *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.

[HL] R. Hassin, A. Levin. An efficient polynomial time approximation scheme for the constrained minimum spanning tree problem. *SIAM J. Comput.*, 33(2): 261–268 (2004).

[HR] H.W. Hamacher and G. Ruhe. On spanning tree problems with multiple objectives. *Annals of Operations Research*, 52, pp. 209–230, 1994.

[HW] D. Haussler, E. Welzl. Epsilon-nets and simplex range queries. *Discrete Computational Geometry*, 2:127–151, 1987.

[Joh] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9: 256–278 (1978).

[KN07] J.A. Kelner and E. Nikolova. On the Hardness and Smoothed Complexity of Quasi-Concave Minimization. In *FOCS*, 2007.

[KP] V. Koltun, C.H. Papadimitirou. Approximately dominating representatives. *Theoretical Computer Science*, 371, pp. 148–154, 2007.

[KPW] J. Komlos, J. Pach, W. Woeginger. Almost tight bounds for Epsilon-Nets. *Discrete and Computational Geometry*, 7, pp. 10–15, 1992.

[KW] K. Klamroth and M. Wiecek. Dynamic programming approaches to the multiple criteria knapsack problem. Naval Research Logistics, 47(1):5776, 2000.

[LB] M. Lindenbaum, A. M. Bruckstein. Blind approximation of planar convex sets. *IEEE Trans. on Robotics and Automation* 10(4), pp. 517-529, 1994.

[Lov] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Math.*, 13: 383–390 (1975).

[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.

[MagSt] T.L. Magnanti, D. Stratila. Separable Concave Optimization Approximately Equals Piece-wise Linear Optimization. *IPCO*, pp. 234–243, 2004.

[MR] B. Manthey, L.S. Ram. Approximation Algorithms for Multi-Criteria Traveling Salesman Problems. *Algorithmica*, 2009.

[Mat1] J.Matoušek. Reporting points in halfspaces. *FOCS*, pp. 207–215, 1991.

[Mat2] J.Matoušek. In J. Pach (editor), *New Trends in Discrete and Computational Geometry*, Springer-Verlag, 1993.

[Meg1] N. Meggido. Combinatorial Optimization with Rational Objective Functions *Math of OR*, pp. 414–424, 1979.

[Meg2] N. Meggido. Applying Parallel Computation Algorithms in the Design of Serial Algo-ritmhs. *JACM*, pp. 852–865, 1983.

[Mit] K. M. Miettinen. *Nonlinear Multiobjective Optimization*, Kluwer, 1999.

[MitS] J. Mitchell and S. Suri. Separation and Approximation of Polyhedral Objects. *SODA*, pp. 296–306, 1992.

[MSW] J.Matoušek, R, Seidel, E. Welzl. How to net a lot with little: small ε-nets for disks and halfspaces. *SoCG*, pp. 16–22, 1990.

[MuSh] Ketan Mulmuley and Pradyut Shah. A Lower Bound for the Shortest Path Problem. *IEEE Conference on Computational Complexity* , pp. 14–21, 2000.

[NK+] E. Nikolova, J.A. Kelner, M. Brand and M. Mitzenmacher. Stochastic Shortest Paths via Quasi-Convex Maximization. *ESA*, pp. 552–563, 2006.

[NU] G.L. Nemhauser and Z. Ullmann. Discrete dynamic programming and capital allocation. *Management Science*, 15(9):494505, 1969.

[PR] E. Pyrga, S. Ray. New Existence Proofs for ϵ-Nets. In *Proc. SoCG*, 2008.

[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.

[PW] J. Pach, W. Woeginger. Some new bounds for Epsilon-Nets. In *Proc. 6th ACM Symposium on Computational Geometry*, pages 10–15, 1990.

[PY1] C.H. Papadimitriou, M. Yannakakis. On the Approximability of Trade-offs and Optimal Access of Web Sources. In *Proc. FOCS*, pp. 86–92, 2000.

[PY2] C.H. Papadimitriou, M. Yannakakis. Multiobjective Query Optimization. In *Proc PODS*, pp. 52–59, 2001.

[Ra] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing* 1, pp. 244-256, 1972.

[Rav+] Many birds with one stone: multi-objective approximation algorithms. R. Ravi, M. V. Marathe, S.S. Ravi, D.J. Rosenkrantz, H.B. Hunt. *STOC*, pp. 852–865, 1993.

[RS] R. Raz and S. Safra. A Sub-Constant Error-Probability Low-Degree Test, and a Sub-Constant Error-Probability PCP Characterization of NP. *STOC*, pp. 475-484, 1997.

[RF] G. Ruhe and B. Fruhwirth. Epsilon-optimality for biciteria programs and its application to minimum cost flows. *Computing* , 44, pp. 21–34, 1990.

[Ro] G. Rote. The convergence rate of the sandwich algorithm for approximating convex functions. *Computing*, 48, pp. 337-361, 1992.

[RT09] H. Röglin and S.H. Teng. Smoothed Analysis of Multiobjective Optimization. In *FOCS*, 2009.

[Ru] G. Ruhe. Complexity results for multicriterial and parametric network flows using a pathological graph of Zadeh. *Zeitschrift fur Operations Research* , 32, pp. 9–27, 1988.

[RW] S. Ruzika, M. M. Wiecek. Approximation Methods in Multiobjective Programming (Survey paper). *J. Opt. Th. and Appl.*, 126(3), pp. 473-501, 2005.

[TZ] G. Tsaggouris, C.D. Zaroliagis. Multiobjective Optimization: Improved FPTAS for Shortest Paths and Non-linear Objectives with Applications. In *Proc. ISAAC*, pp. 389–398, 2006.

[VV] P. Van Mieghen, L. Vandenberghe. Trade-off Curves for QoS Routing. In *Proc. INFOCOM*, 2006.

[VC] V. N. Vapnik, A. Ya. Chervonenkis. On the uniform convergence of relative frequencies to their probabilities. *Theory Probab. Appl.*, 16(2): 264-280, 1971.

[VY] S. Vassilvitskii, M. Yannakakis. Efficiently computing succinct trade-off curves. *Theoretical Computer Science* 348, pp. 334–356, 2005.

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

[Ze] M. Zeleny. *Linear Multiobjective Programming*, Springer, 1974.

[YG] X. Yang, C. Goh. A method for convex curve approximation. *European J. of Oper. Res.* 97, pp. 205-212, 1997.