Theory Comput Syst (2009) 45: 162–186
DOI 10.1007/s00224-007-9096-4

Multiobjective Optimization: Improved FPTAS for Shortest Paths and Non-Linear Objectives with Applications

George Tsaggouris · Christos Zaroliagis

Published online: 28 November 2007

Abstract We provide an improved FPTAS for multiobjective shortest paths—a fundamental (NP-hard) problem in multiobjective optimization—along with a new generic method for obtaining FPTAS to any multiobjective optimization problem with *non-linear* objectives. We show how these results can be used to obtain better approximate solutions to three related problems, multiobjective constrained [optimal] path and non-additive shortest path, that have important applications in QoS routing and in traffic optimization. We also show how to obtain a FPTAS to a natural generalization of the weighted multicommodity flow problem with elastic demands and values that models several realistic scenarios in transportation and communication networks.

Keywords Multiobjective optimization · Multiobjective shortes path · FPTAS · Non-linear objectives · Multiple constrained (optimal) path · Non-additive shortest path · Qos-aware multicommodity flow

This work was partially supported by the Future and Emerging Technologies Unit of EC (IST priority—6th FP), under contracts No. IST-2002-001907 (integrated project DELIS) and No. FP6-021235-2 (project ARRIVAL), and the Action PYTHAGORAS of the Operational Programme for Educational & Vocational Training II, with matching funds from the European Social Fund and the Greek Ministry of Education.

G. Tsaggouris · C. Zaroliagis
R.A. Computer Technology Institute, Patras University Campus, N. Kazantzaki Str, 26500 Patras, Greece

G. Tsaggouris · C. Zaroliagis (✉)
Department of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece
e-mail: zaro@ceid.upatras.gr

G. Tsaggouris
e-mail: tsaggour@ceid.upatras.gr

1 Introduction

Multiobjective shortest paths (MOSP) is a core problem in the area of multiobjective optimization (an area under intense study within Operations Research and Economics in the last 60 years [7, 8]) with numerous applications that span from traffic optimization to quality-of-service (QoS) routing in networks and multicriteria decision making [10]. Informally, the problem consists in finding a set of paths that captures not a single optimum but the trade-off among $d > 1$ objective functions in a digraph whose edges are associated with d-dimensional attribute (cost) vectors.

In general, an instance of a multiobjective optimization problem is associated with a set of feasible solutions Q and a d-vector function $\mathbf{f} = [f_1, \dots, f_d]^T$ (d is typically a constant) associating each feasible solution $q \in Q$ with a d-vector $\mathbf{f}(q)$. Without loss of generality, we assume that all objectives f_i, $1 \le i \le d$, are to be minimized. In a multiobjective optimization problem, we are interested not in finding a single optimal solution, but in computing the trade-off among the different objective functions, called the *Pareto set* or *curve* $\mathcal{P}$. The set $\mathcal{P}$ is the set of all feasible solutions in Q whose vector of the various objectives is *not* dominated by any other solution; a solution p *dominates* another solution q iff $f_i(p) \le f_i(q)$, $\forall 1 \le i \le d$, with a strict inequality for some i. Multiobjective optimization problems are usually NP-hard, as indeed is the case for MOSP. This is due to the fact that the Pareto curve is typically exponential in size (even in the case of two objectives). On the other hand, even if a decision maker is armed with the entire Pareto curve, s/he is left with the problem of which is the "best" solution for the application at hand. Consequently, three natural approaches to solve multiobjective optimization problems are to: (i) study approximate versions of the Pareto curve; (ii) optimize one objective while bounding the rest (*constrained approach*); and (iii) proceed in a normative way and choose the "best" solution by introducing a utility (typically non-linear) function on the objectives (*normalization approach*). In this paper, we investigate all these approaches for the multiobjective shortest path problem. In particular, we provide an improved FPTAS for MOSP along with a result of independent interest: a new generic method for obtaining FPTAS to *any* multiobjective optimization problem with non-linear objectives. We show how these results can be used to obtain efficient approximate solutions to the *multiple constrained* (*optimal*) *path* problems (constrained approach), and to the *non-additive shortest path* problem (normalization approach). We also show how efficient approximate solutions can be obtained in a completely different context; namely, to a natural generalization of the weighted multicommodity flow problem with elastic demands and values that models several realistic scenarios in transportation and communication networks.

1.1 Multiobjective Shortest Paths

Despite so much research in multiobjective optimization [7, 8], only recently a systematic study of the complexity issues regarding the construction of approximate Pareto curves has been initiated [24, 30]. Informally, a $(1+\varepsilon)$-*Pareto curve* $\mathcal{P}_\varepsilon$ is a subset of feasible solutions such that for any Pareto optimal solution, there exists a solution in $\mathcal{P}_\varepsilon$ that is no more than a factor of $(1+\varepsilon)$ away in all objectives. Papadimitriou and Yannakakis show in a seminal work [24] that for any multiobjective optimization problem there exists a $(1+\varepsilon)$-Pareto curve $\mathcal{P}_\varepsilon$ of (polynomial) size $|\mathcal{P}_\varepsilon| = O((4B/\varepsilon)^{d-1})$, where B is the number of bits required to represent the values in the objective functions (bounded by some polynomial in the size of the input); $\mathcal{P}_\varepsilon$ can be constructed by $O((4B/\varepsilon)^d)$ calls to a GAP routine that solves, in time polynomial in the size of the input and $1/\varepsilon$, the following problem: given a vector of values $\mathbf{a}$, either compute a solution that dominates $\mathbf{a}$, or report that there is no solution better than $\mathbf{a}$ by at least a factor of $1+\varepsilon$ in all objectives.

For the case of MOSP (and some other problems with linear objectives), Papadimitriou and Yannakakis [24] show how a GAP routine can be constructed (based on a pseudopolynomial algorithm for computing exact paths), and consequently provide a FPTAS for this problem. Note that FPTAS for MOSP were already known in the case of two objectives [18], as well as in the case of multiple objectives in directed acyclic graphs (DAGs) [32]. In particular, the 2-objective case has been extensively studied [8], while for $d > 2$ very little has been achieved; actually (to the best of our knowledge) the results in [24, 32] are the only and currently best FPTAS known.

Let $C^{\max}$ denote the ratio of the maximum to the minimum edge cost (in any dimension), and let n (resp. m) be the number of nodes (resp. edges) in a digraph. Note that $B = O(\log(nC^{\max}))$ for MOSP and hence $|\mathcal{P}_\varepsilon| = O((\log(nC^{\max})/\varepsilon)^{d-1})$. For the case of DAGs and $d > 2$, the algorithm of [32] runs in $O(nm(\frac{n\log(nC^{\max})}{\varepsilon})^{d-1} \times \log^{d-2}(\frac{n}{\varepsilon}))$ time, while for $d = 2$ this improves to $O(nm\frac{1}{\varepsilon}\log n \log(nC^{\max}))$. For $d = 2$, a FPTAS can be created by repeated applications of a stronger variant of the GAP routine—like a FPTAS for the restricted shortest path (RSP) problem [9, 19, 22]. In [30] it is shown that this achieves a time of $O(nm|\mathcal{P}^*_\varepsilon|(\log\log n + 1/\varepsilon))$ for general digraphs and $O(nm|\mathcal{P}^*_\varepsilon|/\varepsilon)$ for DAGs, where $|\mathcal{P}^*_\varepsilon|$ is the size of the smallest possible $(1+\varepsilon)$-Pareto curve (and which can be as large as $\log_{1+\varepsilon} nC^{\max} \approx \frac{1}{\varepsilon}\ln(nC^{\max})$).[1] All these approaches deal typically with the single-pair version of the problem.

Our first contribution in this work (Sect. 3) is a new and remarkably simple FPTAS for constructing a set of approximate Pareto curves (one for every node) for the *single-source* version of the MOSP problem in *any* digraph. For any $d > 1$, our algorithm runs in time $O(nm(\frac{n\log(nC^{\max})}{\varepsilon})^{d-1})$ for general digraphs, and in $O(m(\frac{n\log(nC^{\max})}{\varepsilon})^{d-1})$ for DAGs. Table 1 summarizes the comparison of our results with the best previous ones. Our results improve significantly upon previous approaches for general digraphs [24, 30] and DAGs [30, 32], for all $d > 2$. For $d = 2$, our running times depend on ε^{-1}, while those based on repeated RSP applications (like in [30]) depend on ε^{-2}. Hence, our algorithm gives always better running times for DAGs, while for general digraphs we improve the dependence on $1/\varepsilon$.

All previous methods are based on converting pseudopolynomial time algorithms to FPTAS using rounding and scaling techniques on the input edge costs. Our approach departs from this line of research. It builds upon a natural iterative process that extends and merges sets of node labels representing paths, while keeping them small by selectively discarding paths in an error controllable way.

[1] We would like to mention that the focus of the work in [30] is to provide approximate Pareto sets that are as small as possible.

Table 1 Comparison of new and previous results for MOSP. T_{GAP} denotes the time of a GAP routine, which is polynomial in the input and $1/\varepsilon$ (but exponential in d)

		Best previous	This work
General	$d=2$	$O(nm\frac{1}{\varepsilon}\log(nC^{\max})(\log\log n+\frac{1}{\varepsilon}))$ [30]	$O(n^2m\frac{1}{\varepsilon}\log(nC^{\max}))$
digraphs	$d>2$	$O((\log(nC^{\max})/\varepsilon)^d\cdot T_{\text{GAP}})$ [24]	$O(nm(\frac{n\log(nC^{\max})}{\varepsilon})^{d-1})$
DAGs	$d=2$	$O(nm\frac{1}{\varepsilon}\log n\log(nC^{\max}))$ [32]	$O(nm\frac{1}{\varepsilon}\log(nC^{\max}))$
		$O(nm\frac{1}{\varepsilon^2}\log(nC^{\max}))$ [30]	
	$d>2$	$O(nm(\frac{n\log(nC^{\max})}{\varepsilon})^{d-1}\log^{d-2}(\frac{n}{\varepsilon}))$ [32]	$O(m(\frac{n\log(nC^{\max})}{\varepsilon})^{d-1})$

1.2 Non-Linear Objectives

Our second contribution in this work concerns two fundamental problems in multi-objective optimization: (i) Construct a FPTAS for the normalized version of a multi-objective optimization problem when the utility function is *non-linear*. (ii) Construct a FPTAS for a multiobjective optimization problem with *non-linear* objectives.

An algorithm for the first problem was given in [27] (earlier version of this work) for $d \geq 2$ objectives and polynomial utility function, and independently in [1] for $d = 2$ objectives and quasi-polynomially bounded utility function. In the latter work [1], the authors also show that if the non-linear utility function grows (sub)exponentially in both attributes, then the first problem cannot be approximated within any polynomial factor unless P = NP. Hence, the restriction to quasi-polynomially bounded utility functions may be crucial. Let $T(1/\varepsilon, m')$ denote the time to generate a $(1+\varepsilon)$-Pareto curve for an instance of a multiobjective optimization problem of size m'. The algorithm in [1] provides a FPTAS with time complexity $T(\Lambda_1/\varepsilon^2, m')$, where Λ_1 is polylogarithmic on the maximum cost in any dimension.

We show in Sect. 4 that we can construct a FPTAS for the normalized version of *any* multiobjective optimization problem with $d \geq 2$ objectives and quasi-polynomially bounded utility function in time $T(\Lambda_2/\varepsilon, m')$, where $\Lambda_2 < \Lambda_1$ is polylogarithmic on the maximum cost in any dimension. Our results are based on a *novel* and simple analysis, and improve upon those in [1] both w.r.t. the running time (better dependence on $1/\varepsilon$ and $\Lambda_2 < \Lambda_1$) and the number of objectives—as well as upon those in [27] w.r.t. the class of utility functions.

The only generic method known for addressing the second problem is that in [24], which assumes the existence of a GAP routine. Such routines for the case of non-linear objectives are not known. The GAP routines given in [24] concern problems with linear objectives only (shortest paths, spanning tree, and perfect matching).

We show in Sect. 4 that a FPTAS for *any* multiobjective optimization problem $\mathcal{M}'$ with quasi-polynomially bounded non-linear objective functions can be constructed from a FPTAS for a much simpler version $\mathcal{M}$ of the problem. $\mathcal{M}$ has the same feasible solution set with $\mathcal{M}'$ and objectives the *identity functions* on the attributes of the non-linear objective functions of $\mathcal{M}'$. In other words, our result suggests that restricting the study of approximate Pareto curves to identity (on the attributes) objectives suffices for treating the non-linear case. Our approach constitutes the first generic method for obtaining FPTAS for any multiobjective optimization problem with quasi-polynomial non-linear objectives.

1.3 Applications

We consider four problems that play a key role in several domains, including QoS routing in communication networks, traffic equilibria, transport optimization, and information dissemination.

Multiple Constrained (Optimal) Paths. A prime research issue in networking is how to satisfy QoS requirements set by various applications running over a network (e.g., bandwidth, delay, packet loss, reliability, etc.) [23]. One of the key issues in providing QoS guarantees is how to determine paths that satisfy QoS constraints, a problem known as QoS routing or constraint-based routing. The two most fundamental problems in QoS routing are the *multiple constrained optimal path* (MCOP) and the *multiple constrained path* (MCP) problems (see e.g., [17, 21, 23]). In MCOP, we are given a d-vector of costs $\mathbf{c}$ on the edges and a $(d-1)$-vector $\mathbf{b}$ of QoS-bounds. The objective is to find an s–t path p that minimizes $c_d(p) = \sum_{e \in p} c_d(e)$, and obeys the QoS-bounds, i.e., $c_i(p) = \sum_{e \in p} c_i(e) \leq b_i$, $\forall 1 \leq i \leq d-1$. MCOP is NP-hard, even when $d = 2$ in which case it is known as the restricted shortest path problem and admits a FPTAS [9, 19, 22]. In MCP, the objective is to find an s–t path p that simply obeys a d-vector $\mathbf{b}$ of QoS-bounds, i.e., $c_i(p) = \sum_{e \in p} c_i(e) \leq b_i$, $\forall 1 \leq i \leq d$. MCP is NP-complete. For both problems, the case of $d = 2$ objectives has been extensively studied and there are also very efficient FPTAS known [9, 22]. For $d > 2$, apart from the generic approach in [24], only heuristic methods and pseudopolynomial time algorithms are known [23]. We are able to show how (quality guaranteed) approximate schemes to both MCOP and MCP can be constructed that have the same complexity with MOSP, thus improving upon all previous approaches for any $d > 2$.

Non-Additive Shortest Paths. In this problem (NASP), we are given a digraph whose edges are associated with d-dimensional cost vectors and the task is to find a path that minimizes a certain d-attribute non-linear utility function. NASP is a fundamental problem in several domains [13, 14, 26], the most prominent of which is finding traffic equilibria [13, 26]. In such applications, a utility function for a path is defined that typically translates edge attributes (e.g., travel time, cost, distance, tolls, etc) to a common utility cost measure (e.g., money). Experience shows that users of traffic networks value certain attributes (e.g., time) non-linearly [20]: small amounts have relatively low value, while large amounts are very valuable. Also, the vast majority of toll road or transit systems have a non-additive (non-linear) toll/fare structure [13]. Consequently, the most interesting theoretical models for traffic equilibria [13, 26] involve minimizing a monotonic non-linear utility function. NASP is an NP-hard problem. By virtue of the results in [1, 27], there exists a FPTAS for $d = 2$ and quasi-polynomial utility function [1], and a FPTAS for any $d \geq 2$ and polynomial utility function [27].

In Sect. 5, we show how our FPTAS for MOSP, along with our generic framework for dealing with non-linear objectives, can be used to obtain a FPTAS for NASP for *any* $d > 1$ and a larger than quasi-polynomially bounded family of utility functions. Actually, our approach allows the NASP utility function to grow exponentially on a single attribute. This does not contradict the inapproximability result in [1]; it merely makes the gap between NASP approximability and inapproximability even tighter. Our results improve considerably upon those in [1, 27] w.r.t. time (dependence on $1/\varepsilon$), number of objectives, and class of utility functions.

QoS-Aware Multicommodity Flow. In the classical weighted multicommodity flow (MCF) problem, demands and commodity values (that multiply the flow in the objective function) are considered fixed. In several realistic network design scenarios, however, encountered in communication and transportation networks [2, 5, 6, 12, 31], this may not be the case, since demands and values are usually *elastic* to certain parameters—typically to the QoS provided by the network. The goal is to compute the maximum weighted MCF subject to the QoS-elastic demands and values. We call this generalized version of the weighted MCF problem as *QoS-aware MCF*. Consider, for instance, network operators in a public transportation network who wish to route various commodities (customers with common origin-destination pairs) to meet certain demands. Studies have shown [25, 31] that customers switch to other transport services when QoS drops, and more customers use a service when its quality is improved. Upgrading or downgrading of QoS has typically an impact on the pricing policy (e.g., the value charged for a worse in QoS route is reduced to minimize the loss of customers). A similar situation is encountered with networking (e.g., multimedia) applications over the Internet or with information dissemination over various communication networks [6]. In such a setting, a "server" (owned by some service provider) sends information to "clients", who retrieve answers to queries they have posed regarding various types of information (or service). Common queries are typically grouped together. Answering a query incurs a cost and a data acquisition time that depends on the communication capacity. When a "client" is provided with a non-optimal service (e.g., long data acquisition time due to capacity restrictions), s/he will most likely switch to another provider. On the other hand, the provider may reduce the cost of such a service in order to minimize the loss. Consequently, network operators or service providers are confronted with the following design issues: which is the maximum profit obtained with the current capacity policy that incurs certain QoS-elastic demands and values? How much will this profit improve if the capacity is increased? Which is the necessary capacity to achieve a profit above a certain threshold? A fast algorithm for the QoS-aware MCF problem would allow network designers to address effectively such issues by identifying capacity bottlenecks and proceed accordingly.

A related problem, called *max-flow with QoS guarantee* (QoS max-flow), has been considered in [3]. The problem asks for computing the maximum (unweighted) MCF routed along a set of paths whose cost does not exceed a specific bound, and has been shown to be NP-hard in [3]. In the same paper [3], a pseudopolynomial time approximation scheme for QoS max-flow is given. It can be easily seen that QoS max-flow is a special case of the QoS-aware MCF problem (Sect. 5).

We show (Sect. 5) that the QoS-aware MCF problem can be formulated (in a non-straightfor-ward manner) as a fractional packing LP, and provide a FPTAS for its approximate solution. Our algorithm builds upon the Garg and Könemann Lagrangian relaxation method for fractional packing LPs [15], combined with the phases technique by Fleischer [11]. A crucial step of the method is to construct an oracle that identifies the most violated constraint of the dual LP. While in the classical weighted MCF problem the construction of the oracle is harmless (reduces to the standard, single objective shortest path problem), this is *not* the case with the QoS-aware MCF problem. The construction turns out to be highly non-trivial, since it reduces to a multiobjective (actually non-additive) shortest path problem due to the QoS-elastic demands and values. Using our FPTAS, we can construct the required oracle and hence provide a FPTAS for the QoS-aware MCF problem. Our approach gives also a FPTAS for the QoS max-flow problem, thus improving upon the result in [3].

The rest of the paper is organized as follows. In Sect. 2, we give some fundamental definitions that will be used throughout the paper. In Sect. 3, we present our algorithm for the single-source multiobjective shortest path problem. The generic methods for constructing FPTAS for multiobjective optimization problems with non-linear objectives are given in Sect. 4. The applications of our results to multiple constrained (optimal) paths, non-additive shortest paths, and QoS-aware multicommodity flows are given in Sect. 5. We conclude in Sect. 6. Preliminary parts of this work appeared in [28, 29].

2 Preliminaries

Recall that an instance of a multiobjective optimization problem is associated with a set of feasible solutions Q and a d-vector function $\mathbf{f} = [f_1, \dots, f_d]^T$ associating each feasible solution $q \in Q$ with a d-vector $\mathbf{f}(q)$. The *Pareto set or curve* $\mathcal{P}$ of Q is defined as the set of all undominated elements of Q. Given a vector of approximation ratios $\boldsymbol{\rho} = [\rho_1, \dots, \rho_d]^T$ ($\rho_i \geq 1$, $1 \leq i \leq d$), a solution $p \in Q$ $\boldsymbol{\rho}$*-covers* a solution $q \in Q$ iff it is as good in each objective i by at least a factor ρ_i, i.e., $f_i(p) \leq \rho_i \cdot f_i(q)$, $1 \leq i \leq d$. A set $\Pi \subseteq Q$ is a $\boldsymbol{\rho}$*-cover* of Q iff for all $q \in Q$, there exists $p \in \Pi$ such that p $\boldsymbol{\rho}$-covers q (note that a $\boldsymbol{\rho}$-cover may contain dominated solutions). A $\boldsymbol{\rho}$-cover is also called $\boldsymbol{\rho}$*-Pareto set*. If all entries of $\boldsymbol{\rho}$ are equal to ρ, we also use the terms ρ-cover and ρ-Pareto set.

A *fully polynomial time approximation scheme* (FPTAS) for computing the Pareto set of an instance of a multiobjective optimization problem is a family of algorithms that, for any fixed constant $\varepsilon > 0$, contains an algorithm that always outputs a $(1+\varepsilon)$-Pareto set and runs in time polynomial in the size of the input and $\frac{1}{\varepsilon}$. W.l.o.g. we make the customary assumption that $\varepsilon \leq 1$, yielding $\ln(1+\varepsilon) = \Theta(\varepsilon)$, which will be used throughout the paper.

If $\mathbf{a} = [a_1, a_2, \dots, a_d]^T$ is a d-dimensional vector and λ a scalar, then we denote by $\mathbf{a}^\lambda = [a_1^\lambda, a_2^\lambda, \dots, a_d^\lambda]^T$. A vector with all its elements equal to zero is denoted by $\mathbf{0}$.

3 Single-Source Multiobjective Shortest Paths

In the multiobjective shortest path problem, we are given a digraph $G = (V, E)$ and a d-dimensional function vector $\mathbf{c} : E \to [\mathbb{R}^+]^d$ associating each edge e with a cost vector $\mathbf{c}(e)$. We extend the cost function vector to handle paths by extending the domain to the power set of E, thus considering the function $\mathbf{c}: 2^E \to [\mathbb{R}^+]^d$, where the cost vector of a path p is the sum of the cost vectors of its edges, i.e., $\mathbf{c}(p) = \sum_{e \in p} \mathbf{c}(e)$. Given two nodes v and w, let $P(v, w)$ denote the set of all v–w paths in G. In the *multiobjective shortest path* problem, we are asked to compute the Pareto set of $P(v, w)$ w.r.t. $\mathbf{c}$. In the *single-source multiobjective shortest path* (SSMOSP) problem, we are given a node s and the task is to compute the Pareto sets of $P(s, v)$ w.r.t. $\mathbf{c}$, $\forall v \in V$.

Given a vector $\boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2, \dots, \varepsilon_{d-1}]^T$ of error parameters ($\varepsilon_i > 0$, $1 \le i \le d-1$) and a source node s, we present below an algorithm that computes, for each node v, a $\boldsymbol{\rho}$-cover of $P(s, v)$, where $\boldsymbol{\rho} = [1+\varepsilon_1, 1+\varepsilon_2, \dots, 1+\varepsilon_{d-1}, 1]^T$. Note that we can be *exact* in one dimension (here w.l.o.g. the d-th one), without any impact on the running time. In the following, let $c_i^{\min} \equiv \min_{e \in E} c_i(e)$, $c_i^{\max} \equiv \max_{e \in E} c_i(e)$, and $C_i = \frac{c_i^{\max}}{c_i^{\min}}$, for all $1 \le i \le d$. Let also $P^i(v, w)$ denote the set of all v–w paths in G with no more than i edges; clearly, $P^{n-1}(v, w) \equiv P(v, w)$.

3.1 The SSMOSP Algorithm

Our algorithm resembles the classical (label correcting) Bellman-Ford method. Previous attempts to straightforwardly apply such an approach [4, 7, 8] had a very poor (exponential) performance, since all undominated solutions (exponentially large sets of labels) have to be maintained. The key idea of our method is that we can implement the label sets as arrays of polynomial size by relaxing the requirements for strict Pareto optimality to that of $\boldsymbol{\rho}$-covering.

We represent a path $p = (e_1, e_2, \dots, e_{k-1}, e_k)$ by a label that is a tuple ($\mathbf{c}(p)$, pred(p), lastedge(p)), where $\mathbf{c}(p) = \sum_{e \in p} \mathbf{c}(e)$ is the d-dimensional cost vector of the path, pred$(p) = \vec{q}$ is a pointer to the label of the subpath $q = (e_1, e_2, \dots, e_{k-1})$ of p, and lastedge$(p) = e_k$ points to the last edge of p. An empty label is represented by ($\mathbf{0}$, *null*, *null*), while a single edge path has a *null* pred pointer. This representation allows us to retrieve the entire path, without implicitly storing its edges, by following the pred pointers. Let $\mathbf{r} = [r_1, \dots, r_{d-1}, 1]^T$ be a vector of approximation ratios. The algorithm proceeds in rounds. In each round i and for each node v the algorithm computes a set of labels Π_v^i, which is an $\mathbf{r}^i$-cover of $P^i(s, v)$. We implement these sets of labels using $(d-1)$-dimensional arrays $\Pi_v^i[0..\lfloor \log_{r_1}(nC_1) \rfloor, 0..\lfloor \log_{r_2}(nC_2) \rfloor, \dots, 0..\lfloor \log_{r_{d-1}}(nC_{d-1}) \rfloor]$, and index these arrays using $(d-1)$-vectors. This is done by defining a function $\mathbf{pos}: 2^E \to [\mathbb{N}_0]^{d-1}$. For a path p, $\mathbf{pos}(p) = [\lfloor \log_{r_1} \frac{c_1(p)}{c_1^{\min}} \rfloor, \lfloor \log_{r_2} \frac{c_2(p)}{c_2^{\min}} \rfloor, \dots, \lfloor \log_{r_{d-1}} \frac{c_{d-1}(p)}{c_{d-1}^{\min}} \rfloor]^T$ gives us the position in Π_v^i corresponding to p. The definition of **pos** along with the fact that for any path p we have $c_i(p) \le (n-1)c_i^{\max}$, $\forall 1 \le i \le d$, justifies the size of the arrays.

Initially, $\Pi_v^0 = \emptyset$, for all $v \in V - \{s\}$, and Π_s^0 contains only the trivial empty path. For each round $i \ge 1$ and for each node v the algorithm computes Π_v^i as follows (see also Fig. 1). Initially, we set Π_v^i equal to Π_v^{i-1}. We then examine the incoming edges of v, one by one, and perform an *Extend-&-Merge* operation for each edge examined. An *Extend-&-Merge* operation takes as input an edge $e = (u, v)$ and the sets Π_u^{i-1} and Π_v^i. It extends all labels $p \in \Pi_u^{i-1}$ by e, and merges the resulting set

Fig. 1 The SSMOSP algorithm

```
SSMOSP($G, s, \mathbf{c}, \mathbf{r}$){
forall $v \in V$ {$\Pi_v^0 = \emptyset$;}
$\Pi_s^0[\mathbf{0}] = \{(\mathbf{0}, null, null)\}$;
for $i = 1$ to $n - 1${
    forall $v \in V${
        $\Pi_v^i = \Pi_v^{i-1}$;
        forall $e = (u, v) \in E$
            $\Pi_v^i$ = Extend-&-Merge($\Pi_v^i, \Pi_u^{i-1}, e$);
    }
}
}
function Extend-&-Merge($R, Q, e$) {
forall $p \in Q$ {
    $q = (\mathbf{c}(p) + \mathbf{c}(e), \vec{p}, e)$;
    $\mathbf{pos}(q) = [\lfloor \log_{r_1} \frac{c_1(q)}{c_1^{\min}} \rfloor, \ldots, \lfloor \log_{r_{d-1}} \frac{c_{d-1}(q)}{c_{d-1}^{\min}} \rfloor]^T$;
    if $R[\mathbf{pos}(q)] = null$ or $c_d(R[\mathbf{pos}(q)]) > c_d(q)$ {
        $R[\mathbf{pos}(q)] = q$;
    }
}
return $R$;
}
```

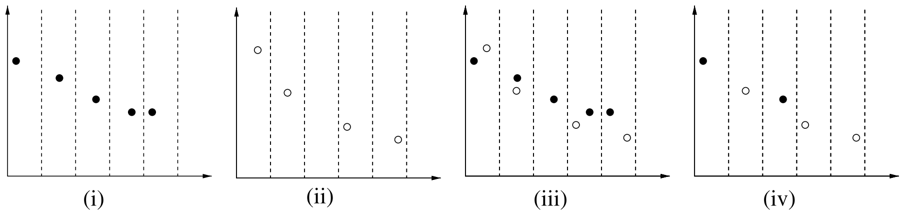

Fig. 2 *Extend-&-Merge* operation for $d = 2$. Horizontal axis $c_1(\cdot)$ and vertical axis $c_2(\cdot)$

of s–v paths with Π_v^i. Since each extension results in a new label (path) $q = (\mathbf{c}(p) + \mathbf{c}(e), \vec{p}, e)$ whose $\mathbf{pos}(q)$ leads to an array position which may not be empty, the algorithm maintains in each array position the (at most one) path that covers all other paths with the same $\mathbf{pos}(\cdot)$ value, which turns out to be the path with the smallest c_d cost. This keeps the size of the sets polynomially bounded. In particular, q is inserted in the position $\mathbf{pos}(q) = [\lfloor \log_{r_1} \frac{c_1(q)}{c_1^{\min}} \rfloor, \lfloor \log_{r_2} \frac{c_2(q)}{c_2^{\min}} \rfloor, \ldots, \lfloor \log_{r_{d-1}} \frac{c_{d-1}(q)}{c_{d-1}^{\min}} \rfloor]^T$ of Π_v^i, unless this position is already filled in with a label q' for which $c_d(q') \leq c_d(q)$.

To illustrate the *Extend-&-Merge* operation, we give, in Fig. 2, an example for $d = 2$.

We map the paths as points in the two dimensional plane with vertical axis $y = c_2(\cdot)$ and horizontal axis $x = c_1(\cdot)$. The horizontal axis is divided into intervals such that the left and the right endpoint of each interval differ by a multiplicative factor of

$(1+\varepsilon)$. Figure 2(i) shows the paths in R, and Fig. 2(ii) shows the paths in Q that are to be extended by e. Figure 2(iii) shows the union of R and Q (before the merge), while Fig. 2(iv) shows the resulting set after the merge. In each interval, at most one path is kept which is the path with the smallest c_2 cost.

The following lemma establishes the correctness of our approach.

Lemma 1 *For all $v \in V$ and for all $i \geq 0$, after the i-th round Π_v^i $\mathbf{r}^i$-covers $P^i(s,v)$.*

Proof It suffices to prove that for all $p \in P^i(s,v)$, there exists $q \in \Pi_v^i$ such that $c_\ell(q) \leq r_\ell^i c_\ell(p)$, $\forall 1 \leq \ell \leq d$. We prove this by induction.

For the basis of the induction ($i = 1$) consider a single edge path $p \equiv (e) \in P^1(s,v)$. At each round all incoming edges of v are examined and an *Extend-&-Merge* operation is executed for each edge. After the first round and due to the **if** condition of the *Extend-&-Merge* operation, position $\mathbf{pos}(p)$ of Π_v^1 contains a path q for which: (i) $\mathbf{pos}(q) = \mathbf{pos}(p)$; and (ii) $c_d(q) \leq c_d(p)$. From (i) it is clear that for all $1 \leq \ell \leq d-1$, we have $\lfloor \log_{r_\ell} \frac{c_\ell(q)}{c_\ell^{\min}} \rfloor = \lfloor \log_{r_\ell} \frac{c_\ell(p)}{c_\ell^{\min}} \rfloor$, and therefore $\log_{r_\ell} \frac{c_\ell(q)}{c_\ell^{\min}} - 1 \leq \log_{r_\ell} \frac{c_\ell(p)}{c_\ell^{\min}}$. This, along with (ii) and the fact that $r_d = 1$, implies that $c_\ell(q) \leq r_\ell c_\ell(p)$, $\forall 1 \leq \ell \leq d$.

For the induction step consider a path $p \equiv (e_1, e_2, \ldots, e_k = (u,v)) \in P^i(s,v)$, for some $k \leq i$. The subpath $p' \equiv (e_1, e_2, \ldots, e_{k-1})$ of p has at most $i-1$ edges and applying the induction hypothesis we get that there exists a path $q' \in \Pi_u^{i-1}$ such that $c_\ell(q') \leq r_\ell^{i-1} c_\ell(p')$, $1 \leq \ell \leq d$. Let now $\bar{q}$ be the concatenation of q' with edge e_k. Then, we have:

$$c_\ell(\bar{q}) \leq r_\ell^{i-1} c_\ell(p), \quad 1 \leq \ell \leq d. \tag{1}$$

It is clear by our algorithm that during the *Extend-&-Merge* operation for edge e_k in the i-th round $\bar{q}$ was examined. Moreover, at the end of the i-th round and due to the **if** condition of the *Extend-&-Merge* operation, position $\mathbf{pos}(\bar{q})$ of Π_v^i contains a path q for which: (iii) $\mathbf{pos}(q) = \mathbf{pos}(\bar{q})$; and (iv) $c_d(q) \leq c_d(\bar{q})$. From (iii) it is clear that $\lfloor \log_{r_\ell} c_\ell(q) \rfloor = \lfloor \log_{r_\ell} c_\ell(\bar{q}) \rfloor$, $\forall\ 1 \leq \ell \leq d-1$, and therefore $\log_{r_\ell} c_\ell(q) - 1 \leq \log_{r_\ell} c_\ell(\bar{q})$, $\forall\ 1 \leq \ell \leq d-1$, which implies that

$$c_\ell(q) \leq r_\ell c_\ell(\bar{q}), \quad 1 \leq \ell \leq d-1. \tag{2}$$

Since $r_d = 1$, combining now (iv) and (2) with (1), we get that $c_\ell(q) \leq r_\ell^i c_\ell(p)$, $\forall\ 1 \leq \ell \leq d$. □

We now turn to the time complexity.

Lemma 2 *Algorithm SSMOSP computes, for all $v \in V$, an $\mathbf{r}^{n-1}$-cover of $P(s,v)$ in total time $O(nm \prod_{j=1}^{d-1}(\lfloor \log_{r_j}(nC_j) \rfloor + 1))$.*

Proof From Lemma 1, it is clear that, for any $v \in V$, Π_v^{n-1} is an $\mathbf{r}^{n-1}$-cover of $P^{n-1}(s,v) \equiv P(s,v)$, since any path has at most $n-1$ edges. The algorithm terminates after $n-1$ rounds. In each round it examines all of the m edges and performs an *Extend-&-Merge* operation. The time of this operation is proportional to the size of the arrays used, which equals $\prod_{j=1}^{d-1}(\lfloor \log_{r_j}(nC_j)\rfloor + 1)$ and therefore the total time complexity is $O(nm \prod_{j=1}^{d-1}(\lfloor \log_{r_j}(nC_j)\rfloor + 1))$. □

Applying Lemma 2 with $\mathbf{r} = [(1+\varepsilon_1)^{\frac{1}{n-1}}, (1+\varepsilon_2)^{\frac{1}{n-1}}, \ldots, (1+\varepsilon_{d-1})^{\frac{1}{n-1}}, 1]^T$, and taking into account that $\ln(1+\delta) = \Theta(\delta)$ for small δ, yields the main result of this section.

Theorem 1 *Given a vector $\boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{d-1}]^T$ of error parameters and a source node s, there exists an algorithm that computes, for all $v \in V$, a $\boldsymbol{\rho}$-cover of $P(s,v)$ (set of all s–v paths), where $\boldsymbol{\rho} = [1+\varepsilon_1, 1+\varepsilon_2, \ldots, 1+\varepsilon_{d-1}, 1]^T$, in total time $O(n^d m \prod_{j=1}^{d-1}(\frac{1}{\varepsilon_j}\log(nC_j)))$.*

Let $C^{\max} = \max_{1\le j\le d-1} C_j$. In the special case, where $\varepsilon_i = \varepsilon, \forall 1 \le i \le d-1$, we have the following result.

Corollary 1 *For any error parameter $\varepsilon > 0$, there exists a FPTAS for the single-source multiobjective shortest path problem with d objectives on a digraph G that computes $(1+\varepsilon)$-Pareto sets (one for each node of G) in total time $O(nm(\frac{n\log(nC^{\max})}{\varepsilon})^{d-1})$.*

3.2 Extensions

Further improvements can be obtained in the case of DAGs by exploiting the topological ordering of such graphs. In particular for each node v we maintain a set Π_v as in the general algorithm. Initially $\Pi_v = \emptyset, \forall v \in V - \{s\}$ and $\Pi_v[\mathbf{0}] = \{(\mathbf{0}, \mathit{null}, \mathit{null})\}$. The algorithm visits all nodes w.r.t. the topological ordering and for each visited node, it performs an *Extend-&-Merge* operation to all its outgoing edges. It is easy to see that in this case the time reduces by a factor of n in all the aforementioned results. For instance, the time of Corollary 1 becomes $O(m(\frac{n\log(nC^{\max})}{\varepsilon})^{d-1})$.

It is also quite easy to see that the algorithm actually computes an approximate Pareto curve w.r.t. the additional objective of minimizing the number of hops (number of edges in the path). Indeed, Lemma 1 implies that for any $v \in V$ the union of all Π_v^i, over all i rounds, constitutes an approximate Pareto curve also w.r.t. that additional objective.

4 Non-Linear Objectives

In this section, we present two generic methods to construct a FPTAS for the normalized version of any multiobjective optimization problem with a non-linear utility function, as well as a FPTAS for any multiobjective optimization problem with non-linear objectives, for a quite general family of non-linear functions. The only precondition is the existence of a FPTAS for a much simpler version of the problems.

Let $\mathcal{M}$ be (an instance of) a multiobjective optimization problem with set of feasible solutions Q and vector of objective functions $\mathbf{c} = [c_1, \ldots, c_d]^T$, associating each feasible solution $q \in Q$ with a d-vector of attributes $\mathbf{c}(q)$; i.e., the i-th objective is the identity function of the i-th attribute.

Let $\mathcal{N}$ be the normalized version of $\mathcal{M}$ w.r.t. a non-decreasing, non-linear utility function $\mathcal{U} : [\mathbb{R}^+]^d \to \mathbb{R}$; i.e., the objective of $\mathcal{N}$ is $\min_{q\in Q} \mathcal{U}(\mathbf{c}(q))$. We will show that a FPTAS for $\mathcal{M}$ can provide a FPTAS for $\mathcal{N}$. To obtain such a FPTAS, we consider a quite general family of non-linear functions $\mathcal{U}(\mathbf{x})$.

A multiattribute function $\mathcal{U}(\mathbf{x})$ is called *quasi-polynomially bounded* (see e.g., [1]) if there exist some constants γ and δ such that

$$\frac{\frac{\partial \mathcal{U}}{\partial x_i}(\mathbf{x})}{\mathcal{U}(\mathbf{x})} \le \gamma \frac{1}{x_i} \prod_{k=1}^{d} \ln^\delta x_k, \quad 1 \le i \le d.$$

For instance, the function $\mathcal{U}([x_1, x_2]^T) = x_1^{\text{polylog}(x_1)} + x_2^{\text{polylog}(x_2)}$ is quasi-polynomially bounded, while the function $\mathcal{U}([x_1, x_2]^T) = 2^{x_1^\mu} + 2^{x_2^\mu}$, for some $\mu > 0$, is not. Note also that this class includes all non-decreasing polynomials.

Let $\mathcal{C}_i = \max_{q\in Q} c_i(q)$ be the maximum cost in the i-th dimension, and let $\log \mathcal{C}_i$ be polynomial to the input size (as indeed is the case for MOSP and other problems, like the multiobjective versions of spanning tree, perfect matching, knapsack, etc.). We can prove the following.

Theorem 2 *Let the objective function $\mathcal{U}$ of $\mathcal{N}$ be quasi-polynomially bounded. If there exists a FPTAS for $\mathcal{M}$ with time complexity $T(1/\varepsilon, m')$, then there exists a FPTAS for $\mathcal{N}$ with complexity $T(\Lambda/\varepsilon, m')$, where m' is the input size of $\mathcal{M}$ and $\Lambda = \gamma d \prod_{i=1}^{d} \ln^\delta \mathcal{C}_i$.*

Proof We construct a $(1 + \varepsilon')$-Pareto set Π for $\mathcal{M}$, where ε' will be chosen later. Pick $q = \text{argmin}_{p\in\Pi}(\mathcal{U}(\mathbf{c}(p)))$. Let p^* denote the optimal solution with cost vector $\boldsymbol{c}^* = \boldsymbol{c}(p^*)$. By the definition of Π, we know that there exists some $p' \in \Pi$ such that $c_i(p') \le \min\{(1 + \varepsilon')c_i^*, \mathcal{C}_i\}$. By the choice of q we have that $\mathcal{U}(\mathbf{c}(q)) \le \mathcal{U}(\mathbf{c}(p'))$, thus it suffices to bound $\frac{\mathcal{U}(\boldsymbol{c}(p'))}{\mathcal{U}(\boldsymbol{c}(p^*))}$.

Let $\boldsymbol{c}'$ be the vector whose elements are given by $c_i' = \min\{(1 + \varepsilon')c_i^*, \mathcal{C}_i\}$, $\forall 1 \le i \le d$. Since $\mathcal{U}(\cdot)$ is non-decreasing, $\frac{\mathcal{U}(\boldsymbol{c}(p'))}{\mathcal{U}(\boldsymbol{c}(p^*))} \le \frac{\mathcal{U}(\boldsymbol{c}')}{\mathcal{U}(\boldsymbol{c}^*)} = \exp[\ln \mathcal{U}(\boldsymbol{c}') - \ln \mathcal{U}(\boldsymbol{c}^*)]$. We write the exponent as a telescopic sum $\ln \mathcal{U}(\boldsymbol{c}') - \ln \mathcal{U}(\boldsymbol{c}^*) = \sum_{k=1}^{d}[F_k(c_k') - F_k(c_k^*)]$, where $F_k(x) = \ln \mathcal{U}([c_1', \ldots, c_{k-1}', x, c_{k+1}^*, \ldots, c_d^*]^T)$. On each term k of the sum, we apply the well-known Mean Value Theorem[2] for $F_k(x)$ on the interval (c_k^*, c_k'). Hence, $\forall 1 \le k \le d$, there exists some ζ_k with $c_k^* < \zeta_k < c_k'$ such that

$$F_k(c_k') - F_k(c_k^*) = F_k'(\zeta_k)(c_k' - c_k^*) \le \frac{\frac{\partial \mathcal{U}}{\partial x_k}(\boldsymbol{c}^{[k]})}{\mathcal{U}(\boldsymbol{c}^{[k]})} \varepsilon' c_k^*,$$

[2]**Mean Value Theorem:** Let $f(x)$ be differentiable on (a, b) and continuous on $[a, b]$. Then, there is at least one point $c \in (a, b)$ such that $f'(c) = (f(b) - f(a))/(b - a)$.

where $\boldsymbol{c}^{[k]}$ are vectors with

$$c_i^{[k]} = \begin{cases} c_i' & \text{if } 1 \leq i < k, \\ \zeta_k & \text{if } i = k, \\ c_i^* & \text{if } k < i \leq d. \end{cases}$$

Consequently,

$$\frac{\mathcal{U}(\boldsymbol{c}')}{\mathcal{U}(\boldsymbol{c}^*)} \leq \exp\left[\varepsilon' \sum_{k=1}^{d} \left[\frac{\frac{\partial \mathcal{U}}{\partial x_k}(\boldsymbol{c}^{[k]})}{\mathcal{U}(\boldsymbol{c}^{[k]})} c_k^*\right]\right].$$

Observe now that the term

$$\sum_{k=1}^{d} \left[\frac{\frac{\partial \mathcal{U}}{\partial x_k}(\boldsymbol{c}^{[k]})}{\mathcal{U}(\boldsymbol{c}^{[k]})} c_k^*\right]$$

is bounded by $\Lambda = \gamma d \prod_{i=1}^{d} \ln^{\delta} \mathcal{C}_i$. Hence, choosing $\varepsilon' = \frac{\ln(1+\varepsilon)}{\Lambda}$, yields a $1+\varepsilon$ approximation in time $T(\Lambda/\varepsilon, m')$. □

The above result improves upon that of [1] both w.r.t. d (number of objectives) and time; the time in [1] ($d = 2$) is $T(\Lambda'/\varepsilon^2, m')$, where $\Lambda' = \gamma 2^{\delta+4} \prod_{i=1}^{2} \ln^{\delta+1} \mathcal{C}_i$.

Now, let $\mathcal{M}'$ be a multiobjective optimization problem, defined on the same with $\mathcal{M}$ set of feasible solutions Q, but having a vector of objective functions $\mathbf{U} = [U_1, \ldots, U_h]^T$ associating each $q \in Q$ with an h-vector $\mathbf{U}(q)$. These objective functions are defined as $U_i(q) = \mathcal{U}_i(\mathbf{c}(q))$, $1 \leq i \leq h$, where $\mathcal{U}_i : [\mathbb{R}^+]^d \to \mathbb{R}$ are non-linear, non-decreasing, quasi-polynomially bounded functions. We can show the following.

Theorem 3 *Let the objective functions of $\mathcal{M}'$ be quasi-polynomially bounded. If there exists a FPTAS for $\mathcal{M}$ with time complexity $T(1/\varepsilon, m')$, then there exists a FPTAS for $\mathcal{M}'$ with complexity $T(\Lambda/\varepsilon, m')$, where m' is the input size of $\mathcal{M}$ and $\Lambda = \gamma d \prod_{i=1}^{d} \ln^{\delta} \mathcal{C}_i$.*

Proof We construct a $(1+\varepsilon')$-Pareto curve Π for $\mathcal{M}$, where $\varepsilon' = \frac{\ln(1+\varepsilon)}{\Lambda}$, and show that Π constitutes a $(1+\varepsilon)$-Pareto curve for $\mathcal{M}'$. To see this, it suffices to prove that for all $q \in Q$, there exists $p \in \Pi$ such that $U_i(p) \leq (1+\varepsilon)U_i(q)$, $\forall 1 \leq i \leq h$. By the construction of Π, we have that for all $q \in Q$ there exists $p \in \Pi$ such that $c_k(p) \leq \min\{(1+\varepsilon')c_k(q), C_k\}$, $\forall 1 \leq k \leq d$. To bound $\frac{U_i(p)}{U_i(q)} = \frac{\mathcal{U}_i(\boldsymbol{c}(p))}{\mathcal{U}_i(\boldsymbol{c}(q))}$, $\forall 1 \leq i \leq h$, we work similarly to Theorem 2 with $\boldsymbol{c}(p)$ and $\boldsymbol{c}(q)$ in place of $\boldsymbol{c}'$ and $\boldsymbol{c}^*$, respectively. □

It is interesting to observe that the time of the above construction is independent of the number h of objective functions.

5 Applications

We show how the results of Sects. 3 and 4 can be used to provide efficient approximate solutions to the MCOP, MCP, NASP, and QoS-aware MCF problems mentioned in the Introduction.

5.1 Multiple Constrained (Optimal) Paths

Let $\boldsymbol{\rho} = [1+\varepsilon_1, 1+\varepsilon_2, \dots, 1+\varepsilon_{d-1}, 1]^T$ and let Π be a $\boldsymbol{\rho}$-cover Π of $P(s,t)$, constructed using the SSMOSP algorithm as implied by Theorem 1. For MCOP, choose $p' = \text{argmin}_{p\in\Pi}\{c_d(p); c_i(p) \leq (1+\varepsilon_i)b_i, \ \forall 1 \leq i \leq d-1\}$. This provides a so-called *acceptable* solution in the sense of [17] by slightly relaxing the QoS-bounds; that is, the path p' is at least as good as the MCOP-optimum and is nearly feasible, violating each QoS-bound b_i, $1 \leq i \leq d-1$, by at most a $1+\varepsilon_i$ factor. For MCP, choose a path $p' \in \Pi$ that obeys the QoS-bounds, or answer that there is no path p in $P(s,t)$ for which $c_i(p) \leq b_i/(1+\varepsilon_i)$, $\forall 1 \leq i < d$. In the latter case, if a feasible solution for MCP exists, then (by the definition of Π) we can find a solution in Π that is nearly feasible (i.e., it violates each QoS-bound b_i, $1 \leq i \leq d-1$, by at most a $1+\varepsilon_i$ factor). By Theorem 1, the required time for both cases is $O(n^d m \prod_{j=1}^{d-1}(\frac{1}{\varepsilon_j}\log(nC_j))$, which can be reduced to $O(n^d m \prod_{j=1}^{d-1}(\frac{1}{\varepsilon_j}\log(\min\{nC_j, b_j/c_j^{\min}\}))$ by observing that it is safe to discard any path p for which $c_j(p) > (1+\varepsilon_j)b_j$ for some $1 \leq j \leq d-1$ (thus reducing the size of the Π_v^i arrays).

5.2 Non-Additive Shortest Paths

In this problem (NASP) we are given a digraph $G = (V, E)$ and a d-dimensional function vector $\mathbf{c} : E \to [\mathbb{R}^+]^d$ associating each edge e with a vector of attributes $\mathbf{c}(e)$ and a path p with a vector of attributes $\mathbf{c}(p) = \sum_{e\in p}\mathbf{c}(e)$. We are also given a d-attribute non-decreasing and *non-linear* utility function $\mathcal{U} : [\mathbb{R}^+]^d \to \mathbb{R}$. The objective is to find a path p^*, from a specific source node s to a destination t, that minimizes the objective function, i.e., $p^* = \text{argmin}_{p\in P(s,t)}\mathcal{U}(\mathbf{c}(p))$. (It is easy to see that in the case where $\mathcal{U}$ is linear, NASP reduces to the classical single-objective shortest path problem.) For the general case of non-linear $\mathcal{U}$, it is not difficult to see that NASP is NP-hard.

Theorem 2 suggests that our FPTAS for MOSP yields an (improved w.r.t. [1, 27]) FPTAS for NASP for the case of quasi-polynomially bounded functions. We show that we can do even better by taking advantage of the fact that our FPTAS for MOSP is *exact* in one dimension (w.l.o.g. the d-th). This allows us to provide a FPTAS for an even more general (than quasi-polynomial) family of functions. Specifically, we consider d-attribute functions for which there exist some constants γ and δ such that

$$\frac{\frac{\partial\mathcal{U}}{\partial x_i}(\mathbf{x})}{\mathcal{U}(\mathbf{x})} \leq \gamma\frac{1}{x_i}\prod_{k=1}^{d}\ln^\delta x_k, \qquad 1 \leq i \leq d-1.$$

The fact that we do not require that this condition holds for the d-th attribute allows $\mathcal{U}$ to be even *exponential* on x_d; e.g., $\mathcal{U}([x_1, x_2]^T) = x_1^{\text{polylog}(x_1)} + 2^{x_2^\mu}$, for any $\mu > 0$.

Note that this does not contradict the inapproximability result in [1], which applies to functions of the form $\mathcal{U}([x_1, x_2]^T) = 2^{x_1^\mu} + 2^{x_2^\mu}$, for $\mu > 0$. Our result makes the gap between NASP approximability and inapproximability even tighter. Let $\mathcal{C}_i$ denote the maximum path cost in the i-th dimension, i.e., $\mathcal{C}_i = (n-1)\max_{e\in E} c_i(e)$.

Theorem 4 *Let $\mathcal{U}$ be a non-decreasing function for which*

$$\frac{\frac{\partial \mathcal{U}}{\partial x_i}(\mathbf{x})}{\mathcal{U}(\mathbf{x})} \le \gamma \frac{1}{x_i} \prod_{k=1}^{d} \ln^\delta x_k, \quad 1 \le i \le d-1.$$

Then, for any $\varepsilon > 0$, there exists an algorithm that computes in time $O(n^d m (\frac{\log(nC^{\max})\Lambda}{\varepsilon})^{d-1})$ a $(1+\varepsilon)$-approximation to the NASP optimum w.r.t. $\mathcal{U}(\mathbf{x})$, where $\Lambda = \gamma(d-1)\prod_{i=1}^{d} \ln^\delta \mathcal{C}_i$.

Proof Apply Theorem 1 and construct a $\boldsymbol{\rho}$-Pareto set Π of $P(s,t)$, with $\rho_i = 1 + \frac{\ln(1+\varepsilon)}{\Lambda}$, $\forall 1 \le i \le d-1$, and $\rho_d = 1$. Pick $p' = \operatorname{argmin}_{p\in\Pi}(\mathcal{U}(\mathbf{c}(p)))$. The rest of the proof follows similarly to that of Theorem 2, taking into account that we are exact in the d-th dimension (i.e., $\rho_d = 1$). □

5.3 QoS-Aware Multicommodity Flow

In this section, we present our FPTAS for the QoS-aware multicommodity flow (MCF) problem. We start with a formal definition of the problem and its LP formulation, proceed with a review of the GK method [15] upon which our algorithm builds, and finally give the details of our FPTAS. We close by presenting extensions of our results to constrained versions of the QoS-aware MCF problem.

5.3.1 Problem Definition and LP Formulation

We are given a digraph $G = (V, E)$, along with a capacity function $u : E \to \mathbb{R}_0^+$ on its edges. We are also given a set of k commodities. A commodity i, $1 \le i \le k$, is a tuple $(s_i, t_i, d_i, wt_i(\cdot), \mathrm{f}_i(\cdot), \mathrm{v}_i(\cdot))$, whose attributes are defined as follows. Attributes $s_i \in V$ and $t_i \in V$ are the source and the target nodes, respectively, while $d_i \in \mathbb{R}_0^+$ is the demand of the commodity. The weight function $wt_i : E \to \mathbb{R}_0^+$ quantifies the *quality of service (QoS)* for commodity i (smaller weight means better QoS). For any s_i–t_i path p, $wt_i(p) := \sum_{e\in p} wt_i(e)$ and let $\delta_i(s_i, t_i)$ be the length of the shortest path from s_i to t_i w.r.t. the weight function $wt_i(\cdot)$. The non-decreasing function $\mathrm{f}_i : [1, \infty) \to [0, 1]$ is the *elasticity function* of i that determines the portion $\mathrm{f}_i(x)$ of the commodity's demand d_i that is lost if the provided path is x times worse than the shortest path w.r.t. $wt_i(\cdot)$; that is, if a units of d_i were supposed to be sent in case the provided path was shortest (optimal), then only $(1 - \mathrm{f}_i(x))a$ units will be shipped through the actually provided (non-optimal) path, while $\mathrm{f}_i(x)a$ units will be lost. Commodity i is also associated with a non-increasing *profit function* $\mathrm{v}_i : [1, \infty) \to \mathbb{R}_0^+$ that gives the profit $\mathrm{v}_i(x)$ from shipping one unit of flow of commodity i through a path that is x times worse than the shortest path w.r.t. $wt_i(\cdot)$. The objective is to maximize the total profit, i.e., the sum over all commodities and over all paths of the flow routed from every commodity on each path multiplied by the commodity's profit, subject to the capacity and demand constraints, and w.r.t. the QoS-elasticity of demands and profits. We call the above the *QoS-aware Multicommodity Flow* (MCF) problem.

Let $P_i = \{p : p \text{ is an } s_i\text{–}t_i \text{ path}\}$ be the set of *candidate paths* along which flow from commodity i can be sent. Consider such a particular path $p \in P_i$ and let $X_i(p) \in \mathbb{R}_0^+$ denote the flow of commodity i routed along p. The definition of the elasticity function implies that for each unit of flow of commodity i routed along p, there are $\frac{1}{1-\mathrm{f}_i(x)}$ units *consumed* from the demand of the commodity. Thus, we define a *consumption function* $\mathrm{h}_i : [1, \infty) \to [1, \infty)$ with $\mathrm{h}_i(x) = \frac{1}{1-\mathrm{f}_i(x)}$. Since f_i is non-decreasing, h_i is also non-decreasing. Accordingly, we define the *consumption* $h_i(p) \geq 1$ of a path p as the amount of demand consumed for each unit of flow routed along p:

$$h_i(p) = \mathrm{h}_i\left(\frac{wt_i(p)}{\delta_i(s_i, t_i)}\right).$$

Similarly, we define the *value* $v_i(p)$ of a path p as the profit from routing one unit of flow of commodity i through p:

$$v_i(p) = \mathrm{v}_i\left(\frac{wt_i(p)}{\delta_i(s_i, t_i)}\right).$$

Using the above definitions, the QoS-aware MCF problem can be described by the following LP

$$\max \quad \sum_{i=1}^{k} \sum_{p \in P_i} v_i(p) X_i(p) \tag{3}$$

$$\text{s.t.} \quad \sum_{i=1}^{k} \sum_{e \in p, p \in P_i} X_i(p) \leq u(e), \quad \forall e \in E, \tag{4}$$

$$\sum_{p \in P_i} X_i(p) h_i(p) \leq d_i, \quad \forall i = 1 \ldots k,$$

$$X_i(p) \geq 0, \quad \forall i = 1 \ldots k, \ \forall p \in P_i. \tag{5}$$

5.3.2 Review of the GK Method

In this section, we review the GK Lagrangian relaxation method for finding approximate solutions to fractional packing LPs. To give more intuition and better understand the method, we start with its basic version that concerns the maximum multicommodity flow problem.

Maximum Multicommodity Flow

In the maximum MCF problem, we are given a digraph $G = (V, E)$ along with a capacity function $u : E \to \mathbb{R}_0^+$ on its edges, and k pairs of terminals (s_i, t_i) with one commodity associated with each pair. The goal is to find a MCF such that the sum of the flows of all commodities is maximized. Let $P_i = \{p : p \text{ is an } s_i\text{–}t_i \text{ path}\}$ be the set of *candidate paths* along which flow from commodity i can be sent, and for a path $p \in P_i$ let $X_i(p) \in \mathbb{R}_0^+$ denote the flow of commodity i sent along p. The LP formulation of maximum MCF is:

$$\begin{aligned} \max \quad & \sum_{i=1}^{k} \sum_{p \in P_i} X_i(p) \\ \text{s.t.} \quad & \sum_{i=1}^{k} \sum_{e \in p, p \in P_i} X_i(p) \le u(e), \quad \forall e \in E, \\ & X_i(p) \ge 0, \quad \forall i = 1 \dots k, \ \forall p \in P_i. \end{aligned}$$

The dual to the above LP is an assignment of lengths $l : E \to \mathbb{R}_0^+$ to the edges such that the length of the shortest s_i–t_i path is at least 1 for all commodities i. The length of an edge e represents the cost of using an additional unit of capacity from e. The dual LP for maximum MCF is as follows:

$$\begin{aligned} \min \quad & \sum_{e \in E} l(e) u(e) \\ \text{s.t.} \quad & \sum_{e \in p} l(e) \ge 1, \quad \forall i = 1 \dots k, \ \forall p \in P_i, \\ & l(e) \ge 0, \quad \forall e \in E. \end{aligned}$$

The GK algorithm starts with length function $l(\cdot) = \delta$, for an appropriately small $\delta > 0$ (depending on n and ε), and with a primal solution $x(\cdot) = 0$. While there is an s_i–t_i path p of length less than 1, the algorithm selects such a path and increases both the primal and the dual variables. For the primal problem, the solution is increased by setting $x(p) = x(p) + c$, where $c = \min_{e \in p} u(e)$ is the bottleneck capacity of p. This solution satisfies the nonnegativity constraints, but it may violate the capacity constraints. Garg and Könemann [15] proved that once the most violated capacity constraint is determined, then the primal solution can be scaled so that it becomes feasible (by dividing all variables by an appropriate scalar). After updating $x(p)$, the dual variables are updated so that the length of an edge becomes exponential on its congestion, i.e., $l(e) = l(e)(1 + \varepsilon c/u(e))$, for all $e \in p$ (the length of the other edges do not change). Consequently, the length of the bottleneck edge is increased by a factor of $(1 + \varepsilon)$. The most violated constraint of the dual is determined by finding the shortest s_i–t_i path among all commodities.

Fleischer [11] improved upon the above algorithm by selecting violating constraints in a less costly manner, while still being able to deliver a similar approximation guarantee. Instead of computing the shortest s_i–t_i path among all commodities, it is sufficient to find a path of length at most $(1 + \varepsilon)$ times the length of the shortest path. This idea reduces the number of shortest path computations, since instead of considering all commodities i to obtain the shortest s_i–t_i path, one can cycle through the commodities, sticking with a specific commodity j until the shortest

s_j–t_j path for that commodity is larger than $(1+\varepsilon)$ times a lower bound estimate of the length of the shortest path. Let l_r denote the length function at the end of iteration r, let $a(r) = \min_{1\le i\le k} \min_{p\in P_i} l_r(p)$, and let $\overline{a}(r)$ be a lower bound on $a(r)$. Initially, $\overline{a}(0) = \delta$. The algorithm proceeds in iterations and considers each commodity one by one. As long as the shortest s_j–t_j path p for commodity j has length less than $\min\{1, (1+\varepsilon)\overline{a}(r)\}$, flow is augmented along p and $\overline{a}(r+1) = \overline{a}(r)$. When the length of p is at least $(1+\varepsilon)\overline{a}(r)$, commodity $j+1$ is considered. After all k commodities are examined, we know that $a(r) \ge (1+\varepsilon)\overline{a}(r)$, and $\overline{a}$ is updated by setting $\overline{a}(r+1) = (1+\varepsilon)\overline{a}(r)$. This idea is implemented by defining phases determined by the values of $\overline{a}$, staring with $\overline{a} = \delta$ and ending with $\overline{a} = \delta(1+\varepsilon)^z$ for some integer z such that $1 \le \delta(1+\varepsilon)^z < (1+\varepsilon)$.

Fractional Packing LP

The linear program given in Sect. 5.3.1 is a (pure) fractional packing LP, i.e., a linear program of the form $\max\{c^T x \mid Ax \le b, x \ge 0\}$, where $A_{M\times N}$, $b_{M\times 1}$ and $c_{N\times 1}$ have positive entries. By scaling we also assume that $A(i,j) \le b(i)$, $\forall i, j$. The dual of that problem is $\min\{b^T y \mid A^T y \ge c, y \ge 0\}$. In [15], Garg and Könemann present a remarkably elegant and simple FPTAS for solving fractional packing LPs. Their algorithm resembles their approach for maximum MCF. It maintains a primal and a dual solution. At each step the most violated constraint in the dual is identified and the corresponding primal and dual variables are increased. The most violated constraint is identified by using an exact oracle.

The algorithm works as follows. Let the *length* of a column j with respect to the dual variables y be $\text{length}_y(j) = \sum_i \frac{A(i,j)}{c(j)} y(i)$. Let $a(y)$ denote the length of the minimum-length column, i.e., $a(y) = \min_j \text{length}_y(j)$. Let also $D(y) = b^T y$ be the dual objective value with respect to y. Then, the dual problem is equivalent to finding an assignment y that minimizes $\frac{D(y)}{a(y)}$. The procedure is iterative. Let y_{k-1} be the dual variables and f_{k-1} be the value of the primal solution at the beginning of the k-th iteration. The initial values of the dual variables are $y_0(i) = \delta/b(i)$, where δ is a constant to be chosen later, and the primal variables are initially zero. In the k-th iteration, a call to an oracle is made that returns the minimum length column q of A, i.e., $\text{length}_{y_{k-1}}(q) = \alpha(y_{k-1})$. Let now $p = \text{argmin}_i \frac{b(i)}{A(i,q)}$ be the “minimum capacity” row. In this iteration, we increase the primal variable $x(q)$ by $\frac{b(p)}{A(p,q)}$, thus the primal objective becomes $f_k = f_{k-1} + c(q)\frac{b(p)}{A(p,q)}$. The dual variables are updated as

$$y_k(i) = y_{k-1}(i)\left(1 + \varepsilon \frac{b(p)/A(p,q)}{b(i)/A(i,q)}\right),$$

where $\varepsilon > 0$ is a constant depending on the desired approximation ratio. For brevity we denote $a(y_k)$ and $D(y_k)$ by $a(k)$ and $D(k)$, respectively. The procedure stops at the first iteration t such that $D(t) \ge 1$. The final primal solution constructed may not be feasible since some of the packing constraints may be violated. However, scaling the final value of the primal variables by $\log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}$ gives a feasible solution (see [15, 28]).

The above algorithm can be straightforwardly extended to work with an approximate oracle.[3] Simply, in the k-th iteration we call an oracle that returns a $(1+w)$-approximation of the minimum length column of A. If q is the column returned by the oracle, then we have that $\text{length}_{y_{k-1}}(q) \le (1+w)a(y_{k-1})$. By working similarly to [15] and choosing $\delta = (1+\varepsilon)((1+\varepsilon)M)^{-1/\varepsilon}$, we can easily show the following theorem (see [28] for a proof).

Theorem 5 *There is an algorithm that computes a $(1-\varepsilon)^{-2}(1+w)$-approximation to the packing LP after at most $M\lceil \log_{1+\varepsilon}\frac{1+\varepsilon}{\delta}\rceil = M\lceil \frac{1}{\varepsilon}(1+\log_{1+\varepsilon} M)\rceil$ iterations, where M is the number of rows.*

5.3.3 The FPTAS for QoS-Aware Multicommodity Flows

In this section, we describe the (non-straightforward) details of solving the QoS-aware MCF problem by building upon the GK method. We start by obtaining the dual of the LP formulation of the QoS-aware MCF problem. We introduce for each edge e a dual variable $l(e)$ that corresponds to the capacity constraint (4) on e, and for each commodity i we introduce a dual variable ϕ_i that corresponds to the demand constraint (5) on i. The dual LP becomes

$$\min \quad D = \sum_{e\in E} l(e)u(e) + \sum_{i=1}^{k} \phi_i d_i \tag{6}$$

$$\begin{aligned} \text{s.t.} \quad & l(p) + \phi_i h_i(p) \ge v_i(p), \quad \forall i = 1\dots k,\ \forall p \in P_i, \\ & l(p) \ge 0, \quad \forall i = 1\dots k,\ \forall p \in P_i, \\ & \phi_i \ge 0, \quad \forall i = 1\dots k, \end{aligned} \tag{7}$$

where $l(p) := \sum_{e\in p} l(e)$.

To apply the GK method, it must hold $u(e) \ge 1$, $\forall e \in E$, and $d_i \ge h_i(p)$, $\forall 1 \le i \le k$, $p \in P_i$. To ensure this, we scale the capacities and demands by $\min\{\min_{e\in E} u(e), \min_{1\le i\le k} \frac{d_i}{h_i^{\max}}\}$, where $h_i^{\max} = h_i(\frac{(n-1)\max_{e\in E} wt_i(e)}{\delta_i(s_i,t_i)})$ is an upper bound on the maximum possible value of $h_i(\cdot)$.

Given an assignment (l,ϕ) for the dual variables, the length of a dual constraint is defined as $\text{length}_{(l,\phi)}(i,p) = \frac{l(p)+\phi_i h_i(p)}{v_i(p)}$ and the length of the most violated constraint is denoted by $a(l,\phi) = \min_{1\le i\le k}\min_{p\in P_i} \text{length}_{(l,\phi)}(i,p)$. The algorithm maintains a dual variable $l(e)$ for each edge e, initially equal to $\frac{\delta}{u(e)}$, and a dual variable ϕ_i for each commodity i, initially equal to $\frac{\delta}{d_i}$, where $\delta = (1+\varepsilon) \times((1+\varepsilon)(m+k))^{-\frac{1}{\varepsilon}}$.

The algorithm proceeds in iterations. Initially all flows are zero. In each iteration, it makes a call to an oracle that returns a commodity i' and a path $p \in P_{i'}$ that approximately minimizes $\text{length}_{(l,\phi)}(i,q)$ over all $1 \le i \le k$ and $q \in P_i$; i.e., we have

[3] Such an extension of the GK approach to work with approximate oracles was known before [16], and its combination with the phases technique of Fleischer [11] for solving packing problems has been first observed by Young [33] for solving the more general case of mixed packing LPs.

$\text{length}_{(l,\phi)}(i', p) \leq (1+\varepsilon)a(l,\phi)$. It then augments $\Delta = \min\{\frac{d_i'}{h_i'(p)}, \min_{e\in p} u(e)\}$ units of flow from commodity i' through p and updates the corresponding dual variables by setting $l(e) = l(e)(1+\varepsilon\frac{\Delta}{u(e)})$, $\forall e \in p$, and $\phi_{i'} = \phi_{i'}(1+\varepsilon\frac{\Delta h_{i'}(p)}{d_{i'}})$. The algorithm terminates at the first iteration for which $D = \sum_{e\in E} l(e)u(e) + \sum_{i=1}^{k}\phi_i d_i > 1$, and scales the final flow by $\log_{1+\varepsilon}\frac{1+\varepsilon}{\delta}$.

We now turn to the most crucial step of the algorithm: to build a suitable approximate oracle to identify the most violated constraint (7) of the dual. Our task is to approximately minimize, overall $1 \leq i \leq k$ and $q \in P_i$, the function

$$\frac{l(q) + \phi_i h_i(q)}{v_i(q)} = \frac{l(q) + \phi_i \cdot \mathrm{h}_i\left(\frac{wt_i(q)}{\delta_i(s_i,t_i)}\right)}{\mathrm{v}_i\left(\frac{wt_i(q)}{\delta_i(s_i,t_i)}\right)}.$$

Note that for a fixed i, this requires the solution of a NASP instance with objective function

$$U([x_1, x_2]^T) = \frac{x_1 + \phi_i \mathrm{h}_i(\frac{x_2}{\delta_i(s_i,t_i)})}{\mathrm{v}_i(\frac{x_2}{\delta_i(s_i,t_i)})}$$

and cost vector $\mathbf{c} = [l, wt_i]^T$. Note also that

$$\frac{\frac{\partial U}{\partial x_1}([x_1,x_2]^T)}{U([x_1,x_2]^T)} \leq \frac{1}{x_1}$$

which implies that we can apply Theorem 4 for any fixed i and make use of a non-additive shortest path routine $\bar{p} = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)$ that returns an s_i–t_i path $\bar{p}$ that approximately (within $(1+\varepsilon)$) minimizes the above function, overall $q \in P_i$, in time $O(n^2 m \frac{\log(nL)}{\varepsilon})$, where $L = \frac{\max_{e\in E} l(e)}{\min_{e\in E} l(e)}$.

To efficiently implement the oracle, we do not call the NASP routine for every value of i. Instead, the oracle proceeds in phases (like in [11]), maintaining a lower bound estimation $\overline{a}$ of $a(l,\phi)$, initially equal to $\overline{a} = \frac{1}{1+\varepsilon}\min_{1\leq i\leq k}\{\frac{l(p_i)+\phi_i h_i(p_i)}{v_i(p_i)} \mid p_i = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)\}$. In each phase, the oracle examines the commodities one by one by performing NASP computations. For each commodity i the oracle returns a path $p = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)\}$ for which $\frac{l(p)+\phi_i h_i(p)}{v_i(p)} \leq \overline{a}(1+\varepsilon)^2$. As long as such a path can be found, the oracle sticks to commodity i. Otherwise, it continues with commodity $i+1$. After all k commodities are considered in a phase, we know that $a(l,\phi) \geq (1+\varepsilon)\overline{a}$ and proceed to the next phase by setting $\overline{a} = (1+\varepsilon)\overline{a}$. The pseudocodes of our algorithm and the oracle are given in Fig. 3.

To discuss correctness and time bounds, we start with the following lemma that establishes an upper bound on the ratio of the lengths of the minimum length column at the start and the end of the GK algorithm.

```
QoS-MCF($G, u, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{d}, \boldsymbol{wt}, \boldsymbol{v}, \varepsilon$) {
  **forall** $e \in E$ { $l(e) = \frac{\delta}{u(e)}$ }
  **for** $i = 1$ to $k$ { $\phi_i = \frac{\delta}{d_i}$ }
  **for** $i = 1$ to $k$ { **forall** $e \in E$ { $X_i(e) = 0$ }}
  $D = (m + k)\delta$;
  **for** $i = 1$ to $k$ { $p_i = \mathrm{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)$ }
  $\overline{a} = \frac{1}{1+\varepsilon} \min_{1 \le i \le k} \left\{ \frac{l(p_i)+\phi_i h_i(p_i)}{v_i(p_i)} \right\}$;
  $i = 1$;
  **while** $D \le 1$ {
    $(p, i, \overline{a})$ = QoS-MCF-oracle($G, \boldsymbol{s}, \boldsymbol{t}, l, \boldsymbol{wt}, \boldsymbol{v}, \boldsymbol{\phi}, \varepsilon, i, \overline{a}$);
    $\Delta = \min\{\frac{d_i}{h_i(p)}, \min_{e \in p} u(e)\}$;
    $X_i(p) = X_i(p) + \Delta$;
    **forall** $e \in p$ **do** $l(e) = l(e)(1 + \varepsilon \frac{\Delta}{u(e)})$;
    $\phi_i = \phi_i(1 + \varepsilon \frac{\Delta h_i(p)}{d_i})$;
    $D = D + \varepsilon \Delta \frac{l(p)+\phi_i h_i(p)}{v_i(p)}$;
  }
  **for** $i = 1$ to $k$ { **forall** $e \in E$ { $X_i(e) = X_i(e)/\log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}$ }}
}
QoS-MCF-oracle($G, \boldsymbol{s}, \boldsymbol{t}, l, \boldsymbol{wt}, \boldsymbol{v}, \boldsymbol{\phi}, \varepsilon, j, \overline{a}$) {
    **while true** {
      **for** $i = j$ to $k$ {
        $p = \mathrm{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)$;
        **if** $\frac{l(p)+\phi_i h_i(p)}{v_i(p)} \le \overline{a}(1 + \varepsilon)^2$
          **return** $(p, i, \overline{a})$;
      }
      $\overline{a} = \overline{a}(1 + \varepsilon)$; /* update rule for next phase */
    }
}
```

Fig. 3 The approximation algorithm for the QoS-MCF problem

Lemma 3 *Let $a(0)$ and $a(t)$ be the lengths of the minimum length column at the start and the end of the algorithm, respectively. Then, $\frac{a(t)}{a(0)} \le \frac{1+\varepsilon}{\delta}$.*

Proof By the initial values of the dual variables, we have $a(0) = \min_j \sum_i \frac{A(i,j)}{c(j)} y_0(i)$ $= \delta \cdot \min_j \sum_i \frac{A(i,j)}{c(j)b(i)}$. Since now the algorithm stops at the first iteration t such that $D(t) > 1$ and the dual variables increase by at most $1 + \varepsilon$ in each iteration, it holds that $D(t) \le 1 + \varepsilon$. Consequently, $\sum_i b(i) y_t(i) \le 1 + \varepsilon$, which implies that $y_t(i) \le (1 + \varepsilon)\frac{1}{b(i)}$, $\forall i$. Hence, $a(t) = \min_j \sum_i \frac{A(i,j)}{c(j)} y_t(i) \le (1 + \varepsilon) \cdot \min_j \sum_i \frac{A(i,j)}{c(j)b(i)} = \frac{1+\varepsilon}{\delta} a(0)$. □

The following lemma establishes the approximation guarantee for the oracle.

Lemma 4 *A call to the oracle returns a* $(1+\varepsilon)^2$*-approximation of the most violated constraint in the dual.*

Proof Let $\overline{a}_j$ be the value of $\overline{a}$ during the j-th phase of the algorithm. It suffices to show that for all phases $j \geq 1, \overline{a}_j \leq a(l,\phi)$.

Initially ($j=1$), we set

$$\overline{a}_1 = \frac{1}{1+\varepsilon} \min_{1\leq i\leq k} \left\{ \frac{l(p_i)+\phi_i h_i(p_i)}{v_i(p_i)} \,\middle|\, p_i = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon) \right\}.$$

By the definition of the NASP routine, we get

$$\overline{a}_1 \leq \frac{1}{1+\varepsilon} \min_{1\leq i\leq k} \left\{ (1+\varepsilon) \min_{p\in P_i} \frac{l(p)+\phi_i h_i(p)}{v_i(p)} \right\} = a(l,\phi).$$

For any subsequent phase $j>1$, consider phase $j-1$. The oracle finishes the examination of a commodity i and proceeds with $i+1$ only when a call to NASP$(G, s_i, t_i, l, wt_i, \varepsilon)$ in phase $j-1$ returns a path p_i for which $\frac{l(p_i)+\phi_i h_i(p_i)}{v_i(p_i)} > \overline{a}_{j-1}(1+\varepsilon)^2$. This inequality and the definition of the NASP routine imply that at the end of phase $j-1$, we have for each commodity i

$$\overline{a}_{j-1}(1+\varepsilon)^2 < (1+\varepsilon) \min_{p\in P_i} \frac{l(p)+\phi_i h_i(p)}{v_i(p)}.$$

Hence, by the definition of $a(l,\phi)$, and since $l(e)$ can only increase during the algorithm, at the end of the phase we have $\overline{a}_{j-1}(1+\varepsilon)^2 < (1+\varepsilon)a(l,\phi)$. Since $\overline{a}_j = \overline{a}_{j-1}(1+\varepsilon)$, we get $\overline{a}_j < a(l,\phi)$. □

To establish a bound on the time complexity of the algorithm, we need to count the number of NASP computations. Clearly, at most one NASP computation is needed per augmentation of flow. The rest of NASP computations (not leading to an augmentation) are bounded by k times the number of phases. The following lemma establishes a bound on the total number of phases.

Lemma 5 *The number of phases of algorithm QoS-MCF is bounded by* $\lceil \frac{1}{\varepsilon}(1+\log_{1+\varepsilon}(m+k)) \rceil + 2$.

Proof Let $a(0)$ and $a(t)$ be the lengths of the most violated constraint at the start and the end of the algorithm, respectively. Let now $\overline{a}_j$ be the value of $\overline{a}$ during the j-th phase of the algorithm, and T be the last phase of the algorithm.

Initially, we set $\overline{a}_1 = \frac{1}{1+\varepsilon}\min_{1\leq i\leq k}\{\frac{l(p_i)+\phi_i h_i(p_i)}{v_i(p_i)} \mid p_i = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)\}$ and by the definition of the NASP routine we get that $a(0) \leq (1+\varepsilon)\overline{a}_1$. From the proof of Lemma 4, we have that $\overline{a}_T \leq a(t)$, and from Lemma 3 we get that $a(t) \leq \frac{1+\varepsilon}{\delta}a(0)$. Combining the last three inequalities we get $\overline{a}_T \leq \frac{(1+\varepsilon)^2}{\delta}\overline{a}_1$. By the update rule for $\overline{a}$ on each phase, we have that $\overline{a}_T = \overline{a}_1(1+\varepsilon)^{T-1}$, and therefore

$\overline{a}_1(1+\varepsilon)^{T-1} \leq \frac{(1+\varepsilon)^2}{\delta}\overline{a}_1$, which implies that $T \leq \log_{1+\varepsilon}\frac{(1+\varepsilon)^3}{\delta}$. Hence, the number of phases is bounded by $\lceil \log_{1+\varepsilon}\frac{(1+\varepsilon)^3}{\delta} \rceil = \lceil \frac{1}{\varepsilon}(1+\log_{1+\varepsilon}(m+k)) \rceil + 2$, since $\delta = (1+\varepsilon)((1+\varepsilon)(m+k))^{-\frac{1}{\varepsilon}}$. □

We are now ready for the main result of this section.

Theorem 6 *There is an algorithm that computes a* $(1-\varepsilon)^{-2}(1+\varepsilon)^2$*-approximation to the QoS-aware MCF problem in time* $O((\frac{1}{\varepsilon})^3(m+k)\log(m+k)mn^2(\frac{1}{\varepsilon}\log(m+k) + \log(nU))$, *where n is the number of nodes, m is the number of edges, k is the number of commodities, and* $U = \frac{\max_{e\in E}u(e)}{\min_{e\in E}u(e)}$.

Proof From Theorem 5 (with $M = m+k$) and Lemma 4 we have that the algorithm computes a $(1-\varepsilon)^{-2}(1+\varepsilon)^2$-approximation to the optimal and terminates after at most $(m+k)\lceil\frac{1}{\varepsilon}(1+\log_{1+\varepsilon}(m+k))\rceil$ augmentations. Since for each phase at most k NASP computations do not lead to an augmentation, we get from Lemma 5 that the oracle performs at most $k\lceil\frac{1}{\varepsilon}(1+\log_{1+\varepsilon}(m+k))\rceil + 2k$ NASP computations not leading to an augmentation. Therefore, the total number of NASP computations during an execution of the algorithm is $O(\frac{1}{\varepsilon}(m+k)\log_{1+\varepsilon}(m+k)) = O((\frac{1}{\varepsilon})^2(m+k)\log(m+k))$.

A NASP computation is carried out in time $O(\frac{1}{\varepsilon}n^2m\log(nL))$, where $L = \frac{\max_{e\in E}l(e)}{\min_{e\in E}l(e)}$. From the initialization of $l(e)$, and since they can only increase during the algorithm, it is clear that $\min_{e\in E}l(e) \geq \frac{\delta}{\max_{e\in E}u(e)}$. Since now the algorithm stops at the first iteration such that $\sum_{e\in E}l(e)u(e) + \sum_{i=1}^{k}\phi_i d_i > 1$ and the dual variables increase by at most $1+\varepsilon$ in each iteration, it holds that $\sum_{e\in E}l(e)u(e) + \sum_{i=1}^{k}\phi_i d_i \leq 1+\varepsilon$. Consequently at the end of the algorithm we have $l(e) \leq \frac{(1+\varepsilon)}{u(e)}$, $\forall e \in E$, and thus $\max_{e\in E}l(e) \leq \frac{1+\varepsilon}{\min_{e\in E}u(e)}$. Hence, $L \leq \frac{1+\varepsilon}{\delta}U$. By our choice of $\delta = (1+\varepsilon)((1+\varepsilon)\ (m+k))^{-\frac{1}{\varepsilon}}$, we have that $L \leq ((1+\varepsilon)(m+k))^{\frac{1}{\varepsilon}}U$, and hence the time required for a NASP computation is $O(\frac{1}{\varepsilon}mn^2(\frac{1}{\varepsilon}\log(m+k) + \log(nU)))$. Thus, we get an algorithm that computes a $(1-\varepsilon)^{-2}(1+\varepsilon)^2$-approximation to the QoS-aware MCF problem in time $O([(\frac{1}{\varepsilon})^2(m+k)\log(m+k)][\frac{1}{\varepsilon}mn^2(\frac{1}{\varepsilon}\log(m+k) + \log(nU)]) = O((\frac{1}{\varepsilon})^3(m+k)\log(m+k)mn^2(\frac{1}{\varepsilon}\log(m+k) + \log(nU))$, which is polynomial to the input and $\frac{1}{\varepsilon}$. □

5.3.4 Extensions

Better bounds can be obtained for the *constrained* version of the QoS-aware MCF problem. In that version all profits are constant (non QoS-elastic), there is an upper bound (constraint) on the QoS per path provided, and the objective is to maximize the total profit. In this case, we can achieve a FPTAS by implementing the oracle using a FPTAS for RSP instead of NASP (using the algorithm in [22]). Arguing as in Theorem 6 and taking into account that the FPTAS for RSP runs in $O(mn(\log\log n + 1/\varepsilon))$

time, we can achieve a running time of

$$O\left(\left(\frac{1}{\varepsilon}\right)^2 (m+k)\log(m+k)mn(\log\log n + 1/\varepsilon)\right).$$

For the version of the problem with unbounded demands, which constitutes the QoS max flow problem defined in [3], a better time bound of

$$O\left(\left(\frac{1}{\varepsilon}\right)^2 nm^2 \log m(\log\log n + 1/\varepsilon)\right)$$

can be achieved since the number of constraints in its corresponding LP is m.

6 Conclusions

We provided an efficient FPTAS for a core problem (multiobjective shortest path) in multiobjective optimization along with a new generic method for obtaining FPTAS for any multiobjective optimization problem with non-linear objectives. These two results formed the building blocks for obtaining better approximate solutions to three related problems: multiobjective constrained path, multiobjective constrained optimal path, and non-additive shortest path. We also considered an important generalization of the weighted multicommodity flow problem with elastic demands and values, and exhibited an interesting connection between the efficient approximate (FPTAS) solution of that problem and non-additive shortest paths. It would be interesting to investigate whether more time-efficient FPTAS for multiobjective shortest paths can be obtained.

Acknowledgements We are indebted to Naveen Garg, Jochen Könemann, Spyros Kontogiannis, Frank Wagner for various useful discussions, and to Christos Papadimitriou for bringing [3] to our attention. We are also indebted to an anonymous referee for several valuable comments that improved the presentation.

References

1. Ackermann, H., Newman, A., Röglin, H., Vöcking, B.: Decision making based on approximate and smoothed Pareto curves. In: Algorithms and Computation, ISAAC 2005. Lecture Notes in Computer Science, vol. 3827, pp. 675–684. Springer, Berlin (2006). Full version as Tech. Report AIB-2005-23, RWTH Aachen, December (2005)
2. Beckmann, M., McGuire, G., Winsten, C.: Studies in the Economics of Transportation. Yale University Press, New Haven (1956)
3. Chaudhuri, K., Papadimitriou, C., Rao, S.: Optimum routing with quality of service constraints. Manuscript (2004)
4. Corley, H., Moon, I.: Shortest paths in networks with vector weights. J. Optim. Theory Appl. **46**(1), 79–86 (1985)
5. Dafermos, S.: The general multimodal network equilibrium problem with elastic demands. Networks **12**, 57–72 (1982)
6. Datta, A., Vandermeer, D., Celik, A., Kumar, V.: Broadcast protocols to support efficient retrieval from databases by mobile users. ACM Trans. Database Syst. **24**(1), 1–79 (1999)
7. Ehrgott, M.: Multicriteria Optimization. Springer, Berlin (2000)

8. Ehrgott, M., Gandibleux, X. (eds.): Multiple Criteria Optimization—State of the Art Annotated Bibliographic Surveys. Kluwer Academic, Boston (2002)
9. Ergun, F., Sinha, R., Zhang, L.: An improved FPTAS for restricted shortest path. Inf. Process. Lett. **83**, 287–291 (2002)
10. Figueira, J., Greco, S., Ehrgott, M. (eds.): Multiple Criteria Decision Analysis—State of the Art Surveys. Springer, Boston (2005)
11. Fleischer, L.K.: Approximating fractional multicommodity flows independent of the number of commodities. SIAM J. Discrete Math. **13**(4), 505–520 (2000)
12. Florian, M., Nguyen, S.: A method for computing network equilibrium with elastic demands. Transp. Sci. **8**, 321–332 (1974)
13. Gabriel, S., Bernstein, D.: The traffic equilibrium problem with nonadditive path costs. Transp. Sci. **31**(4), 337–348 (1997)
14. Gabriel, S., Bernstein, D.: Nonadditive shortest paths: subproblems in multi-agent competitive network models. Comput. Math. Organ. Theory **6**, 29–45 (2000)
15. Garg, N., Könemann, J.: Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In: Proc. 39th IEEE Symposium on Foundations of Computer Science, FOCS'98, pp. 300–309. IEEE CS Press (1998)
16. Garg, N., Könemann, J.: Personal communication (2005)
17. Goel, A., Ramakrishnan, K.G., Kataria, D., Logothetis, D.: Efficient computation of delay-sensitive routes from one source to all destinations. In: Proc. IEEE Conf. Comput. Commun. INFOCOM (2001)
18. Hansen, P.: Bicriterion Path Problems. In: Theory and Applications. Proc. 3rd Conf. Multiple Criteria Decision Making. Lecture Notes in Economics and Mathematical Systems, vol. 117, pp. 109–127. Springer, Berlin (1979)
19. Hassin, R.: Approximation schemes for the restricted shortest path problem. Math. Oper. Res. **17**, 36–42 (1992)
20. Hensen, D., Truong, T.: Valuation of travel times savings. J. Transp. Econ. Policy 237–260 (1985)
21. Korkmaz, T., Kruz, M.: Multiconstrained optimal path selection. In: Proc. IEEE Conf. Comput. Commun. INFOCOM, pp. 834–843 (2001)
22. Lorenz, D.H., Raz, D.: A simple efficient approximation scheme for the restricted shortest path problem. Oper. Res. Lett. **28**, 213–219 (2001)
23. Van Mieghem, P., Kuipers, F.A., Korkmaz, T., Krunz, M., Curado, M., Monteiro, E., Masip-Bruin, X., Sole-Pareta, J., Sanchez-Lopez, S.: Quality of service routing. In: Quality of Future Internet Services. Lecture Notes in Computer Science, vol. 2856, pp. 80–117. Springer, Berlin (2003)
24. Papadimitriou, C., Yannakakis, M.: On the approximability of trade-offs and optimal access of web sources. In: Proc. 41st Symp. on Foundations of Computer Science, FOCS, pp. 86–92 (2000)
25. PIN project (Projekt Integrierte Netzoptimierung): Deutsche Bahn AG (2000)
26. Scott, K., Bernstein, D.: Solving a best path problem when the value of time function is nonlinear. Preprint 980976 of the Annual Meeting of the Transportation Research Board (1997)
27. Tsaggouris, G., Zaroliagis, C.: Improved FPTAS for multiobjective shortest paths with applications. CTI Techn. Report TR-2005/07/03, July (2005)
28. Tsaggouris, G., Zaroliagis, C.: QoS-aware multicommodity flows and transportation planning. In: Proc. 6th Workshop on Algorithmic Methods and Models for Optimization of Railways, ATMOS (2006)
29. Tsaggouris, G., Zaroliagis, C.: Multiobjective optimization: Improved FPTAS for shortest paths and non-linear objectives with applications. In: Algorithms and Computation, ISAAC 2006. Lecture Notes in Computer Science, vol. 4288, pp. 389–398. Springer, Berlin (2006)
30. Vassilvitskii, S., Yannakakis, M.: Efficiently computing succinct trade-off curves. In: Automata, Languages, and Programming, ICALP 2004. Lecture Notes in Computer Science, vol. 3142, pp. 1201–1213. Springer, Berlin (2004)
31. Wagner, F.: Challenging optimization problems at deutsche bahn. AMORE Workshop (invited talk) (1999)
32. Warburton, A.: Approximation of Pareto optima in multiple-objective shortest path problems. Oper. Res. **35**, 70–79 (1987)
33. Young, N.: Sequential and parallel algorithms for mixed packing and covering. In: Proc. 42nd IEEE Symp. on Foundations of Computer Science, FOCS, pp. 538–546 (2001)