Evaluating FPTASes for the Multi-Objective Shortest Path Problem

Bachelor Thesis Operations Research

Thomas Breugem

June 25, 2014

Abstract

We discuss three FPTASes for the Multi-Objective Shortest Path problem. Two of which are known in the literature, the third is a newly proposed FPTAS, aimed at exploiting small Pareto curves. The FPTASes are analyzed, empirically, based on worst case complexity, average case complexity and smoothed complexity. We also analyze the size of the Pareto curve under different conditions.

Keywords: Shortest Path, Multi-Objective Optimization, FPTAS, Complexity Analysis

Contents

1 Introduction

In this thesis we evaluate different FPTASes for the Multi-Objective Shortest Path problem, one of the well known multi-objective optimization problems. We propose a new FPTAS, aimed to be fast when the number of efficient paths is small. We evaluate, empirically, this FPTAS and two other FPTASes from the literature, based on different complexity analyses.

The layout of the thesis is as follows. Because the main topic of the thesis builds on a large body of theory, the first part will be used to give a (brief) review of the relevant topics. In Section 2 we discuss approximation algorithms in general and give the definitions relevant for the remainder of the thesis (e.g., PTAS and FPTAS). We do this using some well known combinatorial problems, at the end we also discuss (in)approximability results. In Section 3 we consider multi-objective optimization. We motivate the study of such problems and discuss relevant literature on the subject. In this section we also formalize the idea of approximate solutions to such problems, i.e., approximate Pareto-optimal frontiers. We conclude Section 3 by showing how these approximate solutions may be used to construct FPTASes for a large class of non-linear single-objective problems, thereby showing that multi-objective optimization is also highly applicable to other types of problems.

We then turn to the main part of the thesis, in which we discuss the multi-objective shortest path problem (MOSP) and FPTASes for this problem. In Section 4 we give the general definition of MOSP and define most of the notation that will be used throughout the remainder of the thesis. We give an algorithm that finds the Pareto-optimal frontier for a given MOSP instance, and also show that the frontier may be exponential in the number of nodes (i.e., the problem becomes intractable). This, of course, is the motivation for our study of approximation algorithms. In Section 5 we discuss three FPTASes for the MOSP; we discuss the FPTASes given in Mittal and Schulz [2008] and Tsaggouris and Zaroliagis [2009], the third FPTAS is a new contribution to the literature. For all three we prove correctness and derive the running time.

In Section 6 we analyze the algorithm that finds the whole Pareto curve and the FPTASes empirically, which is to our knowledge the first time this is done. We base our analysis on three different types of complexity: worst case, average case and smoothed complexity. The former two are well studied, the latter is relatively new (see, for example, Spielman and Teng [2009]). We discuss the different types of complexity and analyze how the algorithms perform.

We conclude the thesis by restating the most import results in Section 7, where we also discuss possible directions for further research.

2 Approximation Algorithms

In this section we briefly discuss approximation algorithms. We give the general definitions of approximation algorithms (and related topics) and highlight some of the well known results in this branch of combinatorial optimization. For detailed descriptions of the problems that are discussed, see, for example, Papadimitriou and Steiglitz [1982].

2.1 Definition of an Approximation Algorithm

An approximation algorithm, also called approximation scheme, is an algorithm which returns, for a given instance of the optimization problem, an 'almost optimal' solution, i.e, a solution that never differs more than a certain factor from the optimal solution (in terms of objective value). Such algorithms may provide alternative approaches to NP-complete problems, which can not be solved efficiently (modulo $\mathsf{P} \neq \mathsf{NP}$). Here efficient means that the algorithm runs in time *polynomial* in the size of the instance (i.e., the size of the input to the algorithm). Clearly, one wants to know how large the difference between the optimal solution and the approximation is, therefore we classify approximation algorithms based on this difference in a worst case scenario.

Formally, let $\mathcal{P}$ be an minimization (maximization) problem. We say algorithm $\mathcal{A}$ is an ε-approximate algorithm for $\mathcal{P}$ if, for every instance of $\mathcal{P}$, $\mathcal{A}$ returns a solution which objective value is no more than $1 + \varepsilon$ (no less than $1 - \varepsilon$) times that of the optimal solution, for some fixed $\varepsilon > 0$. (The case in which ε is a function of the input is defined analogously.) A well known example of such an algorithm is the 1/2-approximate algorithm for the metric TSP (that is, the Traveling Salesman Problem with triangle inequality) given in Christofides [1975]. Obviously, such algorithms are only relevant when they run substantially faster than any (known) exact algorithm.

2.2 PTAS and FPTAS

When for an optimization problem there exists an efficient ε-approximation algorithm for every $\varepsilon > 0$, we say the problem has a Polynomial Time Approximation Scheme (PTAS). Formally, a PTAS for a minimization (maximization) problem $\mathcal{P}$ is a family of algorithms such that for every $\varepsilon > 0$ there is an algorithm $\mathcal{A}_\varepsilon$ which is an ε-approximate algorithm for $\mathcal{P}$, running in time polynomial in the input size. If the running time is also polynomial in $1/\varepsilon$ we call this family of algorithms a Fully Polynomial Time Approximation Scheme (FPTAS). FPTASes are of much practical importance, since they allow substitution between the solution quality and recourses such as computational power and time. For a lot of well known problems, such as Knapsack, FPTASes are known. Woeginger [1999] gives a general framework for designing FPTASes based on dynamic programming techniques. One can see the existence of an FPTAS as the best possible result for an NP-complete problem (again, of course, assuming $\mathsf{P} \neq \mathsf{NP}$).

2.3 Limits on approximability

Unfortunately, not all combinatorial optimization problems can be approximated. The best known example is probably the inapproximability for the TSP for *any* ε (it can be shown that this allows for an efficient algorithm for Hamilitonian Path, which is NP-complete). Of course, this result only holds when $\mathsf{P} \neq \mathsf{NP}$. For the metric TSP, as discussed above, it is shown that no ε-approximation exists for $\varepsilon < 1/122$, see Karpinski et al. [2013]. Arora [1998] gives an extensive survey of (in)approximability results. It should be mentioned that a lot of lower bounds are believed to be much tighter than currently known, though no one has yet succeeded to prove such results (for example, it is not excluded that the 1/2-approximation of Christofides [1975] is the best possible).

3 Multi-Objective Optimization

In this section we discuss multi-objective optimization. First, we give a brief overview of the literature on this subject. Thereafter, we formally define multi-objective problems and solutions to such problems (i.e., Pareto-optimal frontiers). These definitions will be used to develop the FPTASes in the upcoming sections. We conclude this section by showing how the theory of multi-objective optimization can be applied to non-linear single-objective problems.

3.1 Overview

Multi-objective optimization (also called multi-criteria optimization) problems are a well studied branch of combinatorial optimization. The goal is to find optimal solutions based on multiple criteria; one might for example consider designing a communication network between certain cities in which we not only consider cost but also the failure probability of the network. This problem could be formulated as an bi-objective Minimum Spanning Tree problem.

Intuitively, it is clear that the presence of multiple objective functions lead to a more complex problem; it turns out that this is indeed true, for a lot of multi-objective problems it is shown that they are NP-hard or NP-complete (see, for example, Ehrgott [2000]). It is shown that for problems such as Shortest Path and Minimum Spanning Tree, problems for which multiple efficient algorithms are known for the single objective variant, the problem is intractable, even if we consider only two objective functions (we show this for Shortest Path in Section 4.4).

Due to such intractability results one might consider alternative approaches for solving multi-objective problems, such as approximation algorithms and evolutionary algorithms. Both these approaches have different advantages. Approximation algorithms (as already discussed in Section 2) have the advantage that the approximation ratio is *guaranteed* (i.e., it is proven that a certain ratio is always achieved). Evolutionary algorithms, on the other hand, are often fast and produce good solutions. They are, however, much harder (or even impossible) to analyze theoretically, and thus no approximation ratio can be proven. In this thesis, we consider approximation algorithms (for evolutionary algorithms, see, for example, Zitzler [1999]).

3.2 Definitions

Before we can discuss multi-objective optimization in detail, we formalize a multi-objective optimization problem, and a solution to such a problem; in the remainder of the thesis, we consider multi-objective optimization problems defined as follows.

Definition 3.1. *An instance π of a multi-objective optimization problem Π is given by d objective functions, denoted by $f_1, ..., f_d$. Every f_i maps the set of feasible solutions, denoted as X, to the positive real numbers, i.e., $f_i : X \to \mathbb{R}_+$, for $i = 1, ..., d$.*

In general, there is no solution that minimizes *all* objective functions. Therefore, a solution to a multi-objective function is given by a Pareto-optimal frontier.

Definition 3.2. *Let π be an instance of a multi-objective minimization problem. A* Pareto-optimal frontier*, denoted by $P(\pi)$, is a set of solutions such that $x \in P(\pi)$ if and only if there is no $x' \in X$ for which $f_i(x') \leq f_i(x)$, for $i = 1, ..., d$, with strict inequality for at least one i.*

Hence $P(\pi)$ is a set of all undominated solutions (also called 'trade-offs' or efficient solutions). (In the remainder of the thesis, we will use the names Pareto-optimal frontier, Pareto frontier and Pareto curve interchangeably.) The concept of the Pareto-optimal frontier is visualized in Figure 1. The Figure shows a bi-objective optimization problem, where the points in the plane indicate objective values corresponding to feasible solutions. The Pareto-optimal frontier consists of all the undominated solutions, which are the green points.

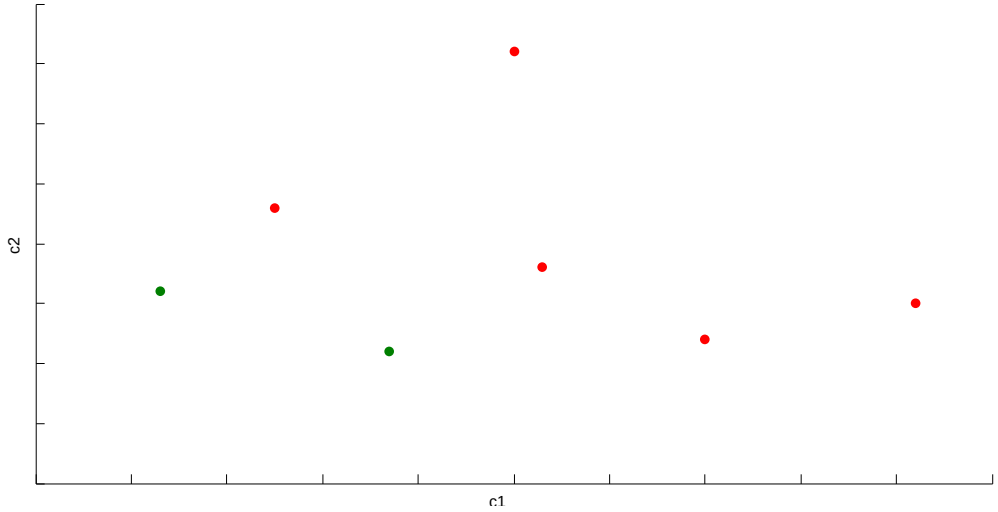

Figure 1: Example Pareto-optimal frontier

It is not uncommon that $P(\pi)$ is exponential in size, and hence the problem becomes intractable (since any algorithm should examine this set at least once). To circumvent this issue, one might consider an approximate Pareto-optimal frontier defined as follows.

Definition 3.3. *Let π be an instance of a multi-objective minimization problem. For $\varepsilon > 0$, an* ε-approximate Pareto-optimal frontier*, denoted by $P_\varepsilon(\pi)$, is a set of solutions, such that for all $x \in X$, there is an $x' \in P_\varepsilon(\pi)$ for which $f_i(x') \leq (1+\varepsilon) f_i(x)$, for all $i = 1, ...d$.*

Loosely speaking, $P_\varepsilon(\pi)$ is a set of solutions that *almost* dominate all solutions. It is shown in Papadimitriou and Yannakakis [2000] that under general conditions $P_\varepsilon(\pi)$ can be constructed in polynomial time (we discuss this procedure in Section 5.2).

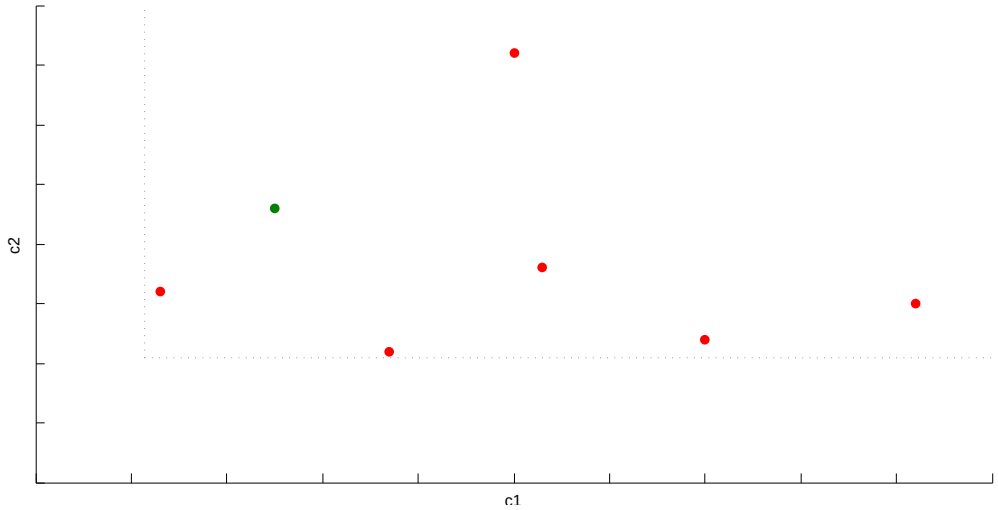

Figure 2: Example approximate Pareto-optimal frontier

Figure 2 gives an example of an approximate Pareto-optimal frontier (the approximation factor ε is chosen arbitrarily). The green point is the only solution in the approximate Pareto-optimal

frontier. The gray lines indicate which solutions are 'covered' by this solution, i.e., the rectangle indicated by these lines contains all points (a_1, a_2) with $f_i(x) \leq (1+\varepsilon)a_i$, for $i = 1, 2$, with x the solution in the frontier. We note that the set $P_\varepsilon(\pi)$ is not unique; the points in the Pareto-optimal curve of Figure 1 qualify as well. Furthermore, as Figure 2 illustrates, the approximate Pareto-optimal frontier can consist entirely of *dominated* solutions.

3.3 Applications to Single-Objective Optimization

As an extra motivation for our study of approximate Pareto-frontiers we conclude this section by showing that approximate Pareto frontiers allow for approximate solutions to a large range of non-linear single-objective optimization problems.

Let π be an instance of a multi-objective optimization problem as in Definition 3.1, i.e., we optimize d objective functions $f_1, ..., f_d$ over the feasible set X. Furthermore, consider a function $h : \mathbb{R}_+^d \to \mathbb{R}_+$, satisfying:

1. $h(y) \leq h(y')$ for all $y, y' \in \mathbb{R}_+^d$, such that $y_i \leq y'_i$ for $i = 1, ..., d$.
2. $h(\lambda y) \leq \lambda^c h(y)$ for all $y \in \mathbb{R}_+^d$ and $\lambda > 1$, for some fixed $c > 0$.

Note that the above conditions are satisfied for numerous well known functions, such as all l_p-norms and the product of multiple linear functions.

We will consider the general single-objective minimization problem:

$$\min \quad g(x) = h(f(x)), \quad x \in X \tag{1}$$

Here, and elsewhere, $f(x)$ is the vector $(f_1(x), ..., f_d(x))$. We will refer to this problem simply as $\mathcal{P}$. Mittal and Schulz [2008] show that if an approximate Pareto-frontier for π can be constructed in time polynomial in $|\pi|$ and $1/\varepsilon$ then we obtain an FPTAS for $\mathcal{P}$.

In order to proof the existence of an FPTAS for $\mathcal{P}$, we first proof the following lemmata. The first concerns the presence of an optimal solution in the Pareto-optimal frontier.

Lemma 3.1 (Mittal and Schulz [2008])**.** *There is at least one optimal solution x' to $\mathcal{P}$ such that $x' \in P(\pi)$.*

Proof. let $\hat{x}$ be an optimal solution of $\mathcal{P}$. Suppose $\hat{x} \notin P(\pi)$, then there exists a solution $x' \in P(\pi)$ such that $f_i(x') \leq f_i(\hat{x})$ for $i = 1, ..., d$. This implies, by property 1 of h, that $h(f(x')) \leq h(f(\hat{x}))$ and hence x' minimizes the function g. □

The second lemma shows how an approximate solution can be obtained from an approximate Pareto-frontier.

Lemma 3.2 (Mittal and Schulz [2008])**.** *Define $\hat{\varepsilon} = (1+\varepsilon)^{1/c} - 1$. Let $\hat{x}$ be a solution in $P_{\hat{\varepsilon}}(\pi)$ that minimizes $g(x)$ over all $x \in P_{\hat{\varepsilon}}(\pi)$. Then $\hat{x}$ is an ε-approximate solution of $\mathcal{P}$.*

Proof. Let x' be an optimal solution of $\mathcal{P}$ in $P(\pi)$. By definition of $P_{\hat{\varepsilon}}(\pi)$ there is a solution $x'' \in P_{\hat{\varepsilon}}(\pi)$ such that $f_i(x'') \leq (1+\hat{\varepsilon}) f_i(x')$ for $i = 1, ..., d$. Using properties 1 and 2 of h we have

$$\begin{aligned} g(x'') &= h(f_1(x''), ..., f_d(x'')) \\ &\leq h((1+\hat{\varepsilon})f_1(x'), ..., (1+\hat{\varepsilon})f_d(x')) \\ &\leq (1+\hat{\varepsilon})^c h(f_1(x'), ..., f_d(x')) = (1+\varepsilon)g(x') \end{aligned}$$

Since $\hat{x}$ was a minimizer of $g(x)$ over the set $P_{\hat{\varepsilon}}(\pi)$, we have $g(\hat{x}) \leq g(x'') \leq (1+\varepsilon)g(x')$. □

Combining lemmata 3.1 and 3.2 we obtain the following theorem.

Theorem 3.1. *Consider the multi-objective version of problem $\mathcal{P}$ (that is, minimizing $f_1, ..., f_d$ over X). If, for every $\varepsilon > 0$, we can construct $P_\varepsilon(\pi)$ for this problem in time polynomial in both $|\pi|$ and $1/\varepsilon$, then there is an FPTAS for $\mathcal{P}$, for d fixed.*

Thus, finding approximations schemes for problems as $\mathcal{P}$ can be reduced to finding approximate Pareto-curves for multi-objective problems.

Summarizing, we laid the framework for FPTASes for the MOSP in the previous two sections. We also showed that the study of approximate Pareto curves enables us to find approximation schemes for other types of problems as well. In the next sections we focus on the Shortest Path problem, and the existence of FPTASes for multi-objective version of this problem.

4 Shortest Path

In this section we discuss the Shortest Path problem. We start by giving the general formulation of the problem, followed by a brief overview of the literature on the subject. Thereafter, we give an algorithm that finds the Pareto curve for a multi-objective shortest path instance. We conclude by showing that the size of the curve can be exponential in the number of nodes, hence motivating our study of approximate algorithms.

4.1 General Formulation

The Shortest Path problem is one of the most well known problems in combinatorial optimization. Given a graph $G = (V, E)$ the problem is finding a path between two given nodes s and t that minimizes a given cost function c. We consider the case where the edge costs are non-negative integers, i.e., the function c maps the set of edges E to the set $\mathbb{Z}_+$. This naturally generalizes to the multi-objective Shortest Path problem (abbreviated as MOSP) by considering a cost function $c : E \to \mathbb{Z}_+^d$.

This version is also known as the *shortest s-t path problem* and is one among multiple variations. One of those variations is the *single-source shortest path problem*. In this problem a node s is given, and we are asked to determine all shortest $s - v$ paths, for $v \in V$. Due to the obvious connection with dynamic programming, this is often the problem that is solved by an algorithm, instead of the shortest $s-t$ path problem. The labeling algorithms we consider in the remainder of the thesis, for example, solve the latter problem. Although we can simply take only the $s-t$ paths from the returned solution to obtain a solution for the shortest $s-t$ path problem, it does influence the running time of the algorithm. This follows from the simple fact that we not only determine the efficient $s - t$ paths, but we determine *all* efficient paths. Therefore, we state explicitly which of the two variants is considered when determining the running time of the algorithms.

We will use the following notation in the remainder of the thesis. For a general graph $G = (V, E)$, define $n \equiv |V|$ (i.e., the *order* of the graph) and $m = |E|$ (i.e., the *size* of the graph). Furthermore, let $P^i(s, v)$ resp. $W^i(s, v)$ denote the set of all $s - v$ paths resp. walks of length at most i. The set of all $s - v$ paths resp. walks is denoted simply as $P(s, v)$ resp. $W(s, v)$. Finally, we extend the cost function to map the power set 2^E to the positive integers by defining $c(p) \equiv \sum_{e \in p} c(e)$, where $p \in 2^E$. We write $c_i(p)$ for the cost of path p in terms of the i^{th} objective function.

4.2 Overview

It is clear that the general Shortest Path problem is highly applicable, with applications ranging from pure navigational problems to, for example, determining the structure of social networks. For the single linear objective version of the problem efficient algorithms exist. The most famous one is probably Dijkstra's algorithm, see Dijkstra [1959], that solves the shortest path problem for non-negative weights in $O(n^2)$ time.

In practice, one might also consider an objective function which is non-linear. For example, if we want to find a path that is not only short but that also has a low probability of delay, then we can consider an objective function of the form $c_1(P) \cdot c_2(P)$, where $c_1(P)$ resp. $c_2(P)$ is the length resp. delay probability of the path. In Section 3.3 we showed that such problems are tightly linked to the multi-objective Shortest Path problem.

In the remainder of the thesis we focus on the MOSP, in particular the bi-objective version, which is well studied. As mentioned in Section 3 the multi-objective version of the shortest path problem is intractable (see Ehrgott [2000]). We will give the proof at the end of this section. Interesting results on the MOSP can be found in Müller-Hannemann and Weihe [2006], who derive conditions under which the Pareto curve is polynomial in size.

Skriver [2000] gives an overview of different exact algorithms for the multi-objective shortest path problem. Tarapata [2007] also gives an extensive survey of recent results, including results on approximating the Pareto curves. Among those are the algorithms discussed in Papadimitriou and Yannakakis [2000] (which we discuss in Section 5.2) and in Tsaggouris and Zaroliagis [2009] (which we discuss in Section 5.3). A different kind of approximation algorithm is given in Kern and Woeginger [2007], which is based on quadratic programming. This method approximates the problem where the objective is the product of two linear functions, such as the one considered above, it is, however, not generalizable to instances where $d > 2$.

4.3 Solving MOSP

MOSP can be solved using a labeling algorithm. The algorithm, given in Ehrgott [2000], is a labeling algorithm that uses the lexographic ordering of cost vectors to efficiently store all the labels. Due to the possible exponentially large Pareto frontier, the worst case performance of the algorithm is exponential. The algorithm finds all efficient $s - v$ paths, for all $v \in V$, we omit the proof of correctness (see Ehrgott [2000]).

A path p is represented by a label $(c(p), pred(p), last(p))$, here $pred(p)$ is a pointer to the label of $p - last(p)$, where $last(p)$ is the last edge added to the path. From here on, we use the names path and label interchangeably. Furthermore, when we consider dominance, we only consider labels that end *at the same node* (one might interpret this as if the lists are kept for every node separately, we omit this for simplicity).

The algorithm keeps two sets of labels; one set, denoted by $\mathcal{L}_T$, contains the *temporary* labels, the other set, denoted by $\mathcal{L}_P$, contains all *permanent* label. While the set $\mathcal{L}_T$ is not empty, the algorithm takes the lexographically smallest label of the set, say label p. We then remove this label from the set of temporary labels and add it to the set of permanent labels. Furthermore, for each edge e incident with the last node of p we add a new label p' to $\mathcal{L}_T$ for the path obtained by extending p with the edge e, unless this label is dominated by one of the labels already made. Furthermore, if a label is dominated by the newly created label, we delete that label.

Let π be the MOSP instance considered. Note that the algorithm finds all efficient paths, and hence the relevant problem for the running time is the single-source shortest path problem, i.e., the running time depends on the *total* number of efficient paths. We can express the running time of the algorithm as follows.

Lemma 4.1. *Let K be the total number of efficient paths. The algorithm finds the Pareto-optimal frontier in $O(n^2K^2)$ time.*

Proof. Since the total number of efficient paths is K, we add K labels to $\mathcal{L}_P$. For each added label we create at most n new labels, hence we create $O(nK)$ labels in total. When creating a new label we have to check whether it dominates/is dominated by any of the labels in $\mathcal{L}_T$ and/or $\mathcal{L}_P$. It is clear that the size of $\mathcal{L}_T$ is $O(nK)$, hence this step takes $O(nK + K) = O(nK)$ time. (We note that is not hard to construct an instance where the size of $\mathcal{L}_T$ is $\Theta(nK)$ at some point in the execution.) Note that while checking for dominance, we can simultaneously keep the set in lexographic order. Since the dominance check is done for all created labels, the total running time becomes $O(n^2K^2)$. □

Although such an expression for the running time is clearly useless *a priori* to solving the problem, we will see in Section 6 that there is empirical evidence for a small number of efficient paths, and hence a good performance of the algorithm.

4.4 MOSP is intractable

We conclude this section by proving the intractability of MOSP. The proof is from Ehrgott [2000]. We note that this proof can be easily extended to have all edge costs strictly positive.

Theorem 4.1. *MOSP is intractable, even for only two dimensions.*

Proof. We consider a directed graph $D = (V, A)$ with n nodes, where n is odd (note that for the undirected case, the proof is similar). For $i = 1, 3, 5, ..., n-2$, we add the arc $v_i v_{i+2}$ with cost $(0, 2^{\frac{i-1}{2}})$, the arc $v_i v_{i+1}$ with cost $(2^{\frac{i-1}{2}}, 0)$ and the arc $v_{i+1} v_{i+2}$ with cost $(0, 0)$, see Figure 3.

From the construction of D it is clear that for each path $p \in P(v_1, v_n)$ we have $c_1(p) + c_2(p) = 2^{\frac{n-1}{2}} - 1$. Furthermore, in each dimension, the full range of costs $\left\{0, ..., 2^{\frac{n-1}{2}} - 1\right\}$ can be achieved. This implies that there are $2^{\frac{n-1}{2}}$ efficient paths, which shows that the instance is intractable. □

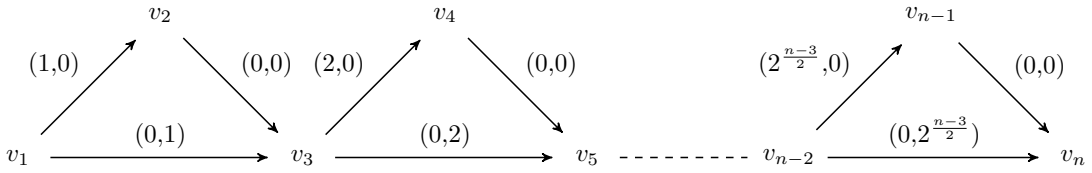

Figure 3: MOSP instance with exponential number of efficient paths.

5 Description of the FPTASes

In this section we give three different methods for constructing an approximate Pareto frontier for the multi-objective shortest path problem. The first one is given in Mittal and Schulz [2008] and is also applicable to different problems. The second one, given in Tsaggouris and Zaroliagis [2009], is tailor made for the shortest path problem. The third one we consider is a new FPTAS, based on the MOSP algorithm described in Section 4.3 and the work of Tsaggouris and Zaroliagis [2009].

5.1 Preliminaries

We develop the FPTASes for multi-objective optimization instances under the following assumption.

Assumption 5.1. *Let π be an instance of a multi-objective optimization problem as in Definition 3.1. It is assumed that every f_i, $i = 1, .., d$, takes value in the interval $\left[2^{-p(|\pi|)}, 2^{p(|\pi|)}\right]$ for some polynomial $p(\cdot)$. Here $|\pi|$ denotes the size of the instance π.*

This assumption is used by Mittal and Schulz [2008] to develop the FPTAS for general multi-objective optimization problems. Note that Assumption 5.1 holds for the MOSP: the cost of each path is bounded above by n times the maximal edge cost.

Furthermore, the FPTASes we discuss in fact find the approximate Pareto-optimal frontier for all $s - t$ *walks* of size at most $n - 1$, instead of all paths. (Note that not all returned walks are necessarily paths, due to the fact that we consider an *approximate* Pareto curve.) The reason for this simplification differs, but the underlying reason is that walks are 'easier' combinatorial objects, in the sense that they are less restrictive when it comes to extending a walk with an extra edge. Note that the length restriction is necessary to make the feasible region finite. That this simplification is allowed follows from the simple observation that $P(s,t)$ is a subset of $W^{n-1}(s,t)$. Furthermore, any returned walk can be truncated to obtain a path which cost is never higher (due to the non-negativity of the edge costs), hence we obtain an approximate Pareto-optimal frontier for the original problem by truncating all returned walks.

For convenience, we introduce some extra notation in this section. Given an instance of the MOSP problem as discussed in Section 4.1, let $c_i^{min} \equiv \min_{e \in E}\ c_i(e)$, $c_i^{max} \equiv \max_{e \in E}\ c_i(e)$ and $C_i \equiv c_i^{max}/c_i^{min}$, for $i = 1, ..., d$. Furthermore, define $C^{max} \equiv \max_{1 \leq i \leq d}\ C_i$.

We now turn to discussing the three different FPTASes.

5.2 FPTAS Mittal and Schulz [2008]

Mittal and Schulz [2008] construct the approximate Pareto frontier using the method proposed in Papadimitriou and Yannakakis [2000]. They show that $P_\varepsilon(\pi)$ can be constructed in fully polynomial time under very general conditions. Their approach is roughly as follows. First, they show that constructing an approximate Pareto curve can be reduced to solving the so called *'gap problem'* for a polynomial number (in terms of $|\pi|$ and $1/\varepsilon$) of different points. This gap problem can then be reduced to solving the *exact problem* corresponding to π a polynomial number of times. Here the exact problems consists of finding a feasible solution which cost is equal to a given constant.

We now discuss this method in detail. First, we define the gap problem and proof how this problem can be used to construct the approximate Pareto curve. We then define the exact problem and show how this can be used to solve the gap problem. We end by showing how the results are combined to show the existence of an FPTAS for constructing the approximate Pareto curve.

Papadimitriou and Yannakakis [2000] define the gap problem as follows, and proof how the approximate Pareto frontier can be constructed by solving this problem.

Theorem 5.1 (Papadimitriou and Yannakakis [2000])**.** *Let d be fixed and $\varepsilon > 0$. One can determine an $P_\varepsilon(\pi)$ in time polynomial in $|\pi|$ and $1/\varepsilon$ if the following gap problem can be solved in polynomial time: Given an d-vector of values $(v_1, ..., v_d)$ either*

1. *return a solution $x \in X$ such that $f_i(x) \leq v_i$ for $i = 1, ..., d$; or*
2. *assert there is no $x \in X$ such that $f_i(x) \leq (1 - \varepsilon')v_i$, for $i = 1, ...d$, where ε' satisfies $(1 - \varepsilon')(1 + \varepsilon)^{1/2} = 1$.*

Proof. If the above problem can be solved in polynomial time, we construct an approximate Pareto-optimal frontier as follows.

Consider the box in $\mathbb{R}^d$ given by $\left\{(v_1, ..., v_d) : 2^{-p(|\pi|)} \leq v_i \leq (1 + \varepsilon)^{1/2} 2^{p(|\pi|)}\right\}$. We successively divide this box into smaller boxes such that the ratio of division equals $(1 + \varepsilon)^{1/2}$. It is clear that this results in $O((p(\pi)/\varepsilon)^d)$ boxes. For all corner points of these boxes we solve the gap problem. Among all obtained solutions, we keep the ones that are not Pareto dominated (with respect to the found solutions). It is clear that this can be done in polynomial time in both $|\pi|$ and $1/\varepsilon$.

We claim that this set of solutions is the desired $P_\varepsilon(\pi)$. Suppose not, then there is a solution $x \in X$ for which there is no x' among the found solutions such that $f_i(x') \leq (1 + \varepsilon)f_i(x)$, for $i = 1, ..., d$. Let $v' = (v'_1, ..., v'_d)$ be the corner point such that $(1 + \varepsilon)^{1/2} f_i(x) \leq v'_i \leq (1 + \varepsilon)f_i(x)$, for $i = 1, ..., d$. When the gap problem was called at this corner point it is clear that, due to the presence of x, a solution must have been returned. This implies that a solution x' is returned with $f_i(x') \leq v'_i$, and thus $f_i(x') \leq (1 + \varepsilon)f_i(x)$, for $i = 1, ..., d$, which is a contradiction. □

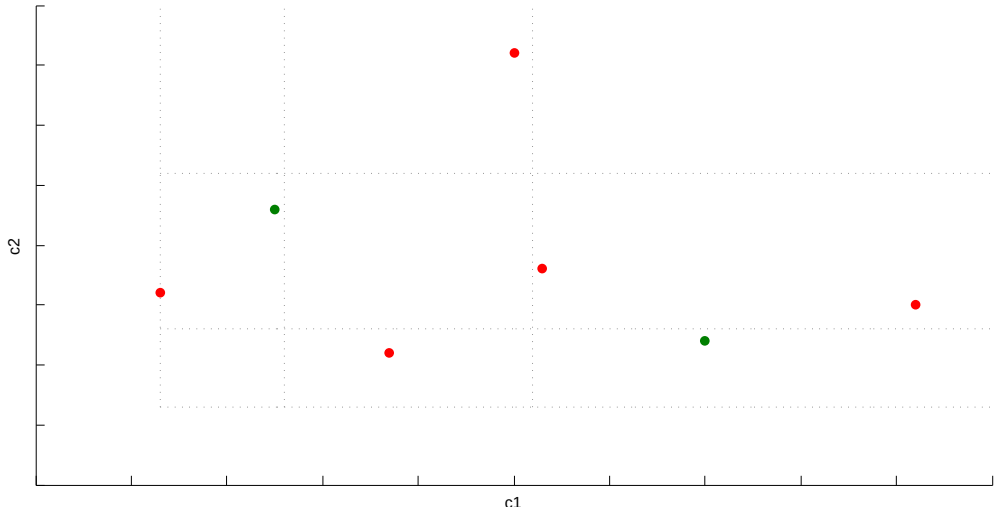

Figure 4: FPTAS Mittal and Schulz [2008]

To illustrate this approach, we consider the set of points in Figure 4, also used in Section 3. The dashed gray lines indicate the boxes that are constructed. Note that the size increases. Furthermore, note that the smallest objective function value (in this case the c_1 value of the left most point) determines the region that is divided into smaller boxes. For each box the gap problem is called for the point in the top right corner. The green points indicate solutions that are returned, the red points those that are not. Note, for example, that the point in the lower left corner is not included in the curve, this is valid since the green point to the right of it is close enough (recall that the ratio of division was $(1 + \varepsilon)^{1/2}$).

We now turn to solving the gap problem. First, we formalize the exact problem.

Definition 5.1. *Consider an instance π as in Definition 3.1, with $X \subseteq \mathbb{Z}_+^m$. The exact version of the problem is as follows. Given $K \in \mathbb{Z}_+$ and $c_i \in \mathbb{Z}_+$ for $i = 1, ..., m$, is there a solution $x \in X$ such that $\sum_{i=1}^m c_i x_i = K$?*

Papadimitriou and Yannakakis [2000] show that if there exists a pseudo polynomial time algorithm for the exact problem, then the gap problem can be solved in polynomial time.

Theorem 5.2 (Papadimitriou and Yannakakis [2000])**.** *Consider an instance π of a multi-objective optimization problem. If there is a pseudo polynomial algorithm for the exact problem corresponding to π, then, for every $\varepsilon > 0$, there is an algorithm for constructing $P_\varepsilon(\pi)$ in time polynomial in both $|\pi|$ and $1/\varepsilon$.*

Proof. We restrict our attention to the case where the value of all variables is bounded by n, which is polynomial in $|\pi|$. Furthermore, we consider linear objective functions f_i, $i = 1, ..., d$. Suppose we want to solve the gap problem for the vector $(v_1, ..., v_d)$. Let $r = \lceil nm/\varepsilon' \rceil$, where ε' satisfies $(1-\varepsilon')(1+\varepsilon)^{1/2} = 1$. We first define a 'truncated' objective function. For all $j = 1, ..., m$ if for some $i = 1, .., d$ we have $a_{ij} > v_i$ we drop the variable x_j from all objective functions (and set $x_j = 0$). Let $\mathcal{I}$ be the index set of the remaining variables. Next we define, for each of the objective functions, a new function $f_i'(x) = \sum_{j \in \mathcal{I}} a_{ij}' x_j$, where $a_{ij}' = \lceil a_{ij} r / v_i \rceil$. Note that the maximal value of a coefficient is now r.

From Lemma A.1 (see Appendix A) it follows that:

1. if $f_i'(x) \leq r$, then $f_i(x) \leq v_i$.
2. if $f_i(x) \leq (1-\varepsilon')v_i$, then $f_i'(x) \leq r$.

Hence, to solve the gap problem, it suffices to find an $x \in X$ such that $f_i(x) \leq r$ for $i = 1, ..., d$, or assert that no such x exists. Because all coefficients of $f_i(x)$ are nonnegative integers, there are $(r+1)^d$ ways in which the inequalities $f_i(x) \leq r$, $i = 1, ..., d$, can be satisfied. To check if there is a solution $x \in X$ such that $f_i'(x) = c_i$ for $i = 1, ...d$ we solve the exact problem given by $\sum_{i=1}^d M^{i-1} f_i'(x) = \sum_{i=1}^d M^{i-1} c_i$, where $M = nmr + 1$ (i.e., M is one greater than the maximum value of $f_i(x)$). It follows that the coefficients are of order $O(((nm)^2/\varepsilon)^d)$, and hence the pseudo polynomial time algorithm runs in polynomial time. This implies that the gap problem can be solved in polynomial time. The Theorem now follows from Theorem 5.1. □

We now turn to our main point of exercise, obtaining an FPTAS for the multi-objective shortest path. Let π be a multi-objective shortest path instance as discussed in Section 4.1. We obtain the following corollary to Theorem 5.2.

Corollary 5.1. *An ε-approximate Pareto-optimal frontier for an instance π of the multi-objective shortest path, satisfying Assumption 5.1, can be constructed in time polynomial in both $|\pi|$ and $1/\varepsilon$.*

Proof. From Theorem 5.2 it follows that we can construct the ε-approximate Pareto-optimal frontier in polynomial time if the exact problem can be solved in pseudo polynomial time. Although the exact version of Shortest Path is strongly NP-complete (since Hamiltonian Path is a special case), we can construct the frontier by relaxing the problem to finding a walk of length at most $n-1$, as discussed in Section 5.1.The exact walk problem can be solved in $O(nmK)$ time using dynamic programming, hence the result follows. □

Next, we determine the runtime of the algorithm.

Lemma 5.1. *The algorithm given in Mittal and Schulz [2008] constructs an ε-approximate Pareto frontier for π in $O\left(nm\left(n^3 m^3 \log(nC^{max})/\varepsilon^3\right)^d\right)$ time.*

Proof. From the proof of Theorem 5.1 it is clear that we solve the gap problem $O((p(\pi)/\varepsilon)^d)$ times. By the definition of C^{max} this is $O((\log(nC^{max})/\varepsilon)^d)$. In order to solve the gap problem, we call the exact problem $O(r^d)$ times, which is $O((nm/\varepsilon)^d)$, with input of order $O((nm)^2/\varepsilon)^d)$. The exact shortest walk problem can be solved in $O(nmK)$ time. This implies that the total runtime of the algorithm is

$$O\left((\log(nC^{max})/\varepsilon)^d \cdot (nm/\varepsilon)^d \cdot nm((nm)^2/\varepsilon)^d\right) = O\left(nm\left(n^3m^3\log(nC^{max})/\varepsilon^3\right)^d\right).$$

□

We note that although the theoretical runtime is polynomial, the runtime of the FPTAS can be tremendous. To illustrate this, we observe that, using the fact that $m = O(n^2)$, the dependency on the number of nodes alone already is $O(n^{21})$ for the bi-objective shortest path. As a conclusive remark we note that Theorem 5.2 allows for FPTASes for numerous other well known problems, such as Minimum Spanning Tree and Knapsack.

5.3 FPTAS Tsaggouris and Zaroliagis [2009]

The second FPTAS we consider is given in Tsaggouris and Zaroliagis [2009]. Their algorithm resembles the Bellman-Ford labeling algorithm (see Ehrgott [2000]), and is remarkably simple. The main idea of their algorithm is that they are able to store all labels in arrays of polynomial size by relaxing the assumption of undominated solutions. That is, by looking at $P_\varepsilon(\pi)$ instead of $P(\pi)$ they are able to keep the number of labels that need to be stored polynomial. As in Mittal and Schulz [2008], the algorithm given in Tsaggouris and Zaroliagis [2009] creates the approximate Pareto-curve for all $s-t$ walks (in fact, for all $s-v$ walks, for all $v \in V$).

As in Section 4.3, we represent a walk w by a label $(c(w), pred(w), last(w))$, here $pred(w)$ is a pointer to the label of $w - last(w)$, where $last(w)$ is the last edge added to the walk.

We consider a multi-objective shortest path instance π as before. To store the labels, we consider $d-1$ dimensional arrays $\mathcal{L}_v^i$ for $i = 1, ..., n-1$ and $v \in V$, with length $\lfloor \log_r(nC_j) \rfloor$ in the j^{th} dimension, for $j = 1, ..., d-1$. Here r determines the approximation factor (see Lemma 5.2). For a walk w we determine the position as

$$pos(w) = \left[\left\lfloor \log_r \frac{c_1(w)}{c_1^{min}} \right\rfloor, ..., \left\lfloor \log_r \frac{c_{d-1}(w)}{c_{d-1}^{min}} \right\rfloor \right].$$

It is clear that the size of the arrays is sufficient to store any $w \in W^{n-1}$, where $W^{n-1} \equiv \bigcup_{v \in V} W^{n-1}(s,v)$.

The algorithm proceeds as follows. Initially, we have $\mathcal{L}_v^0 = \emptyset$ for all $v \in V - \{s\}$, and $\mathcal{L}_s^0$ contains only the trivial empty path. In each round $1 \leq i \leq n-1$ we compute $\mathcal{L}_v^i$ for all v as follows. First we set $\mathcal{L}_v^i$ equal to $\mathcal{L}_v^{i-1}$. Then, for every $uv \in E$ we extend all labels $\mathcal{L}_u^{i-1}$ with the edge uv, i.e., for every label w we create a new label $w' = (c(w) + c(uv), w, uv)$. We then add this label to $\mathcal{L}_v^i$ at position $pos(w')$, unless for the current label at that position, say w'' it holds that $c_d(w'') \leq c_d(w')$. If there is no label at the position, we always add w'.

We proof the correctness of the algorithm for a one parameter approximation, i.e., we consider the same approximation factor in every dimension. The algorithm, however, also allows for different factors (see Tsaggouris and Zaroliagis [2009] for details). We should note that the algorithm is always exact in one dimension (in this case, the d^{th} dimension).

Lemma 5.2 (Tsaggouris and Zaroliagis [2009])**.** *For all $i \leq n-1$ and $v \in V$, after the i^{th} round $\mathcal{L}_v^i$ is an r^i-approximate Pareto frontier for $W^i(s,v)$.*

Proof. It suffices to proof that for all $w \in W^i(s,v)$ there is $w' \in \mathcal{L}_v^i$ such that $c_l(w') \leq r^i c_l(w)$ for $l = 1, ..., d$. We prove this by induction.

For $i = 1$ consider a walk $w \in W^1(s, v)$. At each round all elements are examined, hence there must be $w' \in \mathcal{L}_v^1$ such that (i) $pos(w') = pos(w)$ and (ii) $c_d(w') \leq c_d(w)$. From (i) it follows that for all $l = 1, ..., d-1$ we have $\left\lfloor \log_r \frac{c_l(w')}{c_l^{min}} \right\rfloor = \left\lfloor \log_r \frac{c_l(w)}{c_l^{min}} \right\rfloor$, which implies $\log_r \frac{c_l(w')}{c_l^{min}} - 1 \leq \log_r \frac{c_l(w)}{c_l^{min}}$. Along with (ii) this implies that $c_l(w') \leq r c_l(w)$ for $l = 1, ..., d$.

For the induction step we consider a walk $w \in W^i(s, v)$. Removing the last edge of w, say uv, results in a new walk $w' \in W^{i-1}(s, u)$. Using the induction hypothesis we get that here is a $w'' \in \mathcal{L}_u^{i-1}$ such that $c_l(w'') \leq r^{i-1} c_l(w')$ for $l = 1, ..., d$. By adding edge uv to w'', which gives a walk q, we have

$$c_l(q) \leq r^{i-1} c_l(w) \quad l = 1, ..., d \tag{2}$$

Clearly, in the i^{th} round of the algorithm q is considered. Therefore, there is a walk $q' \in \mathcal{L}_v^i$ such that (iii) $pos(q') = pos(q)$ and (iv) $c_d(q') \leq c_d(q)$. From (iii) it follows that for all $l = 1, ..., d-1$ we have $\left\lfloor \log_r \frac{c_l(q')}{c_l^{min}} \right\rfloor = \left\lfloor \log_r \frac{c_l(q)}{c_l^{min}} \right\rfloor$, which implies $\log_r \frac{c_l(q')}{c_l^{min}} - 1 \leq \log_r \frac{c_l(q)}{c_l^{min}}$. Along with (iv) this implies that

$$c_l(q') \leq r c_l(q) \quad l = 1, ..., d \tag{3}$$

Combining (2) and (3) we find that $c_l(q') \leq r^i c_l(w)$ for $l = 1, ..., d$. □

To illustrate the idea, we consider Figure 5, where the same set of points as in Figure 4 is shown. The dotted lines indicate the value of the position function. Note that the steps are smaller than in Figure 4, since we use $r = (1 + \varepsilon)^{1/(n-1)}$. Furthermore, note that the algorithm is exact in one dimension (in this case c_1), therefore, for each position value, the leftmost point is added. We also note that the algorithm does not consider dominance when creating the labels, as we observe from the selected points in Figure 5.

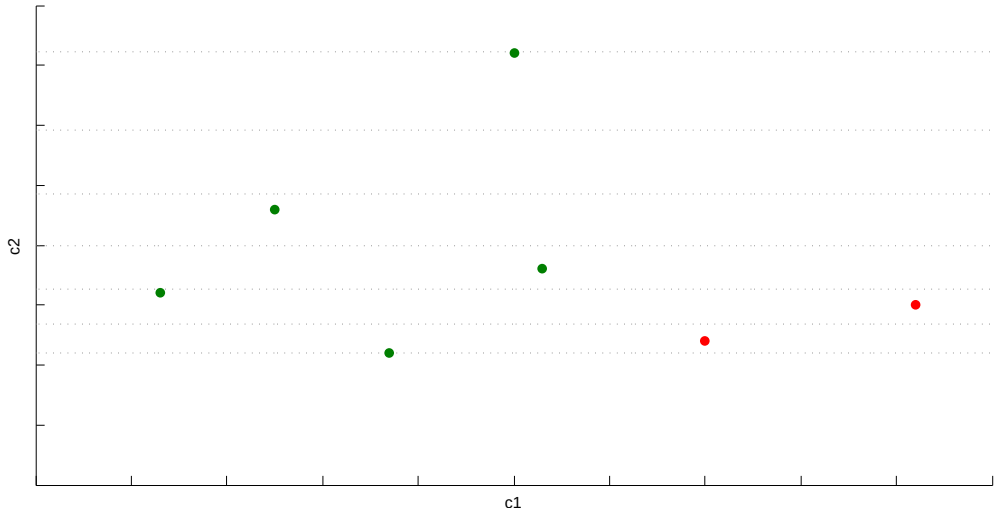

Figure 5: FPTAS Tsaggouris and Zaroliagis [2009]

We now consider the runtime of this algorithm.

Lemma 5.3. *The algorithm given in Tsaggouris and Zaroliagis [2009] constructs an ε-approximate Pareto frontier for π in $O\left(nm\left(n\log(nC^{max})/\varepsilon\right)^{d-1}\right)$ time.*

Proof. From Lemma 5.2 it is clear that, by setting $r = (1+\varepsilon)^{1/n-1}$, $\mathcal{L}_t^{n-1}$ is the desired ε-approximate Pareto frontier. The algorithms terminates after $n-1$ rounds and in each round it examines all m edges. For each edge, we consider extending all labels. The time this takes is equal to the size of the arrays used, i.e., is $O\left((\log_r(nC^{max}))^{d-1}\right)$. Using the fact that $\log_r(x) = \log(x)/\log(r)$ and $\log(r) \approx \varepsilon/(n-1)$ for ε small gives the result. □

5.4 New FPTAS

For the third algorithm, we propose an FPTAS that combines features of both the MOSP algorithm of Section 4.3 and the algorithm of Tsaggouris and Zaroliagis [2009]. Motivation for this algorithm is the fact that experiments showed that the set of efficient paths is often very small. The MOSP algorithm of Section 4.3 is able to be highly efficient in such a situation, because of the lexographic lists and dominance checks. The algorithm of Tsaggouris and Zaroliagis [2009], on the other hand, seems to profit very little of this fact (as already mentioned, for example, it does not consider dominance among the kept solutions).

Therefore, we propose a new algorithm which extends the MOSP algorithm of Section 4.3 with the position function used in Tsaggouris and Zaroliagis [2009] to construct an FPTAS that is able to exploit small sets of efficient paths.

The algorithm is as follows. We follow the exact same procedure as the algorithm of Section 4.3, with one crucial difference. Whenever a label w is to be made permanent, we do not add it to $\mathcal{L}_P$ immediately. Instead we check if there is a label $w' \in \mathcal{L}_P$, with the same end node, such that (i) $pos(w') \leq pos(w)$ and (ii) $c_d(w') \leq c_d(w)$. If both hold, we do not add w and continue the algorithm. If (i) holds exact, i.e., $pos(w') = pos(w)$, we keep the one with the lowest value $c_d(\cdot)$. Note that the algorithm considers a dominance check when a new label is created (among the labels in $\mathcal{L}_P$ and $\mathcal{L}_T$). This check is still *exact*, i.e., we do not consider $pos(\cdot)$ here. The lexographic ordering of the cost vectors first considers the d^{th} dimension. This small fact is important, as the following useful Lemma shows.

Lemma 5.4. *For every value $pos(\cdot)$ can take and for each $v \in V$, we add at most one label to $\mathcal{L}_P$.*

Proof. Suppose we first add a label w and then a label w' at the same position, both ending at the same node. Because w' is added, we have $c_d(w') < c_d(w)$, and thus $w' <_{lex} w$ (recall that lexographic order was established starting at dimension d).

Let u be the last node w and w' have in common, and let w_{sl} denote the subwalk of w from s to l. It is clear that $w'_{sl} <_{lex} w$, for all l visited after u. Because w' is made permanent it follows that all subwalks are made permanent before. Therefore, after w_{su} is made permanent, at least one of the subwalks w_{sl} is contained in $\mathcal{L}_T$ in each iteration, until w' is made permanent. Combining this with the fact that w is made permanent before w' implies that at least one of the subwalks was in $\mathcal{L}_T$ when w was made permanent. This leads to a contradiction, since the lexographic smallest path is made permanent. □

We are now able to proof the correctness of the above algorithm, the proof is closely related to the proof of Lemma 5.2.

Lemma 5.5. *For all $i \leq n-1$ and $v \in V$, the algorithm finds an r^i-approximate Pareto frontier for $W^i(s,v)$.*

Proof. It suffices to proof that for all $w \in W^i(s,v)$ there is a label $w' \in W(s,v)$ made permanent such that $c_l(w') \leq r^i c_l(w)$ for $l = 1, ..., d$. We prove this by induction.

For $i = 1$, consider any efficient walk $w \in W^1(s,v)$, $v \in V$. it is clear that we add w to $\mathcal{L}_T$. Since w is efficient it is never deleted from $\mathcal{L}_T$ (recall that the dominance check is *exact*). Therefore at some point this label is lexographic smallest, and is thus considered to be made permanent. This

implies that there is a label $w' \in W(s,v)$ made permanent with (i) $pos(w') \leq pos(w)$ and (ii) $c_d(w') \leq c_d(w)$. From (i) it follows that for all $l = 1, ..., d-1$ we have $\left\lfloor \log_r \frac{c_l(w')}{c_l^{min}} \right\rfloor \leq \left\lfloor \log_r \frac{c_l(w)}{c_l^{min}} \right\rfloor$, which implies $\log_r \frac{c_l(w')}{c_l^{min}} - 1 \leq \log_r \frac{c_l(w)}{c_l^{min}}$. Along with (ii) this implies that $c_l(w') \leq r c_l(w)$ for $l = 1, ..., d$. By Lemma 5.4 this label will be in the final set of permanent labels, which gives the result for $i = 1$.

For the induction step we consider a walk $w \in W^i(s,v)$. Removing the last edge of w, say uv, results in a new walk $w' \in W^{i-1}(s,u)$. By the induction hypothesis we get that there is a label $w'' \in W(s,u)$ made permanent such that $c_l(w'') \leq r^{i-1} c_l(w')$ for $l = 1, ..., d$. By adding edge uv to w'', which gives a walk $q \in W(s,v)$, we have

$$c_l(q) \leq r^{i-1} c_l(w) \quad l = 1, ..., d. \tag{4}$$

Clearly, when w'' is made permanent walk q is considered. Now either q is considered to be made permanent or it is deleted from $\mathcal{L}_T$. The latter can only occur if we find a label $q' \in W(s,v)$ with $c_l(q') \leq c_l(q)$, $l = 1, ..., d$. Repeating this argument, it is clear that we will always consider a label $q' \in W(s,v)$ to be made permanent with

$$c_l(q') \leq c_l(q) \quad l = 1, ..., d. \tag{5}$$

Therefore, there is a walk $q'' \in W(s,v)$ made permanent such that (iii) $pos(q'') \leq pos(q')$ and (iv) $c_d(q'') \leq c_d(q')$. From (iii) it follows that for all $l = 1, ..., d-1$ we have $\left\lfloor \log_r \frac{c_l(q'')}{c_l^{min}} \right\rfloor \leq \left\lfloor \log_r \frac{c_l(q')}{c_l^{min}} \right\rfloor$, which implies $\log_r \frac{c_l(q'')}{c_l^{min}} - 1 \leq \log_r \frac{c_l(q')}{c_l^{min}}$. Along with (iv) this implies that

$$c_l(q'') \leq r c_l(q') \quad l = 1, ..., d \tag{6}$$

Combining (4), (5) and (6) we find that $c_l(q'') \leq r^i c_l(w)$ for $l = 1, ..., d$. By Lemma 5.4 this label will be in the final set of permanent labels, which completes the induction step.

□

This gives the main result regarding the new FPTAS.

Theorem 5.3. *The algorithm constructs an ε-approximate Pareto frontier for π in $O\left(\left(n^2 \left(n \log(nC^{max})/\varepsilon\right)^{d-1}\right)^2\right)$ time.*

Proof. It is clear that the returned set of permanent labels is the desired ε-approximate Pareto frontier. Due to Lemma 5.4 it is clear that the number of permanent labels created is at most $O\left(n \left(n \log(nC^{max})/\varepsilon\right)^{d-1}\right)$, i.e., n times the number of different values $pos(\cdot)$ can take. For every label made permanent we create at most n labels, hence the total number of labels created is order $O\left(n^2 \left(n \log(nC^{max})/\varepsilon\right)^{d-1}\right)$. For each of the created labels we need to check dominance, which is proportional to the number of labels, and hence takes $O\left(n^2 \left(n \log(nC^{max})/\varepsilon\right)^{d-1}\right)$ time. Note that the extra check when considering making a label permanent can be seen as an extra dominance check, hence the run time increases by at most a factor two. The result follows. □

Although the asymptotic running time of the algorithm seems substantially higher than the FPTAS of Tsaggouris and Zaroliagis [2009], we should note that this only occurs in the worst possible scenarios. To illustrate this: when we have $pos(p) \neq pos(p')$ for each pair p, p' of $s - v$ paths, $v \in V$ (i.e., the position function is irrelevant) the new FPTAS is almost as fast as the algorithm of Section 4.3, while the run time of the FPTAS of Tsaggouris and Zaroliagis [2009] might not improve much (or might not improve at all).

6 Performance Analysis

In this section we analyze the performance of the algorithms on different MOSP instances. Doing this we consider, empirically, three different types of analysis.

The first one we consider is worst case analysis, which is probably the most well known form of analyzing algorithms; one conisders the performance of algorithms on the *worst possible* inputs (e.g., MOSP instances with an exponential number of efficient paths). Secondly, we consider average case analysis. In this type of analysis a probability distribution over a set of inputs is considered. The running time of the algorithm is then evaluated based on the average running time considering this distribution. The third type of analysis we consider is smoothed analysis. This type of analysis, which is relatively new, can be seen as a hybrid of the above two methods: one evaluates the performance of the algorithms on worst case instances, where some attributes are randomly perturbed (e.g., small random cost changes to edge weights). We consider two different frameworks for such perturbations.

We will analyze the exact algorithm, given in Section 4.3, the FPTAS of Tsaggouris and Zaroliagis [2009], described in 5.3 and the new FPTAS, given in Section 5.4. We will at times simply refer to them as exact, first FPTAS and second FPTAS. The FPTAS given in Mittal and Schulz [2008] will not be analyzed. As already mentioned in Section 5.2, the theoretical running time, although polynomial, is extremely high. We were unable to solve instances with as little as 5 nodes using this algorithm. We may conclude that the algorithm is impractical, although its theoretical value is still important (since it is an FPTAS for a large class of problems).

We observed that for both FPTASes the *actual* approximation factors were very close to one, unless the number of efficient paths was exponential (which is obvious). Varying parameters such as density or the range of edge costs did not change this observation, we therefore omit analysis on this subject in the thesis.

6.1 Worst Case Analysis

Worst case analysis is the most common way of analyzing the performance of algorithms. It is the form of analysis on which, for example, the famous $\mathsf{P} = \mathsf{NP}$ question is based. As mentioned above, worst case analysis consists on evaluating algorithms on the *worst* possible input. Arora and Barak [2009] give a detailed treatise on the subject and related topics.

We evaluate the worst case performance based on the artificial instance used to proof intractability in Section 4.4. We solve this problem for $n = 15, 17, ..., 35$. For the FPTASes we used an approximation factor $\varepsilon = 0.1$. The results are shown in Figure 6. Since the theoretical worst case performance of the algorithms is already established in Sections 4 and 5, we will only discuss it briefly.

Figure 6 uses two different y-axes, the left one for the exact algorithm (ranging from 0 to 4000 seconds) and the right one for the two FPTASes (ranging from 0 to 40 seconds). We observe the steep curve in the running time of the exact algorithm compared to the two FPTASes. This is of course to be expected, due to the reasoning in Section 4.4. Furthermore, we see that the FPTAS of Tsaggouris and Zaroliagis [2009] has far less trouble with the instances than the FPTAS proposed in Section 5.4. This too is to be expected; the dominance checks are costly, since the set is large, and lead to no improvement, since all paths are efficient. The first FPTAS omits these steps and is therefore able to solve the instances more efficiently.

6.2 Average Case Analysis

Average case analysis is another well known form of analysis (see, for example, Arora and Barak [2009] for a formal treatment). The main idea of this approach is to consider instances that are

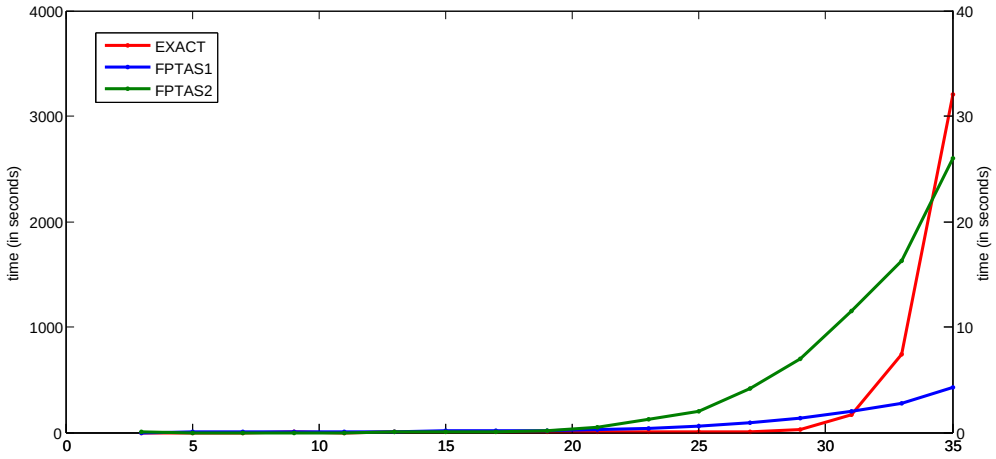

Figure 6: Worst case analysis

'likely to occur', instead of the, often artificial, worst case instances. This method thus tries to analyze an algorithm from a more practical point of view.

The main problem with this type of analysis is deciding which distribution should be used, since 'average' cases can differ substantially in practice (as an example, an average MOSP problem occurring in a pure navigational setting might be very different from one occurring when optimizing databases). We will consider the probability space $\mathcal{G}(n, \rho)$, i.e., the space of graphs of n nodes and density ρ. We consider graphs of 30 nodes, where the costs of the edges are independent uniform distributed between 1 and 2^{20}. The results for $\varepsilon = 0.1$ are shown in Figure 7, the results for $\varepsilon = 0.01$ are shown in Figure 10 (see Appendix B). For each of the densities $0.1, 0.2, ..., 1$ we solved 20 instances. The dotted lines indicate 95% confidence intervals. The orange line indicates the number of found efficient $s - t$ paths. In the remaining figures, the left axis will indicate the number of paths, while the right axis indicates the running times.

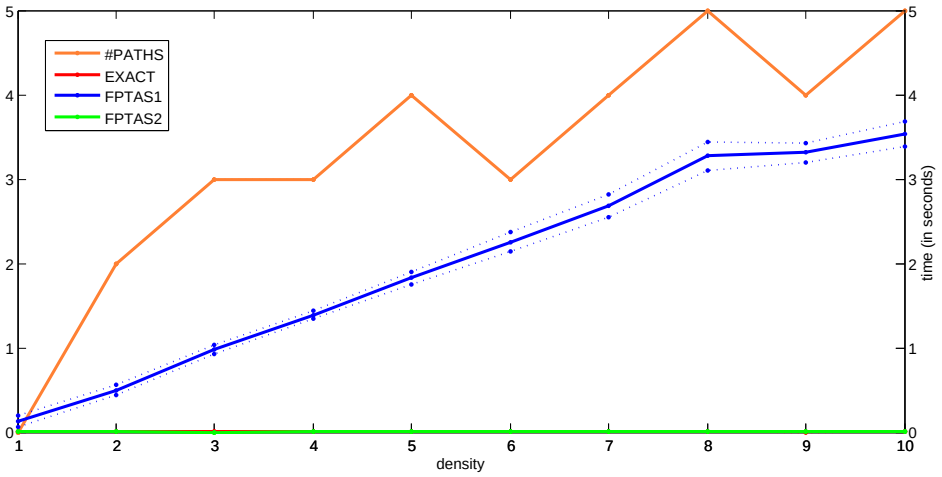

Figure 7: Average case complexity with $\varepsilon = 0.1$

The results in Figure 7, as well as Figure 10, are clear: both the exact algorithm and the second

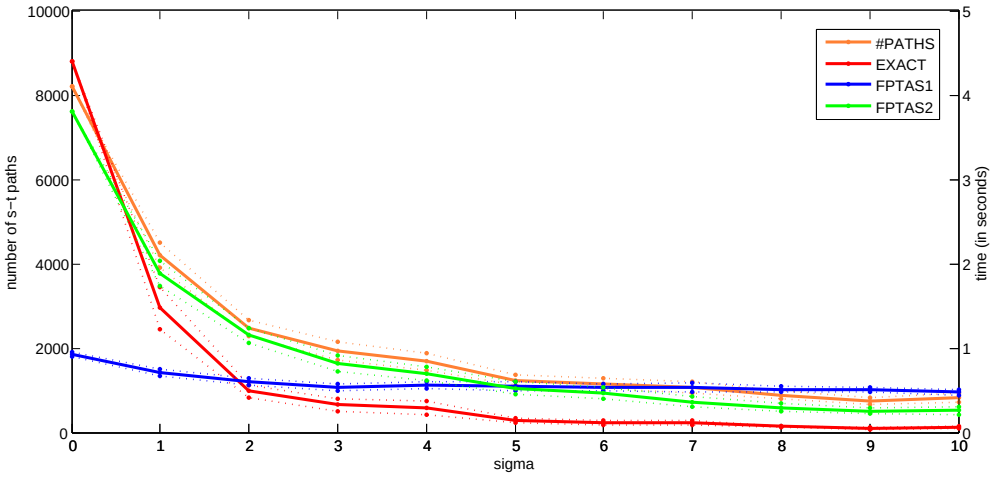

Figure 8: Smoothed analysis based on cost perturbations

FPTAS have no trouble with the instances (their running time seems not distinguishable from zero in both figures). This is in line with the motivation for the second FPTAS; the algorithm seems to be highly efficient when the Pareto-curve is small, as was the case for the random graphs. The exact algorithm is also highly efficient on the random instances, due to the small size of the Pareto-curve.

We see that the FPTAS of Tsaggouris and Zaroliagis [2009] takes considerably longer time to solve the instances, especially when ε becomes smaller. Furthermore, the running time steadily increases. This is due to the fixed size of the arrays that are considered, and the fact that, if the number of edges increase, the expected maximum cost value will increase as well. As mentioned before, we also observe that the running time of the first FPTAS increases substantially as ε decreases (as expected, of course), but that this does not hold for the second FPTAS; it seems that the runtime of the second FPTAS is independent of the chosen value of ε, when the Pareto curve is small (note that this is in line with the conclusive remarks of Section 5.4).

6.3 Smoothed Analysis

The third type of analysis we consider is smoothed analysis. As mentioned above, this type of analysis is relatively new, and is based on small 'errors' around worst case instances. Such errors seem very plausible when one considers, for example, small rounding errors or human mistakes. The idea behind smoothed analysis is to bridge the gap between average case and worst case analysis. Spielman and Teng [2009] give an in depth treatise on the subject. Interesting results are shown in Spielman and Teng [2003]; they show that the smoothed complexity of the well known Simplex Algorithm (see Papadimitriou and Steiglitz [1982]) is polynomial (under a certain pivot rule) in the input size and the standard deviation of the perturbations, thereby providing new insights on the fact that the Simplex Algorithm is fast in practice, although the worst case running time is exponential. Inspired by this idea, we consider two different frameworks for analyzing the algorithms.

The first one we consider is based on random perturbations of the costs of the edges. We consider the worst case scenario used in Section 4.4 and analyze what happens if we add independent $N(0, \sigma^2)$ distributed errors to the edge costs (rounding to the nearest integer afterwards to keep costs integer). The results of this method are shown in Figure 8, for the instance with $n = 27$ nodes and $\varepsilon = 0.1$. The values are based on solving 50 instances, the dotted lines indicate 95% confidence intervals. As before, the orange line shows the number of efficient $s - t$ paths.

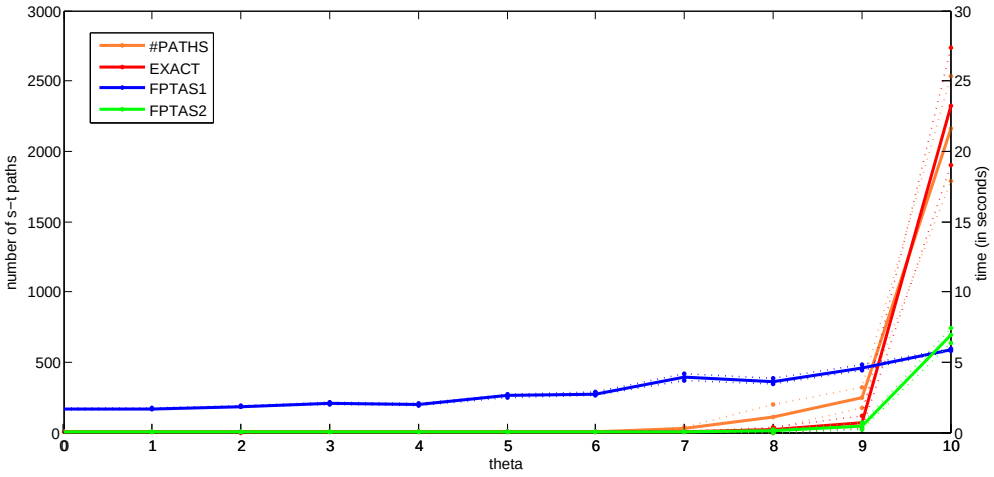

Figure 9: Transition analysis for $\rho = 0.6$.

As can be observed in Figure 8, even small perturbations decrease the size of the efficient set (and hence the run time of the exact algorithm and the second FPTAS) drastically. In fact, for $\sigma = 10$ we see that the running times start to look like the ones observed on random instances, even though the number of efficient paths is still substantial. We note that the value of σ is small compared to the costs in the graph: the largest cost is $2^{(n-3)/2} = 4096$. From these results it seems unlikely that, considering instances with some randomness, we will observe very high running times.

The second framework we consider is somewhat more involved. Because a MOSP instance has multiple components (there is the topology of the graph and the cost vectors) we also consider changes in terms of the graph. The idea of the method is that we simulate the *transition* of a random instance to the worst case instance (therefore, we will simply refer to it as transition analysis). The procedure is as follows.

1. Given input θ and ρ, both between 0 and 1, and the number of nodes n, generate a random graph $G = (V, E) \in \mathcal{G}(n, \rho)$, with integer costs uniformly distributed between 1 to 2^n. Furthermore, construct a worst case instance (as in Section 4.4) on n nodes, say $G' = (V', E')$.
2. For each edge $e' \in E'$, select an edge $e \in E$ at random. With probability θ, remove e from E and add e' to E.
3. return G.

Note that the procedure is fully specified by a pair (ρ, θ), for fixed n. The two parameter specification allows us to look at multiple interesting situations. For example, by setting $\theta = 1$, we consider what happens when we add extra edges to the worst case instance. By varying ρ we are then able to analyze if it matters *how many* edges are added. On the other hand, by setting $\theta < 1$ we are able to see the effect of a 'partial occurrence' of the worst case instance.

In Figure 9 the results are shown for $\rho = 0.6$, as a function of θ (for $\rho = 0.4$ and $\rho = 0.8$, see Appendix B), with $n = 27$. As before we used $\varepsilon = 0.1$ and based our results on 50 instances. We chose ρ not too small, so that the effect of extra edges could be observed (note that by setting $\theta = 1$ and ρ sufficiently small we get precisely the worst case instance on 27 nodes).

The results show some striking phenomena. First, there seems to be a sharp transition around $\theta = 0.9$, when the run times increase substantially. For θ small the results look similar to those on random instances (which is no surprise), but also for θ as large as 0.8 the number of efficient paths

is still very small, and both the exact algorithm as the first FPTAS seem to have no problem at all. This indicates that the number of efficient paths due to the worst case instance depends heavily on its *exact* structure: the removal of a small fraction of the edges combined with the addition of (randomly chosen) new edges greatly decreases the number of efficient paths. Furthermore, for θ large, the variance increases substantially, indicating that the running time depends heavily on the generated graph. We also note that the average runtime for $\theta = 1$ is higher than for the worst case analysis; the addition of extra edges seems to result in higher run times. This can be explained by the fact that the average degree is higher, and hence the possible number of paths that need to be explored is larger. This agrees with the fact that although the number of efficient paths is close to what is observed in Figure 8, the running time is substantially higher.

Another interesting aspect are the differences observed when varying ρ. We see that the number of efficient $s-t$ paths decreases as ρ increases, due to the fact that the number of edges increases and hence the probability of short (efficient) paths increases. Note that this is a different phenomena than in the average case analysis, since we are now adding edges to an instance with an already large set of efficient paths, instead of considering an entirely random graph. The differences in terms of running time are harder to interpret. Although the variation is high, the running time for $\theta = 1$ seems substantially higher for $\rho = 0.6$ compared to the other two cases. This indicates that the running time is a trade-off between the number of efficient paths on one hand, and the number of edges (or the average degree) on the other hand.

Summarizing, we analyzed the algorithms based on different frameworks. We observed small Pareto curves for the random graphs and the smoothed analysis (excluding the transition analysis with $\theta = 1$). For such instances the exact algorithm and the second FPTAS easily outperformed the first FPTAS. Furthermore, increasing the density or approximation ratio increased the running time of the FPTAS of Tsaggouris and Zaroliagis [2009] substantially, while the other run times remained almost unchanged. When considering worst case analysis, on the other hand, the FPTAS of Tsaggouris and Zaroliagis [2009] outperformed the others, especially when the number of nodes got larger. This was also the case for the transition analysis with $\theta = 1$, although the observed difference in running time with the second FPTAS was small (this difference will probably grow, however, when the number of nodes increases).

7 Conclusion

In this thesis we evaluated three different FPTASes for the multi-objective shortest path problem. The first two were the FPTASes of Mittal and Schulz [2008] and Tsaggouris and Zaroliagis [2009]. The third FPTAS is a new addition to the literature, intended to benefit from small Pareto-optimal frontiers. This was achieved by using elements of both the exact algorithm and the FPTAS of Tsaggouris and Zaroliagis [2009].

In this light the results of Section 6 were promising; we observed a very small number of efficient paths for random graphs, and also in the more involved smoothed analysis we saw evidence for small Pareto curves. In the case of random perturbations the number of efficient paths sharply fell, even for a standard deviation of less than one percent of the maximum cost. In the transition analysis the observed size of the Pareto curve was small as well, unless we considered the case $\theta = 1$. In this analysis we also observed that increasing the density resulted in a smaller number of efficient paths, although this did not directly result in a shorter running time.

When we considered instances with a very large number of efficient paths (e.g., worst case analysis and transition analysis with $\theta = 1$), the FPTAS of Tsaggouris and Zaroliagis [2009] outperformed all others. Therefore, the FPTASes seem to complement each other; when one expects a high number of efficient paths, the FPTAS of Tsaggouris and Zaroliagis [2009] is probably the best choice, while otherwise the new FPTAS is recommended. Especially when the graph has high density combined with large costs, but few efficient paths, the use of the second FPTAS greatly reduces the run time. As a negative result, we observed that the running time of the FPTAS of Mittal and Schulz [2008] was too high to be included in the analysis; the applicability of this FPTAS seems to be restricted to theory.

For further research the results on smoothed complexity seem promising. It would be interesting to derive an expression for the run time, in terms of the standard deviation of the edge costs. The two parameter analysis seems too complicated for theoretical analysis, although results might be obtained when considering one parameter fixed. A further empirical study of the performance of the FPTASes also seems fruitful; the results in Section 6 gave some interesting insights, but it should be validated if these results also hold in different settings.

Appendix A

Lemma A.1. *Suppose* $f(x) = \sum_{j=1}^{m} a_j x_j$, *with* $0 \leq a_j \leq v$ *and* $x_j \in \{0, ..., n\}$. *Furthermore, let* $r = \lceil nm/\varepsilon \rceil$ *and* $f'(x) = \sum_{j=1}^{m} a'_j x_j$, *where* $a'_j = r = \lceil a_j/v \rceil$. *Then*

1. *if* $f'(x) \leq r$, *then* $f(x) \leq v$.
2. *if* $f(x) \leq v(1-\varepsilon)$, *then* $f'(x) \leq r$.

Proof. 1. Given that $f'(x) \leq r$, we have

$$f(x) = \sum_{j=1}^{m} a_j x_j = \frac{v}{r} \sum_{j=1}^{m} \frac{a_j r}{v} x_j \leq \frac{v}{r} \sum_{j=1}^{m} \left\lceil \frac{a_j r}{v} \right\rceil x_j = \frac{v}{r} f'(x) \leq v.$$

2. If $f(x) \leq v(1-\varepsilon)$, we have

$$\sum_{j=1}^{m} \frac{a_j r}{v} x_j = \frac{r}{v} f(x) \leq \frac{r}{v} v(1-\varepsilon) = r(1-\varepsilon).$$

Rounding up all terms on the left side we obtain

$$\sum_{j=1}^{m} \left\lceil \frac{a_j r}{v} \right\rceil x_j = f'(x) \leq r(1-\varepsilon) + nm,$$

which implies that

$$f'(x) \leq r(1-\varepsilon) + nm = r - \left\lceil \frac{nm}{\varepsilon} \right\rceil \varepsilon + nm \leq r.$$

□

Appendix B

Figure 10 shows the average case analysis (discussed in Section 6.2) for $\varepsilon = 0.01$.

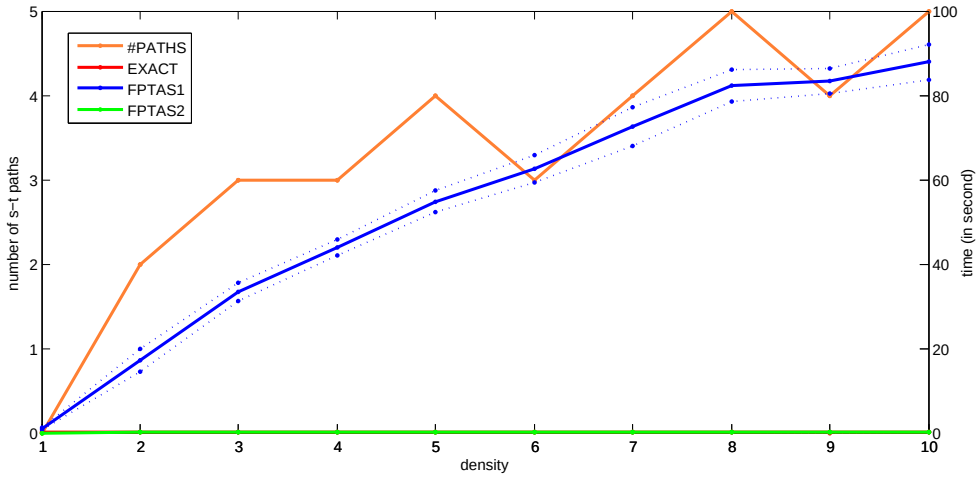

Figure 10: Average case complexity with $\varepsilon = 0.01$

Figures 11 and 12 show the transition analysis (discussed in Section 6.3) for $\rho = 0.4$ and $\rho = 0.8$.

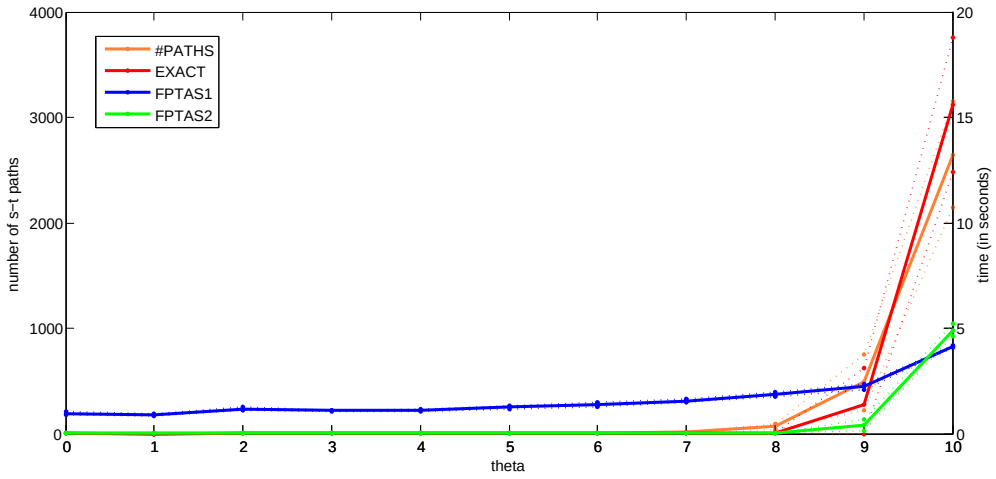

Figure 11: Transition analysis for $\rho = 0.4$.

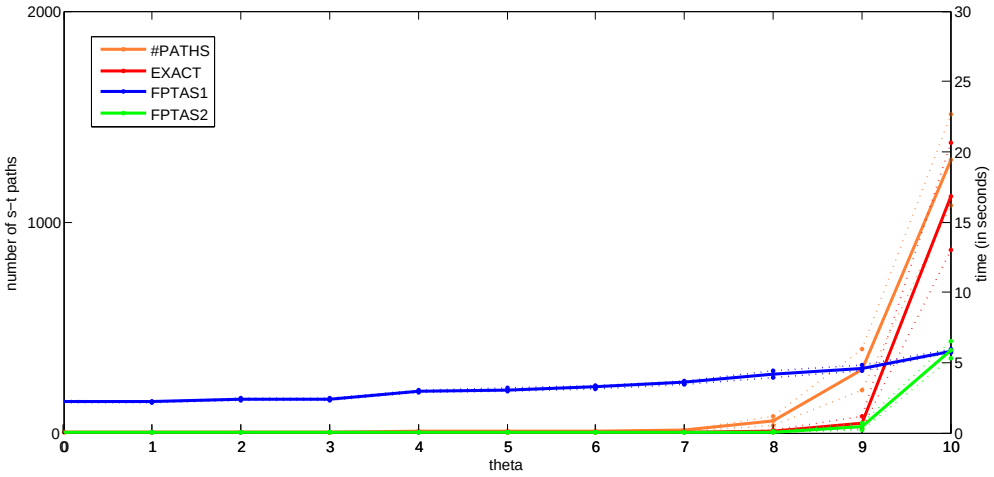

Figure 12: Transition analysis for $\rho = 0.8$.

References

S. Arora. The approximability of NP-hard problems. In *In Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 337–348, 1998.

S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009. ISBN 0521424267, 9780521424264.

N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. 1975.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. ISSN 0029-599X. doi: 10.1007/BF01386390. URL `http://dx.doi.org/10.1007/BF01386390`.

M. Ehrgott. *Multicriteria optimization*. Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, 2000.

M. Karpinski, M. Lampis, and R. Schmied. New Inapproximability Bounds for TSP. 2013.

W. Kern and G. J. Woeginger. Quadratic programming and combinatorial minimum weight product problems. *Mathematical Programming*, 110(3):641–649, 2007. ISSN 0025-5610. doi: 10.1007/s10107-006-0047-7. URL `http://dx.doi.org/10.1007/s10107-006-0047-7`.

S. Mittal and A. S. Schulz. A General Framework for Designing Approximation Schemes for Combinatorial Optimization Problems with Many Objectives Combined into One. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, volume 5171 of *Lecture Notes in Computer Science*, pages 179–192. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-85362-6.

M. Müller-Hannemann and K. Weihe. On the cardinality of the Pareto set in bicriteria shortest path problems. *Annals of Operations Research*, 147(1):269–286, 2006. ISSN 0254-5330. doi: 10.1007/s10479-006-0072-1. URL `http://dx.doi.org/10.1007/s10479-006-0072-1`.

C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982. ISBN 0-13-152462-3.

C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources (Extended Abstract). In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 86–92, 2000.

A. J. V. Skriver. A classification of bicriterion shortest path (BSP) algorithms. *Asia Pacific Journal of Operational Research*, 17(2):199–212, 2000.

D. A. Spielman and S. Teng. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time, 2003.

D. A. Spielman and S. Teng. Smoothed analysis: an attempt to explain the behavior of algorithms in practice. *Commun. ACM*, 2009.

Z. Tarapata. Selected Multicriteria Shortest Path Problems: An Analysis of Complexity, Models and Adaptation of Standard Algorithms. *Int. J. Appl. Math. Comput. Sci.*, 17(2):269–287, June 2007. ISSN 1641-876X. doi: 10.2478/v10006-007-0023-2. URL `http://dx.doi.org/10.2478/v10006-007-0023-2`.

G. Tsaggouris and C. Zaroliagis. Multiobjective Optimization: Improved FPTAS for Shortest Paths and Non-Linear Objectives with Applications. *Theor. Comp. Sys.*, 45 (1):162–186, April 2009. ISSN 1432-4350. doi: 10.1007/s00224-007-9096-4. URL `http://dx.doi.org/10.1007/s00224-007-9096-4`.

G. J. Woeginger. When Does a Dynamic Programming Formulation Guarantee the Existence of an FPTAS? In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '99, pages 820–829, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics. ISBN 0-89871-434-6. URL `http://dl.acm.org/citation.cfm?id=314500.314921`.

E. Zitzler. Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications, 1999.