Research Academic
COMPUTER TECHNOLOGY INSTITUTE

TECHNICAL REPORT No. TR 2005/04/01

MULTIOBJECTIVE SHORTEST PATHS
AND QOS-AWARE MULTICOMMODITY FLOWS

George Tsaggouris Christos D. Zaroliagis

April 2005

Multiobjective Shortest Paths and QoS-aware Multicommodity Flows*

George Tsaggouris† Christos Zaroliagis†

April 12, 2005

Abstract

The QoS-aware Multicommodity Flow (MCF) problem is an important generalization of the weighted MCF problem that models realistic network design scenarios where commodity demands and profits (for routing flow) are elastic to the Quality-of-Service of the paths along which the commodity is routed. We provide a FPTAS for this problem by employing a Lagrangian relaxation method, which assumes the existence of an oracle that identifies the most violated constraint of the dual. This identification is a non-trivial task, since it involves the solution of an NP-hard non-linear combinatorial optimization problem, known as non-additive shortest path, which is a variant of the multiobjective shortest path problem. Both multiobjective and non-additive shortest path problems are fundamental in multiobjective optimization and hence of independent interest. We also provide efficient FPTAS for these two problems that improve significantly upon previous approaches, especially in the case of more than two objectives. To the best of our knowledge, this is the first FPTAS for non-additive shortest paths.

1 Introduction

Consider a capacitated directed network $G = (V, E)$ in which we wish to route k commodities to meet certain initial demands. Each commodity i is associated with a specific origin-destination pair (s_i, t_i), a demand d_i, and a value v_i representing the *profit* of routing one unit of flow from that commodity. Also, for each commodity i, a weight $wt_i : E \to \mathbb{R}_0^+$ is defined that quantifies the provided *quality of service (QoS)* when this commodity is routed along an edge e or a path p, where $wt_i(p) = \sum_{e \in p} wt_i(e)$. Smaller weight means better QoS. When a commodity is not routed along its shortest w.r.t. wt_i (optimal w.r.t. the QoS) path due to capacity restrictions, then (i) a portion of its demand d_i drops (the worse the QoS of the path, the larger the portion of d_i that is lost), and (ii) its value v_i is reduced (the worse the QoS, the larger the reduction). In other words, demands and values are *elastic* to the provided QoS. The objective in the above problem 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 value, subject to the capacity and demand constraints, and w.r.t. the QoS-elasticity of demands and values. We call the above the *QoS-aware Multicommodity Flow* (MCF) problem.

The QoS-aware MCF problem is a natural generalization of the weighted MCF problem, modelling realistic scenarios encountered in important applications in transportation[1] and communication networks [4, 23]. For instance, network operators in a public transportation network would

*This work was partially supported by the IST Programme (6th FP) of EC under contract No. IST-2002-001907 (integrated project DELIS), 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.

†Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece, and Dept of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece. Emails: {tsaggour,zaro}@ceid.upatras.gr

[1]Elasticity of demands has received considerable attention in the transportation literature; see e.g., [1, 3, 9].

like to route various commodities (customers with common origin-destination pairs) to meet certain demands [23, 24]. A customer, when provided with a non-optimal path (route) due to unavailable capacity, s/he will most likely switch to another operator or even other means of transport and the probability in doing so increases as the QoS drops (actually, as a result of statistical measurements over several years, major European railway companies know quite accurately the percentage of customers they lose in such cases as a function of the path's QoS [24]). To minimize the loss of customers, the value charged for the requested service is usually reduced to make the alternative (worse in QoS) path, offered for that service, attractive. A similar situation is encountered with information dissemination over various communication networks [4]. 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. 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 an 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.

The QoS-aware MCF problem can be described as a linear program (LP). To capture the QoS-characteristics of the paths, however, an exponential number of variables is required, since the paths are exponentially many. Moreover, the standard ways (arc-flow formulations) of writing MCF LPs as polynomial size programs do not work here, because of the need to capture the QoS-characteristics along paths. Consequently, standard LP solving methods are out of consideration. For this reason, one has to resort to more sophisticated *Lagrangian-relaxation* methods (see e.g., [8, 12, 20, 26], and [2] for a comprehensive survey and further references). These methods turn to be particularly useful for solving sparse problems with exponentially many variables, like the QoS-aware MCF problem, by exploiting structural properties of the LP formulations. An important category includes (pure) *fractional packing* LPs, which are of the form $\max\{c^T x | Ax \leq b, x \geq 0\}$, where $A_{M\times N}$, $b_{M\times 1}$, and $c_{N\times 1}$ have positive entries. The QoS-aware MCF problem can be formulated as a fractional packing LP.

A remarkably simple and elegant Lagrangian-relaxation method for solving fractional packing LPs was given by Garg and Könemann [12] – henceforth the *GK approach*. This approach assumes the existence of an oracle that identifies the most violated constraint of the dual LP. Finding an efficient way to implement such an oracle constitutes the *prime challenge* when applying this (or any other Lagrangian-relaxation) method. In the case of the QoS-aware MCF problem, the implementation of the oracle for identifying the most violated constraint involves the solution of an NP-hard *non-linear* combinatorial optimization problem, known as the *non-additive shortest path* (NASP) – a variant of the multiobjective shortest path problem. In NASP, we are given a digraph whose edges are associated with a d-dimensional cost vector and the task is to find a path that minimizes a certain d-attribute non-linear cost function.

Multiobjective shortest paths and NASP are core problems in multiobjective optimization with numerous applications; see e.g., [10, 11, 14, 17]. In multiobjective optimization (an area under intense study within Operations Research and Economics in the last 60 years [5, 6]), we are not interested in a single optimal solution, but in the trade-off among the different objective functions, called the *Pareto curve* (the set of all feasible solutions whose vector of the various objectives is not dominated by any other solution). Since the Pareto curve is typically exponential in size (even

in the case of two objectives), it is natural to study approximate versions of this concept. 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, an alternative (complementary) approach is to proceed in a normative way and choose the "best" solution by introducing a utility function (as in the case of NASP).

Despite so much research [5, 6], only recently a systematic study of the complexity issues regarding the construction of approximate Pareto curves has been initiated [19, 22]. Informally, an $(1+\varepsilon)$-approximate Pareto curve P_ε is a subset of feasible solutions such that for any Pareto optimal solution, there exists a solution in P_ε that is no more than $(1+\varepsilon)$ away in all objectives. Papadimitriou and Yannakakis in a seminal work [19] show that a polynomial in size P_ε exists for any multiobjective optimization problem, has size $|P_\varepsilon| = O((4B/\varepsilon)^{d-1})$, where B is the number of bits required to represent the input and d is the number of objective functions (typically a constant), and it can be constructed by $O((4B/\varepsilon)^d)$ calls to a GAP routine that solves (in time polynomial in B 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. Improvements on the number of GAP calls to produce a constant approximation to the smallest possible P_ε for the cases of 2 and 3 objectives are presented in [22], while for $d > 3$ objectives inapproximability results are shown for such a constant approximation.

For the case of multiobjective shortest paths, Papadimitriou and Yannakakis [19] 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 multiobjective shortest paths were already known in the case of two objectives [15], as well as in the case of multiple objectives in directed acyclic graphs (DAGs) [25]. In particular, the 2-objective case has been extensively studied [6], while for $d > 2$ very little has been achieved; actually (to the best of our knowledge) the results in [19, 25] are the only and currently best FPTAS known. Let C^{max} denote the ratio of the maximum to the minimum edge weight (in any dimension). For the case of DAGs and $d > 2$, the algorithm of [25] runs in $O(nm(\frac{n\log(nC^{max})}{\varepsilon})^{d-1}\log^{d-2}(\frac{n}{\varepsilon}))$ 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 [7, 16, 18]. In [22] it is shown that this achieves a time of $O(nm|P^*_\varepsilon|(\log\log n + 1/\varepsilon))$ for general digraphs and $O(nm|P^*_\varepsilon|/\varepsilon)$ for DAGs, where $|P^*_\varepsilon|$ is the size of the smallest possible P_ε (and which can be as large as $\log_{1+\varepsilon} nC^{max} \approx \frac{1}{\varepsilon}\ln(nC^{max})$). Note that all these approaches deal typically with the single-pair version of the problem. Regarding NASP, no FPTAS is known (to the best of our knowledge). Previous approaches deal with exact solutions, thus resulting in pseudopolynomial algorithms, or concern the efficient solution of the Lagrangian relaxation of NASP (best point in the convex hull of the Pareto set); see [21] and the references in that paper.

Our first contribution in this work (Section 3) is a new and remarkably simple FPTAS for constructing a set of approximate Pareto curves for the *single-source* version of the multiobjective shortest path 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 time $O(m(\frac{n\log(nC^{max})}{\varepsilon})^{d-1})$ for DAGs. These results improve significantly upon the previous approaches for general digraphs [19, 22] and DAGs [22, 25], for all $d > 2$. For $d = 2$, our running times depend on ε^{-1}, while those based on repeated-RSP applications (like in [22]) depend on ε^{-2}. This always gives us better running times than those in [22, 25] for DAGs, while for general digraphs we get a better running time than [22] when $n = O(\frac{1}{\varepsilon})$. The same bounds apply to NASP, when the utility function has a quite general form – that includes *any* polynomial of bounded degree with non-negative coefficients – and constitute the *first* FPTAS for this problem. For example, an important special case (required for the solution of the QoS-aware MCF problem) concerns the utility function $\mathcal{U}([x_1, x_2]^T) = x_1\mathcal{U}_1(x_2) + \mathcal{U}_2(x_2)$ (where $\mathcal{U}_1$ and $\mathcal{U}_2$ are non-linear and non-decreasing). In this case, our FPTAS runs in $O(n^2m\frac{\log(nC^{max})}{\varepsilon})$ time, i.e.,

in time *independent* of the particular form of $\mathcal{U}_1$ and $\mathcal{U}_2$. A similar statement holds for the general case. Unlike previous methods that are based on converting pseudopolynomial time algorithms to FPTAS using rounding and scaling techniques, our approach for multiobjective shortest paths builds upon a natural iterative process that extends and merges sets of node labels representing partial solutions, while keeping them small by discarding some solutions in an error controlable way.

Armed with the above efficient FPTAS for NASP, we are able to address effectively the prime challenge of the Lagrangian relaxation method and present an efficient implementation of the required oracle, thus achieving a FPTAS for the QoS-aware MCF problem. This constitutes the second contribution of this work (Section 4). In particular, our algorithm is based on a straightforward, albeit powerful, extension of the GK approach that uses approximate (instead of exact) oracles, combined with the phases technique by Fleischer [8]. Such an extension of the GK approach to work with approximate oracles was known before [13], and its combination with the phases technique of Fleischer [8] for solving packing problems has been first observed by Young [26] for solving the more general case of mixed packing LPs. Our approach is not based on that of Young [26], but builds directly upon those of [8, 12], resulting in a simpler and crisp algorithm.

2 Preliminaries and Problem Formulation

2.1 QoS-aware Multicommodity Flow

We are given a digraph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges, 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, $1 \leq i \leq k$, 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. 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 gives the portion $\mathrm{f_i}(x)$ of the commodity's demand d_i that we lose if the provided path is x times worse than the shortest path w.r.t $wt_i(\cdot)$. 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 $wt_i(\cdot)$.

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 $\mathrm{f_i}$ is non-decreasing, $\mathrm{h_i}$ 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:

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

The objective is to maximize the total profit, which is the sum over all commodities of the flow routed from each commodity multiplied by the corresponding QoS-elastic value, subject to the

capacity and demand constraints and w.r.t. the QoS-elasticity of the demands. To formulate the QoS-aware MCF problem as a linear program, for each commodity i and each path $p \in P_i$, we introduce a variable $X_i(p)$ denoting the flow of commodity i routed along p. Using the above definitions, the QoS-aware MCF problem can be described as follows.

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

$$s.t. \quad \sum_{i=1}^{k} \sum_{e \in p, p \in P_i} X_i(p) \leq u(e), \forall e \in E \tag{2}$$

$$\sum_{p \in P_i} X_i(p) h_i(p) \leq d_i, \forall i = 1 \ldots k \tag{3}$$

2.2 Multiobjective Optimization

An instance of a multiobjective optimization problem is associated with a set of feasible solutions Q and a d-vector function $\mathbf{f} = [f_1, \ldots, f_d]^T$ (d is the number of objectives, which is typically a small 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, $1 \leq i \leq d$, are to be minimized.

Let p, q be two feasible solutions. A solution p *dominates* q iff it is at least as good in all objectives, i.e., $f_i(p) \leq f_i(q)$, $1 \leq i \leq d$. The *Pareto set* $\mathcal{P}$ of Q is defined as the set of all undominated elements of Q. Given a vector of approximation ratios $\boldsymbol{\rho} = [\rho_1, \ldots, \rho_d]^T$ ($\rho_i \geq 1$, $1 \leq i \leq d$), 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). An $\boldsymbol{\rho}$-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 an $(1+\varepsilon)$-Pareto set and runs in time polynomial in the size of the input and $\frac{1}{\varepsilon}$.

If $\mathbf{a} = [a_1, a_2, \cdots, a_d]^T$ and $\mathbf{b} = [b_1, b_2, \cdots, b_d]^T$ are d-dimensional vectors and λ a scalar, then we denote by $\mathbf{ab} = [a_1 b_1, a_2 b_2, \cdots, a_d b_d]^T$ and $\mathbf{a}^\lambda = [a_1^\lambda, a_2^\lambda, \cdots, a_d^\lambda]^T$. 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 $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges, 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 powerset 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* (SSMSP) 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 $\boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_{d-1}]^T$ of error parameters ($0 < \varepsilon_i < 1$, $1 \leq i \leq d-1$) and a source node s, in this section we present an algorithm that computes, for each node v, a $\boldsymbol{\rho}$-cover of

$P(s,v)$, where $\boldsymbol{\rho} = [1+\varepsilon_1, 1+\varepsilon_2, \cdots, 1+\varepsilon_{d-1}, 1]^T$. 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 $1 \leq i \leq d$. 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 SSMSP algorithm

In our algorithm, a path $p = (e_1, e_2, \cdots, e_{k-1}, e_k)$ is represented by a label that is a tuple of the form $(\mathbf{c}(p), \text{pred}(p), \text{lastedge}(p))$, where $\mathbf{c}(p) = \sum_{e\in p} \mathbf{c}(e)$ is the d-dimensional cost vector of the path, $\text{pred}(p) = \vec{q}$ is a pointer to the label of the subpath $q = (e_1, e_2, \cdots, e_{k-1})$ of p, and $\text{lastedge}(p) = e_k$ 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.

Our single-source multiobjective shortest path algorithm can be viewed as a generalization of the classical Bellman-Ford algorithm. Let $\mathbf{r} = [r_1, \ldots, r_{d-1}, 1]$ 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 $\Pi_v^i[0..\lfloor\log_{r_1}(nC_1)\rfloor, 0..\lfloor\log_{r_2}(nC_2)\rfloor, \cdots, 0..\lfloor\log_{r_{d-1}}(nC_{d-1})\rfloor]$, 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, $\mathbf{pos}(p) = [\lfloor\log_{r_1}\frac{c_1(p)}{c_1^{min}}\rfloor, \lfloor\log_{r_2}\frac{c_2(p)}{c_2^{min}}\rfloor, \cdots, \lfloor\log_{r_{d-1}}\frac{c_{d-1}(p)}{c_{d-1}^{min}}\rfloor]^T$ 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 $c_i(p) \leq (n-1)c_i^{max}$, $\forall 1 \leq i \leq d$, 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 $i \geq 1$ and for each node v the algorithm computes Π_v^i as follows. 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 the labels in Π_u^{i-1} by e, and merges the resulting set of s-v paths with Π_v^i, while maintaining at most one label (the one with the smallest c_d cost) for each position of the Π_v^i array, thus keeping the size of the sets polynomially bounded. In particular, the *Extend-&-Merge* operation on an edge $e = (u,v)$ is implemented as follows. We iterate through all labels $p \in \Pi_u^{i-1}$ and extend each p by e forming a new label (path) $q = (\mathbf{c}(p) + \mathbf{c}(e), \vec{p}, e)$. We then insert q in the position $\mathbf{pos}(q) = [\lfloor\log_{r_1}\frac{c_1(q)}{c_1^{min}}\rfloor, \lfloor\log_{r_2}\frac{c_2(q)}{c_2^{min}}\rfloor, \cdots, \lfloor\log_{r_{d-1}}\frac{c_{d-1}(q)}{c_{d-1}^{min}}\rfloor]^T$ of Π_v^i, unless this position is already filled in with a label q' for which $c_d(q') \leq c_d(q)$. The pseudocode of our algorithm is given in Fig. 1.

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 $c_\ell(q) \leq r_\ell c_\ell(p)$, $\forall 1 \leq \ell \leq d$.

SSMSP$(G, s, \mathbf{c}, \mathbf{r})$\{
  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 $p \in Q$ \{
    $q = (\mathbf{c}(p) + \mathbf{c}(e), \vec{p}, e)$;
    $\mathbf{pos}(q) = [\lfloor \log_{r_1} \frac{c_1(q)}{c_1^{min}} \rfloor, \cdots, \lfloor \log_{r_{d-1}} \frac{c_{d-1}(q)}{c_{d-1}^{min}} \rfloor]^T$;
    if $R[\mathbf{pos}(q)] = null$ **or** $c_d(R[\mathbf{pos}(q)]) > c_d(q)$ \{
      $R[\mathbf{pos}(q)] = q$;
    \}
  return R;
\}

Figure 1: The SSMSP algorithm

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:

$$c_\ell(\bar{q}) \leq r_\ell^{i-1} c_\ell(p), \quad 1 \leq \ell \leq d \tag{4}$$

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

$$c_\ell(q) \leq r_\ell c_\ell(\bar{q}), \quad 1 \leq \ell \leq d-1. \tag{5}$$

Since $r_d = 1$, combining now (iv) and (5) with (4), 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 SSMSP 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 $\mathbf{r} = [(1+\varepsilon_1)^{\frac{1}{n-1}}, (1+\varepsilon_2)^{\frac{1}{n-1}}, \cdots, (1+\varepsilon_{d-1})^{\frac{1}{n-1}}, 1]$, and taking into account that $\ln(1+\delta) \approx \delta$ for small δ, yields the main result of this section.

Theorem 1 *Given a vector $\boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_{d-1}]^T$ 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 $\boldsymbol{\rho} = [1+\varepsilon_1, 1+\varepsilon_2, \cdots, 1+\varepsilon_{d-1}, 1]^T$, in total time $O(n^d m \prod_{j=1}^{d-1} (\frac{1}{\varepsilon_j} \log(nC_j)))$.*

Let $C^{max} = \max_{1 \leq j \leq d-1} C_j$. In the special case, where $\varepsilon_i = \varepsilon$, $\forall 1 \leq i \leq d-1$, 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 $\Pi_v[\mathbf{0}] = \{(\mathbf{0}, null, null)\}$. The algorithm visits all the nodes w.r.t. the topological order and for each visited node, it performs an *Extend-&-Merge* operation to all its outgoing edges. We can show 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.

Our results for multiobjective shortest paths can be also applied to the *constrained versions* of the problem that have important applications in computer networks (see e.g., [14, 17]); namely, to the *Multiple Constrained Optimal Path* (MCOP) problem and to the *Multiple Constrained Path* (MCP) problem. 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., $c_i(p) = \sum_{e\in p} c_i(e) \leq b_i$, $\forall 1 \leq i \leq d-1$. For the case where $d = 2$, MCOP is known as the *Restricted Shortest Path* problem, which is NP-hard, but admits a FPTAS [7, 16, 18]. 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., $c_i(p) = \sum_{e\in p} c_i(e) \leq b_i$, $\forall 1 \leq i \leq d$.

Let $\boldsymbol{\rho} = [1+\varepsilon_1, 1+\varepsilon_2, \cdots, 1+\varepsilon_{d-1}, 1]^T$ and let Π be a $\boldsymbol{\rho}$-cover Π of $P(s,t)$, constructed using the SSMSP algorithm as implied by Theorem 1. For MCOP, we choose $p' = \text{argmin}_{p\in\Pi}\{c_d(p); c_i(p) \leq (1+\varepsilon_i)b_i,\ \forall 1 \leq i \leq d-1\}$ and get an *acceptable* solution in the sense of [14] 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 $1 \leq i \leq d-1$ by at most an $1+\varepsilon_i$ factor. For MCP, we 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 $c_i(p) \leq (1+\varepsilon_i)b_i$, $\forall 1 \leq i \leq d$. 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).

3.3 Non-Additive Shortest Paths

The above framework for multiobjective shortest paths can be applied to obtain FPTASs for the non-additive shortest path problem, which is of independent interest and of particular importance in this work (cf. Section 4.2).

In the *Non-Additive Shortest Path* (NASP) problem we are given a digraph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges and a d-dimensional function vector $\mathbf{c} : E \to [\mathbb{R}^+]^d$ associating each edge e with a vector of attributes $\mathbf{c}(e)$. A path p is also associated with a vector of attributes and $\mathbf{c}(p) = \sum_{e\in p} \mathbf{c}(e)$. We are also given a d-attribute non-decreasing and *non-linear* cost 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 (via a reduction

from the restricted shortest path problem).

To obtain our FPTAS, we consider a quite general family of non-linear functions $\mathcal{U}(\mathbf{x})$. Specifically, let $\mathcal{U}(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \mathcal{U}_j(x_d) \prod_{k=1}^{d-1} {x_k}^{\beta_{jk}}$, where $\alpha_j, \beta_{jk} \geq 0$, and each $\mathcal{U}_j$ is any non-negative and non-decreasing function. Typically, N, α_j, and β_{jk} are considered constants, and we assume that the value of $\mathcal{U}(\mathbf{x})$, for any $\mathbf{x}$, can be computed in constant time. We can prove the following.

Theorem 2 *Let $\mathcal{U}(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \mathcal{U}_j(x_d) \prod_{k=1}^{d-1} {x_k}^{\beta_{jk}}$ be a cost function, where $\alpha_j, \beta_{jk} \geq 0$, and $\mathcal{U}_j$ be any non-negative and non-decreasing functions. Then, for any $\varepsilon > 0$, there exists an algorithm that computes a $(1+\varepsilon)$-approximation to the NASP optimum with respect to $\mathcal{U}(\mathbf{x})$ in time $O(n^d m(\frac{\log(nC^{max})\Delta}{\varepsilon})^{d-1})$, where $\Delta = \max_{1 \leq j \leq N} \sum_{k=1}^{d-1} \beta_{jk}$.*

Proof. Apply Theorem 1 and construct a $\boldsymbol{\rho}$-Pareto set Π of $P(s,t)$, with $\rho_i = (1+\varepsilon)^{\frac{1}{\Delta}}$, $\forall 1 \leq i \leq d-1$, and $\rho_d = 1$. Pick $p' = \text{argmin}_{p \in \Pi}(\mathcal{U}(\mathbf{c}(p)))$. Let p^* denote the optimal solution.

By the definition of Π we know that there exists some $q \in \Pi$ such that $c_k(q) \leq (1+\varepsilon)^{\frac{1}{\Delta}} c_k(p^*)$, $\forall 1 \leq k \leq d-1$, and $c_d(q) \leq c_d(p^*)$. Since $\alpha_j, \beta_{jk} \geq 0$, $\forall j,k$, we get:

$$\begin{aligned}
\mathcal{U}(\mathbf{c}(q)) &= \sum_{j=1}^{N} \alpha_j \mathcal{U}_j(c_d(q)) \prod_{k=1}^{d-1} (c_k(q))^{\beta_{jk}} \leq \sum_{j=1}^{N} \alpha_j \mathcal{U}_j(c_d(p^*)) \prod_{k=1}^{d-1} ((1+\varepsilon)^{\frac{1}{\Delta}} c_k(p^*))^{\beta_{jk}} \\
&\leq \sum_{j=1}^{N} ((1+\varepsilon)^{\frac{1}{\Delta}})^{\sum_{k=1}^{d-1} \beta_{jk}} \alpha_j \mathcal{U}_j(c_d(p^*)) \prod_{k=1}^{d-1} (c_k(p^*))^{\beta_{jk}} \leq ((1+\varepsilon)^{\frac{1}{\Delta}})^{\Delta} \mathcal{U}(\mathbf{c}(p^*))
\end{aligned}$$

Since $p' = \text{argmin}_{p \in \Pi}(\mathcal{U}(\mathbf{c}(p)))$, we get $\mathcal{U}(\mathbf{c}(p')) \leq \mathcal{U}(\mathbf{c}(q)) \leq (1+\varepsilon)\mathcal{U}(\mathbf{c}(p^*))$. ∎

It is clear from Theorem 2 that we can still have a FPTAS for NASP, when N and β_{jk} are bounded by some polynomial in the size of the input, and $\mathcal{U}(\mathbf{x})$, $\forall \mathbf{x}$, can be computed in time polynomial in the size of the input. A straightforward application of Theorem 2 gives the following.

Corollary 2 *If the cost function is of the form $\mathcal{U}([x_1, x_2]^T) = x_1 \mathcal{U}_1(x_2) + \mathcal{U}_2(x_2)$, with $\mathcal{U}_1, \mathcal{U}_2$ non-negative and non-decreasing, then, for any $\varepsilon > 0$, there is an algorithm that computes an $(1+\varepsilon)$-approximation to the optimum of NASP in time $O(n^2 m \frac{\log(nC_1)}{\varepsilon})$.*

4 A FPTAS for the QoS-aware Multicommodity Flow Problem

4.1 Review of the GK Approach

A (pure) fractional packing LP is a linear program of the form $\max\{c^T x | Ax \leq b, x \geq 0\}$, where $A_{M \times N}$, $b_{M \times 1}$ and $c_{N \times 1}$ have positive entries. By scaling we also assume that $A(i,j) \leq b(i)$, $\forall i,j$. The dual of that problem is $\min\{b^T y | A^T y \geq c, y \geq 0\}$. In [12], Garg and Könemann present a remarkably simple and elegant FPTAS for solving fractional packing LPs. Their algorithm maintains a primal and a dual solution. At each step they identify the most violated constraint in the dual and increase the corresponding primal variable, and the dual variables. The most violated constraint is identified by using an exact oracle. Their 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 $D(t) \geq 1$. 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 Lemma 7 in the Appendix).

The above algorithm can be straightforwardly extended to work with an approximate oracle. Simply, in the k-th iteration we call an oracle that returns an $(1+w)$-approximation of the minimum length column of A. If q is the column returned by the oracle, then we have that $\text{length}_{y_{k-1}}(q) \leq (1+w)a(y_{k-1})$. By working similarly to [12] and choosing $\delta = (1+\varepsilon)((1+\varepsilon)M)^{-1/\varepsilon}$, we can show the following theorem (whose proof is in the Appendix for the sake of completeness).

Theorem 3 *There is an algorithm that computes an $(1-\varepsilon)^{-2}(1+w)$-approximation to the packing LP after at most $M\lceil \log_{1+\varepsilon} \frac{1+\varepsilon}{\delta} \rceil = M\lceil \frac{1}{\varepsilon} \log_{1+\varepsilon} M \rceil$ iterations, where M is the number of rows.*

The following lemma establishes an upper bound on the ratio of the lengths of the minimum length column at the start and the end of the algorithm.

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, $\frac{a(t)}{a(0)} \leq \frac{1+\varepsilon}{\delta}$.*

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 $D(t) \leq 1+\varepsilon$. Consequently, $\sum_i b(i)y_t(i) \leq 1+\varepsilon$, which implies that $y_t(i) \leq (1+\varepsilon)\frac{1}{b(i)}$, $\forall i$. Hence, $a(t) = \min_j \sum_i \frac{A(i,j)}{c(j)} y_t(i) \leq (1+\varepsilon) \cdot \min_j \sum_i \frac{A(i,j)}{c(j)b(i)} = \frac{1+\varepsilon}{\delta} a(0)$. ∎

4.2 The FPTAS for QoS-aware Multicommodity Flows

Recall the LP formulation of the QoS-aware MCF problem from Section 2.1. To obtain its dual, we introduce for each edge e a dual variable $l(e)$ that corresponds to the capacity constraint (2) on e, and for each commodity i we introduce a dual variable ϕ_i that corresponds to the demand constraint (3) on i. The dual LP becomes:

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

$$s.t. \quad l(p) + \phi_i h_i(p) \geq v_i(p), \forall i = 1 \ldots k, \forall p \in P_i \tag{7}$$

where $l(p) := \sum_{e \in p} l(e)$. It can be easily seen that the primal is a (pure) fractional packing LP. We solve this problem using the algorithm described in Section 4.1, which boils down in constructing a suitable approximate oracle to identify the most violated constraint (7) of the dual. To apply the algorithm, we have to scale the capacities and demands by $\min\{\min_{e \in E} u(e), \min_{1 \leq i \leq k} \frac{d_i}{h_i^{max}}\}$,

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 $h_i(\cdot)$, so that $u(e) \geq 1$, $\forall e \in E$, and $d_i \geq h_i(p)$, $\forall 1 \leq i \leq k$, $p \in P_i$.

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 by $a(l,\phi) = \min_{1\leq i\leq k} \min_{p\in P_i} \text{length}_{(l,\phi)}(i,p)$ we denote the length of the most violated constraint. 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 $\delta = (1+\varepsilon)((1+\varepsilon)(m+k))^{-\frac{1}{\epsilon}}$.

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 $1 \leq i \leq k$ and $q \in P_i$; i.e., we have $\text{length}_{(l,\phi)}(i',p) \leq (1+\varepsilon)a(l,\phi)$. 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 proceed to describe the oracle. Observe that 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, the above function is of the form required by Corollary 2 with $\mathbf{c} = [l, wt_i]^T$, $\mathcal{U}_1(x) = \frac{1}{\mathrm{v}_i\left(\frac{x}{\delta_i(s_i,t_i)}\right)}$ and $\mathcal{U}_2(x) = \phi_i \cdot \frac{\mathrm{h}_i\left(\frac{x}{\delta_i(s_i,t_i)}\right)}{\mathrm{v}_i\left(\frac{x}{\delta_i(s_i,t_i)}\right)}$. Consequently, for a fixed i we can 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 $q \in P_i$.

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 [8]), maintaining a lower bound estimation $\overline{a}$ of $a(l,\phi)$, initially equal to $\overline{a} = \frac{1}{1+\varepsilon}\min_{1\leq i\leq k}\{\frac{l(p_i)+\phi_i h_i(p_i)}{v_i(p_i)} | p_i = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)\}$. In each phase, the oracle examines the commodities one by one by performing NASP computations. For each commodity i the oracle returns any path $p = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)\}$ for which $\frac{l(p)+\phi_i h_i(p)}{v_i(p)} \leq \overline{a}(1+\varepsilon)^2$. It then 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. 2.

The following lemma establishes the approximation guarantee for the oracle.

Lemma 4 *A call to the oracle returns an $(1+\varepsilon)^2$-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 $\overline{a}_1 = \frac{1}{1+\varepsilon}\min_{1\leq i\leq k}\left\{\frac{l(p_i)+\phi_i h_i(p_i)}{v_i(p_i)} | p_i = \text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)\right\}$. 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 $\text{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)$ 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

```
QoS-MCF(G, u, s, t, d, wt, v, ε) {
  forall e ∈ E { l(e) = δ/u(e) }
  for i = 1 to k { φ_i = δ/d_i }
  for i = 1 to k { forall e ∈ E { X_i(e) = 0 }}
  D = (m + k)δ;
  for i = 1 to k { p_i = NASP(G, s_i, t_i, l, wt_i, ε) }
  ā = 1/(1+ε) min_{1≤i≤k} { (l(p_i)+φ_i h_i(p_i))/v_i(p_i) };
  i = 1;
  while D ≤ 1 {
    (p, i, ā) = QoS-MCF-oracle(G, s, t, l, wt, v, φ, ε, i, ā);
    Δ = min{ d_i/h_i(p), min_{e∈p} u(e) };
    X_i(p) = X_i(p) + Δ;
    forall e ∈ p do l(e) = l(e)(1 + ε Δ/u(e));
    φ_i = φ_i(1 + ε Δ h_i(p)/d_i);
    D = D + εΔ (l(p)+φ_i h_i(p))/v_i(p);
  }
  for i = 1 to k { forall e ∈ E { X_i(e) = X_i(e)/log_{1+ε} (1+ε)/δ }}
}
```

```
QoS-MCF-oracle(G, s, t, l, wt, v, φ, ε, j, ā) {
  while true {
    for i = j to k {
      p = NASP(G, s_i, t_i, l, wt_i, ε);
      if (l(p)+φ_i h_i(p))/v_i(p) ≤ ā(1 + ε)^2
        return (p, i, ā);
    }
    ā = ā(1 + ε); /* update rule for
                     next phase */
  }
}
```

Figure 2: The approximation algorithm for the QoS-MCF problem.

NASP routine imply that at the end 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* $\lceil \frac{1}{\varepsilon}\log_{1+\varepsilon}(m+k)\rceil + 2$.

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 $\overline{a}_1 = \frac{1}{1+\varepsilon}\min_{1\le i\le k}\left\{\frac{l(p_i)+\phi_i h_i(p_i)}{v_i(p_i)} | p_i = \mathrm{NASP}(G, s_i, t_i, l, wt_i, \varepsilon)\right\}$ and by the definition of the NASP routine we get that $a(0) \le (1+\varepsilon)\overline{a}_1$. From the proof of Lemma 4, we have that $\overline{a}_T \le a(t)$, and from Lemma 3 we get that $a(t) \le \frac{1+\varepsilon}{\delta}a(0)$. Combining the last three inequalities we get $\overline{a}_T \le \frac{(1+\varepsilon)^2}{\delta}\overline{a}_1$. 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 $\overline{a}_1(1+\varepsilon)^{T-1} \le \frac{(1+\varepsilon)^2}{\delta}\overline{a}_1$, which implies that $T \le \log_{1+\varepsilon}\frac{(1+\varepsilon)^3}{\delta}$. Hence, the number of phases is bounded by $\lceil\log_{1+\varepsilon}\frac{(1+\varepsilon)^3}{\delta}\rceil = \lceil\frac{1}{\varepsilon}\log_{1+\varepsilon}(m+k)\rceil + 2$, since $\delta = (1+\varepsilon)((1+\varepsilon)(m+k))^{-\frac{1}{\varepsilon}}$. ∎

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

Theorem 4 *There is an algorithm that computes an $(1-\varepsilon)^{-2}(1+\varepsilon)^2$-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 3 (with $M = m+k$) and Lemma 4 we have that the algorithm computes an $(1-\varepsilon)^{-2}(1+\varepsilon)^2$-approximation to the optimal and terminates after at most $(m+k)\lceil\log_{1+\varepsilon}(m+k)\rceil$ 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 $k\lceil\frac{1}{\varepsilon}\log_{1+\varepsilon}(m+k)\rceil+2k$ NASP computations not leading to an augmentation. Therefore, the total number of NASP computations during an execution of the algorithm is $O((m+k)\log_{1+\varepsilon}(m+k))$.

A NASP computation is carried out in time $O(\frac{1}{\varepsilon}n^2 m\log(nL))$, 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 an $(1-\varepsilon)^{-2}(1+\varepsilon)^2$-approximation to the QoS-aware MCF problem in time $O((\frac{1}{\varepsilon})^3(m+k)\log(m+k)mn^2(\frac{1}{\varepsilon}\log(m+k)+\log(nU))$, which is polynomial to the input and $\frac{1}{\varepsilon}$. ∎

4.3 FPTAS for Multicommodity Flows with QoS-guarantees

An interesting variant of the QoS-aware MCF problem concerns the case where all profits are constant (i.e., $v_i(p) = v_i$, $\forall i, p \in P_i$), and there is an upper bound on the QoS per path provided; that is, the consumption functions are now defined as:

$$h_i(p) = \begin{cases} 1 & \text{if } wt_i(p) \leq b_i \\ +\infty & \text{otherwise} \end{cases}, \quad \forall i, p \in P_i.$$

We call this the *MCF with QoS-guarantees problem* and an FPTAS can be obtained by our algorithm in Section 4.2. Its complexity, however, can be reduced by observing that instead of NASP it suffices to perform Restricted Shortest Path (RSP) computations. The currently best FPTAS for general digraphs is due to Lorenz and Raz [18], and runs in $O(mn(\log\log n + 1/\varepsilon))$ time. Following the approach of Section 4.2, but implementing the oracle using a FPTAS for RSP instead of NASP (using the algorithm in [18]), and arguing as in Theorem 4, we get the following.

Theorem 5 *There is an algorithm that computes an $(1-\varepsilon)^{-2}(1+\varepsilon)^2$-approximation to the MCF with QoS-guarantees problem in time $O((\frac{1}{\varepsilon})^2(m+k)\log(m+k)mn(\log\log n+1/\varepsilon))$, where n is the number of nodes, m is the number of edges, and k is the number of commodities.*

5 Conclusions

We considered an important generalization of the weighted multicommodity flow problem with elastic demands and profits. We exhibited an interesting connection between the efficient approximate (FPTAS) solution of that problem and multiobjective shortest paths, which are of independent interest. In addition, we provided efficient FPTAS for multiobjective shortest paths and several variants of them that improve upon previous approaches.

Acknowledgments. We are indebted to Naveen Garg, Jochen Könemann, Spyros Kontogiannis, and Frank Wagner for various useful discussions.

References

[1] M. Beckmann, G. McGuire, and C. Winsten, *Studies in the Economics of Transportation*, Yale University Press, 1956.

[2] D. Bienstock, *Potential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice*, Kluwer Academic Publishers, Boston, 2002.

[3] S. Dafermos, "The General Multimodal Network Equilibrium Problem with Elastic Demands", *Networks*, 12 (1982), pp. 57-72.

[4] A. Datta, D. Vandermeer, A. Celik, and V. Kumar, "Broadcast Protocols to Support Efficient Retrieval from Databases by Mobile Users", *ACM Transactions on Database Systems*, 24:1 (1999), pp. 1-79.

[5] M. Ehrgott, *Multicriteria Optimization*, Springer, 2000.

[6] M. Ehrgott and X. Gandibleux, *Multiple Criteria Optimization - state of the art annotated bibliographic surveys*, Kluwer Academic Publishers, Boston, 2002.

[7] F. Ergun, R. Sinha and L. Zhang "An improved FPTAS for restricted shortest path", *Information Processing Letters*, 83 (2002) pp.287-291.

[8] L.K. Fleischer, "Approximating fractional multicommodity flows independent of the number of commodities", *SIAM Journal on Discrete Mathematics*, 13:4 (2000), pp. 505-520.

[9] M. Florian and S. Nguyen, "A Method for Computing Network Equilibrium with Elastic Demands", *Transportation Science*, 8 (1974), pp. 321-332.

[10] S. Gabriel and D. Bernstein, "The Traffic Equilibrium Problem with Nonadditive Path Costs", *Transportation Science* 31:4(1997), pp. 337-348.

[11] S. Gabriel and D. Bernstein, "Nonadditive Shortest Paths: Subproblems in Multi-Agent Competitive Network Models", *Computational & Mathematical Organization Theory* 6(2000), pp. 29-45.

[12] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems", in *Proc. 39th IEEE Symposium on Foundations of Computer Science* – FOCS'98, (IEEE CS Press, 1998), pp.300-309.

[13] N. Garg and J. Könemann, personal communication, 2005.

[14] A. Goel, K. G. Ramakrishnan, D. Kataria, and D. Logothetis, "Efficient Computation of Delay-Sensitive Routes from One Source to All Destinations", in *Proc. IEEE Conf. Comput. Commun.* – INFOCOM 2001.

[15] P. Hansen, "Bicriterion Path Problems", *Proc. 3rd Conf. Multiple Criteria Decision Making – Theory and Applications*, LNEMS Vol. 117 (Springer, 1979), pp. 109-127.

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

[17] T. Korkmaz and M. Kruz, "Multiconstrained optimal path selection", in *Proc. IEEE Conf. Comput. Commun.* – INFOCOM 2001, pp. 834-843.

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

[19] C. Papadimitriou and M. Yannakakis, "On the Approximability of Trade-offs and Optimal Access of Web Sources", in *Proc. 41st Symp. on Foundations of Computer Science* – FOCS 2000, pp. 86-92.

[20] S. Plotkin, D. Shmoys, and E. Tardos, "Fast Approximation Algorithms for Fractional Packing and Covering Problems", *Mathematics of Operations Research* 20 (1995), pp. 257-301.

[21] G. Tsaggouris and C. Zaroliagis, "Non-Additive Shortest Paths", in *Algorithms* – ESA 2004, LNCS Vol. 3221 (Springer-Verlag, 2004), pp. 822-834.

[22] S. Vassilvitskii and M. Yannakakis, "Efficiently Computing Succinct Trade-off Curves", in *Automata, Languages, and Programming* – ICALP 2004, LNCS Vol. 3142 (Springer, 2004), pp. 1201-1213.

[23] F. Wagner, "Challenging Optimization Problems at Deutsche Bahn", AMORE Workshop (invited talk), 1999.

[24] F. Wagner (Deutsche Bahn AG), personal communication, 2004.

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

[26] N. Young, "Sequential and Parallel Algorithms for Mixed Packing and Covering", in *Proc. 42nd IEEE Symp. on Foundations of Computer Science* – FOCS 2001, pp. 538-546.

A APPENDIX

Proof of Theorem 3

The analysis is straightforward from [12]. We only consider an approximate oracle. In order to prove Theorem 3 we need the next two lemmata. In the first lemma, we establish a bound on the ratio of the optimal dual value to the primal objective value at the end of the algorithm.

Lemma 6 *Let $\beta = \min_y \frac{D(y)}{a(y)}$ be the optimal dual value and let t be the last iteration of the algorithm. The ratio of the optimal dual value to the primal objective value at the end of the algorithm is bounded by $\frac{\varepsilon(1+w)}{\ln(1/M\delta)}$.*

Proof. For each iteration $k \geq 1$ it is

$$\begin{aligned} D(k) &= \sum_i b(i)y_k(i) \\ &= \sum_i b(i)y_{k-1}(i) + \varepsilon\frac{b(p)}{A(p,q)}\sum_i A(i,q)y_{k-1}(i) \\ &\leq D(k-1) + (1+w)\varepsilon(f_k - f_{k-1})a(k-1) \end{aligned}$$

which implies that

$$D(k) \leq D(0) + (1+w)\varepsilon\sum_{l=1}^{k}(f_l - f_{l-1})a(l-1).$$

Since $\beta = \min_y D(y)/a(y)$ it is $\beta \leq D(l-1)/a(l-1), \forall l = 1 \ldots k$, and thus

$$D(k) \leq M\delta + \frac{(1+w)\varepsilon}{\beta}\sum_{l=1}^{k}(f_l - f_{l-1})D(l-1).$$

Observe now that for fixed k, this right hand side is maximized by setting $D(l-1)$ to its maximum possible value for all $1 \leq l-1 < k$, and let us denote this maximum value by $D'(k)$, i.e.,

$$D'(k) = M\delta + \frac{(1+w)\varepsilon}{\beta}\sum_{l=1}^{k}(f_l - f_{l-1})D'(l-1).$$

Consequently,

$$\begin{aligned} D(k) &\leq D'(k) \\ &= D'(k-1) + \frac{(1+w)\varepsilon}{\beta}(f_k - f_{k-1})D'(k-1) \\ &= D'(k-1)\left(1 + \frac{(1+w)\varepsilon}{\beta}(f_k - f_{k-1})\right) \\ &\leq D'(k-1)e^{\frac{(1+w)\varepsilon}{\beta}(f_k - f_{k-1})} \\ &\leq D'(0)e^{\frac{(1+w)\varepsilon}{\beta}\sum_{l=1}^{k}(f_l - f_{l-1})} \\ &\leq D'(0)e^{\frac{(1+w)\varepsilon}{\beta}(f_k - f_0)} \end{aligned}$$

Since now $D'(0) = M\delta$ and $f_0 = 0$ it follows that

$$D(k) \leq M\delta e^{(1+w)\varepsilon f_k/\beta}.$$

From our stopping condition it is $1 \leq D(t) \leq M\delta e^{(1+w)\varepsilon f_t/\beta}$ and hence

$$\frac{\beta}{f_t} \leq \frac{\varepsilon(1+w)}{\ln(1/M\delta)}.$$

∎

The final primal solution constructed may not be feasible since some of the packing constraints may be violated. The second lemma shows that the final primal assignment can be appropriately scaled as to obtain a feasible solution.

Lemma 7 *Scaling the final primal assignment by* $\log_{1+\varepsilon}\left(\frac{1+\varepsilon}{\delta}\right)$*, we obtain a feasible solution to the fractional packing LP.*

Proof. When we pick a column q and increase the left-hand-side of the i-th constraint by $\frac{A(i,q)b(p)}{A(p,q)b(i)}$. Simultaneously we increase the dual variable $y(i)$ by a multiplicative factor of $1 + \varepsilon\frac{A(i,q)b(p)}{A(p,q)b(i)}$. By the definition of p it follows that $\frac{A(i,q)b(p)}{A(p,q)b(i)} \leq 1$ and thus increasing the left-hand-side of the i-th constraint by one causes an increase in $y(i)$ by a multiplicative factor of $1 + \varepsilon$. Since t is the first iteration for which $D(t) > 1$, it is $y_{t-1}(i) < 1/b(i)$ and thus $y_t(i) < (1+\varepsilon)/b(i)$. Since now $y_0(i) < \delta/b(i)$ it follows that the left-hand-side of the i-th constraint is no more than $\log_{1+\varepsilon}\left(\frac{1+\varepsilon}{\delta}\right)$ for any i. Thus scaling the primal solution by $\log_{1+\varepsilon}\left(\frac{1+\varepsilon}{\delta}\right)$ gives a feasible solution. ∎

We now proceed with the proof of Theorem 3.

Proof of Theorem 3: In the k-th iteration we increase the dual variable of the "minimum capacity" row by a factor of $1 + \varepsilon$. Since we stop the algorithm at the first iteration t such that $D(t) > 1$ it follows that $D(t) < 1 + \varepsilon$ and thus $y_t(i) < \frac{1+\varepsilon}{b(i)}$ for any row. Since now $y_0(i) = \frac{\delta}{b(i)}$ and $y_t(i) < \frac{1+\varepsilon}{b(i)}$ and there are M rows the total number of iterations is at most $M\lceil\log_{1+\varepsilon}\frac{1+\varepsilon}{\delta}\rceil = M\lceil\frac{1}{\varepsilon}\log_{1+\varepsilon}M\rceil$, by choosing $\delta = (1+\varepsilon)((1+\varepsilon)M)^{-1/\varepsilon}$.

The ratio of the optimal dual value to objective value of the scaled final primal assignment is $\gamma = \frac{\beta}{f_t}\log_{1+\varepsilon}\left(\frac{1+\varepsilon}{\delta}\right)$ by substituting the bound on $\frac{\beta}{f_t}$ from Lemma 6 we get

$$\gamma \leq \frac{\varepsilon(1+w)}{\ln(1/M\delta)}\log_{1+\varepsilon}\left(\frac{1+\varepsilon}{\delta}\right) = \frac{\varepsilon(1+w)}{\ln(1+\varepsilon)}\frac{\ln\frac{1+\varepsilon}{\delta}}{\ln(1/M\delta)}$$

For $\delta = (1+\varepsilon)((1+\varepsilon)M)^{-1/\varepsilon}$, the ratio $\frac{\ln \frac{1+\varepsilon}{\delta}}{\ln(1/M\delta)}$ equals $(1-\varepsilon)^{-1}$, hence we have

$$\gamma \leq \frac{\varepsilon(1+w)}{(1-\varepsilon)\ln(1+\varepsilon)} \leq \frac{\varepsilon(1+w)}{(1-\varepsilon)(\varepsilon-\varepsilon^2/2)} \leq (1-\varepsilon)^{-2}(1+w)$$

■