Polynomial Time Approximation Algorithms for Multi-Constrained QoS Routing

Guoliang Xue, *Senior Member, IEEE*, Weiyi Zhang, Jian Tang and Krishnaiya Thulasiraman, *Fellow, IEEE*

***Abstract*— We study the multi-constrained Quality-of-Service (QoS) routing problem where one seeks to find a path from a source to a destination in the presence of $K \geq 2$ additive end-to-end QoS constraints. This problem is NP-hard and is commonly modeled using a graph with n vertices and m edges with K additive QoS parameters associated with each edge. For the case of $K = 2$, the problem has been well studied, with several provably good polynomial time approximation algorithms reported in the literature, which enforce one constraint while approximating the other. We first focus on an *optimization version* of the problem where we enforce the first constraint and approximate the other $K - 1$ constraints. We present an $O(mn \log\log\log n + mn/\epsilon)$ time $(1+\epsilon)(K-1)$-approximation algorithm and an $O(mn \log\log\log n + m(\frac{n}{\epsilon})^{K-1})$ time $(1+\epsilon)$-approximation algorithm, for any $\epsilon > 0$. *When K is reduced to 2, both algorithms produce an $(1+\epsilon)$-approximation with a time complexity better than that of the best-known algorithm designed for this special case.* We then study the *decision version* of the problem, and present an $O(m(\frac{n}{\epsilon})^{K-1})$ time algorithm which either finds a feasible solution or confirms that there does not exist a source-destination path whose first weight is bounded by the first constraint and whose every other weight is bounded by $(1-\epsilon)$ times the corresponding constraint. If there exists an $\mathcal{H}$-hop source-destination path whose first weight is bounded by the first constraint and whose every other weight is bounded by $(1-\epsilon)$ times the corresponding constraint, our algorithm finds a feasible path in $O(m(\frac{\mathcal{H}}{\epsilon})^{K-1})$ time. *This algorithm improves previous best-known algorithms with $O((m+n\log n)n/\epsilon)$ time for $K = 2$ and $O(mn(n/\epsilon)^{K-1})$ time for $K \geq 2$.*

***Index Terms*— QoS routing, multiple additive constraints, efficient approximation algorithms.**

1. INTRODUCTION

In the multi-constrained Quality-of-Service (QoS) routing problem, one seeks for a path from a source node to a destination node that satisfies multiple QoS constraints, where the constraints could be cost, delay, and reliability of the path [3], [11], [16], [17], [23]. We model the network by a directed graph with n vertices and m edges where the *vertices* represent *computers* or *routers* and the *edges* represent *communication links*. Each edge has K weights associated with it, representing the edge cost, edge delay, and edge reliability, etc. Weights on edges can be extended to weights on paths in a natural way. If the edge weights represent cost and delay, then the corresponding path weight is the summation of the weights of the edges on the path. If the edge weight represents reliability, then the corresponding path weight is the product of the weights of the edges on the path. Note that the logarithm of the product of H positive numbers is equal to the sum of the logarithms of the H positive numbers. Therefore, these QoS parameters are said to be *additive*. QoS parameters such as bandwidth are known as *bottleneck* parameters where the corresponding weight of a path is the smallest of the weights of the edges on the path [8], [23]. Problems involving bottleneck QoS constraints can be solved efficiently by considering only those edges whose weights are no less than a chosen value. Problems involving two or more additive QoS constraints have been shown to be NP-hard [23]. In this paper, we restrict our attention to the *multi-constrained path* problem (MCP) with $K \geq 2$ additive QoS parameters.

Due to its important applications, the MCP problem has been studied extensively. Most of the existing works concentrate on an optimization version of MCP for the special case of $K = 2$, known as the *delay-constrained least cost* path problem (DCLC) where the two edge weights are *cost* and *delay*, and one seeks for a least-cost path under the constraint that the delay of the path is within a given delay constraint. Warburton in [24] first developed a *fully polynomial time approximation scheme* (FPTAS) [4] for the DCLC problem on acyclic graphs. In [5], Ergun *et al.* presented an FPTAS for the case of acyclic graphs with a time complexity of $O(m(\frac{n}{\epsilon}))$. For the problem on general graphs, Hassin in [9] presented an FPTAS with a time complexity of $O(mn(\frac{n}{\epsilon})\log(\frac{n}{\epsilon}))$, where ϵ is the approximation parameter. Lorenz and Raz in [15] presented a faster FPTAS with a time complexity of $O(mn(\log\log n + 1/\epsilon))$. *All these* FPTAS*s share the following feature. Given a delay constraint $\mathcal{D}$, an approximation parameter $\epsilon > 0$, and a pair of source-destination nodes, the* FPTAS*s find a source-destination path whose delay is at most $\mathcal{D}$ and whose cost is no more than $(1+\epsilon)$ times the cost of the least-cost delay-constrained path, provided that there is a source-destination path whose delay is at most $\mathcal{D}$.* We wish to emphasize that if every source-destination path has delay greater then $\mathcal{D}$, then all these algorithms will terminate declaring that the problem is infeasible.

Chen and Nahrstedt [3] studied the decision version of the DCLC problem where we want to find a path which satisfies both the delay constraint and the cost constraint. They proposed a polynomial time heuristic algorithm based on scaling and rounding of the delay parameter so that the delay parameter of each edge is approximated by a bounded integer. For any given $\epsilon > 0$, if there is a path whose cost is within the cost constraint and whose delay is within $(1-\epsilon)$ times the delay constraint, the heuristic guarantees finding a feasible path in $O((m + n\log n)\frac{n}{\epsilon})$ time.

Guoliang Xue and Weiyi Zhang are with the Department of Computer Science and Engineering at Arizona State University, Tempe, AZ 85287-8809. Email: {xue, weiyi.zhang}@asu.edu. Jian Tang is with the Department of Computer Science at Montana State University, Bozeman, MT 59717-3880. Email: tang@cs.montana.edu. Krishnaiya Thulasiraman is with the School of Computer Science at the University of Oklahoma, Norman, OK 73019. The research of Xue, Zhang and Tang was supported in part by ARO grant W911NF-04-1-0385 and NSF grants CCF-0431167 and ANI-0312635. The research of Thulasiraman was supported in part by NSF grant ANI-0312435.

In [32], Yuan presented a *limited granularity heuristic* and a limited path heuristic for the decision version of the general MCP problem ($K \geq 2$) with a time complexity of $O(mn(\frac{n}{\epsilon})^{K-1})$. Similar to the algorithm of Chen and Nahrstedt [3], Yuan's algorithm guarantees finding a feasible path, provided that there exists a source-destination path whose first path weight is bounded by the first constraint and whose every other path weight is bounded by $(1-\epsilon)$ times the corresponding constraint.

The aforementioned works are closely related to the current paper. Our results can be viewed as improvements/extensions of the works of Hassin [9], Lorenz and Raz [15], Chen and Nahrstedt [3], and Yuan [32].

There are many other works related to this topic. Most of them deal with the MCP problem with two constraints. Goel *et al.* [7] presented an approximation algorithm for the single source all destinations *delay sensitive routes* problem. Given a delay constraint $\mathcal{D}$, an approximation parameter $\epsilon > 0$, and a source node, the algorithm finds a source-destination path for every destination node such that the delay of the path is no more than $(1+\epsilon)\mathcal{D}$ and the cost of the path is no more than the cost of the delay-constrained least cost path for that source-destination pair. In other words, if the delay-constrained (path delay bounded by $\mathcal{D}$) least cost path has a cost of $\mathcal{C}$ (note that $\mathcal{C}$ cannot be computed in polynomial time, unless P = NP), the algorithm computes a path whose cost is bounded by $\mathcal{C}$ and whose delay is bounded by $(1+\epsilon)\mathcal{D}$. The time complexity of this algorithm is $O((m + n\log n)\mathcal{H}/\epsilon)$, where $\mathcal{H}$ is the hop-count of the longest computed path. The authors of [12], [28], [29] proposed to use a linear combination of the two weights and presented simple algorithms for finding a good linear combination of the two weights. Liu *et al.* [14] proposed a *select-function-based* heuristic algorithm. Xiao *et al.* [27] presented a primal simplex approach. Orda and Sprintson [18] presented a precomputation scheme for QoS routing with two additive parameters. Guerin and Orda [8] presented efficient approximation algorithms for QoS routing with inaccurate information. Orda and Sprintson [19] presented efficient approximation algorithms for computing a pair of disjoint QoS paths.

For the general MCP problem with $K \geq 2$, Korkmaz and Krunz [13] proposed a *randomized heuristic* for the MCP problem. Van Mieghem *et al.* [21], [22] proposed a *self-adaptive multiple constraints routing algorithm*. Xue *et al.* [30] presented an FPTAS for an optimization version of the K-constrained QoS routing problem with a worst-case running time fully polynomial, but not strongly polynomial (the running time depends on the encoding size of the values of link weights). In a recent paper, Xue *et al.* [31] studied the MCP problem with $K \geq 2$ and presented an efficient K-approximation algorithm and an FPTAS. In the optimization problems studied in [31], *all K constraints are approximated, with no constraint being enforced*. This is different from the common practice for the case of $K = 2$, where one constraint is enforced, while the other constraint is approximated.

In this paper, we study an optimization version the MCP problem with $K \geq 2$ (to be called OMCP), where the first constraint is enforced while the other $K-1$ constraints are approximated. We also study the decision version of the problem, where we seek for a path which satisfies all K constraints. We make the following contributions: (1) For the DCLC problem, we present an $O(mn\log\log\log n + mn/\epsilon)$ time $(1+\epsilon)$-approximation algorithm, improving the current best $O(mn\log\log n + mn/\epsilon)$ time algorithm of [15]. (2) For the OMCP problem, we present an $(1+\epsilon)(K-1)$-approximation algorithm with a time complexity of $O(mn\log\log\log n + mn/\epsilon)$, where $\epsilon > 0$ is any given constant. We also present an $(1+\epsilon)$-approximation scheme for OMCP with a time complexity of $O(mn\log\log\log n + m(\frac{n}{\epsilon})^{K-1})$. This contribution differs from the work of Lorenz and Raz [15] in that we deal with the more general case of the MCP problem with $K \geq 2$, while Lorenz and Raz [15] deal with the DCLC problem, which is equivalent to the special case of OMCP with $K = 2$. Our work also differs from that of Xue *et al.* [31] in that we enforce one constraint and approximate the other $K-1$ constraints, while Xue *et al.* [31] approximate all K constraints without enforcing any of the constraints. (3) For the decision version of the problem, we present an $O(m(\frac{n}{\epsilon})^{K-1})$ time algorithm which either finds a feasible solution or confirms that there does not exist a source-destination path whose first path weight is bounded by the first constraint and whose every other path weight is bounded by $(1-\epsilon)$ times the corresponding constraint. If there exists an $\mathcal{H}$-hop source-destination path whose first path weight is bounded by the first constraint and whose every other path weight is bounded by $(1-\epsilon)$ times the corresponding constraint, our algorithm finds a feasible path in $O(m(\frac{\mathcal{H}}{\epsilon})^{K-1})$ time. This contribution improves previously best-known $O((m+n\log n)n/\epsilon)$ time algorithm of Chen and Nahrstedt [3] for the special case of $K = 2$ and previously best-known $O(mn(n/\epsilon)^{K-1})$ time algorithm of Yuan [32] for the general case of $K \geq 2$. In other words, our algorithm has the same performance guarantee while having a better time complexity.

The rest of this paper is organized as follows. In Section 2, we define the problems and some notations. In Section 3, we present some basic results that will be used in later sections. In Section 4, we present our improved approximation scheme for the DCLC problem. In Section 5, we present our $(1+\epsilon)(K-1)$-approximation algorithm and our FPTAS for OMCP. In Section 6, we present our algorithm for the decision version of the MCP problem. In Section 7, we present numerical results. We conclude this paper in Section 8.

2. Definitions and Notations

We use an *integer constant* $K \geq 2$ to denote the number of QoS parameters. Unless specified otherwise, all constants, functions, and variables are assumed to have *real values*. We refer readers to [4], [26] for graph theoretic notations not defined here. We refer readers to [4], [6] for definitions of "NP-hard" and other concepts in complexity theory that are not defined here. We use the symbol □ to denote the *end of the description* of a definition/lemma/theorem, and use the symbol ■ to denote the *end of the proof* of a lemma/theorem. All logarithms are base-2 logarithms.

We model a computer network by an edge weighted directed graph $G = (V, E, \vec{\omega})$, where V is the set of n *vertices*, E is

the set of m *edges*, and $\vec{\omega} = (\omega_1, \ldots, \omega_K)$ is an *edge weight vector* so that $\omega_k(e) \geq 0$ is the k^{th} *weight* of edge e, $\forall\ e \in E,\ 1 \leq k \leq K$. For a path π in G, the k^{th} *weight* of π, denoted by $\omega_k(\pi)$, is the sum of the k^{th} weights over the edges on π: $\omega_k(\pi) \triangleq \sum_{e\in\pi} \omega_k(e)$. We study the following Decision version of the Multi-Constrained Path problem (DMCP).

*Definition 2.1 (*DMCP$(G, s, t, K, \vec{W}, \vec{\omega})$*):* INSTANCE: An edge weighted directed graph $G = (V, E, \vec{\omega})$, with K nonnegative real-valued edge weights $\omega_k(e), 1 \leq k \leq K$, associated with each edge $e \in E$; a constraint vector $\vec{W} = (W_1, \ldots, W_K)$ where each W_k is a positive constant; and a source-destination node pair (s, t). QUESTION: Is there an s–t path π such that $\omega_k(\pi) \leq W_k,\ 1 \leq k \leq K$? □

In the above definition, the inequality $\omega_k(\pi) \leq W_k$ is called *the k^{th} QoS constraint*. An s–t path π satisfying all K QoS constraints is called a *feasible path* or a *feasible solution* of DMCP$(G, s, t, K, \vec{W}, \vec{\omega})$. We say that DMCP$(G, s, t, K, \vec{W}, \vec{\omega})$ is *feasible* if it has a feasible path, and *infeasible* otherwise. We may simply use DMCP to denote DMCP$(G, s, t, K, \vec{W}, \vec{\omega})$, if no confusion can be caused. The DMCP problem is known to be NP-hard [6], [23], even for the case of $K = 2$. We will also consider a *tightened version* of DMCP, defined in the following, for a given $\epsilon \in (0, 1)$.

*Definition 2.2 (*DMCP$_\epsilon(G, s, t, K, \vec{W}, \vec{\omega})$*):* INSTANCE: A constant $\epsilon \in (0, 1)$, an edge weighted directed graph $G = (V, E, \vec{\omega})$, with K nonnegative real-valued edge weights $\omega_k(e), 1 \leq k \leq K$, associated with each edge $e \in E$; a constraint vector $\vec{W} = (W_1, \ldots, W_K)$ where each W_k is a positive constant; and a source-destination node pair (s, t). QUESTION: Is there an s–t path π such that $\omega_1(\pi) \leq W_1$ and $\omega_k(\pi) \leq (1 - \epsilon)W_k,\ 2 \leq k \leq K$? □

An s–t path π satisfying the above K QoS constraints is called a *feasible path* or a *feasible solution* of DMCP$_\epsilon(G, s, t, K, \vec{W}, \vec{\omega})$. The DMCP$_\epsilon$ problem is also NP-hard, as it is equivalent to the DMCP problem.

In Section 6, we will present a polynomial time algorithm which either finds a feasible solution of DMCP or verifies that DMCP$_\epsilon$ is infeasible, for any given constant $\epsilon \in (0, 1)$.

We also study the OMCP problem, an Optimization version of MCP, which is defined in the following.

*Definition 2.3 (*OMCP$(G, s, t, K, \vec{W}, \vec{\omega})$*):* INSTANCE: An edge weighted directed graph $G = (V, E, \vec{\omega})$, with K nonnegative real-valued edge weights $\omega_k(e), 1 \leq k \leq K$, associated with each edge $e \in E$; a constraint vector $\vec{W} = (W_1, \ldots, W_K)$ where each W_k is a positive constant; and a source-destination node pair (s, t). PROBLEM: Find an s–t path π such that $\max_{2\leq k\leq K} \frac{\omega_k(\pi)}{W_k}$ is minimized, subject to the constraint $\omega_1(\pi) \leq W_1$. □

We use π^{OMCP} to denote an *optimal solution* (also called an *optimal path*) to OMCP$(G, s, t, K, \vec{W}, \vec{\omega})$, and call $\max_{2\leq k\leq K} \frac{\omega_k(\pi^{\mathsf{OMCP}})}{W_k}$ (denoted by ν^{OMCP}) the *optimal value* of OMCP$(G, s, t, K, \vec{W}, \vec{\omega})$.

A β-approximation algorithm for a minimization problem is an algorithm that, for any instance of the problem, finds a solution whose value is at most β times the optimal value of the instance, in time bounded by a polynomial in the *encoding size* of the instance [6]. $\mathcal{A}_\epsilon$ is called a *fully polynomial time approximation scheme* (FPTAS), if for any fixed $\epsilon > 0$, $\mathcal{A}_\epsilon$ is an $(1 + \epsilon)$-approximation algorithm whose running time is a polynomial in the encoding size of the instance, and in $\frac{1}{\epsilon}$.

When $K = 2$, the OMCP problem becomes the well-known Delay-Constrained Least-Cost path problem (DCLC). In this case, for an edge $e \in G$, $\omega_1(e)$ denotes the *delay* of e and $\omega_2(e)/W_2$ denotes the *cost* of e. We are seeking a least cost s–t path in G with path delay no more than W_1. Although DCLC is a special case of OMCP, in this paper, we use slightly different notations to define the DCLC problem. We will use $\delta(e)$ (rather than $\omega_1(e)$) to denote the *delay* of edge e, use $\kappa(e)$ (rather than $\omega_2(e)/W_2$) to denote the *cost* of edge e, and use $\mathcal{D}$ (rather than W_1) to denote the *delay constraint*. These notations help simplify the presentation of our algorithms for OMCP (in Section 5) where we need to transform an instance of OMCP to an instance of DCLC such that $\delta(e) = \omega_1(e)$, $\mathcal{D} = W_1$, and $\kappa(e) = \max_{2\leq k\leq K} \frac{\omega_k(e)}{W_k}$.

*Definition 2.4 (*DCLC$(G, s, t, \mathcal{D}, \delta, \kappa)$*):* INSTANCE: An edge weighted directed graph $G = (V, E, \delta, \kappa)$ where each edge $e \in E$ is associated with a nonnegative real-valued *delay* $\delta(e)$ and a nonnegative real-valued *cost* $\kappa(e)$; a positive constant $\mathcal{D}$ (the *delay constraint*); and a source-destination node pair (s, t). PROBLEM: Find an s–t path π such that $\kappa(\pi) \triangleq \sum_{e\in\pi} \kappa(e)$ is minimized, subject to the constraint $\delta(\pi) \triangleq \sum_{e\in\pi} \delta(e) \leq \mathcal{D}$. □

An s–t path π in G is called a $\mathcal{D}$-delay-constrained s–t path if $\delta(\pi) \leq \mathcal{D}$. DCLC seeks for a least-cost $\mathcal{D}$-delay-constrained s–t path, which we denote by π^{DCLC}. We also use ν^{DCLC} to denote $\kappa(\pi^{\mathsf{DCLC}})$ and call it the *optimal value* of DCLC$(G, s, t, \mathcal{D}, \delta, \kappa)$.

In Section 4, we will present an $O(mn \log\log\log n + m(\frac{n}{\epsilon}))$ time FPTAS for DCLC. In Section 5, we will present an $O(mn \log\log\log n + m(\frac{n}{\epsilon}))$ time $(1+\epsilon)(K-1)$-approximation algorithm and an $O(mn \log\log\log n + m(\frac{n}{\epsilon})^{K-1})$ time FPTAS for OMCP. Our approximation algorithm and FPTASs for OMCP$(G, s, t, K, \vec{W}, \vec{\omega})$ and for DCLC need to solve instances of the following two restricted versions of DMCP, denoted by MCPP and MCPN, respectively.

*Definition 2.5 (*MCPP$(G, s, t, K, \mathcal{D}, \mathcal{C}, \vec{\omega})$*):* INSTANCE: An edge weighted directed graph $G = (V, E, \vec{\omega})$, with K nonnegative edge weights $\omega_k(e), 1 \leq k \leq K$, associated with each edge $e \in E$ such that $\omega_k(e)$ is a **p***ositive integer* for $e \in G$ and $k = 2, \ldots, K$; a positive constant $\mathcal{D}$ and a *positive integer constant* $\mathcal{C}$; and a source-destination node pair (s, t). QUESTION: Is there an s–t path π such that $\omega_1(\pi) \leq \mathcal{D}$ and $\omega_k(\pi) \leq \mathcal{C},\ 2 \leq k \leq K$? □

*Definition 2.6 (*MCPN$(G, s, t, \mathcal{D}, \mathcal{C}, \delta, \kappa)$*):* INSTANCE: An edge weighted directed graph $G = (V, E, \delta, \kappa)$, with nonnegative real-valued weight $\delta(e)$ and **n***onnegative integer-valued weight* $\kappa(e)$ associated with each edge $e \in E$; a positive constant $\mathcal{D}$ and a *positive integer constant* $\mathcal{C}$; and a source-destination node pair (s, t). QUESTION: Is there an s–t path π such that $\delta(\pi) \leq \mathcal{D}$ and $\kappa(\pi) \leq \mathcal{C}$? □

In MCPP, the edge weights $\omega_2, \ldots, \omega_K$ all take **p***ositive integer-values*, $\mathcal{D}$ corresponds to W_1 in DMCP, and $\mathcal{C}$ corresponds to $W_2, \ldots, W_k$ (with equal values) in DMCP. In MCPN, the edge weight κ takes **n***onnegative integer-values*.

Before moving on, we use the simple network in Fig. 1 to

illustrate the concepts (DMCP, OMCP and DMCP$_\epsilon$) defined in this section. In this network an edge e has $K = 3$ QoS parameters $(\omega_1(e), \omega_2(e), \omega_3(e))$. Assume $(W_1, W_2, W_3) = (3, 5, 7)$. DMCP is infeasible, as is DMCP$_\epsilon$ for any $\epsilon \in (0, 1)$. OMCP has an optimal value $\nu^{\mathsf{OMCP}} = \frac{9}{7}$ with $s \rightarrow y \rightarrow t$ as the only optimal path. If we change the constraints to $(W_1, W_2, W_3) = (5, 6, 11)$, then all three paths are feasible for DMCP. In this case, OMCP has an optimal value of $\frac{5}{11}$, with $s \rightarrow z \rightarrow t$ as the only optimal path. The path $s \rightarrow y \rightarrow t$ is not a feasible solution of DMCP$_\epsilon$ for any $\epsilon \in (0, 1)$. The path $s \rightarrow x \rightarrow t$ is a feasible solution of DMCP$_\epsilon$ for any $\epsilon \in (0, 1/11]$. The path $s \rightarrow z \rightarrow t$ is a feasible solution of DMCP$_\epsilon$ for any $\epsilon \in (0, 6/11]$.

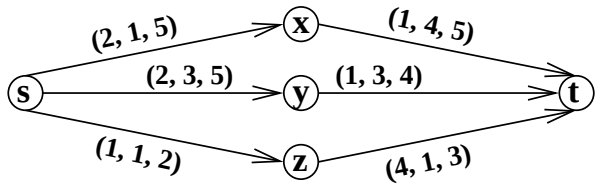


Fig. 1. A network where each edge e has 3 QoS parameters $(\omega_1(e), \omega_2(e), \omega_3(e))$. s is the source node and t is the destination node.

Note that in the problems DMCP, OMCP and DMCP$_\epsilon$, we are enforcing the first constraint $\omega_1(\pi) \leq W_1$. Therefore we assume throughout this paper that there exists an s–t path π such that $\omega_1(\pi) \leq W_1$ ($\delta(\pi) \leq \mathcal{D}$ in the case of DCLC). This condition can be verified in $O(m+n \log n)$ time by Dijkstra's shortest path algorithm using ω_1 as the metric. We also assume that $s \neq t$. Table 1 lists frequently used notations.

TABLE 1
FREQUENTLY USED NOTATIONS

K	number of additive QoS parameters
s, t	source and destination nodes
$\vec{\omega} = (\omega_1, \ldots, \omega_K)$	edge weight vector
$\vec{W} = (W_1, \ldots, W_K)$	the K QoS constraints
$\delta(e)$, $\kappa(e)$	edge delay, and edge cost (used in DCLC)
MCP	multi-constrained path problem
OMCP$(G, s, t, K, \vec{W}, \vec{\omega})$	optimization version of MCP
π^{OMCP}, ν^{OMCP}	optimal path and value of OMCP
DMCP$(G, s, t, K, \vec{W}, \vec{\omega})$	decision version of MCP
DMCP$_\epsilon(G, s, t, K, \vec{W}, \vec{\omega})$	tightened version of DMCP
DCLC$(G, s, t, \mathcal{D}, \delta, \kappa)$	the DCLC problem
π^{DCLC}, ν^{DCLC}	optimal path and value of DCLC
MCPP$(G, s, t, K, \mathcal{D}, \mathcal{C}, \vec{\omega})$	DMCP w. $\omega_k(e)$ positive integers, $k > 1$
MCPN$(G, s, t, \mathcal{D}, \mathcal{C}, \delta, \kappa)$	DMCP w. $\kappa(e)$ nonnegative integers
PseudoMCPP	algorithm for solving MCPP
PseudoMCPN	algorithm for solving MCPN
TEST$_P$	approximate testing w. PseudoMCPP
TEST$_N$	approximate testing w. PseudoMCPN

3. EXACT ALGORITHMS FOR MCPP AND MCPN, SCALING AND ROUNDING, AND APPROXIMATE TESTING

In this section, we present some building blocks that will be used in later sections. These include an $O(m\mathcal{C}^{K-1})$ time algorithm for MCPP, an $O((m + n \log n)\mathcal{C})$ time algorithm for MCPN, two scaling and rounding techniques and their corresponding polynomial time approximate testing procedures.

A. A Pseudo-Polynomial Time Algorithm for MCPP

We first present an $O(m\mathcal{C}^{K-1})$ time algorithm for MCPP$(G, s, t, K, \mathcal{D}, \mathcal{C}, \vec{\omega})$. The algorithm is named PseudoMCPP and listed as Algorithm 1. When the MCPP problem is feasible, our algorithm outputs YES, together with an s–t path π such that $\omega_1(\pi) \leq \mathcal{D}$ and $\omega_k(\pi) \leq \mathcal{C},\ 2 \leq k \leq K$. When the MCPP problem is infeasible, our algorithm outputs NO. This algorithm is well-known and has been used in [7] and [15] in the case of $K = 2$. We provide a formal presentation for the sake of completeness. It is assumed that at each vertex $v \in V$, there is a $(K-1)$-dimensional array $d[c_2, \ldots, c_K]$ (where $c_k \in \{0, 1, \ldots, \mathcal{C}\}$) which is used to hold the least delay (measured by ω_1) among s–v paths π such that $\omega_k(\pi) \leq c_k,\ 2 \leq k \leq K$. At each vertex $v \in V$, there is also a $(K-1)$-dimensional array $\pi[c_2, \ldots, c_K]$ which holds the predecessor of v on the s–v path π such that $\omega_1(\pi) = d[c_2, \ldots, c_K]$ and $\omega_k(\pi) \leq c_k,\ 2 \leq k \leq K$. In the description of the algorithm, we will use $d[v; c_2, \ldots, c_K]$ and $\pi[v; c_2, \ldots, c_K]$ to denote the two arrays at node v.

Algorithm 1 PseudoMCPP$(G, s, t, K, \mathcal{D}, \mathcal{C}, \vec{\omega})$

1: **for** $c_k := 0$ to $\mathcal{C},\ 2 \leq k \leq K$ **do**
2: $d[v; c_2, \ldots, c_K] := \infty$, $\pi[v; c_2, \ldots, c_K] :=$ null, $\forall\ v \in V$, $d[s; c_2, \ldots, c_K] := 0$.
3: **end for**
4: **for** all $(c_2, \ldots, c_K) \in \{0, 1, \ldots, \mathcal{C}\}^{K-1}$ in increasing lexicographic order **do**
5: **for** each $(u, v) \in E$ s.t. $b_k \triangleq c_k - \omega_k(u, v) \geq 0,\ 2 \leq k \leq K$ **do**
6: **if** $(d[v; c_2, \ldots, c_K] > d[u; b_2, \ldots, b_K] + \omega_1(u, v))$ **then**
7: $d[v; c_2, \ldots, c_K] := d[u; b_2, \ldots, b_K] + \omega_1(u, v)$, $\pi[v; c_2, \ldots, c_K] := u$.
8: **end if**
9: **end for**
10: **end for**
11: **if** $(d[t; \mathcal{C}, \ldots, \mathcal{C}] > \mathcal{D})$ **then**
12: output NO and stop: MCPP is infeasible.
13: **end if**
14: Find the smallest integer $c \leq \mathcal{C}$ s.t. $d[t; c, \ldots, c] \leq \mathcal{D}$.
15: output YES and an s–t path π s.t. $\omega_1(\pi) \leq \mathcal{D}$ and $\omega_k(\pi) \leq c,\ 2 \leq k \leq K$;
16: stop: Path π is feasible for MCPP.

Lemma 3.1: The worst-case time complexity of Algorithm 1 is $O(m\mathcal{C}^{K-1})$. If MCPP$(G, s, t, K, \mathcal{D}, \mathcal{C}, \vec{\omega})$ is feasible, the algorithm computes a feasible path π for MCPP$(G, s, t, K, \mathcal{D}, c, \vec{\omega})$ where c is the smallest nonnegative integer less than or equal to $\mathcal{C}$ such that MCPP$(G, s, t, K, \mathcal{D}, c, \vec{\omega})$ is feasible. □

PROOF. The for-loop in lines 1-3 takes $O(n\mathcal{C}^{K-1})$ time to initialize the table entries. The inner for-loop in lines 5-9 spends $O(m)$ time for the corresponding tuple $(c_2, \ldots, c_K) \in \{0, 1, \ldots, \mathcal{C}\}^{K-1}$, trying to update the entries $d[v; c_2, \ldots, c_K]$ and $\pi[v; c_2, \ldots, c_K]$. Hence the total time spent by the outer for-loop in lines 4-10 is bounded by $O(m\mathcal{C}^{K-1})$. The if-statement in lines 11-13 takes constant time. Lines 14-16 take $O(\mathcal{C} + n)$ time. Therefore the time complexity of Algorithm 1

is $O(m\mathcal{C}^{K-1})$. We prove the correctness of the algorithm next.

We call an s–v path π a $(c_2,\ldots,c_K)$-*constrained s–v path*, if $\omega_k(\pi) \leq c_k$ for $2 \leq k \leq K$. We call an s–v path π a $(c_2,\ldots,c_K)$-*constrained shortest s–v path*, if π is a $(c_2,\ldots,c_K)$-constrained s–v path and $\omega_1(\pi) \leq \omega_1(\pi')$ for any $(c_2,\ldots,c_K)$-constrained s–v path π'. The for-loop in lines 4-10 computes a $(c_2,\ldots,c_K)$-constrained shortest s–v path for each $v \in V$ and each $(c_2,\ldots,c_K) \in \{0,1,\ldots,\mathcal{C}\}^{K-1}$ (where the length is stored in $d[v;c_2,\ldots,c_K]$ and the predecessor of v on this path is stored in $\pi[v;c_2,\ldots,c_K]$).

Recall that in the MCPP problem, $\omega_k(e)$ is a *positive integer* for each edge e and $2 \leq k \leq K$. Therefore if π is a $(c_2,\ldots,c_K)$-constrained shortest s–v path (measured in $\omega_1(\pi)$) and that u is the predecessor of v on the path, we must have $\omega_k(\pi_u) = \omega_k(\pi) - \omega_k(u,v) < \omega_k(\pi)$ for $2 \leq k \leq K$, where π_u is the subpath of π from s to u. Since the outer for-loop in lines 4-10 loops over all $(c_2,\ldots,c_K) \in \{0,1,\ldots,\mathcal{C}\}^{K-1}$ in increasing lexicographic order, the inner for-loop in lines 5-9 correctly computes the $(c_2,\ldots,c_K)$-constrained shortest s–v path, since before the relaxation for edge (u,v) is performed (for the chosen $(c_2,\ldots,c_K)$), the $(b_2,\ldots,b_K)$-constrained shortest s–u path has already been correctly computed. Therefore $\mathsf{MCPP}(G,s,t,K,\mathcal{D},\mathcal{C},\omega)$ is feasible if and only if $d[t;\mathcal{C},\ldots,\mathcal{C}] \leq \mathcal{D}$ at the end of the outer for-loop of the algorithm. When $\mathsf{MCPP}(G,s,t,K,\mathcal{D},\mathcal{C},\omega)$ is feasible, lines 14-16 of the algorithm find the smallest integer $c \leq \mathcal{C}$ such that $\mathsf{MCPP}(G,s,t,K,\mathcal{D},c,\omega)$ is feasible and outputs the corresponding path. This proves the correctness of Algorithm 1. ■

B. A Pseudo-Polynomial Time Algorithm for MCPN

We now present an $O((m + n\log n)\mathcal{C})$ time algorithm for MCPN. The algorithm, named PseudoMCPN, is listed as Algorithm 2. When the MCPN problem is feasible, our algorithm outputs YES, together with an s–t path π such that $\delta(\pi) \leq \mathcal{D}$ and $\kappa(\pi) \leq \mathcal{C}$. When the MCPN problem is infeasible, our algorithm outputs NO. The arrays $d[v;c]$ and $\pi[v;c]$ are similarly defined as in PseudoMCPP. The separation of relaxations of edges with positive cost (lines 5-9) from relaxations of edges with zero cost (lines 10-17) makes the algorithm very fast in practice. The idea of PseudoMCPN is also well-known, and has been used in [3], [7]. We list it here for the sake of completeness. We will see later that novel combinations of PseudoMCPP and PseudoMCPN lead to our new algorithms with improved time complexities.

Lemma 3.2: The worst-case time complexity of PseudoMCPN is $O((m+n\log n)\mathcal{C})$. If $\mathsf{MCPN}(G,s,t,\mathcal{D},\mathcal{C},\delta,\kappa)$ is feasible, the algorithm computes a feasible path π for $\mathsf{MCPN}(G,s,t,\mathcal{D},c,\delta,\kappa)$ where c is the smallest non-negative integer less than or equal to $\mathcal{C}$ such that $\mathsf{MCPN}(G,s,t,\mathcal{D},c,\delta,\kappa)$ is feasible. □

PROOF. The for-loop in lines 1-3 of the algorithm spends $O(n\mathcal{C})$ time to initialize the arrays. The outer for-loop in lines 4-20 of the algorithm loops the value c from 0 to $\mathcal{C}$ to compute the table entries for $d[v;c]$ and $\pi[v;c]$. For each value of c, we first spend $O(m)$ time performing relaxations via edges (u,v) with $\kappa(u,v) > 0$. We then perform relaxations on edges (u,v) with $\kappa(u,v) = 0$. This round of relaxations is performed in a non-decreasing order of $d[u;c]$, and is accomplished in $O(m + n\log n)$ time by applying Dijkstra's algorithm on the auxiliary graph G^c. At this time, the table entries $d[v;c]$ and $\pi[v;c]$ are correctly computed. Therefore the algorithm solves the MCPN problem in $O((m+n\log n)\mathcal{C})$ time. ■

Algorithm 2 $\mathsf{PseudoMCPN}(G,s,t,\mathcal{D},\mathcal{C},\delta,\kappa)$

1: **for** $c := 0$ to $\mathcal{C}$ **do**
2: $d[v;c] := \infty, \pi[v;c] := \mathsf{null},\ \forall\ v \in V.\ d[s;c] := 0.$
3: **end for**
4: **for** $c := 0$ to $\mathcal{C}$ **do**
5: **for** each $(u,v) \in E$ s.t. $\kappa(u,v) > 0$ and $b \triangleq c - \kappa(u,v) \geq 0$ **do**
6: **if** $(d[v;c] > d[u;b] + \delta(u,v))$ **then**
7: $d[v;c] := d[u;b] + \delta(u,v)$, $\pi[v;c] := u$.
8: **end if**
9: **end for**
10: Construct an auxiliary graph $G^c(V^c,E^c,\delta)$, where V^c is the same as V, E^c contains all edges $e \in E$ such that $\kappa(e) = 0$ (with the same weight $\delta(e)$ as in G). For each $v \in V \setminus \{s\}$ such that $d[v;c] < \infty$, E^c also contains an edge (s,v) with weight $\delta(s,v) = d[v;c]$.
11: Apply Dijkstra's shortest path algorithm to compute shortest paths from s to all other nodes in G^c, with respect to weights defined below.
12: **for** all $v \in V$ **do**
13: Let $d'[v;c]$ denote the weight of the shortest s–v path in G^c. Let $\pi'[v;c]$ denote the predecessor of v on the shortest s–v path in G^c.
14: **if** $d[v;c] > d'[v;c]$ **then**
15: $d[v;c] := d'[v;c]$, $\pi[v;c] := \pi'[v;c]$.
16: **end if**
17: **end for**
18: **if** $(d[t;c] \leq \mathcal{D})$ **then**
19: output YES and the s–t path π s.t. $\delta(\pi) \leq \mathcal{D}$ and $\kappa(\pi) \leq c$; stop: Path π is feasible for MCPN.
20: **end if**
21: **end for**
22: output NO and stop: MCPN is infeasible.

C. Scaling, Rounding, Polynomial Time Approximate Testing

In this section, we describe the scaling, rounding, and polynomial time approximate testing procedure used by Hassin [9] for DCLC. We also describe a scaling, rounding and polynomial time approximate testing procedure for OMCP that was used by Lorenz and Raz [15] (for DCLC).

For a given positive real number θ and an instance of $\mathsf{DCLC}(G,s,t,\mathcal{D},\delta,\kappa)$, we construct an auxiliary graph $G^\theta_N = (V,E,\delta,\kappa^\theta_N)$ which is the same as $G = (V,E,\delta,\kappa)$ except that the edge weight κ is changed to κ^θ_N such that for each edge $e \in G$, $\kappa^\theta_N(e) = \lfloor \kappa(e)\cdot\theta \rfloor$. For given real numbers $\mathbf{C} > 0$ and $\zeta \in (0, n-1]$, we define $\mathsf{TEST}_N(\mathbf{C},\zeta) = \mathsf{YES}$ if $\mathsf{MCPN}(G^\theta_N,s,t,\mathcal{D},\lfloor\frac{n-1}{\zeta}\rfloor,\delta,\kappa^\theta_N)$ is feasible (where $\theta = \frac{n-1}{\mathbf{C}\cdot\zeta}$) and define $\mathsf{TEST}_N(\mathbf{C},\zeta) = \mathsf{NO}$ otherwise. This is the rounding and approximate testing technique used by Hassin [9]

for DCLC, and can be implemented using PseudoMCPN. Using the techniques of [9], [10], [20], one can prove the following lemma (proof in Appendix A).

Lemma 3.3: Let ν^{DCLC} be the optimal value of $\mathsf{DCLC}(G, s, t, \mathcal{D}, \delta, \kappa)$. Let $\mathbf{C}$ and ζ be two fixed positive numbers such that $0 < \zeta \leq n-1$. Then

- $\mathsf{TEST}_N(\mathbf{C}, \zeta) = \mathsf{YES}$ implies $\nu^{\mathsf{DCLC}} < \mathbf{C} \cdot (1+\zeta)$;
- $\mathsf{TEST}_N(\mathbf{C}, \zeta) = \mathsf{NO}$ implies $\nu^{\mathsf{DCLC}} > \mathbf{C}$.

$\mathsf{TEST}_N(\mathbf{C}, \zeta)$ has $O((m + n\log n)\frac{n}{\zeta})$ time complexity. □

For a given positive real number θ and an instance of $\mathsf{OMCP}(G, s, t, K, \vec{W}, \vec{\omega})$, we construct an auxiliary graph $G_P^\theta = (V, E, \vec{\omega}_P^\theta)$ which is the same as $G = (V, E, \vec{\omega})$ except that the edge weight vector $\vec{\omega}$ is changed to $\vec{\omega}_P^\theta$ such that for each $e \in G$, $\omega_{P1}^\theta(e) = \omega_1(e)$, $\omega_{Pk}^\theta(e) = \lfloor \frac{\omega_k(e)\cdot\theta}{W_k} \rfloor + 1$, $2 \leq k \leq K$. For given real numbers $\mathbf{C} > 0$ and $\zeta \in (0, n-1]$, we define $\mathsf{TEST}_P(\mathbf{C}, \zeta) = \mathsf{YES}$ if $\mathsf{MCPP}(G_P^\theta, s, t, K, \mathcal{D}, \lfloor \frac{n-1}{\zeta} \rfloor + n - 1, \vec{\omega}_P^\theta)$ is feasible (where $\theta = \frac{n-1}{\mathbf{C}\cdot\zeta}$) and define $\mathsf{TEST}_P(\mathbf{C}, \zeta) = \mathsf{NO}$ otherwise. This is a generalization of the rounding and approximate testing technique for DCLC used by Lorenz and Raz [15], and can be implemented using PseudoMCPP. Using the techniques of [15], one can prove the following lemma (proof in Appendix B).

Lemma 3.4: Let ν^{OMCP} be the optimal value of $\mathsf{OMCP}(G, s, t, K, \vec{W}, \vec{\omega})$. Let $\mathbf{C}$ and ζ be two fixed positive numbers such that $0 < \zeta \leq n-1$. Then

- $\mathsf{TEST}_P(\mathbf{C}, \zeta) = \mathsf{YES}$ implies $\nu^{\mathsf{OMCP}} < \mathbf{C} \cdot (1+\zeta)$;
- $\mathsf{TEST}_P(\mathbf{C}, \zeta) = \mathsf{NO}$ implies $\nu^{\mathsf{OMCP}} > \mathbf{C}$.

$\mathsf{TEST}_P(\mathbf{C}, \zeta)$ has $O(m\cdot(\max\{\frac{n}{\zeta}, n\})^{K-1})$ time complexity.□

When TEST_P is applied to DCLC (equivalent to the special case of OMCP with $K = 2$), we adopt the convention that $\omega_{P1}^\theta(e) = \delta(e)$, $\omega_{P2}^\theta(e) = \lfloor \kappa(e)\cdot\theta \rfloor + 1$, $\forall\, e \in E$. Note also that $\nu^{\mathsf{OMCP}} = \nu^{\mathsf{DCLC}}$ in this case. Both TEST_N and TEST_P work for any positive constant $\zeta \leq n-1$. The time complexity of $\mathsf{TEST}_N(\mathbf{C}, \zeta)$ can be made as low as $O(m + n\log n)$ by increasing the value of ζ. However, the time complexity of $\mathsf{TEST}_P(\mathbf{C}, \zeta)$ cannot be made lower than $O(mn^{K-1})$ by increasing the value of ζ. For the case of $K = 2$ in particular, if we use $\zeta = (\log n)^2$, TEST_N has a time complexity of $O(\frac{mn}{\log n})$, while TEST_P has a time complexity of $O(mn)$. If we set $\zeta \leq 1$, TEST_P has a time complexity of $O(\frac{mn}{\zeta})$, while TEST_N has a time complexity of $O(\frac{mn+n^2\log n}{\zeta})$. These observations lead to our improved FPTAS for DCLC using a novel combination of existing techniques.

4. An Improved FPTAS for DCLC

The best-known FPTAS for DCLC is due to Lorenz and Raz [15], which has a worst-case running time of $O(mn\log\log n + mn/\epsilon)$ for computing an $(1+\epsilon)$-approximation for any given $\epsilon > 0$. Their scheme first spends $O(mn\log\log n)$ time to compute a lower bound LB and an upper bound UB of ν^{DCLC} such that $\mathsf{LB} \leq \nu^{\mathsf{DCLC}} \leq \mathsf{UB} \leq 4\mathsf{LB}$. It then spends $O(mn/\epsilon)$ time to compute an $(1+\epsilon)$-approximation of DCLC by solving $\mathsf{MCPP}(G^\theta, s, t, 2, \mathcal{D}, \lfloor \frac{\mathsf{UB}\cdot(n-1)}{\mathsf{LB}\cdot\epsilon} \rfloor + n - 1, \delta, \kappa_P^\theta)$ with $\theta = \frac{n-1}{\mathsf{LB}\cdot\epsilon}$.

In this section, we use a novel combination of the techniques of Hassin [9] and the techniques of Lorenz and Raz [15] to obtain a new FPTAS for DCLC, listed as Algorithm 3 below, with a time complexity of $O(mn\log\log\log n + mn/\epsilon)$.

Algorithm 3 $\mathsf{FPTAS\text{-}DCLC}(G, s, t, \mathcal{D}, \delta, \kappa)$

1: Find the smallest $c \in \{\kappa(e) \mid e \in E\}$ such that any s–t path π with $\delta(\pi) \leq \mathcal{D}$ must contain at least one edge e with $\kappa(e) \geq c$. Set $\mathsf{LB}^{[0]} := c$, $\mathsf{UB}^{[0]} := c \times n$, Set $i := 0$.
2: Set $\zeta_N := (\log n)^2$.
3: **while** $\mathsf{UB}^{[i]} \geq 2 \times (1+\zeta_N) \times \mathsf{LB}^{[i]}$ **do**
4: Let $\mathbf{C} := \sqrt{\frac{\mathsf{LB}^{[i]}\times\mathsf{UB}^{[i]}}{1+\zeta_N}}$.
5: **if** $\mathsf{TEST}_N(\mathbf{C}, \zeta_N) = \mathsf{YES}$ **then**
6: $\mathsf{UB}^{[i+1]} := \mathbf{C}\cdot(1+\zeta_N)$, $\mathsf{LB}^{[i+1]} := \mathsf{LB}^{[i]}$.
7: **else**
8: $\mathsf{UB}^{[i+1]} := \mathsf{UB}^{[i]}$, $\mathsf{LB}^{[i+1]} := \mathbf{C}$.
9: **end if**
10: $i := i + 1$.
11: **end while**
12: Set $\zeta_P := 1$.
13: **while** $\mathsf{UB}^{[i]} \geq 2 \times (1+\zeta_P) \times \mathsf{LB}^{[i]}$ **do**
14: Let $\mathbf{C} := \sqrt{\frac{\mathsf{LB}^{[i]}\times\mathsf{UB}^{[i]}}{1+\zeta_P}}$.
15: **if** $\mathsf{TEST}_P(\mathbf{C}, \zeta_P) = \mathsf{YES}$ **then**
16: $\mathsf{UB}^{[i+1]} := \mathbf{C}\cdot(1+\zeta_P)$, $\mathsf{LB}^{[i+1]} := \mathsf{LB}^{[i]}$.
17: **else**
18: $\mathsf{UB}^{[i+1]} := \mathsf{UB}^{[i]}$, $\mathsf{LB}^{[i+1]} := \mathbf{C}$.
19: **end if**
20: $i := i + 1$.
21: **end while**
22: Set $\mathsf{LB} := \mathsf{LB}^{[i]}$, $\mathsf{UB} := \mathsf{UB}^{[i]}$.
23: Set $\theta := \frac{n-1}{\mathsf{LB}\cdot\epsilon}$. Apply Algorithm 1 to $\mathsf{MCPP}(G_P^\theta, s, t, 2, \mathcal{D}, \lfloor \frac{\mathsf{UB}(n-1)}{\mathsf{LB}\epsilon} \rfloor + n - 1, \delta, \kappa_P^\theta)$ to compute a path π.
output π as an $(1+\epsilon)$-approximation of DCLC.

The basic idea of FPTAS-DCLC is as follows. First, in line 1, we spend $O(m\log n + n(\log n)^2)$ time to obtain an initial lower bound LB and an initial upper bound UB of ν^{DCLC} such that $\mathsf{LB} \leq \nu^{\mathsf{DCLC}} \leq \mathsf{UB} \leq n\mathsf{LB}$ using the technique of Lorenz and Raz [15]. Second, in lines 2-11, we spend $O(mn)$ time to refine the pair of lower and upper bounds such that $\mathsf{LB} \leq \nu^{\mathsf{DCLC}} \leq \mathsf{UB} \leq 2(1+(\log n)^2)\mathsf{LB}$ using the technique of Hassin [9]. This is accomplished by repeated applications of TEST_N with ζ set to $(\log n)^2$). Note that Hassin [9] used $\zeta = \epsilon$ (a small value) in TEST_N, while we use $\zeta = (\log n)^2$ (a large value) in TEST_N to achieve the reduced time complexity. Note that with $\zeta = (\log n)^2$, TEST_N has a lower time complexity than TEST_P. Third, in lines 12-21, we spend $O(mn\log\log\log n)$ time to further refine the pair of lower and upper bounds such that $\mathsf{LB} \leq \nu^{\mathsf{DCLC}} \leq \mathsf{UB} \leq 4\mathsf{LB}$ using the technique of Lorenz and Raz [15]. This is accomplished by repeated applications of TEST_P with ζ set to 1. Note that with $\zeta = 1$, TEST_P has a lower time complexity than TEST_N. Finally, in lines 22-23, we spend $O(mn/\epsilon)$ time to compute an $(1+\epsilon)$-approximation of DCLC using the technique of Lorenz and Raz [15].

Theorem 4.1: For any given $\epsilon > 0$, Algorithm 3 computes an s–t path π that is an $(1+\epsilon)$-approximation of $\mathsf{DCLC}(G, s, t, \mathcal{D}, \delta, \kappa)$ in $O(mn\log\log\log n + mn/\epsilon)$ time.□

PROOF. In line 1 of the algorithm, we find the bottleneck edge cost c such that (1) there is an s–t path π with $\delta(\pi) \leq \mathcal{D}$ and that $\kappa(e) \leq c,\ \forall\ e \in \pi$; (2) any s–t path p with $\delta(p) \leq \mathcal{D}$ must contain at least one edge e with $\kappa(e) \geq c$. This can be accomplished by $O(\log m) = O(\log n)$ executions of Dijkstra's shortest path algorithm, after the edges are sorted according their cost values in $O(m \log n)$ time. Therefore the time required by line 1 is $O((m + n \log n) \log n)$, which is bounded by $O(mn)$. By definition of the bottleneck edge cost, we know that $c \leq \nu^{\mathsf{DCLC}} \leq c \cdot n$. Therefore at the end of line 1, we have

$$\mathsf{LB}^{[0]} \leq \nu^{\mathsf{DCLC}} \leq \mathsf{UB}^{[0]} \leq n \cdot \mathsf{LB}^{[0]}. \tag{4.1}$$

By the choices of ζ_N and $\mathbf{C}$ in lines 2 and 4 of the algorithm, we have $\frac{\mathbf{C}}{\mathsf{LB}^{[i]}} = \frac{\mathsf{UB}^{[i]}}{\mathbf{C}} = \sqrt{(1+\zeta_N)\frac{\mathsf{UB}^{[i]}}{\mathsf{LB}^{[i]}}}$ throughout the while-loop in lines 3-11. It follows from Lemma 3.3 that $\mathsf{TEST}_N(\mathbf{C}, \zeta_N) = \mathsf{YES}$ implies that $(1+\zeta_N)\cdot\mathbf{C}$ is an upper bound of ν^{DCLC} and $\mathsf{TEST}_N(\mathbf{C}, \zeta_N) = \mathsf{NO}$ implies that $\mathbf{C}$ is a lower bound of ν^{DCLC}. Therefore we have

$$\begin{aligned}\frac{\mathsf{UB}^{[i]}}{\mathsf{LB}^{[i]}} &= (1+\zeta_N)^{\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2^i}} \cdot \left(\frac{\mathsf{UB}^{[0]}}{\mathsf{LB}^{[0]}}\right)^{\frac{1}{2^i}} \\ &\leq (1+\zeta_N)\cdot\left(\frac{\mathsf{UB}^{[0]}}{\mathsf{LB}^{[0]}}\right)^{\frac{1}{2^i}}, i = 1, 2, \ldots. \end{aligned} \tag{4.2}$$

Therefore the while-loop in lines 3-11 is executed at most $O(\log\log\frac{\mathsf{UB}^{[0]}}{\mathsf{LB}^{[0]}}) = O(\log\log n)$ times. Since each execution of $\mathsf{TEST}_N(\mathbf{C}, \zeta_N)$ takes $O((m + n\log n)\frac{n}{\zeta_N}) = O((m + n\log n)\frac{n}{(\log n)^2})$ time, the time complexity of all executions of this while-loop is bounded by $O(mn)$. Let i_N denote the value of i when Algorithm 3 enters line 12. We have

$$\begin{aligned}\mathsf{LB}^{[i_N]} \leq \nu^{\mathsf{DCLC}} \leq \mathsf{UB}^{[i_N]} &\leq 2(1+\zeta_N)\mathsf{LB}^{[i_N]} \\ &= 2(1+(\log n)^2)\mathsf{LB}^{[i_N]}. \end{aligned} \tag{4.3}$$

By the choices of ζ_P and $\mathbf{C}$ in lines 12 and 14 of the algorithm, we have $\frac{\mathbf{C}}{\mathsf{LB}^{[i]}} = \frac{\mathsf{UB}^{[i]}}{\mathbf{C}} = \sqrt{(1+\zeta_P)\frac{\mathsf{UB}^{[i]}}{\mathsf{LB}^{[i]}}}$ throughout the while-loop in lines 13-21. It follows from Lemma 3.4 that $\mathsf{TEST}_P(\mathbf{C}, \zeta_P) = \mathsf{YES}$ implies that $(1+\zeta_P)\cdot\mathbf{C}$ is an upper bound of ν^{DCLC} and $\mathsf{TEST}_P(\mathbf{C}, \zeta_P) = \mathsf{NO}$ implies that $\mathbf{C}$ is a lower bound of ν^{DCLC}. Therefore we have

$$\begin{aligned}\frac{\mathsf{UB}^{[i_N+i]}}{\mathsf{LB}^{[i_N+i]}} &= (1+\zeta_P)^{\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2^i}} \cdot \left(\frac{\mathsf{UB}^{[i_N]}}{\mathsf{LB}^{[i_N]}}\right)^{\frac{1}{2^i}} \\ &\leq (1+\zeta_P)\cdot\left(\frac{\mathsf{UB}^{[i_N]}}{\mathsf{LB}^{[i_N]}}\right)^{\frac{1}{2^i}}, i = 1, 2, \ldots. \end{aligned} \tag{4.4}$$

Therefore the while-loop in lines 13-21 is executed at most $O(\log\log\frac{\mathsf{UB}^{[i_N]}}{\mathsf{LB}^{[i_N]}}) = O(\log\log\log n)$ times. Since each execution of $\mathsf{TEST}_P(\mathbf{C}, \zeta_P)$ takes $O(mn/\zeta_P) = O(mn)$ time, the time complexity of all executions of this while-loop is bounded by $O(mn\log\log\log n)$. Let i_P denote the value of i when Algorithm 3 enters line 22. We have

$$\mathsf{LB}^{[i_P]} \leq \nu^{\mathsf{DCLC}} \leq \mathsf{UB}^{[i_P]} \leq 2(1+\zeta_P)\mathsf{LB}^{[i_P]} = 4\cdot\mathsf{LB}^{[i_P]}. \tag{4.5}$$

We will now prove that the algorithm finds a path π in line 23 of the algorithm and that π is indeed an $(1+\epsilon)$-approximation of DCLC. Note that when the algorithm enters line 23, we have

$$\mathsf{LB} \leq \nu^{\mathsf{DCLC}} \leq \mathsf{UB} \leq 4\cdot\mathsf{LB}. \tag{4.6}$$

Let π^{DCLC} denote an optimal solution of $\mathsf{DCLC}(G, s, t, \mathcal{D}, \delta, \kappa)$, i.e., π^{DCLC} is an s–t path s.t.

$$\delta(\pi^{\mathsf{DCLC}}) \leq \mathcal{D},\ \kappa(\pi^{\mathsf{DCLC}}) \leq \nu^{\mathsf{DCLC}}. \tag{4.7}$$

We have (note that the hop-count of π^{DCLC} is $|\pi^{\mathsf{DCLC}}| \leq n-1$ and that $\theta = \frac{n-1}{\mathsf{LB}\cdot\epsilon}$)

$$\begin{aligned}\kappa_P^\theta(\pi^{\mathsf{DCLC}}) &= \sum_{e\in\pi^{\mathsf{DCLC}}} (\lfloor\kappa(e)\theta\rfloor + 1) \leq n-1+\theta\cdot\sum_{e\in\pi^{\mathsf{DCLC}}}\kappa(e) \\ &= n-1+\theta\kappa(\pi^{\mathsf{DCLC}}) \leq n-1+\frac{\mathsf{UB}(n-1)}{\mathsf{LB}\cdot\epsilon}. \end{aligned} \tag{4.8}$$

Since $\kappa_P^\theta(\pi^{\mathsf{DCLC}})$ is an integer, (4.8) also implies that

$$\kappa_P^\theta(\pi^{\mathsf{DCLC}}) \leq \lfloor\frac{\mathsf{UB}(n-1)}{\mathsf{LB}\cdot\epsilon}\rfloor + n - 1. \tag{4.9}$$

This shows that π^{DCLC} is a feasible solution of $\mathsf{MCPP}(G_P^\theta, s, t, 2, \mathcal{D}, \lfloor\frac{\mathsf{UB}(n-1)}{\mathsf{LB}\epsilon}\rfloor + n - 1, \delta, \kappa_P^\theta)$. Therefore we are guaranteed to find a path π in line 23 of the algorithm. Note that π may be different from π^{DCLC}.

Next we prove that the path π found in line 23 is guaranteed to be an $(1+\epsilon)$-approximation of DCLC. Since π is computed in line 23 of the algorithm, we have

$$\delta(\pi) \leq \mathcal{D},\ \text{and}\ \kappa_P^\theta(\pi) \leq \lfloor\frac{\mathsf{UB}(n-1)}{\mathsf{LB}\cdot\epsilon}\rfloor + n - 1 \tag{4.10}$$

and (refer to lines 14-15 of Algorithm 1)

$$\kappa_P^\theta(\pi) \leq \kappa_P^\theta(\pi^{\mathsf{DCLC}}). \tag{4.11}$$

Therefore we have

$$\begin{aligned}\kappa(\pi) &\leq \frac{1}{\theta}\kappa_P^\theta(\pi) \leq \frac{1}{\theta}\kappa_P^\theta(\pi^{\mathsf{DCLC}}) \leq \frac{1}{\theta}(\theta\kappa(\pi^{\mathsf{DCLC}}) + n - 1) \\ &\leq \frac{1}{\theta}(\theta\nu^{\mathsf{DCLC}} + n - 1) \leq \nu^{\mathsf{DCLC}}(1+\epsilon). \end{aligned} \tag{4.12}$$

(4.10) and (4.12) together show that π is an $(1+\epsilon)$-approximation of DCLC. ■

5. APPROXIMATION SCHEMES FOR OMCP

In this section, we present an $O(mn\log\log\log n + mn/\epsilon)$ time $(1+\epsilon)(K-1)$-approximation algorithm and an $O(mn\log\log\log n + m(n/\epsilon)^{K-1})$ time FPTAS for $\mathsf{OMCP}(G, s, t, K, \vec{W}, \vec{\omega})$, where $\epsilon > 0$ is the approximation parameter. Our approximation algorithm is based on a transformation from the K-constrained QoS routing problem OMCP to the DCLC problem (which has two QoS parameters).

Theorem 5.1: Let an instance ℓ of $\mathsf{OMCP}(G, s, t, K, \vec{W}, \vec{\omega})$ be given by an edge weighted directed graph $G = (V, E, \vec{\omega})$, a constraint vector $\vec{W} = (W_1, \ldots, W_K)$, and a source-destination node pair (s, t). Define a corresponding instance ℓ' of $\mathsf{DCLC}(G, s, t, \mathcal{D}, \delta, \kappa)$ by the following rules:

- G and (s, t) are the same as in ℓ;
- $\mathcal{D} = W_1$;

- $\delta(e) = \omega_1(e),\ \forall\ e \in G$;
- $\kappa(e) = \max\limits_{2\le k\le K} \dfrac{\omega_k(e)}{W_k},\ \forall\ e \in G.$

Assume that the optimal value of ℓ is ν_ℓ^{OMCP} and the optimal value of ℓ' is $\nu_{\ell'}^{\mathsf{DCLC}}$. Then $\nu_{\ell'}^{\mathsf{DCLC}} \le \nu_\ell^{\mathsf{OMCP}} \cdot (K-1)$. For any given $\psi \ge 1$, an s–t path π is a ψ-approximation of $\mathsf{DCLC}(G, s, t, \mathcal{D}, \delta, \kappa)$ implies that π is a $\psi \cdot (K-1)$-approximation of $\mathsf{OMCP}(G, s, t, K, \vec{W}, \vec{\omega})$. □

PROOF. As discussed at the end of Section 2, we assume that there is an s–t path in G whose first weight is no more than W_1, as otherwise neither of the two problems has a feasible solution.

Use π_ℓ^{OMCP} to denote an optimal solution of ℓ. We have $\delta(\pi_\ell^{\mathsf{OMCP}}) = \omega_1(\pi_\ell^{\mathsf{OMCP}}) \le W_1 = \mathcal{D}$. In addition, we have

$$\omega_k(\pi_\ell^{\mathsf{OMCP}}) \triangleq \sum_{e\in\pi_\ell^{\mathsf{OMCP}}} \omega_k(e) \le \nu_\ell^{\mathsf{OMCP}} \cdot W_k,\ 2 \le k \le K. \quad (5.1)$$

Inequality (5.1) implies

$$\sum_{e\in\pi_\ell^{\mathsf{OMCP}}} \frac{\omega_k(e)}{W_k} \le \nu_\ell^{\mathsf{OMCP}},\ 2 \le k \le K. \quad (5.2)$$

Summing (5.2) over $k = 2, 3, \ldots, K$, we obtain

$$\sum_{e\in\pi_\ell^{\mathsf{OMCP}}} \sum_{k=2}^{K} \frac{\omega_k(e)}{W_k} \le (K-1) \cdot \nu_\ell^{\mathsf{OMCP}}. \quad (5.3)$$

Therefore

$$\sum_{e\in\pi_\ell^{\mathsf{OMCP}}} \max_{2\le k\le K} \frac{\omega_k(e)}{W_k} \le \sum_{e\in\pi_\ell^{\mathsf{OMCP}}} \sum_{k=2}^{K} \frac{\omega_k(e)}{W_k} \le (K-1)\nu_\ell^{\mathsf{OMCP}}. \quad (5.4)$$

Inequality (5.4) implies

$$\kappa(\pi_\ell^{\mathsf{OMCP}}) \triangleq \sum_{e\in\pi_\ell^{\mathsf{OMCP}}} \max_{2\le k\le K} \frac{\omega_k(e)}{W_k} \le (K-1) \cdot \nu_\ell^{\mathsf{OMCP}}. \quad (5.5)$$

Since $\delta(\pi_\ell^{\mathsf{OMCP}}) \le W_1 = \mathcal{D}$, π_ℓ^{OMCP} is a feasible solution to ℓ'. Therefore we have $\kappa(\pi_\ell^{\mathsf{OMCP}}) \ge \nu_{\ell'}^{\mathsf{DCLC}}$. This leads to

$$\nu_{\ell'}^{\mathsf{DCLC}} \le \kappa(\pi_\ell^{\mathsf{OMCP}}) \le (K-1) \cdot \nu_\ell^{\mathsf{OMCP}}. \quad (5.6)$$

Let a ψ-approximation of $\mathsf{DCLC}(G, s, t, \mathcal{D}, \delta, \kappa)$ be denoted by an s–t path π. Thus we have $\omega_1(\pi) = \delta(\pi) \le \mathcal{D} = W_1$ and $\kappa(\pi) \le \psi \cdot \nu_{\ell'}^{\mathsf{DCLC}}$. Using inequality (5.6), we have

$$\kappa(\pi) \le \psi\nu_{\ell'}^{\mathsf{DCLC}} \le \psi(K-1)\nu_\ell^{\mathsf{OMCP}} = \psi(K-1)\nu_\ell^{\mathsf{OMCP}}. \quad (5.7)$$

On the other hand, we have

$$\kappa(\pi) \triangleq \sum_{e\in\pi} \max_{2\le k\le K} \frac{\omega_k(e)}{W_k} \ge \max_{2\le k\le K} \sum_{e\in\pi} \frac{\omega_k(e)}{W_k} = \max_{2\le k\le K} \frac{\omega_k(\pi)}{W_k} \ge \frac{\omega_k(\pi)}{W_k},\ 2 \le k \le K. \quad (5.8)$$

Combining (5.8), the fact that $\kappa(\pi) \le \psi \cdot \nu_{\ell'}^{\mathsf{DCLC}}$, and (5.6), we know that for $2 \le k \le K$, we have

$$\omega_k(\pi) \le \kappa(\pi) W_k \le \psi\nu_{\ell'}^{\mathsf{DCLC}} W_k \le \psi(K\text{–}1)\nu_\ell^{\mathsf{OMCP}} W_k. \quad (5.9)$$

Therefore π is a $\psi \cdot (K-1)$-approximation of $\mathsf{OMCP}(G, s, t, K, \vec{W}, \vec{\omega})$. ■

Algorithm 4 $(1+\epsilon)(K-1)$-$\mathsf{Approx}(G, s, t, K, \vec{W}, \vec{\omega})$

1: Construct the instance ℓ' of DCLC corresponding to the given instance ℓ of OMCP as in Theorem 5.1.
2: Apply FPTAS-DCLC to compute an s–t path π_ϵ which is an $(1+\epsilon)$-approximation of ℓ'.
3: Output π_ϵ as an $(1+\epsilon)(K-1)$-approximation of ℓ.

Based on Theorem 5.1 and FPTAS-DCLC presented in Section 4, we present an $(1+\epsilon)(K-1)$-approximation algorithm for $\mathsf{OMCP}(G, s, t, K, \vec{W}, \vec{\omega})$, which is listed as Algorithm 4.

The basic idea of Algorithm 4 is as follows. In line 1, we spend $O(Km) = O(m)$ time to construct an instance ℓ' of DCLC corresponding to the instance ℓ of OMCP as in Theorem 5.1. In line 2, we apply FPTAS-DCLC to compute an s–t path π_ϵ which is an $(1+\epsilon)$-approximation to the newly constructed instance ℓ' of DCLC, in $O(mn \log\log\log n + \frac{mn}{\epsilon})$ time. According to Theorem 5.1, π_ϵ is a $(1+\epsilon)(K-1)$-approximation to the instance ℓ of OMCP. Therefore we have proved the following theorem.

Theorem 5.2: The path π_ϵ found by Algorithm 4 is an $(1+\epsilon)(K-1)$-approximation of $\mathsf{OMCP}(G, s, t, K, \vec{W}, \vec{\omega})$, i.e., $\omega_1(\pi_\epsilon) \le W_1$ and $\omega_k(\pi_\epsilon) \le (1+\epsilon)(K-1)\nu^{\mathsf{OMCP}} \cdot W_k,\ 2 \le k \le K$, where ν^{OMCP} is the optimal value of $\mathsf{OMCP}(G, s, t, K, \vec{W}, \vec{\omega})$. Algorithm 4 has $O(mn \log\log\log n + \frac{mn}{\epsilon})$ time complexity. □

Using the above approximation algorithm, we design an $O(mn \log\log\log n + m(\frac{n}{\epsilon})^{K-1})$ time FPTAS for OMCP, named FPTAS-OMCP, and presented as Algorithm 5.

Algorithm 5 FPTAS-OMCP$(G, s, t, K, \vec{W}, \vec{\omega})$

1: Compute a $1.5(K-1)$-approximation p of OMCP by Algorithm 4 with $\epsilon = 0.5$.
2: **if** $\omega_k(p) = 0,\ 2 \le k \le K$ **then**
3: output p and stop: p is an optimal solution of $\mathsf{OMCP}(G, s, t, K, \vec{W}, \vec{\omega})$.
4: **end if**
5: Set $\mathsf{LB} := \mathsf{LB}^{[0]} := \max_{2\le k\le K} \frac{\omega_k(p)}{1.5(K-1)W_k}$.
Set $\mathsf{UB} := \mathsf{UB}^{[0]} := \max_{2\le k\le K} \frac{\omega_k(p)}{W_k}$.
6: **if** $\mathsf{UB} \le 4 \cdot \mathsf{LB}$ **then**
7: goto line 14.
8: **else**
9: Let $\mathbf{C} := \sqrt{\frac{\mathsf{UB}\cdot\mathsf{LB}}{2}}$.
10: if $\mathsf{TEST}_P(\mathbf{C}, 1) = \mathsf{NO}$, set $\mathsf{LB} := \mathbf{C}$.
11: if $\mathsf{TEST}_P(\mathbf{C}, 1) = \mathsf{YES}$, set $\mathsf{UB} := 2\cdot\mathbf{C}$.
12: goto line 6.
13: **end if**
14: Set $\theta := \frac{n-1}{\mathsf{LB}\cdot\epsilon}$. Apply Algorithm 1 to $\mathsf{MCPP}(G_P^\theta, s, t, K, W_1, \lfloor\frac{\mathsf{UB}(n-1)}{\mathsf{LB}\epsilon}\rfloor + n - 1, \vec{\omega}_P^\theta)$ and output the corresponding feasible path π^G.

The basic idea of FPTAS-OMCP is as follows. First, in line 1, we apply Algorithm 4 with $\epsilon = 0.5$ to compute an s–t path p. According to Theorem 5.2, $\max_{2\le k\le K} \frac{\omega_k(p)}{1.5(K-1)W_k}$ is a lower bound for ν^{OMCP} and $\max_{2\le k\le K} \frac{\omega_k(p)}{W_k}$ is an upper bound for ν^{OMCP}. Second, in lines 2-4, we check to

see whether the path p is actually an optimal solution for $\mathsf{OMCP}(G, s, t, K, \vec{W}, \vec{\omega})$. If so, the algorithm stops with an optimal solution p. If not, in line 5, we set the initial lower bound of ν^{OMCP} to $\mathsf{LB}^{[0]} \equiv \max_{2\le k\le K} \frac{\omega_k(p)}{1.5(K-1)W_k}$ and the initial upper bound of ν^{OMCP} to $\mathsf{UB}^{[0]} \equiv \max_{2\le k\le K} \frac{\omega_k(p)}{W_k}$. Then, in lines 6-13, we use the approximate testing procedure TEST_P to generate a sequence of refined lower bound-upper bound pairs so that the ratio of the upper bound over the corresponding lower bound becomes sufficiently small (less than or equal to 4). Finally, in line 14, we solve an instance of MCPP to obtain an $(1+\epsilon)$-approximation of OMCP.

Following the techniques of [15] and [31], we can prove the following theorem, whose proof is left in Appendix C.

Theorem 5.3: Algorithm 5 finds an $(1+\epsilon)$-approximation of $\mathsf{OMCP}(G, s, t, K, \vec{W}, \vec{\omega})$ in $O(mn \log\log\log n + m(\frac{n}{\epsilon})^{K-1})$ time. □

6. A Fast Algorithm to Decide the Feasibility of DMCP or the Infeasibility of DMCP_ϵ

The DMCP problem has been studied by Chen and Nahrstedt [3] for the case $K = 2$ and by Yuan [32] for the case $K \ge 2$. Using the technique of scaling all but the first QoS parameters, they presented polynomial time algorithms such that, for any given constant $\epsilon > 0$, the algorithms either find a feasible path for DMCP, or conclude that DMCP_ϵ is infeasible. The algorithm of Chen and Nahrstedt has a time complexity of $O((m + n\log n)\frac{n}{\epsilon})$ (for the case $K = 2$). The algorithm of Yuan has a time complexity of $O(mn(\frac{n}{\epsilon})^{K-1})$. We present a faster algorithm for solving the same problem. Our algorithm, called FAST-DMCP, is listed in Algorithm 6.

Algorithm 6 $\mathsf{FAST\text{-}DMCP}(G, s, t, K, \vec{W}, \vec{\omega})$

1: Set $H := 1$.
2: Set $\theta := \frac{H}{\epsilon}$.
3: **if** ($\mathsf{MCPP}(G_P^\theta, s, t, K, W_1, \lfloor\frac{H}{\epsilon}\rfloor, \vec{\omega}_P^\theta)$ is feasible) **then**
4: Let π be the s–t path returned by Algorithm 1.
5: **if** $\omega_k(\pi) \le W_k,\ 2 \le k \le K$ **then**
6: output path π and stop: π is a feasible solution of $\mathsf{DMCP}(G, s, t, K, \vec{W}, \vec{\omega})$.
7: **end if**
8: **end if**
9: **if** $H < n - 1$ **then**
10: Set $H := \min\{2H, n - 1\}$, goto line 2.
11: **else**
12: stop: $\mathsf{DMCP}_\epsilon(G, s, t, K, \vec{W}, \vec{\omega})$ is infeasible.
13: **end if**

The basic idea of Algorithm 6 is based on the fact that if DMCP_ϵ has a feasible solution π with a hop count of $\mathcal{H}$, then we can compute a feasible solution of DMCP in $O(m(\frac{H}{\epsilon})^{K-1})$ time by solving an instance of MCPP obtained by scaling G with $\theta = \frac{H}{\epsilon}$, where H is any integer between $\mathcal{H}$ and $n - 1$. However, we do not know the exact value of $\mathcal{H}$ *a priori*. Therefore the algorithm uses a parameter H to *estimate* $\mathcal{H}$. H is initially set to 1 and is doubled every time we find that it is not large enough.

The doubling technique has been used by Goel *et al.* [7] in their delay-scaling algorithm for computing delay-sensitive routes. The delay-scaling algorithm of [7] uses the scaling and rounding technique of Hassin [9], which leads to *nonnegative* edge weights. Therefore a straightforward extension of the delay-scaling algorithm to the case of $K \ge 2$ additive QoS parameters would lead to an algorithm with a time complexity of $O((m + n\log n)(\frac{\mathcal{H}}{\epsilon})^{K-1})$. Our algorithm has a time complexity of $O(m(\frac{\mathcal{H}}{\epsilon})^{K-1})$, since we use the scaling and rounding technique of Lorenz and Raz [15], which leads to *positive* edge weights. It is the combination of the doubling technique of [7] and the scaling and rounding technique of [15] that leads to the reduced time complexity.

Theorem 6.1: FAST-DMCP either finds a feasible path of DMCP or confirms the infeasibility of DMCP_ϵ, in $O(m(\frac{n}{\epsilon})^{K-1})$ worst-case time. If DMCP_ϵ has an $\mathcal{H}$-hop feasible path, then FAST-DMCP finds a feasible path for DMCP in $O(m(\frac{\mathcal{H}}{\epsilon})^{K-1})$ time. □

PROOF. Let us first analyze the time complexity of Algorithm 6, which is dominated by the solution of MCPP instances in line 3. Suppose that FAST-DMCP solves t instances of MCPP using Algorithm 1, with H taking the values $1 = H_1 < H_2 < \cdots < H_t$. Then $H_j = 2^{j-1}$ for $j = 1, 2, \ldots, t-1$ and $H_t = \min\{2^t, n-1\}$. Therefore the total time required for these t calls to Algorithm 1 is bounded by (recall that $K \ge 2$ is a constant)

$$O(m(\frac{H_1}{\epsilon})^{K-1}) + \cdots + O(m(\frac{H_t}{\epsilon})^{K-1}) = O(m(\frac{H_t}{\epsilon})^{K-1}). \quad (6.1)$$

Since $H_t \le n - 1$, Algorithm 6 stops either in line 6 or in line 12, with a worst-case time complexity of $O(m(\frac{H_t}{\epsilon})^{K-1})$, which is bounded by $O(m(\frac{n}{\epsilon})^{K-1})$.

In the following, we will prove that if DMCP_ϵ has a feasible solution with $\mathcal{H}$ hops, then Algorithm 6 finds a feasible solution π of DMCP and stops in line 6 with $H = H_t$ such that $H_t < 2\mathcal{H}$. Let π^{opt} be a feasible solution of DMCP_ϵ which has $\mathcal{H}$ hops. We must have

$$\omega_1(\pi^{opt}) \le W_1,\ \omega_k(\pi^{opt}) \le (1-\epsilon)W_k,\ 2 \le k \le K. \quad (6.2)$$

Due to the condition checking in line 5, we know that if Algorithm 6 stops in line 6, the computed path π is a feasible solution of DMCP. Therefore if $H_t < \mathcal{H}$, Algorithm 6 must have found a feasible solution π of DMCP with running time bounded by $O(m(\frac{H_t}{\epsilon})^{K-1}) = O(m(\frac{\mathcal{H}}{\epsilon})^{K-1})$. Assume that we enter line 2 with $H \ge \mathcal{H}$. We will prove that π^{opt} is a feasible solution of the instance of $\mathsf{MCPP}(G_P^\theta, s, t, K, W_1, \lfloor\frac{H}{\epsilon}\rfloor, \vec{\omega}_P^\theta)$ and that the path π (which could be different from π^{opt}) found in this step is a feasible solution of DMCP. Since $\omega_{P1}^\theta = \omega_1$, (6.2) implies

$$\omega_{P1}^\theta(\pi^{opt}) \triangleq \omega_1(\pi^{opt}) \le W_1. \quad (6.3)$$

Since the hop-count of π^{opt} is $|\pi^{opt}| = \mathcal{H} \le H$, (6.2) also implies

$$\begin{aligned}\omega_{Pk}^\theta(\pi^{opt}) &\triangleq \sum_{e\in\pi^{opt}} (\lfloor\frac{\theta\cdot\omega_k(e)}{W_k}\rfloor + 1) \le \sum_{e\in\pi^{opt}} (\frac{\theta\cdot\omega_k(e)}{W_k} + 1)\\ &= \frac{\theta\cdot\omega_k(\pi^{opt})}{W_k} + |\pi^{opt}|\\ &\le (1-\epsilon)\theta + H,\ 2 \le k \le K. \end{aligned} \quad (6.4)$$

Recall that $\theta = \frac{H}{\epsilon}$. Therefore (6.4) implies

$$\omega^{\theta}_{P_k}(\pi^{opt}) \leq (1-\epsilon)\theta + H = \frac{H}{\epsilon},\ 2 \leq k \leq K. \tag{6.5}$$

Since $\omega^{\theta}_{P_k}(\pi^{opt})$ is integer-valued, (6.5) implies

$$\omega^{\theta}_{P_k}(\pi^{opt}) \leq \lfloor \frac{H}{\epsilon} \rfloor,\ 2 \leq k \leq K. \tag{6.6}$$

(6.3) and (6.6) imply that π^{opt} is a feasible solution of MCPP$(G^{\theta}_P, s, t, K, W_1, \lfloor \frac{H}{\epsilon} \rfloor, \vec{\omega}^{\theta}_P)$. Therefore with this value of H, the algorithm is guaranteed to find a path π (which may be different from π^{opt}) in line 3, which is a feasible solution of MCPP$(G^{\theta}, s, t, K, W_1, \lfloor \frac{H}{\epsilon} \rfloor, \vec{\omega}^{\theta})$. Therefore we have

$$\omega_1(\pi) = \omega^{\theta}_{P_1}(\pi) \leq W_1 \tag{6.7}$$

and

$$\max_{2 \leq k \leq K} \omega^{\theta}_{P_k}(\pi) \leq \frac{H}{\epsilon} = \theta. \tag{6.8}$$

It follows from the definition of $\omega^{\theta}_{P_k}(e)$ that $\omega^{\theta}_{P_k}(\pi) \geq \frac{\theta}{W_k}\omega_k(\pi)$. Therefore (6.8) implies

$$\omega_k(\pi) \leq \omega^{\theta}_{P_k}(\pi)\frac{W_k}{\theta} \leq W_k,\ 2 \leq k \leq K. \tag{6.9}$$

(6.7) and (6.9) imply that π is a feasible solution of DMCP$(G, s, t, K, \vec{W}, \vec{\omega})$ and that Algorithm 6 must stop in line 6 after π is computed with $H \geq \mathcal{H}$. Since a feasible path for DMCP is guaranteed to be found when we enter line 2 with $H \geq \mathcal{H}$, we must have $H < 2 \cdot \mathcal{H}$ when this π (feasible to DMCP) is computed. Therefore the running time of the algorithm is $O(m(\frac{\mathcal{H}}{\epsilon})^{K-1})$ in this case. ∎

When DMCP$_\epsilon$ is feasible, FAST-DMCP finds a feasible solution of DMCP in $O(m(\frac{\mathcal{H}}{\epsilon})^{K-1})$ time, where $\mathcal{H}$ is the minimum hop-count of a feasible solution of DMCP$_\epsilon$. Note that in practice, $\mathcal{H}$ could be much smaller than n. Therefore FAST-DMCP runs faster than FPTAS-OMCP when DMCP$_\epsilon$ is feasible. When FAST-DMCP fails to find a feasible path to DMCP, we can infer that DMCP$_\epsilon$ is infeasible. Note that the running time of the algorithm is $O(m(\frac{n}{\epsilon})^{K-1})$ in this case.

Chen and Nahrstedt [3] (for the case of $K = 2$) and Yuan [32] (for the general case of $K \geq 2$) also presented polynomial time algorithms that are guaranteed to find a feasible solution of DMCP when DMCP$_\epsilon$ is feasible. Compared with the $O(mn/\epsilon + n^2 \log n/\epsilon)$ time algorithm of Chen and Nahrstedt [3] (for the case of $K = 2$), our algorithm is faster both when DMCP$_\epsilon(G, s, t, 2, \vec{W}, \vec{\omega})$ is feasible (with a time complexity of $O(m\mathcal{H}/\epsilon)$) and when DMCP$_\epsilon(G, s, t, 2, \vec{W}, \vec{\omega})$ is infeasible (with a time complexity of $O(mn/\epsilon)$). Compared with the $O(mn(\frac{n}{\epsilon})^{K-1})$ time algorithm of Yuan [32] (for the general case of $K \geq 2$), our algorithm is faster by a factor of n at least.

7. Numerical Results

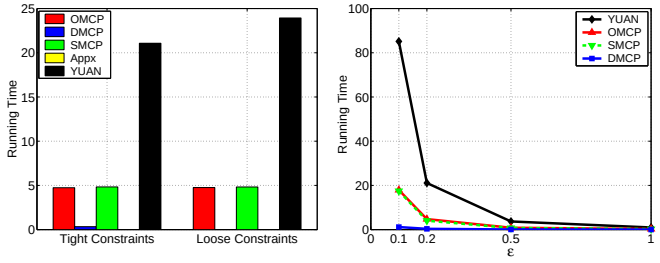
In this section, we present some numerical results to confirm our theoretical analysis. We implemented FPTAS-OMCP of this paper (denoted by OMCP in the figures), FAST-DMCP of this paper (denoted by DMCP in the figures), the $(1+\epsilon)(K-1)$-approximation algorithm of this paper (denoted by Appx in the figures), and compared them with Yuan's heuristic [32] (denoted by YUAN in the figures), as well as the FPTAS of Xue *et al.* [31] (denoted by SMCP in the figures). All tests were performed on a 2.4GHz Linux PC with 2G bytes of memory.

We used well-known Internet topologies to verify the suitability of the algorithms, and randomly generated topologies to verify the computational scalability of the algorithms. The well-known Internet topologies used for our tests are ArpaNet (20 nodes and 32 edges) and ItalianNet (33 nodes and 67 edges). These topologies can be found in Andersen *et al.* [1]. As in Xue *et al.* [31], we used BRITE, a well-known Internet topology generator [2], to generate random topologies. BRITE provides several well-known models (including the Waxman model [25]) for generating reasonable network topologies. We adopted the Waxman model to generate random networks, using the parameters provided by BRITE. In the Waxman model, nodes are randomly inserted one by one into a square field of size $1000 \times 1000\ m^2$. Let $d(u, v)$ denote the Euclidean distance between nodes u and v. The probability of having a bidirected edge (u, v) connecting nodes u and v is $\beta \times e^{-\frac{d(u,v)}{\alpha \cdot L}}$, where e is the base for natural logarithms, L is the maximum distance between two nodes, and $\alpha,\ \beta$ are two parameters in the interval $(0, 1]$. We have used $\alpha = 0.15$ and $\beta = 0.2$ (the default parameters set by BRITE). We used five different numbers of nodes: $80, 100, 120, 140, 160$. Correspondingly, BRITE generated five network topologies with the following sizes: (1) 80 nodes with 314 edges, (2) 100 nodes with 390 edges, (3) 120 nodes with 474 edges, (4) 140 nodes with 560 edges, (5) 160 nodes with 634 edges.

As in [3], [7], [12], [13], [31], [32], the edge weights were uniformly generated in a given range (we used the range $[1, 10]$). From our analysis, one should expect our algorithms to perform similarly on various edge weights. We report numerical results for the case of $K = 3$. For each network topology, and each given value of ϵ, we generated 10 source-destination pairs for testing. For each such test case (topology, ϵ, source-destination pair), we considered two scenarios: *tight constraint* and *ϵ-loose constraint*. A constraint vector (W_1, W_2, W_3) is *tight*, if (1) there is an s–t path p such that $\omega_1(p) \leq W_1$ and (2) DMCP is infeasible, i.e., there does not exist an s–t path p such that $\omega_k(p) \leq W_k,\ 1 \leq k \leq 3$. A constraint vector (W_1, W_2, W_3) is *ϵ-loose*, if DMCP$_\epsilon$ is feasible, i.e., there exists an s–t path p such that $\omega_1(p) \leq W_1$ and $\omega_k(p) \leq (1-\epsilon) \cdot W_k,\ 2 \leq k \leq 3$. We studied these two scenarios because (1) FAST-DMCP and Yuan's algorithm *guarantee* finding a feasible of DMCP if and only if the constraint vector is ϵ-loose; (2) FAST-DMCP has a time complexity $O(m(\frac{\mathcal{H}}{\epsilon})^{K-1})$ when the constraint vector is ϵ-loose and has a time complexity $O(m(\frac{n}{\epsilon})^{K-1})$ otherwise. The performance of FPTAS-OMCP, FPTAS-SMCP and Appx should be independent of the tightness or looseness of the constraint vector. Our numerical results are presented in Figs. 2 through 5, where each figure shows the average of 10 runs.

Fig. 2(a) illustrates the running times (in seconds) of the different algorithms (illustrated in the order OMCP, DMCP, SMCP, Appx, YUAN), as well as their dependency on the tightness of the constraint vector, using the case of $\epsilon = 0.2$ for ItalianNet. As expected, FAST-DMCP and Appx are always

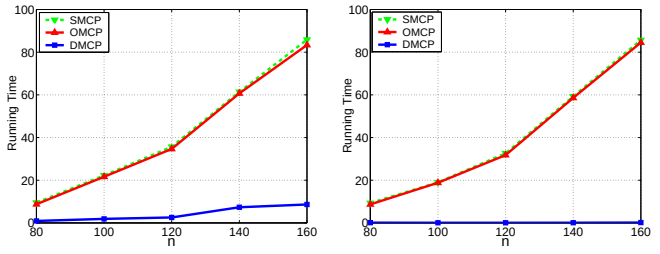
the fastest, FPTAS-OMCP and FPTAS-SMCP have similar running times, and YUAN takes the longest time. The running times of Appx, FPTAS-OMCP and FPTAS-SMCP are independent of the tightness of the constraint, while the running time of FAST-DMCP is very small for loose constraint, and is about 10% of that of FPTAS-OMCP for tight constraint. The running time of FAST-DMCP can be explained as follows. When the constraint is loose, FAST-DMCP takes $O(m(\frac{\mathcal{H}}{\epsilon})^2)$ time, where $\mathcal{H}$ is often very small. This leads to the very small running time. When the constraint is tight, time running time of FAST-DMCP is dominated by the solution of an instance of MCPP with $\mathcal{C} \approx \frac{n}{\epsilon}$, while the running time of FPTAS-OMCP is dominated by the solution of an instance of MCPP with $\mathcal{C} \approx \frac{3n}{\epsilon} + n$. Therefore the ratio of the running time of FPTAS-OMCP over that of FAST-DMCP is roughly equal to $(\frac{3n}{\epsilon} + n)^2/(\frac{n}{\epsilon})^2 = (3+\epsilon)^2 \approx 10.24$. We also observe that the running time of YUAN may increase with W slightly, but not significantly. This is due to the fact that more edge relaxations may be performed by YUAN for larger values of W.



(a) ItalianNet, time vs $\vec{W}$ (b) ItalianNet, time vs ϵ

Fig. 2. Running time vs various factors.

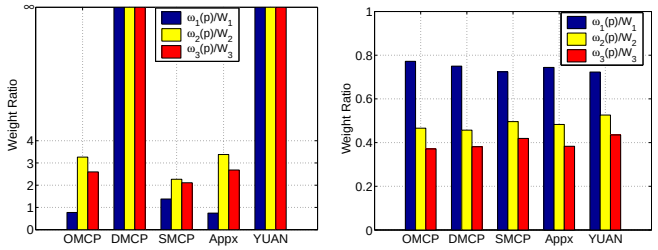
Fig. 2(b) illustrates the running times (in seconds) of FAST-DMCP, FPTAS-OMCP, FPTAS-SMCP and YUAN as functions of ϵ, using the case of tight constraint for ItalianNet. As expected, the running times increase with $\frac{1}{\epsilon}$.



(a) Time vs n, tight constraints (b) Time vs n, loose constraints

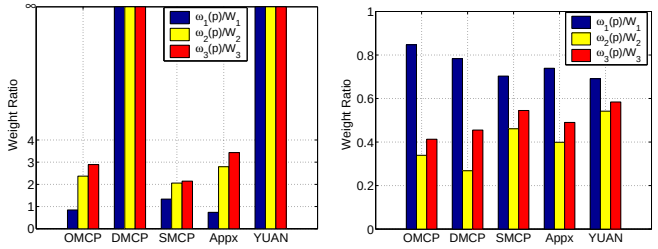
Fig. 3. Running time vs network size

To study the scalability of FAST-DMCP and FPTAS-OMCP with the network size, we also tested FAST-DMCP FPTAS-OMCP and FPTAS-SMCP on large, randomly generated network topologies as described in the second paragraph of this section. Here we have used $\epsilon = 0.5$ for the test cases. The running times of our algorithms are shown in Fig. 3(a) and Fig. 3(b). For the case of tight constraint, we observe that the running times of all three algorithms increase with the network size. FAST-DMCP required the least amount of time (but did not return any path, other than claiming that DMCP is infeasible). For the case of loose constraint, we observe that FPTAS-OMCP and FPTAS-SMCP have similar running times as in the case of tight constraint. However, FAST-DMCP is very efficient in this case. This shows that the numerical results match very well with our theoretical analysis.



(a) ArpaNet, tight constraint (b) ArpaNet, loose constraint

Fig. 4. ArpaNet, Ratio of path weight vs constraint.



(a) ItalianNet, tight constraint (b) ItalianNet, loose constraint

Fig. 5. ItalianNet, Ratio of path weight vs constraint.

Fig. 4(a) shows the vector $(\frac{\omega_1(p)}{W_1}, \frac{\omega_2(p)}{W_2}, \frac{\omega_1(p)}{W_3})$ for the path p computed by each of the five algorithms for ArpaNet with tight constraint such that DMCP is infeasible. We have used $\epsilon = 0.5$ for the cases in this figure. When an algorithm failed to find a path, we use the vector (∞, ∞, ∞) in the figure. We observe that FPTAS-OMCP, Appx and FPTAS-SMCP all found source-destination paths, while FAST-DMCP and YUAN failed to find a source-destination path. This is expected because there is no feasible solution to the decision version of the problem. We also observe that FPTAS-SMCP found paths with the minimum-maximum ratio $\omega_k(p)/W_k$ among the three constraints $1 \leq k \leq 3$, without strictly enforcing the first constraint, while both FPTAS-OMCP and Appx strictly enforce the first constraint. Fig. 4(b) shows the result for ArpaNet with loose constraint such that DMCP_ϵ is feasible. As expected, all five algorithms were able to find a source-destination path in this case. Fig. 5 shows the corresponding results on ItalianNet, where we can make similar observations.

8. Conclusions

In this paper, we have studied the multi-constrained QoS routing problem with $K \geq 2$ additive constraints. We studied an optimization version of this problem (called the OMCP problem) by approximating $K-1$ constraints, while enforcing

one of the constraints. The OMCP problem contains the well-known DCLC problem as a special case when $K = 2$. We first presented an FPTAS for DCLC of time complexity $O(mn \log\log\log n + mn/\epsilon)$, which is better than that of the best-known algorithm due to Lorenz and Raz [15]. Next we presented a theorem characterizing the performance ratio of approximating an instance of OMCP using a corresponding instance of DCLC. Based on this theorem, we presented an $(1+\epsilon)(K-1)$-approximation algorithm for OMCP with time complexity $O(mn \log\log\log n + mn/\epsilon)$, for any constant $\epsilon > 0$. We then presented an FPTAS for OMCP with a time complexity of $O(mn \log\log\log n + m(\frac{n}{\epsilon})^{K-1})$, where $\epsilon > 0$ is the approximation precision. When K is reduced to 2, our approximation algorithms become $(1+\epsilon)$-approximation algorithms for the DCLC problem. For the decision version of the problem, we presented an $O(m(\frac{n}{\epsilon})^{K-1})$ time algorithm which either finds a feasible solution or confirms that there does not exist a source-destination path whose first path weight is bounded by the first constraint and whose every other path weight is bounded by $(1-\epsilon)$ times the corresponding constraint. If there exists an $\mathcal{H}$-hop source-destination path whose first path weight is bounded by the first constraint and whose every other path weight is bounded by $(1-\epsilon)$ times the corresponding constraint, our algorithm finds a feasible path in $O(m(\frac{\mathcal{H}}{\epsilon})^{K-1})$ time. This algorithm improves previously best-known algorithms with $O((m + n\log n)n/\epsilon)$ time for $K = 2$ and $O(mn(n/\epsilon)^{K-1})$ time for $K \geq 2$.

ACKNOWLEDGMENT

We thank the associate editor and the anonymous reviewers whose comments on an earlier version of this paper have helped to significantly improve the presentation of this paper.

REFERENCES

[1] R. Andersen, F. Chung, A. Sen, G. Xue; On disjoint path pairs with wavelength continuity constraint in WDM networks; *IEEE Infocom'2004*; pp. 524–535.
[2] BRITE; http://www.cs.bu.edu/brite/.
[3] S. Chen and K. Nahrstedt; On finding multi-constrained paths; *IEEE ICC'1998*; pp. 874–879.
[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein; *Introduction to Algorithms*; second edition; McGraw Hill, 2001.
[5] F. Ergun, R. Sinha and L. Zhang; An improved FPTAS for restricted shortest path; *Information Processing Letters*; Vol. 83(2002), pp. 287–291.
[6] M.R. Garey and D.S. Johnson; *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W.H. Freeman, 1979.
[7] A. Goel, K.G. Ramakrishnan, D. Kataria and D. Logothetis; Efficient computation of delay-sensitive routes from one source to all destinations; *IEEE Infocom'2001*; pp. 854–858.
[8] R. Guerin and A. Orda; QoS routing in networks with inaccurate information: theory and algorithms; *IEEE/ACM Transactions on Networking*; Vol. 7(1999), pp. 350–364.
[9] R. Hassin; Approximation schemes for the restricted shortest path problem; *Mathematics of Operations Research*; Vol. 17(1992), pp. 36–42.
[10] O.H. Ibarra and C.E. Kim; Fast approximation algorithms for the knapsack and sum of subset problems; *Journal of the ACM*; Vol. 22(1975), pp. 463–468.
[11] J.M. Jaffe; Algorithms for finding paths with multiple constraints; *Networks*; Vol. 14(1984), pp. 95–116.
[12] A. Juttner, B. Szviatovszki, I. Mecs and Z. Rajko; Lagrange relaxation based method for the QoS routing problem; *IEEE Infocom'2001*; pp. 859–868.
[13] T. Korkmaz and M. Krunz; A randomized algorithm for finding a path subject to multiple QoS requirements; *Computer Networks*; Vol. 36(2001), pp. 251-268.
[14] W. Liu, W. Lou, and Y. Fang; An efficient quality of service routing algorithm for delay-sensitive applications; *Computer Networks*; Vol. 47(2005), pp. 87–104.
[15] D.H. Lorenz and D. Raz; A simple efficient approximation scheme for the restricted shortest path problem; *Operations Research Letters*; Vol. 28(2001), pp. 213–219.
[16] Q. Ma and P. Steenkiste; Quality-of-service routing for traffic with performance guarantees; *IWQoS'97*, May 1997.
[17] A. Orda; Routing with end-to-end QoS guarantees in broadband networks; *IEEE/ACM Transactions on Networking*; Vol. 7(1999), pp. 365–374.
[18] A. Orda and A. Sprintson; Precomputation schemes for QoS routing; *IEEE/ACM Transactions on Networking*; Vol. 11(2003), pp. 578–591.
[19] A. Orda and A. Sprintson; Efficient algorithms for computing disjoint QoS paths; *IEEE Infocom'2004*; pp. 727–738.
[20] S. Sahni; General techniques for combinatorial approximation; *Operations Research*; Vol. 25(1977), pp. 920–936.
[21] P. Van Mieghem and F.A. Kuipers, Concepts of exact QoS routing algorithms; *IEEE/ACM Transactions on Networking*; Vol. 12(2004), pp. 851–864.
[22] P. Van Mieghem, H.D. Neve and F.A. Kuipers; Hop-by-hop quality of service routing; *Computer Networks*; Vol. 37(2001), pp. 407–423.
[23] Z. Wang and J. Crowcroft; Quality-of-service routing for supporting multimedia applications; *IEEE Journal on Selected Areas in Communications*; Vol. 14(1996), pp. 1228–1234.
[24] A. Warburton; Approximation of Pareto optima in multiple-objective, shortest path problem; *Operations Research*; Vol. 35(1987), pp. 70–79.
[25] B.M. Waxman; Routing of multipoint connections; *IEEE Journal on Selected Areas in Communications*; Vol. 6(1988), pp. 1617-1622.
[26] D.B. West; *Introduction to Graph Theory*; Prentice Hall, 1996.
[27] Y. Xiao, K. Thulasiraman and G. Xue; QoS routing in communication networks: approximation algorithms based on the primal simplex method of linear programming; *IEEE Transactions on Computers*; Vol. 55(2006), pp. 815–829.
[28] G. Xue; Primal-dual algorithms for computing weight-constrained shortest paths and weight-constrained minimum spanning trees; *IEEE IPCCC'2000*; pp. 271–277.
[29] G. Xue; Minimum cost QoS multicast and unicast routing in communication networks; *IEEE Transactions on Communications*; Vol. 51(2003), pp. 817–824.
[30] G. Xue, A. Sen and R. Banka; Routing with many additive QoS constraints; *ICC'2003: IEEE International Conference on Communications*; pp. 223–227.
[31] G. Xue, A. Sen, W. Zhang, J. Tang and K. Thulasiraman; Finding a path subject to many additive QoS constraints, *IEEE/ACM Transactions on Networking*; Vol. 15(2007), pp. 201–211.
[32] X. Yuan; Heuristic algorithms for multiconstrained quality-of-service routing; *IEEE/ACM Transactions on Networking*; Vol. 10(2002), pp. 244–256.

APPENDIX

A. Proof of Lemma 3.3.

When $\mathsf{TEST}_N(\mathbf{C}, \zeta)$ returns YES, we have an s–t path π that is a feasible solution of $\mathsf{MCPN}(G_N^\theta, s, t, \mathcal{D}, \lfloor\frac{n-1}{\zeta}\rfloor, \delta, \kappa_N^\theta)$. Therefore we have

$$\delta(\pi) \leq \mathcal{D},\ \kappa_N^\theta(\pi) \leq \lfloor\frac{n-1}{\zeta}\rfloor \leq \frac{n-1}{\zeta}. \tag{A.1}$$

Recall that $\theta = \frac{n-1}{\mathbf{C}\cdot\zeta}$ and $\kappa_N^\theta(e) = \lfloor\kappa(e)\cdot\theta\rfloor > \kappa(e)\cdot\theta - 1$, $\forall\ e \in E$. Hence $\kappa_N^\theta(\pi) > \theta\cdot\kappa(\pi) - (n-1)$, since π has at most $n-1$ hops. Therefore we have

$$\frac{n-1}{\zeta} \geq \frac{n-1}{\mathbf{C}\cdot\zeta}\kappa(\pi) - (n-1). \tag{A.2}$$

This leads to

$$\kappa(\pi) \leq (1+\zeta)\mathbf{C}. \tag{A.3}$$

Therefore $\nu^{\mathsf{DCLC}} \leq (1+\zeta)\mathbf{C}$. Recall that ν^{DCLC} is the optimal value of DCLC.

Now assume that $\mathsf{TEST}_N(\mathbf{C},\zeta) = \mathsf{NO}$. This means that for *any* s–t path π, $\delta(\pi) \leq \mathcal{D}$ implies

$$\kappa_N^\theta(\pi) > \lfloor \frac{n-1}{\zeta} \rfloor. \tag{A.4}$$

Since $\kappa_N^\theta(\pi)$ is an integer, we have

$$\kappa_N^\theta(\pi) \geq \lfloor \frac{n-1}{\zeta} \rfloor + 1 > \frac{n-1}{\zeta}. \tag{A.5}$$

On the other hand, we have $\kappa_N^\theta(\pi) \leq \theta \cdot \kappa(\pi)$. Therefore we have

$$\frac{n-1}{\mathbf{C}\cdot\zeta} \cdot \kappa(\pi) > \frac{n-1}{\zeta}, \tag{A.6}$$

which implies $\kappa(\pi) > \mathbf{C}$. Given the arbitrariness of π, we conclude that $\nu^{\mathsf{DCLC}} > \mathbf{C}$. ∎

B. Proof of Lemma 3.4.

When $\mathsf{TEST}_P(\mathbf{C},\zeta)$ returns YES, we have an s–t path π that is a feasible solution of $\mathsf{MCPP}(G_P^\theta, s, t, K, \mathcal{D}, \lfloor \frac{n-1}{\zeta} \rfloor + n - 1, \vec{\omega}_P^\theta)$. Therefore we have $\omega(\pi) \leq \mathcal{D} = W_1$ and

$$\omega_{P_k}^\theta(\pi) \leq \lfloor \frac{n-1}{\zeta} \rfloor + n - 1,\ 2 \leq k \leq K. \tag{B.1}$$

Recall that $\theta = \frac{n-1}{\mathbf{C}\cdot\zeta}$ and $\omega_{P_k}^\theta(e) = \lfloor \frac{\kappa(e)\cdot\theta}{W_k} \rfloor + 1 > \frac{\kappa(e)\cdot\theta}{W_k}$, $\forall\, e \in E$. Hence $\omega_{P_k}^\theta(\pi) > \frac{\theta\cdot\kappa(\pi)}{W_k}$, $2 \leq k \leq K$. Therefore we have

$$\frac{n-1}{\zeta} + n - 1 \geq \frac{n-1}{\mathbf{C}\cdot\zeta} \cdot \frac{\omega_k(\pi)}{W_k},\ 2 \leq k \leq K. \tag{B.2}$$

This leads to

$$\omega_k(\pi) \leq (1+\zeta)\mathbf{C}\cdot W_k,\ 2 \leq k \leq K. \tag{B.3}$$

Therefore $\nu^{\mathsf{OMCP}} \leq (1+\zeta)\mathbf{C}$. Recall that ν^{OMCP} is the optimal value of OMCP.

Now assume that $\mathsf{TEST}_P(\mathbf{C},\zeta) = \mathsf{NO}$. This means that for *any* s–t path π, $\omega_1(\pi) \leq W_1$ implies

$$\omega_{P_k}^\theta(\pi) > \lfloor \frac{n-1}{\zeta} \rfloor + n - 1. \tag{B.4}$$

Since $\omega_{P_k}^\theta(\pi)$ is an integer ($2 \leq k \leq K$), we have

$$\omega_{P_k}^\theta(\pi) > \frac{n-1}{\zeta} + n - 1,\ 2 \leq k \leq K. \tag{B.5}$$

On the other hand, we have $\omega_{P_k}^\theta(\pi) \leq \frac{\theta\cdot\kappa(\pi)}{W_k} + n - 1$, since π has at most $n-1$ hops. Therefore we have

$$n - 1 + \frac{n-1}{\mathbf{C}\cdot\zeta} \cdot \frac{\kappa(\pi)}{W_k} > \frac{n-1}{\zeta} + n - 1,\ 2 \leq k \leq K. \tag{B.6}$$

which implies $\omega_k(\pi) > \mathbf{C}\cdot W_k$, $2 \leq k \leq K$. Given the arbitrariness of π, we conclude that $\nu^{\mathsf{OMCP}} > \mathbf{C}$. ∎

C. Proof of Theorem 5.3.

Let $\{\mathsf{LB}^{[i]}\}$ and $\{\mathsf{UB}^{[i]}\}$ denote the sequences of lower bounds and upper bounds generated by the algorithm. We know that

$$\mathsf{LB}^{[i]} \leq \nu^{\mathsf{OMCP}} \leq \mathsf{UB}^{[i]} \tag{C.1}$$

is true for $i = 0$. Assume that (C.1) is true for $i = l \geq 0$. If $\mathsf{TEST}_P(\sqrt{\frac{\mathsf{LB}^{[l]}\cdot\mathsf{UB}^{[l]}}{2}}, 1) = \mathsf{NO}$, we set $\mathsf{LB}^{[l+1]} := \sqrt{\frac{\mathsf{LB}^{[l]}\cdot\mathsf{UB}^{[l]}}{2}}$ and $\mathsf{UB}^{[l+1]} := \mathsf{UB}^{[l]}$. If $\mathsf{TEST}_P(\sqrt{\frac{\mathsf{LB}^{[l]}\cdot\mathsf{UB}^{[l]}}{2}}, 1) = \mathsf{YES}$, we set $\mathsf{LB}^{[l+1]} := \mathsf{LB}^{[l]}$ and $\mathsf{UB}^{[l+1]} := 2\cdot\sqrt{\frac{\mathsf{LB}^{[l]}\cdot\mathsf{UB}^{[l]}}{2}}$. It follows from Lemma 3.4 that (C.1) is also true for $i = l+1$. Also from the definition of the sequences $\{\mathsf{LB}^{[i]}\}$ and $\{\mathsf{UB}^{[i]}\}$, we have

$$\begin{aligned}\frac{\mathsf{UB}^{[i]}}{\mathsf{LB}^{[i]}} &= 2^{\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2^i}} \cdot \left(\frac{\mathsf{UB}^{[0]}}{\mathsf{LB}^{[0]}}\right)^{\frac{1}{2^i}} \\ &\leq 2 \cdot \left(\frac{\mathsf{UB}^{[0]}}{\mathsf{LB}^{[0]}}\right)^{\frac{1}{2^i}},\ i = 1, 2, \ldots.\end{aligned} \tag{C.2}$$

Therefore the condition in line 6 can be false (hence the statements in lines 9-12 be executed) for no more than $\lceil \log(\log \mathsf{UB}^{[0]} - \log \mathsf{LB}^{[0]}) \rceil$ times. However, $\log \mathsf{UB}^{[0]} - \log \mathsf{LB}^{[0]} \leq \log(1.5\cdot(K-1)) = O(1)$ according to Theorem 5.2. As a result, the worst case running time required by lines 2-13 of the algorithm is bounded by $O(mn^{K-1})$.

In the rest of this proof, we will use θ to denote $\frac{n-1}{\mathsf{LB}\cdot\epsilon}$ (as in line 14) to simplify notations within the proof. Let π^{OMCP} denote an optimal solution of $\mathsf{OMCP}(G, s, t, K, \vec{W}, \vec{\omega})$, i.e., π^{OMCP} is an s–t path such that $\omega_1(\pi^{\mathsf{OMCP}}) \leq W_1$ and $\omega_k(\pi^{\mathsf{OMCP}}) \leq \nu^{\mathsf{OMCP}}\cdot W_k$ for $k = 2, \ldots, K$. Since $\omega_{P_k}^\theta(e) = \lfloor \omega_k(e)\cdot\frac{n-1}{\mathsf{LB}\cdot W_k\cdot\epsilon} \rfloor + 1 \leq \omega_k(e)\cdot\frac{n-1}{\mathsf{LB}\cdot W_k\cdot\epsilon} + 1$ for every edge $e \in E$, we have (noting that π^{OMCP} has at most $n-1$ edges)

$$\begin{aligned}\omega_{P_k}^\theta(\pi^{\mathsf{OMCP}}) &\leq \omega_k(\pi^{\mathsf{OMCP}}) \cdot \frac{n-1}{\mathsf{LB}\cdot W_k\cdot\epsilon} + n - 1 \\ &\leq \nu^{\mathsf{OMCP}} \cdot \frac{n-1}{\mathsf{LB}\cdot\epsilon} + n - 1 \\ &\leq \frac{\mathsf{UB}(n-1)}{\mathsf{LB}\epsilon} + n - 1,\ 2 \leq k \leq K.\end{aligned} \tag{C.3}$$

Since $\omega_{P_k}^\theta$ (for $k = 2, \ldots, K$) always have integer values, (C.3) implies

$$\omega_{P_k}^\theta(\pi^{\mathsf{OMCP}}) \leq \lfloor \frac{\mathsf{UB}(n-1)}{\mathsf{LB}\epsilon} \rfloor + n - 1,\ 2 \leq k \leq K. \tag{C.4}$$

This implies that π^{OMCP} is a feasible solution of $\mathsf{MCPP}(G_P^\theta, s, t, K, W_1, \lfloor \frac{\mathsf{UB}(n-1)}{\mathsf{LB}\epsilon} \rfloor + n - 1, \vec{\omega}_P^\theta)$. Therefore line 14 of the algorithm is guaranteed to find a feasible path. Note that (C.3) also implies

$$\max_{2\leq k\leq K} \omega_{P_k}^\theta(\pi^{\mathsf{OMCP}}) \leq \nu^{\mathsf{OMCP}} \cdot \frac{n-1}{\mathsf{LB}\cdot\epsilon} + n - 1. \tag{C.5}$$

Let π^G be the s–t path found in line 14 of the algorithm. It follows from Lemma 3.1 that π^G is a feasible solution of $\mathsf{MCPP}(G_P^\theta, s, t, K, W_1, c, \vec{\omega}_P^\theta)$, where c is the smallest integer less than or equal to $\lfloor \frac{\mathsf{UB}(n-1)}{\mathsf{LB}\epsilon} \rfloor + n - 1$ such that $\mathsf{MCPP}(G_P^\theta, s, t, K, W_1, c, \vec{\omega}_P^\theta)$ is feasible. Since π^G is *optimal* for the instance of MCPP while π^{OMCP} is only *feasible*

for the same instance, the maximum path weight of π^G cannot exceed the maximum path weight of π^{OMCP}:

$$\max_{2\le k\le K} \omega^{\theta}_{P_k}(\pi^G) \le \max_{2\le k\le K} \omega^{\theta}_{P_k}(\pi^{\mathsf{OMCP}}). \quad \text{(C.6)}$$

Combining (C.6) with (C.5), we obtain

$$\max_{2\le k\le K} \omega^{\theta}_{P_k}(\pi^G) \le \nu^{\mathsf{OMCP}} \cdot \frac{n-1}{\mathsf{LB}\cdot\epsilon} + n - 1. \quad \text{(C.7)}$$

On the other hand, we also have

$$\begin{aligned}\omega^{\theta}_{P_k}(\pi^G) &= \sum_{e\in p^G} \omega^{\theta}_{P_k}(e) \ge \sum_{e\in\pi^G} \frac{\omega_k(e)\cdot(n-1)}{\mathsf{LB}\cdot W_k\cdot\epsilon} \\ &= \omega_k(\pi^G)\cdot\frac{n-1}{\mathsf{LB}\cdot W_k\cdot\epsilon},\ 2\le k\le K. \quad \text{(C.8)}\end{aligned}$$

Combining (C.7) and (C.8), we have

$$\omega_k(\pi^G)\cdot\frac{n-1}{\mathsf{LB}\cdot W_k\cdot\epsilon} \le \nu^{\mathsf{OMCP}}\cdot\frac{n-1}{\mathsf{LB}\cdot\epsilon} + n - 1,\ 2\le k\le K. \quad \text{(C.9)}$$

Some algebraic manipulations on (C.9) yield the following.

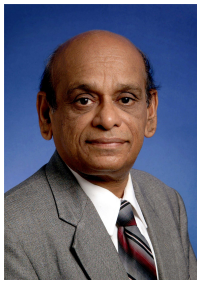
$$\begin{aligned}\omega_k(\pi^G) &\le \nu^{\mathsf{OMCP}}\cdot W_k + \mathsf{LB}\cdot W_k\cdot\epsilon \\ &\le (1+\epsilon)\cdot\nu^{\mathsf{OMCP}}\cdot W_k,\ 2\le k\le K. \quad \text{(C.10)}\end{aligned}$$

Therefore π^G is an $(1+\epsilon)$-approximation to $\mathsf{OMCP}(G,s,t,K,\vec{W},\vec{\omega})$. It follows from Lemma 3.1 that the worst case time complexity of line 14 is $O(m(\frac{n}{\epsilon})^{K-1})$. Since $K\ge 2$ is a constant, the time complexity of line 14 dominates the time complexity of lines 2-13. Since the time complexity of line 1 is $O(mn\log\log\log n)$, the overall time complexity of Algorithm 5 is $O(mn\log\log\log n + m(\frac{n}{\epsilon})^{K-1})$. ■

Guoliang (Larry) Xue (SM'99/ACM'93) received the BS degree (1981) in mathematics and the MS degree (1984) in operations research from Qufu Teachers University, Qufu, China, and the PhD degree (1991) in computer science from the University of Minnesota, Minneapolis, USA. He is a Full Professor in the Department of Computer Science and Engineering at Arizona State University, and has held previous positions at Qufu Teachers University, the Army High Performance Computing Research Center, and the University of Vermont. Dr. Xue's research interests include efficient algorithms for optimization problems in networking, with applications to QoS, survivability, security, privacy, and energy efficiency issues in networks ranging from WDM optical networks to wireless ad hoc and sensor networks. He has published over 150 papers in these areas, including many papers in journals such as IEEE Transactions on Circuits and Systems, IEEE Transactions on Communications, IEEE Transactions on Computers, IEEE Transactions on Vehicular Technology, IEEE/ACM Transactions on Networking, SIAM Journal on Computing, SIAM Journal on Optimization, and conferences such as ACM MobiHoc, ACM/SIAM SODA, and IEEE Infocom. His research has been continuously supported by federal agencies including NSF and ARO. Xue received the *Graduate School Doctoral Dissertation Fellowship* from the University of Minnesota in 1990, a *Third Prize from the Ministry of Education of P.R. China* in 1991, an *NSF Research Initiation Award* in 1994, and an *NSF-ITR Award* in 2003. He is an Editor of Computer Networks (COMNET), an Editor of IEEE Network, and the Journal of Global Optimization. He has served on the executive/program committees of many IEEE conferences, including Infocom, Secon, Icc, Globecom and QShine. He is a TPC co-chair of *IEEE Globecom'2006 Symposium on Wireless Ad Hoc and Sensor Networks*, a TPC co-chair of *IEEE ICC'2007 Symposium on Wireless Ad Hoc and Sensor Networks*, and a TPC co-chair of *QShine* in 2007. He also serves on many NSF grant panels and is a reviewer for many DOD grant proposals.

Weiyi Zhang (S'02) received the B.E. and M.E. degrees from Southeast University, China, in 1999 and 2002 respectively. Currently he is a Ph.D student in the Department of Computer Science and Engineering at Arizona State University. His research interests include reliable communication in networking, protection and restoration in WDM networks, and QoS provisioning in communication networks.

Jian Tang (S'04) received the B.E. and M.E. degrees from Beijing University of Posts and Telecommunications, China, in 1998 and 2001 respectively. He received the PhD degree in computer science from Arizona State University in 2006. Since then, he has been an Assistant Professor in the Department of Computer Science at Montana State University, Bozeman, Montana. His research interest is in the area of networking, with emphases on routing, scheduling, cross-layer design and QoS provisioning in communication networks. This work was done while he was a graduate student at Arizona State University.



Krishnaiyan Thulasiraman (F'90/ACM'95) received the Bachelor's degree (1963) and Master's degree (1965) in electrical engineering from the University of Madras, India, and the Ph.D degree (1968) in electrical engineering from IIT, Madras, India. He holds the Hitachi Chair and is Professor in the School of Computer Science at the University of Oklahoma, Norman, where he has been since 1994. Prior to joining the University of Oklahoma, Thulasiraman was professor (1981-1994) and chair (1993-1994) of the ECE Department in Concordia University, Montreal. He was on the faculty in the EE and CS departments of the IITM during 1965-1981.

Dr. Thulasiraman's research interests have been in graph theory, combinatorial optimization, algorithms and applications in a variety of areas in CS and EE: electrical networks, VLSI physical design, systems level testing, communication protocol testing, parallel/distributed computing, telecommunication network planning, fault tolerance in optical networks, interconnection networks etc. He has published more than 100 papers in archival journals, coauthored with M. N. S. Swamy two text books "Graphs, Networks, and Algorithms" (1981) and "Graphs: Theory and Algorithms" (1992), both published by Wiley Inter-Science, and authored two chapters in the Handbook of Circuits and Filters (CRC and IEEE, 1995) and a chapter on "Graphs and Vector Spaces " for the handbook of Graph Theory and Applications (CRC Press,2003).

Dr. Thulasiraman has received several awards and honors: 2006 IEEE Circuits and Systems Society Technical Achievement Award, Endowed Gopalakrishnan Chair Professorship in CS at IIT, Madras (Summer 2005), Elected member of the European Academy of Sciences (2002), IEEE CAS Society Golden Jubilee Medal (1999), Fellow of the IEEE (1990) and Senior Research Fellowship of the Japan Society for Promotion of Science (1988). He has held visiting positions at the Tokyo Institute of Technology, University of Karlsruhe, University of Illinois at Urbana-Champaign and Chuo University, Tokyo.

Dr. Thulasiraman has been Vice President (Administration) of the IEEE CAS Society (1998, 1999), Technical Program Chair of ISCAS (1993, 1999), Deputy Editor-in-Chief of the IEEE Transactions on Circuits and Systems I (2004-2005), Co-Guest Editor of a special issue on "Computational Graph Theory: Algorithms and Applications" (IEEE Transactions on CAS , March 1988), , Associate Editor of the IEEE Transactions on CAS (1989-91, 1999-2001), and Founding Regional Editor of the Journal of Circuits, Systems, and Computers and an editor of the AKCE International Journal of Graphs and Combinatorics. Recently, he founded the Technical Committee on "Graph theory and Computing" of the IEEE CAS Society.