

# Approximation Algorithms for QoS Routing with Multiple Additive Constraints

Ronghui Hou, King-Shan Lui, Ka-Cheong Leung

Department of Electrical and Electronic Engineering

The University of Hong Kong

Hong Kong SAR, China

E-mail: {rhhou, kslui, kcleung}@eee.hku.hk

Fred Baker

Cisco Research Center

170 West Tasman Dr.

San Jose, CA 95134, USA

E-mail: fred@cisco.com

## Abstract

QoS routing is one of the major building blocks for supporting QoS in network. Precomputing-based QoS routing possesses several metrics, such as avoiding the overload of each node, effectively utilizing the network resources, and facilitates the scalability. Precomputing-based QoS routing composes two stages. In the first stage, each node precomputes the supported QoS of routing from itself to a destination prior to receiving the requests and stores the QoS information in its routing. In the second stage, when receiving a connection request, the source looks up its routing table and finds a feasible path for this request (or determines this request infeasible). In this paper, we study the problem of precomputing the supported QoS from a source to a destination with multiple additive constraints. The problem has been shown to be NP-complete and many approximation algorithms have been developed. We propose a new technique called *multi-dimensional relaxation mechanism* to improve the accuracy of the estimated supported QoS. We formally prove that the multi-dimensional relaxation mechanism can produce smaller approximation error than the existing algorithms. We then propose a set of approximation algorithms applying the multi-dimensional relaxation mechanism. We further verify the performance by extensive simulations.<sup>1</sup>

## Index Terms

Approximation algorithm, multiple additive constraints, QoS routing.

## I. INTRODUCTION

The existing network traffic is mainly composed by multi-media applications, such as webcasting, VoIP, etc, which have diverse quality-of-service (QoS) requirements, such as transmission delay, bandwidth, packet loss, cost, etc. The future network has to support guaranteed QoS. QoS routing is one of the building blocks for supporting QoS in network, which focuses on finding a path subject to a set of constraints. The QoS metrics, such as delay, cost, bandwidth, or loss rate, can be divided into two categories: additive and bottleneck. The delay of a path is

<sup>1</sup>This work is supported in part by the Cisco Research Initiative Award.

the sum of the delay of all the links on this path, and we call delay an additive metric. The loss rate of a path is the product of the loss rate of all the links on this path. However, we can use the logarithm of loss rate to denote the loss rate. In that case, we can also consider the loss rate as an additive metric. Since the bandwidth of a path is the minimum of the bandwidth of all the links on this path, bandwidth is a bottleneck metric. In this work, we consider the problem of finding a best path, say, the cheapest, satisfying a set of additive constraints, which has been proved to be NP-complete [26]. If a connection request has additional bottleneck constraints, before the process of path computation, all the links not satisfying any bottleneck constraint can be pruned, so that the path computation is basically to find a best path with additive constraints.

Most of the existing works on routing with multiple additive constraints focus on finding a path satisfying a set of constraints imposed by a given connection request. In this case, each connection request initiates one process of path computation. When the incoming requests are frequent, the network elements may be overloaded by the path computation. Moreover, the response time for a connection request is relative long. Accordingly, Precomputing-base QoS routing is proposed [16]. Each node precomputes the supported QoS from itself to a destination. When a connection request arrives, the source looks up the QoS routing table and determines immediately whether it can find a path satisfying the given requirements. The process of precomputation can be performed when the network elements are idle or not busy, so that the network resources can be effectively utilized. Besides, precomputing the supported QoS is the necessary component for supporting QoS in the hierarchical networks, such the Internet.

In order for the scalability, the nodes are grouped into different domains (also called autonomous system in the Internet). Each node has no topology information about the other domains. Some nodes connecting the nodes in other domains are called the border nodes. For example, in Fig. 1, nodes  $E$ ,  $B$  and  $G$  are border nodes.  $E$  does not know the topology of the domain  $D_2$ . We now consider the problem of  $E$  computing a feasible path subject to two constraints from itself to  $G$ . In the traditional routing, where each link is associated with just one metric, We can find a best path from  $B$  to  $G$ , so that  $E$  considers that there exists a logical link from  $B$  to  $G$  with the QoS metric of the best path. However, when each link is associated with multiple metrics, we cannot find the best path. In Fig. 1, there are two paths from  $B$  to  $G$  with the QoS parameters  $(3, 4)$  and  $(5, 3)$ . We cannot say which one is better, so that  $B$  should inform the both paths to  $E$ . We can see that before a border computing a feasible path to the border in other domains, each border in the network should have computed the supported QoS to another border in the same domain. Afterwards, the source obtains an aggregated topology and performs the QoS routing of interdomain in the hierarchical networks [12], [20].

Suppose the QoS parameters of paths  $p_1$ ,  $p_2$ , and  $p_3$  in a certain network are  $(7, 5)$ ,  $(7, 9)$  and  $(4, 7)$ , respectively. We can say that  $p_1$  is better than  $p_2$  because  $p_1$  is better in both metrics. We call  $p_2$  is dominated by  $p_1$ . However, neither  $p_1$  is dominated by  $p_3$  nor  $p_1$  dominates  $p_3$  as  $p_1$  is better in terms of metric  $\omega_1$  while  $p_3$  is better in terms of metric  $\omega_2$ . If there is no path that can dominate  $p_1$ , we call  $p_1$  a *non-dominated path*. Different non-dominated

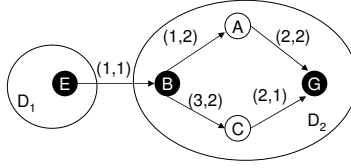


Fig. 1. A simple two-layer hierarchical network topology.

paths define different supported QoSes. For example, given a request with the requirements  $(4, 8)$ ,  $p_3$  is feasible but  $p_1$  is not. On the other hand,  $p_1$  can support the request  $(7, 5)$  but  $p_3$  cannot. The supported QoS is actually defined by all the non-dominated paths. Precomputing the supported QoS is actually to find all the non-dominated paths, which is quite challenging. Even finding one non-dominated path is NP-complete, since this problem is equal to finding a feasible path subject to a set of constraints. For example, if each link is associated with three additive metrics, such as delay, cost, and loss. Denote  $(d, c, l)$  as the QoS parameter of a non-dominated path  $p$ , where  $d$ ,  $c$ , and  $l$  are delay, cost, and lost, respectively. Finding path  $p$  actually is equal to finding the cheapest path with the delay and lost metrics no greater than  $d$  and  $l$ , respectively. The best path should be  $p$ , since there does not exist any path  $p'$  with the delay, cost, and lost no greater than  $d$ ,  $c$ , and  $l$ , respectively. In this case, finding all the non-dominated paths means that we compute the best paths subject to all possible sets of constraints.

#### A. Our contributions

The focus of this work is to devise approximation algorithms for estimating the supported QoS. In other words, our algorithms aims to estimate the QoS parameters of all the non-dominated paths. Although the multi-constrained routing has been paid much attention in the research community, to the best of our knowledge, our work is the first to consider the problem of precomputing the supported QoS with multiple additive constraints. We resort to approximation method since it provides performance guarantee. A good approximation algorithm should have the lower approximation error and the lower computational overhead. However, in the traditional  $\epsilon$ -optimal approximation method, both the approximation error and the computational overhead are directly related to  $\epsilon$ . The smaller  $\epsilon$  produces smaller approximation error but higher computational overhead. Accordingly, we propose a novel method, *multi-dimensional relaxation*, which improves the accuracy performance independently of  $\epsilon$ , so that the computational overhead cannot be affected. By using the traditional approximation method, we can obtain an approximated QoS parameter of a path. We proved that the estimated QoS metric of a path is no less than the real one. The main philosophy of the multi-dimensional relaxation is to approach the estimated QoS metric to the real one. We proved that the estimated QoS parameter of a path found by the multi-dimensional mechanism does be the real one. Since the existing approximation algorithms cannot guarantee this feature, we can conclude that our approach achieves a better estimation. Moreover, this feature denotes that the performance of the multi-dimensional relaxation is not affected by the number of constraints but that of the existing mechanisms do, which is discussed

later. We also give the theoretical analysis for the probability that our algorithm can find the real supported QoS. Afterwards, we argue that the approximation algorithms using the multi-dimensional relaxation mechanism have the same computational and spacial complexities with the existing algorithms. By doing extensive simulations, we show that our algorithms outperform the existing algorithms.

### B. Organization of the paper

The rest of the paper is organized as follows. Section II compares and discusses some existing works on QoS routing with multiple additive constraints. The strengths and weaknesses of these algorithms are highlighted. Section III introduces the network model and gives the problem formulation. In Section IV, we first analyze the performance of the algorithms applying the existing approximation mechanisms for precomputing the supported QoS. We then present our approach and analyze its performance. Finally, we analyze the computational and spacial complexities of our algorithms. Our simulation results are discussed in Section V. Finally, we conclude our work in Section VI.

## II. RELATED WORKS

There have been a relative works on QoS routing in networks, and we just consider the works on the QoS routing with multiple additive constraints. Most of the works consider the problem of finding the feasible path subject to a given set of constraints. We are going to discuss some of them and analyze whether they can be applied for precomputing the supported QoS.

The work in [11] was the first proposal for the QoS routing with two additive constraints. Denote  $c(p)$  and  $d(p)$  as the cost and delay of the path  $p$ , respectively. Jaffe gives an objective function  $f(p) = \max(d(p), d_{req}) + \max(c(p), c_{req})$  and proposes a heuristic algorithm for finding a path minimizing this objective function, where  $d_{req}$  and  $c_{req}$  are the delay and cost requirements imposed by a given request, respectively. This work applies a weighted linear function  $g(p) = \alpha w_1 + \beta w_2$  to denote the fixed weight of a path, where  $w_1$  and  $w_2$  are the two additive metrics of a path. By running the shortest-path algorithm, one path with the minimum linear function value can be found. Denote  $p^*$  and  $p'$  as the paths found by minimizing  $f(p)$  and  $g(p)$  for all possible  $p$  in network, respectively. The key contribution of the work in [11] is that the author give a upper bound of  $\frac{f(p')}{f(p^*)}$ . The work in [1] considers multiple additive constraints by using the similar idea in [11]. This means that the objective function  $f(p)$  and the linear function  $g(p)$  is extended to more than two metrics. The main contribution in this work is also to give the bound of  $\frac{f(p')}{f(p^*)}$ . Many works, as referred to [21], use another objective function  $\max_{1 \leq k \leq \mathcal{K}} \frac{w_k(p)}{C_k}$ , where  $w_k(p)$  is the  $k^{\text{th}}$  metric of path  $p$  and  $C_k$  denotes that  $w_k(p)$  should not be greater than  $C_k$ , for all possible  $1 \leq k \leq \mathcal{K}$ .  $\mathcal{K}$  denotes the number of QoS metrics. Such works develop difference heuristic algorithms for finding a path such that this path provides the minimum objective function value. The work in [21] analyzes the relationship between these two different objective functions. This work also cites almost all of the existing heuristic methods

for routing with multiple additive constraints. Whatever techniques they apply, the common characteristics of all these works is that the path selection process is based on a given set of constraints. However, in the problem of precomputing the supported QoS, we have no specific QoS requirements, so that all of these works cannot be applied for precomputing the supported QoS.

Recently, several approximation algorithms have been proposed for the routing with multiple additive constraints. The works in [2], [3], [9], [7], [13] applies the same approximation technique, called scaling and rounding [17]. These works consider two metrics, say, delay and cost. If the QoS metrics of each link are integer, we can develop the polynomial algorithm for finding the path subject to a set of constraints. These works thus scale one metric, say, delay, to integer, so that the problem is converted into solvable. For instance, denote  $d(u, v)$  as the delay metric of the link  $(u, v)$  and  $d_{req}$  as the given delay requirement. In these works,  $d(u, v)$  for each possible  $(u, v)$  in network is scaled as  $\lceil \frac{d(u, v) \cdot \lambda}{d_{req}} \rceil$ . We can round the metric to either the floor or the ceiling. Since the scaling error of each link is at most  $\frac{d_{req}}{\lambda}$ , a larger  $\lambda$  produces smaller error but higher computational overhead. The time complexity of the algorithm in [9] is  $\mathcal{O}(mn(\log \log(\frac{UB}{LB}) + \frac{1}{\epsilon}))$ , where  $m$  and  $n$  are the number of links and nodes, respectively,  $UB$  and  $LB$  are the upper bound and lower bound of the QoS metric of each link, and  $\epsilon$  is the upper bound of the deviation of the estimated QoS metric of a path. The complexities of the algorithms in [13] and [3] are  $\mathcal{O}(mn(\log \log n + \frac{1}{\epsilon}))$  and  $\mathcal{O}((m + n \log n) \frac{n}{\epsilon})$ . The algorithm in [7] has the best-known complexity of  $\mathcal{O}((m + n \log n) \frac{L}{\epsilon})$ , where  $L$  is the hop count of the longest path in network. In [2], two techniques are proposed to improve the accuracy performance of the algorithm in [7]. We can see that these works are fully polynomial with the number of nodes. However, the path selection of all these works is also based on the specific constraints, moreover, these algorithms cannot be implemented in distributed way.

Some approximation algorithms in [10], [16], [23] apply another approximation technique, called interval partition [17]. For instance, each link is associated with two additive metrics, delay and cost. The general idea of these works is that we discretize one metric of each link, say, delay, by using a sampling sequence  $\{s_1, s_2, \dots, s_n\}$ , where  $s_{i-1} < s_i < s_{i+1}$  for all  $i = 2, \dots, n-1$ , so that we assume that the delay metric of each link is one one sample in the sample sequence. We can consider each sample as an integer, then, the problem is converted to solvable. The works in [16] and [23] apply different sampling techniques. This mean that they use different sampling sequence. For instance, the work in [23] uses uniform sampling. The sampling sequence is  $\{0, \delta, 2\delta, \dots\}$ , where  $\delta$  is called the sampling parameter. The work in [16] uses logarithmic sampling. The sampling sequence is  $\{1, 1 + \delta, (1 + \delta)^2, \dots\}$ . It is obvious that a smaller  $\delta$  produces smaller approximation error but higher computational overhead. In [10], the authors analyze the approximation error produced by uniform sampling and logarithmic sampling with two additive constraints, and also proposes a new approach, *two-dimensional sampling*, to improve the performance of the sampling approximation method. It formally proves that the two-dimensional sampling mechanism produces smaller approximation error than the existing algorithms. The complexity (computational or spacial) of the sampling

approximation method is directly related to the number of samples in the sampling sequence. The number of samples depends on  $\delta$  and the maximum value of the cost metric of each link  $B$ , since we know that the maximum cost metric of a path is less than  $nB$ , where  $n$  is the number of nodes. Although the sampling approximation algorithm is actually pseudo-polynomial, the path selection of the sampling approximation method is independent on the specific constraints, so that it can be suited for precomputing the supported QoS, which is discussed later.

There have been several works applying the heuristic method to precompute the supported QoS [6], [5], [25]. The common technique used by these works is to design a formula combining all the metrics into a mixed metric. A shortest path is then found by using the shortest-path algorithm with the mixed metric. By changing the mixed formula, we will get different paths, and the supported QoS is thus defined by all the found paths. The work in [6] considers two additive constraints. It introduces a linear formula. For instance, let  $w_1$  and  $w_2$  be the first and the second QoS metrics of a path, respectively. The weight of this path is defined by  $\alpha w_1 + (1 - \alpha)w_2$ , where  $0 < \alpha < 1$ . The work in [5] also use the linear function but consider more than two additive constraints. The work in [25] applies non-linear function and argues that the performance of non-linear function is higher than that of linear function. The heuristic algorithm runs very quickly, however, they cannot provide performance guarantee.

There are also some works considering other issues for supporting QoS, such as the works in [8], which considers the QoS routing with inaccurate information and introduces an approximation algorithm, where each metric is represented by a statistics. In this work, we assume that the network information is accurate, and the update of network information [18] is outside the topic in this paper.

### III. NETWORK MODEL AND PROBLEM FORMULATION

This section formulates the general model addressed in this paper. A network is modeled as a directed graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ .  $\mathcal{V}$  is the set of nodes and  $\mathcal{E}$  is the set of directed links among the nodes in  $\mathcal{V}$ .  $s \in \mathcal{V}$  is a source and  $d \in \mathcal{V}$  is a destination. If  $(v_i, v_j) \in \mathcal{E}$ ,  $v_j$  is  $v_i$ 's neighbor. Denote  $\mathcal{A}(v_i)$  as the neighbor set of  $v_i$ . Each link  $e = (v_i, v_j) \in \mathcal{E}$ , where  $v_i, v_j \in \mathcal{V}$ , is associated with  $\mathcal{K}$  independent weights, where  $\mathcal{K} \geq 2$ .  $\omega_k(e) > 0$  is the  $k^{\text{th}}$  weight for link  $e$ .  $\vec{\omega}(e) = (\omega_1(e), \omega_2(e), \dots, \omega_{\mathcal{K}}(e))$  is the weight vector for link  $e$ , and is also called the QoS parameter for link  $e$ . For example, in Fig. 2, link  $e = (A, B)$  is associated with two weights such that  $\omega_1(e) = 3$ ,  $\omega_2(e) = 2$ , and  $\vec{\omega}(e) = (3, 2)$ . Let  $p$  be a path in  $\mathcal{G}$ . The  $k^{\text{th}}$  weight of  $p$ , denoted as  $\omega_k(p)$ , is the sum of the  $k^{\text{th}}$  weight for all links along  $p$ .  $\vec{\omega}(p) = (\omega_1(p), \omega_2(p), \dots, \omega_{\mathcal{K}}(p))$  is the weight vector for path  $p$ . Given path  $p=A \rightarrow B \rightarrow E \rightarrow G$  in Fig. 2, we have  $\omega_1(p) = \omega_1(A, B) + \omega_1(B, E) + \omega_1(E, G) = 8$ ,  $\omega_2(p) = \omega_2(A, B) + \omega_2(B, E) + \omega_2(E, G) = 4$ , and hence  $\vec{\omega}(p) = (8, 4)$ .

*Definition 1:* Given two different paths  $p_1$  and  $p_2$  from  $s$  to  $d$ , if  $\omega_i(p_1) \leq \omega_i(p_2)$  for all  $i = 1, 2, \dots, \mathcal{K}$ ,  $\vec{\omega}(p_1)$  is more representative than  $\vec{\omega}(p_2)$ , denoted by  $\vec{\omega}(p_1) \preceq \vec{\omega}(p_2)$ .

For example, in Fig. 2, define paths  $p_1=A \rightarrow C \rightarrow F \rightarrow G$  and  $p_2=A \rightarrow D \rightarrow C \rightarrow F \rightarrow G$ . Since  $\vec{\omega}(p_1) = (5, 6)$  and  $\vec{\omega}(p_2) = (5, 10)$ , we say that  $p_1$  is better than  $p_2$  and  $\vec{\omega}(p_1) \prec \vec{\omega}(p_2)$ .

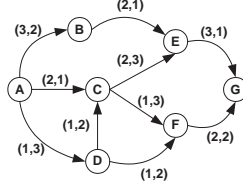


Fig. 2. A simple network where  $(x, y)$  represents an additive QoS parameter  $(\omega_1, \omega_2)$ .

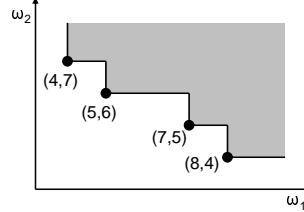


Fig. 3. An optimal weight function on  $\omega_1$ - $\omega_2$  plane.

*Definition 2:* The QoS parameter of a path  $p$  is a *representative vector* if there does not exist the QoS parameter of another path  $p'$  such that  $\vec{\omega}(p') \prec \vec{\omega}(p)$ .  $p$  is also called a *non-dominated paths*

In Fig. 2,  $\mathbb{P}_{A \rightarrow G}$  contains six paths with the QoS parameters  $\{(4, 7), (5, 6), (5, 10), (7, 5), (7, 9), (8, 4)\}$ , where the QoS parameters  $\{(4, 7), (5, 6)\}, (7, 5), (8, 4)\}$  are representative vectors. We plot all these representative vectors on the  $\omega_1$ - $\omega_2$  plane, as illustrated in Fig. 3. The shaded area denotes the feasible region supported by the paths from  $A$  to  $G$ . All the requests falling in the feasible region can be supported by at least one path from  $s$  to  $d$ . Denote  $\mathcal{RP}_{s,d}^{\text{opt}}$  as the set of the QoS parameters of all non-dominated paths from  $s$  to  $d$ , which is also called the representative vector set. Therefore, the supported QoS from  $s$  to  $d$  is defined by  $\mathcal{RP}_{s,d}^{\text{opt}}$ .

We denote the requirement on the  $j^{\text{th}}$  constraint as  $c_j$ , where  $j = 1, \dots, \mathcal{K}$ . That is, a request is represented as  $(c_1, c_2, \dots, c_{\mathcal{K}})$ . A path  $p$  can support request  $(c_1, c_2, \dots, c_{\mathcal{K}})$  if  $\omega_j(p) \leq c_j$  for all  $j = 1, \dots, \mathcal{K}$ . If a request has no constraint on the  $i^{\text{th}}$  weight, we set  $c_i$  to be  $\infty$ . We thus give the routing with multi-constrained problem as follows.

*Definition 3: Multi-constrained optimal path problem (MCOP):* Given any request  $(c_1, \dots, c_{i-1}, \infty, c_{i+1}, \dots, c_{\mathcal{K}})$ , the source needs to find the minimum  $i^{\text{th}}$  weight provided by all the paths from itself to a destination, which satisfy each constraint  $c_i$ .

We let  $\mathbf{W}_{s,d}^{\text{opt},i}(c_1, \dots, c_{i-1}, c_{i+1}, \dots, c_{\mathcal{K}})$  be  $\omega_i(p)$  where  $p$  is the optimal path for the MCOP problem. For example, let each link  $e$  in network be associated with the three additive metrics  $(\omega_1(e), \omega_2(e), \omega_3(e))$ . Suppose the constraints are delay, cost, and hop count, respectively.  $\mathbf{W}^{\text{opt},3}(c_1, c_2)$  represents the minimum hop count of the paths with the path delay no greater than  $c_1$  and the cost no greater than  $c_2$ . Since our work focuses on precomputing-based QoS routing, each source needs to precompute  $\mathbf{W}_{s,d}^{\text{opt},i}(c_1, \dots, c_{i-1}, c_{i+1}, \dots, c_{\mathcal{K}})$  for given any

request  $(c_1, \dots, c_{i-1}, \infty, c_{i+1}, \dots, c_{\mathcal{K}})$ . Denote  $\mathbf{W}_{s,d}^{\text{opt},i}(\bullet)$ <sup>2</sup> as the set of function values with all possible requests. We also call  $\mathbf{W}_{s,d}^{\text{opt},i}(\bullet)$  the minimum  $i^{\text{th}}$  weight function, where  $i = 1, \dots, \mathcal{K}$ . For example, in Fig. 2, we can compute the functions  $\mathbf{W}_{A,G}^{\text{opt},1}(\bullet)$  and  $\mathbf{W}_{A,G}^{\text{opt},2}(\bullet)$  as follows.

$$\mathbf{W}_{A,G}^{\text{opt},1}(c_2) = \begin{cases} 4 & \text{if } c_2 \geq 7 \\ 5 & \text{if } 6 \leq c_2 < 7 \\ 7 & \text{if } 5 \leq c_2 < 6 \\ 8 & \text{if } 4 \leq c_2 < 5 \\ \infty & \text{if } c_2 < 4 \end{cases} \quad (1)$$

$$\mathbf{W}_{A,G}^{\text{opt},2}(c_1) = \begin{cases} 4 & \text{if } c_1 \geq 8 \\ 5 & \text{if } 7 \leq c_1 < 8 \\ 6 & \text{if } 5 \leq c_1 < 7 \\ 7 & \text{if } 4 \leq c_1 < 5 \\ \infty & \text{if } c_1 < 4 \end{cases} \quad (2)$$

Assume that the source has found all the non-dominated paths. Given any request  $(c_1, \dots, c_{i-1}, \infty, c_{i+1}, \dots, c_{\mathcal{K}})$ , if it can find the feasible paths, the optimal path must be a non-dominated paths with the minimum  $i^{\text{th}}$  weight. For example, each function value in (1) or (2) is the 1<sup>st</sup> weight or the 2<sup>nd</sup> weight of a non-dominated path, respectively. In other words,  $\mathbf{W}^{\text{opt},i}(\bullet)$  is defined by  $\mathcal{SR}^{\text{opt}}$ , and vice versa. Therefore, both  $\mathbf{W}^{\text{opt},i}(\bullet)$  and  $\mathbf{W}^{\text{opt},j}(\bullet)$ , where  $i \neq j$ , define the same supported QoS. We thus give our problem statement as follows.

*Definition 4: Precomputing-based Multi-constrained optimal path problem (PMCOP):* is to find all the non-dominated paths from a source  $s$  to a destination  $d$ , or to compute  $\mathbf{W}_{s,d}^{\text{opt},i}(\bullet)$  for any  $i$ , where  $i = 1, \dots, \mathcal{K}$ .

Since computing  $\mathbf{W}^{\text{opt},i}(\bullet)$  is NP-Complete, in the following section, we will discuss how to compute an approximated  $i^{\text{th}}$  weight function, denoted by  $\mathbf{W}_{s,d}^i(\bullet)$ . Define  $\mathcal{SR}_{s,d}^i$  as the set of representative vector defined by  $\mathbf{W}_{s,d}^i(\bullet)$ . In other words,  $\mathcal{SR}_{s,d}^i$  and  $\mathbf{W}_{s,d}^i(\bullet)$  can represent each other. For the ease of discussion, each link weight is bounded above by  $B$ , i.e.,  $\omega_i(e) \leq B$  for all  $i = 1, \dots, \mathcal{K}$  and  $e \in \mathcal{E}$ . Then,  $\omega_i(p)$  is bounded by  $|\mathcal{V}|B$  for all  $i$  and  $p$ . We further denote  $|\mathcal{V}|B$  by  $\mathcal{UB}$ .

## IV. APPROXIMATION ALGORITHMS

### A. Algorithm for links with integral weights

We first assume that the weights of each link are integers. We give (3), which represents the iteration of each node  $s \in \mathcal{V}$  for computing  $\mathbf{W}_{s,d}^1(\bullet)$ .

<sup>2</sup>When the context is clear, we drop the subscripts  $s$  and  $d$  and just use  $\mathbf{W}^{\text{opt},i}$

$$\begin{aligned}
\mathbf{W}_{u,d}^1(c_2, \dots, c_{\mathcal{K}}) &\rightarrow \infty, \quad u \in \mathcal{V}; \\
\mathbf{W}_{d,d}^1(c_2, \dots, c_{\mathcal{K}}) &= 0, \quad c_k \geq 0 \quad \forall k = 2, \dots, \mathcal{K}; \\
\mathbf{W}_{u,d}^1(c_2, \dots, c_{\mathcal{K}}) &= \min_{v \in \mathcal{A}(u)} \\
&\quad \{ \mathbf{W}_{v,d}^1(c_2 - \omega_2(u, v), \dots, c_{\mathcal{K}} - \omega_{\mathcal{K}}(u, v)) + \omega_1(u, v), \\
&\quad \mathbf{W}_{u,d}^1(c_2, \dots, c_{\mathcal{K}}) \} \\
c_k &= 0, 1, \dots, \mathcal{UB} \quad \forall k = 2, \dots, \mathcal{K}, \quad u \in \mathcal{V}.
\end{aligned} \tag{3}$$

To compute  $\mathbf{W}_{s,d}^1(\bullet)$  for all  $s \in \mathcal{V}$ , we keep a distance table of  $|\mathcal{V}|$  rows and  $(\mathcal{UB}+1)^{\mathcal{K}-1}$  columns, where one row for each node and one column for each possible constraint tuple  $(\infty, c_2, \dots, c_{\mathcal{K}})$ . Since there are  $\mathcal{UB} + 1$  different possible values for each  $c_j$ , where  $j = 1, \dots, \mathcal{K}$ , and there are  $\mathcal{K}$  constraints, there are totally  $(\mathcal{UB}+1)^{\mathcal{K}-1}$  different constraint tuples. To ease our discussion, we label the nodes as  $1, 2, \dots, |\mathcal{V}|$  and each column as  $(\infty, c_2, \dots, c_{\mathcal{K}})$ . The entry on row  $s$  and column  $(\infty, c_2, \dots, c_{\mathcal{K}})$  represents the estimated minimum 1<sup>st</sup> weight from  $s$  to  $d$  with the constraint  $c_k$  for all  $k = 2, \dots, \mathcal{K}$ . Initially, all the entries on row  $d$  are set to zero while all the other entries are set to infinity. In the first step, each neighbor  $s$  of  $d$  sets each entry on row  $s$  and column  $(\infty, c_2, \dots, c_{\mathcal{K}})$  to  $\omega_1(s, d)$ , where  $c_j \geq \omega_j(s, d)$  for all  $j = 2, \dots, \mathcal{K}$ . In step  $n$ , we update the entries on each row  $s$  that can be  $n$  hops away from  $d$ . After  $|\mathcal{V}| - 1$  steps, the algorithm terminates since no path can have more than  $|\mathcal{V}| - 1$  hops. The implementation of the above discussion is illustrated in Procedure PMCOPI, where  $\mathbf{W}_{s,d}^i(\bullet)$  is represented by  $\mathcal{SR}_{s,d}^i$ . Procedure APPWCF first builds each entry  $(q[0], \dots, q[i-1], \infty, q[i+1], \dots, q[\mathcal{K}])$  of the distance table on each node, then calls Procedure RELAX to compute the minimum  $i^{\text{th}}$  weight  $q[i]$  for each entry. Afterwards, it will get a vector  $(q[0], \dots, q[\mathcal{K}])$ , and also determine whether the new vector is a representative vector. The procedure of RELAX is actually the implementation of (3). It finds all the paths satisfying the given request, and then selects the one providing the minimum  $i^{\text{th}}$  weight. In the traditional shortest-path algorithm, in each step, each node computes the minimum cost path. While in this algorithm, each node needs to compute a set of minimum cost paths which satisfy all possible requests. Since each requirement is integer, the number of all possible requests are finite. The following lemma shows that Algorithm 1 can compute the *precise* supported QoS.

*Lemma 1:* When the metric of each link is integer, (3) can compute the *precise* QoS information in the network.

*Proof:* We prove that (3) can compute the QoS parameter of each non-dominated path  $p$ . Since  $\omega_i(p)$  is integer for all possible  $i$ , there must exist a entry  $\mathbf{e} = (\infty, \omega_2(p), \dots, \omega_{\mathcal{K}}(p))$  on  $s$ .

For the basic step, suppose that  $p$  is with one-hop. the entry  $\mathbf{e}$  is set to  $\omega_1(p)$ .

As the inductive step, assume that the QoS parameters of all the  $k$ -hop non-dominated paths are found. For a  $k + 1$ -hop path  $p$ , the entry  $\mathbf{e}$  should be set to the sum of the 1<sup>st</sup> weights of a  $k$ -hop non-dominated path and a link, which is the real  $\omega_1(p)$ . ■

## B. The sampling approximation mechanism

When link weights are not integers, we can sample the  $j^{\text{th}}$  weight values in the range  $(0, \mathcal{UB}]$  for all  $j = 2, \dots, \mathcal{K}$ . For example, if we use uniform sampling [10], the sample sequence is  $\mathbf{S} = \{0, \delta, 2\delta, \dots, m\delta, \mathcal{UB}\}$ , where  $\delta$  is called the *sampling parameter* and  $m = \max\{t\delta < \mathcal{UB}, t \in \mathbb{Z}^+\}$ . If we use logarithmic sampling [10], the sample sequence is  $\mathbf{S} = \{1, 1 + \delta, (1 + \delta)^2, \dots, (1 + \delta)^n, \mathcal{UB}\}$ , where  $n = \max\{(1 + \delta)^t < \mathcal{UB}, t \in \mathbb{Z}^+\}$ . The process of estimating the minimum weight function by using uniform sampling is same as that with integral constraints. We just give the iteration of each node shown in (4). We can see that the difference between (3) and (4) is that  $c_k$  in (3) should be chosen from the sample sequence, where  $k = 2, \dots, \mathcal{K}$ . The algorithm of sampling approximation is same as Algorithm 1 except that, in Line 17 of the procedure of APPWCF,  $j$  should be chosen from a sample sequence (uniform or logarithmic).

$$\begin{aligned}
\mathbf{W}_{u,d}^1(c_2, \dots, c_{\mathcal{K}}) &\rightarrow \infty, \quad u \in \mathcal{V}; \\
\mathbf{W}_{d,d}^1(c_2, \dots, c_{\mathcal{K}}) &= 0, \quad c_k \geq 0 \quad \forall k = 2, \dots, \mathcal{K}; \\
\mathbf{W}_{u,d}^1(c_2, \dots, c_{\mathcal{K}}) &= \min_{v \in \mathcal{A}(u)} \\
&\quad \{ \mathbf{W}_{v,d}^1(c_2 - \omega_2(u, v), \dots, c_{\mathcal{K}} - \omega_{\mathcal{K}}(u, v)) + \omega_1(u, v), \\
&\quad \mathbf{W}_{u,d}^1(c_2, \dots, c_{\mathcal{K}}) \} \\
c_k &= 0, \delta, 2\delta, \dots, m\delta, \mathcal{UB} \quad \forall k = 2, \dots, \mathcal{K}, \quad u \in \mathcal{V}.
\end{aligned} \tag{4}$$

As we can only estimate the actual supported QoS by sampling, a source may not find the feasible path for a request, even a feasible path does exist. However, the following lemma shows that the  $i^{\text{th}}$  weight estimated by (3) must be an overestimation of the real one of a certain path. That is,  $\omega_i(p)$  must not be greater than the estimated one computed by (3) for all possible  $i$  and  $p$ .

*Lemma 2:* The estimated  $i^{\text{th}}$  weight of a path is larger than or equal to the real one for all  $i = 1, \dots, \mathcal{K}$ .

*Proof:* We prove by using contradiction. Given a path  $p$  denoted by  $\{s, v_1, \dots, v_h, d\}$ , denote  $\bar{\omega}_i(p)$  as the estimated  $i^{\text{th}}$  weight of  $p$ . Without loss of generality, assume that  $\mathcal{K} = 2$ . By (4),  $\bar{\omega}_1(p) = \omega_1(p)$ , since  $\bar{\omega}_1(p)$  is the sum of the 1<sup>st</sup> weight of all the links on  $p$ . Assume that  $\bar{\omega}_2(p) < \omega_2(p)$ . Let  $c_2 = \bar{\omega}_2(p)$ . By (4), we have  $\mathbf{W}_{s,d}^1(c_2) = \mathbf{W}_{v_1,d}^1(c_2 - \omega_1(s, v_1)) + \omega_2(s, v_1)$ . This means that  $v_1$  must find that path  $(v_1, v_2, \dots, v_h, d)$  satisfies the request  $(\infty, c_2 - \omega_1(s, v_1))$ . Finally, we can deduce that  $v_h$  must find that link  $(v_h, d)$  satisfies the request  $(\infty, c_2 - (\omega_2(p) - \omega_2(v_h, d)))$ , which contradicts our assumption since  $c_2 < \omega_2(p)$ . ■

Suppose that  $s$  finds  $w_1 = \mathbf{W}_{s,d}^1(w_2, \dots, w_{\mathcal{K}})$  using (4). Then,  $s$  must find a path  $p$  satisfying the request  $(w_1, w_2, \dots, w_{\mathcal{K}})$ . We call  $(w_1, w_2, \dots, w_{\mathcal{K}})$  *corresponds* to  $p$ .

The approximated delay function  $\mathbf{W}_{s,d}^i(\bullet)$  can be defined by a set of approximated representative vectors,  $\mathcal{RP}_{s,d}^i$ . We mentioned that  $\mathbf{W}^{\text{opt},i}(\bullet)$  and  $\mathbf{W}^{\text{opt},j}(\bullet)$  define the same supported QoS for different  $i$  and  $j$ . However,  $\mathbf{W}^i(\bullet)$  and  $\mathbf{W}^j(\bullet)$ , the approximated functions, may define different QoSes. For example, Fig. 4(a) illustrates a real supported QoS. Let  $\delta = 0.2$ , then the sample sequence is  $\{1, 1.2, \dots\}$ . in Fig. 4(a), the values at  $w_1$  when  $w_2 = 1.4$

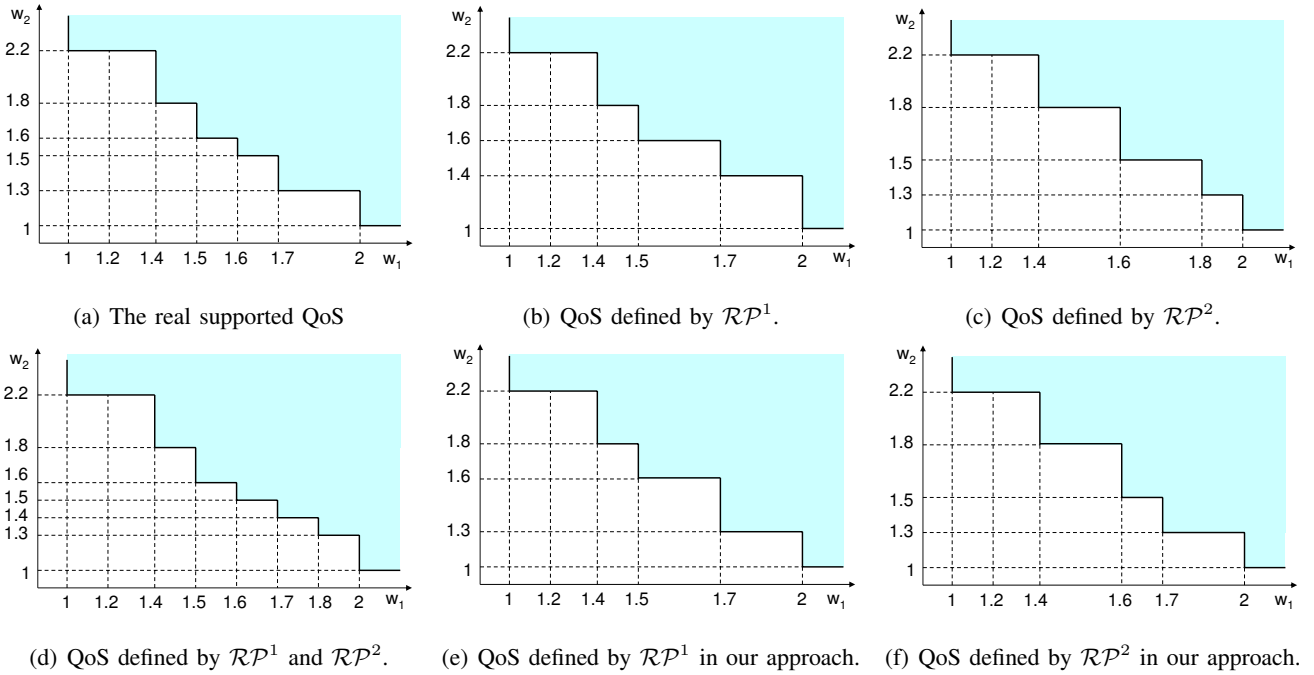


Fig. 4. The construction of the approximated functions.

and  $w_2 = 1.6$  are 1.7 and 1.5, respectively. Then,  $\mathcal{RP}^1$  becomes a staircase as illustrated in Fig. 4(b). Similarly, Fig. 4(c) illustrates the approximated supported QoS defined by  $\mathcal{RP}_{s,d}^2$ . We can see that  $\mathbf{W}_{A,G}^1(\bullet)$  and  $\mathbf{W}_{A,G}^2(\bullet)$  define different QoSes. We would like to analyze the approximation error caused by the sampling mechanism. Without loss of generality, we assume that the approximated supported QoS is defined by  $\mathbf{W}_{s,d}^1(\bullet)$ . We first introduce the following lemma.

*Lemma 3:* Given a real representative vector  $(w_1, \dots, w_{\mathcal{K}})$  in  $\mathcal{RP}_{s,d}^{\text{opt}}$ , if there exists a vector  $(v_1, \dots, v_{\mathcal{K}})$  in  $\mathcal{RP}_{s,d}^1$ , such that  $v_i \leq (1 + \epsilon)w_i$  for all  $i = 1, \dots, \mathcal{K}$ , the algorithm for finding  $\mathcal{RP}_{s,d}^1$  is an  $\epsilon$ -approximation algorithm.

*Proof:* [19] mentioned that an algorithm is an  $\epsilon$ approximation algorithm iff the algorithm generates a path  $p$  that satisfies the request  $(c_1, \dots, c_j)$  whenever the network has a path  $p'$  such that  $\omega_j(p') \leq (1 - \epsilon)c_j$  for all  $j = 1, \dots, \mathcal{K}$ . Assume that the QoS parameter of  $p'$  is  $(w_1, \dots, w_{\mathcal{K}})$ , we thus have

$$\begin{aligned}
 v_j &\leq (1 + \epsilon)w_j \\
 &\leq (1 + \epsilon)(1 - \epsilon)c_j \\
 &= (1 - \epsilon^2)c_j \\
 &\leq c_j
 \end{aligned}$$

Therefore, the algorithm for finding  $\mathcal{RP}_{s,d}^1$  is  $\epsilon$ -approximation algorithm. ■

Given  $\epsilon$ , we need to analyze how to select the sampling parameter  $\delta$  so that the approximated supported QoS satisfies the condition in Lemma 3. We give the following lemma, which is the extension of Lemma 2 in [10].

*Lemma 4:* Given a vector  $(w_1, \dots, w_{\mathcal{K}})$  in  $\mathcal{RP}_{s,d}^{\text{opt}}$ , which is the QoS parameter of a non-dominated path  $p_{s,d}$ . By using uniform sampling, there must exist a vector  $(v_1, \dots, v_{\mathcal{K}})$  in  $\mathcal{RP}_{s,d}^1$  produced by (4) such that  $v_i \leq w_i + \mathcal{H}\delta$  for all  $i = 1, \dots, \mathcal{K}$ , where  $\mathcal{H}$  is the hop count of  $p_{s,d}$ .

*Proof:* Note that  $v_i$ , where  $i = 2, \dots, \mathcal{K}$ , is a sample chosen from the sample sequence  $\{\delta, 2\delta, \dots\}$ . We prove by using induction on the hop count of  $p$ , denoted by  $\mathcal{H}$ .

As the basic step, let  $\mathcal{H} = 1$ . Define the sample  $v_i$  such that  $q_i - \delta < w_i \leq q_i$  for all  $i = 2, \dots, \mathcal{K}$ . We thus have  $\mathbf{W}_{s,d}^1(q_2, \dots, q_{\mathcal{K}}) \leq \omega_1(p)$ . This means that  $s$  can get a vector  $(\omega_1(p), q_2, \dots, q_{\mathcal{K}})$ . We thus have  $v_i \leq q_i < w_i + \delta$  for all  $i = 1, \dots, \mathcal{K}$ .

For the inductive step, suppose the lemma holds for  $\mathcal{H} < k$ . When  $p_{s,d}$  consists of  $k + 1$  hops, there must exist a neighbor  $j$  of  $s$  on  $p_{s,d}$  such that

$$\begin{aligned} w_1 &= \mathbf{W}_{s,d}^{\text{opt},1}(w_2, \dots, w_{\mathcal{K}}) \\ &= \mathbf{W}_{j,d}^{\text{opt},1}(w_2 - \omega_2(s, j), \dots, w_{\mathcal{K}} - \omega_{\mathcal{K}}(s, j)) + \omega_1(s, d) \end{aligned}$$

Let  $w'_j = w_j - \omega_j(s, j)$  and  $w'_1 = \mathbf{W}_{j,d}^{\text{opt},1}(w'_2, \dots, w'_{\mathcal{K}})$ , then, the vector  $(w'_1, w'_2, \dots, w'_{\mathcal{K}})$  must be the QoS parameter of a non-dominated  $k$ -hop path.

Let  $c'_j$  be a sample chosen from sample sequence  $\{\delta, 2\delta, \dots\}$  such that  $c'_j \leq w'_j + k\delta$  for all  $j = 2, \dots, \mathcal{K}$ , based on the inductive assumption, we have

$$\mathbf{W}_{j,d}^1(c'_2, \dots, c'_{\mathcal{K}}) \leq \mathbf{W}_{j,d}^{\text{opt},1}(w'_2, \dots, w'_{\mathcal{K}}) + k\delta$$

Let  $c''_j$  be a sample such that  $c''_j - \delta < \omega_j(s, d) \leq c''_j$ , we thus have

$$\begin{aligned} c_1 &= \mathbf{W}_{s,d}^1(c'_2 + c''_2, \dots, c'_{\mathcal{K}} + c''_{\mathcal{K}}) \\ &\leq \mathbf{W}_{j,d}^1(c'_2 + c''_2 - \omega_2(s, j), \dots, c'_{\mathcal{K}} + c''_{\mathcal{K}} - \omega_{\mathcal{K}}(s, d)) \\ &\quad + \omega_1(s, j) \\ &\leq \mathbf{W}_{j,d}^1(c'_2, \dots, c'_{\mathcal{K}}) + \omega_1(s, j) \\ &\leq \mathbf{W}_{s,j}^{\text{opt},1}(w'_2, \dots, w'_{\mathcal{K}}) + \omega_1(s, d) + k\delta \\ &= w_1 + k\delta \end{aligned}$$

Let  $c_j = c'_j + c''_j$ , which is a sample, we thus have  $c_j \leq w'_j + k\delta + \omega_j(s, j) + \delta = w_j + (k + 1)\delta$ . We thus get a vector  $(c_1, c_2, \dots, c_{\mathcal{K}})$  such that  $c_j \leq w_j + (k + 1)\delta$  for all  $j = 1, \dots, \mathcal{K}$ . ■

By using similar deduction, we can extend Lemma 3 in [10] to the following lemma.

*Lemma 5:* Given a vector  $(w_1, \dots, w_{\mathcal{K}})$  in  $\mathcal{RP}_{s,d}^{\text{opt}}$ , which is the QoS parameter of a non-dominated path  $p_{s,d}$ . By using uniform sampling, there must exist a vector  $(v_1, \dots, v_{\mathcal{K}})$  in  $\mathcal{RP}_{s,d}^1$  produced by (4) such that  $v_i \leq (1 + \delta)^{\mathcal{H}} w_i$  for all  $i = 1, \dots, \mathcal{K}$ , where  $\mathcal{H}$  is the hop count of  $p_{s,d}$ .

By Lemma 5 in [10], given  $\epsilon$ , let  $\delta = \frac{\epsilon}{|\mathcal{V}|}$ , uniform sampling is thus an  $\epsilon$ -approximation algorithm. Let  $\delta = \frac{\epsilon}{2^{|\mathcal{V}|}}$ , logarithmic sampling is thus an  $\epsilon$ -approximation algorithm.

If the approximation algorithm just estimates the 1<sup>st</sup> minimum weight function, we call it the one-dimensional sampling algorithm, which is the algorithm in [23]. In [10], the authors consider two additive constraints and proposed a new approach called the *two-dimensional sampling mechanism*. It has been shown that the two-dimensional sampling mechanism produces smaller approximation error than one-dimensional sampling. Moreover, the time complexities of them are the same. The two-dimensional sampling mechanism can be easily extended to consider more than two additive constraints. We call the corresponding mechanism the *multi-dimensional sampling mechanism*.

The implementation of the multi-dimensional sampling mechanism can be found in Algorithm 5. Each node  $s$  first computes finds  $\mathcal{RP}_{s,d}^i$  for all  $i = 1, \dots, \mathcal{K}$ . In Line 9-25,  $s$  finds all the representative vector in  $\bigcup_{i=1}^{\mathcal{K}} \mathcal{RP}_{s,d}^i$ , denoted by  $\mathcal{SR}_{s,d}$ .  $s$  then uses  $\mathcal{SR}_{s,d}$  to define the estimated supported QoS. It is noteworthy that if we use the multi-dimensional sampling mechanism, Algorithm 3 should be changed a little. In Line 13, we should check each vector in  $\mathcal{SR}_{u,d}$  but not  $\mathcal{SR}_{u,d}^i$ , since  $\mathcal{SR}_{u,d}$  defines a larger feasible region, so that the accumulated approximation error can be reduced. Fig. 4(d) illustrates the supported QoS estimated by the multi-dimensional sampling mechanism, which is defined by the union of  $\mathcal{RP}^1$  and  $\mathcal{RP}^2$ .

We now analyze the time complexities of one-dimensional sampling and multi-dimensional sampling. Denote  $n$  as the number of samples in the sample sequence  $\mathbf{S}$ , where  $n = \mathcal{O}(\frac{UB}{\delta})$  by using uniform sampling. As mentioned in Section IV-A, there are totally  $n^{\mathcal{K}-1}$  entries in the routing table of each node. Therefore, the computational complexity of one-dimensional sampling is  $\mathcal{O}(n^{\mathcal{K}-1})$ , and the computational complexity of multi-dimensional sampling is  $\mathcal{O}(\mathcal{K}n^{\mathcal{K}-1})$ . The spacial complexity is the same as the computational complexity [10]. To reduce the overhead, a larger  $\delta$  can be applied. Unfortunately, the complexities of multi-dimensional sampling are still  $\mathcal{K}$  times of those of one-dimensional sampling. The additional complexity produced by multi-dimensional sampling increases exponentially as  $\mathcal{K}$  increases. It is possible to use a larger sampling parameter to make the complexities of multi-dimensional sampling comparable to the one-dimensional sampling. However, larger sampling parameter produces greater approximation error. Therefore, the performance enhancement of the multi-dimensional sampling method over one-dimensional sampling decreases as the number of constraints increases. Our simulation results also reflect this phenomenon if we restrict the overhead by employing a larger sampling parameter, multi-dimensional sampling cannot perform better than one-dimensional sampling. Accordingly, we introduce a new approach to improve the performance of the approximation algorithm.

### C. The proposed algorithm

In this subsection, we discuss the proposed approach, the *multi-dimensional relaxation mechanism*. We first introduce the following lemma.

*Lemma 6:* Let  $w_1 = \mathbf{W}_{s,d}^1(c_2, \dots, c_{\mathcal{K}})$  and  $w_2 = \mathbf{W}_{s,d}^2(w_1, c_3, \dots, c_{\mathcal{K}})$ . It holds that  $w_2 \leq c_2$ .

*Proof:* Since  $w_1 = \mathbf{W}_{s,d}^1(c_2, \dots, c_{\mathcal{K}})$ ,  $s$  must have found a path  $p$  satisfying the request  $(w_1, c_2, \dots, c_{\mathcal{K}})$ . It is obvious that  $p$  also satisfies the request  $(w_1, \infty, c_3, \dots, c_{\mathcal{K}})$ . Therefore,  $w_2$  is not greater than the estimated 2<sup>nd</sup> weight of  $p$ . Moreover, the estimated 2<sup>nd</sup> weight of  $p$  is not greater than  $c_2$ . We thus have  $w_2 \leq c_2$ . ■

In our approach, node  $s$  first finds  $\mathcal{RP}_{s,d}^i$  for all  $i = 1, \dots, \mathcal{K}$ . Let a vector  $(w_1, c_2, \dots, c_{\mathcal{K}})$  be in  $\mathcal{RP}_{s,d}^1$ . We have  $w_1 = \mathbf{W}^1(c_2, \dots, c_{\mathcal{K}})$ . We then compute  $w_2 = \mathbf{W}_{s,d}^2(w_1, c_3, \dots, c_{\mathcal{K}})$ . Lemma 6 shows that  $w_2 \leq c_2$ . We can refine our estimation by iteratively computing  $w_i = \mathbf{W}^i(w_1, \dots, w_{i-1}, c_{i+1}, \dots, c_{\mathcal{K}})$ . Lemma 6 can be easily extended to show that  $w_i \leq c_i$ . After  $\mathcal{K} - 1$  steps, we will get a new approximated QoS parameter  $(w_1, \dots, w_{\mathcal{K}})$  which defines a larger feasible region than  $(w_1, c_2, \dots, c_{\mathcal{K}})$ . The similar operation is performed for each vector in each  $\mathcal{RP}_{s,d}^i$ . We call such operation as *multi-dimensional relaxation*. Algorithm 6 gives the pseudocode of the multi-dimensional relaxation for a vector. The implementation of our approach is shown in Algorithm 5, which is similar to Algorithm 4. The only difference is that, in Line 8 of Algorithm 5, our approach needs to do the multi-dimensional relaxation for each  $\mathcal{RP}_{s,d}^i$ . After operating the multi-dimensional relaxation for each vector in each  $\mathcal{RP}_{s,d}^i$ , node  $s$  finds all the representative vectors in  $\bigcup_{i=1}^{\mathcal{K}} \mathcal{RP}_{s,d}^i$ , denoted by  $\mathcal{RP}_{s,d}$ , to define the estimated supported QoS. It is noteworthy that, when we apply the multi-dimensional relaxation, in Line 13 of Algorithm 3, we should check each vector in  $\mathcal{SR}_{u,d}$  but not  $\mathcal{SR}_{u,d}^i$ . For example, For approximating the supported QoS illustrated in Fig. 4(a), our approach first computes  $\mathcal{RP}^1$  and  $\mathcal{RP}^2$  illustrated by Fig. 4(b) and Fig. 4(c), respectively. There are five representative vectors in  $\mathcal{RP}^1$ , and we compute  $\mathbf{W}^2(c_1)$ , where  $c_1 \in \{1, 1.4, 1.5, 1.7, 2\}$ . We thus obtain the supported QoS defined by  $\mathcal{RP}^1$  operated by the multi-dimensional relaxation, which is illustrated by Fig. 4(e). Similarly, we also obtain the supported QoS defined by  $\mathcal{RP}^2$  operated by the multi-dimensional relaxation, which is illustrated by Fig. 4(f). Combining Fig. 4(e) and Fig. 4(f), the supported QoS estimated by our approach is the same as the real QoS in Fig. 4(a).

*Lemma 7:* Given a vector  $(w_1, \dots, w_{\mathcal{K}})$  in  $\mathcal{SR}_{s,d}$  computed by our approach,  $w_i$  must be the  $i^{\text{th}}$  weight of a certain path for all  $i = 1, \dots, \mathcal{K}$ .

*Proof:* By the description of our approach,  $w_i$  is computed as the estimated minimum  $i^{\text{th}}$  weight of all the paths satisfying a given request. We prove by induction.

As the basic step, all the nodes 1 hop away from  $d$  compute the supported QoS from themselves to  $d$ . By (3), if a 1-hop path satisfies a given request,  $w_i$  must be the  $i^{\text{th}}$  weight of a link.

For the inductive step, assume that the  $i^{\text{th}}$  value of each vector in  $\mathcal{RP}$  computed by each node  $h - 1$  hop away from  $d$  must be the  $i^{\text{th}}$  weight of a  $(h - 1)$ -hop path in network for all possible  $i$ . If a node  $h$  hop away from  $d$  finds a  $h$ -hop path satisfying a given request, the estimated  $i^{\text{th}}$  weight  $w_i$  is the sum of a  $(h - 1)$ -hop path and a link. Based on the inductive assumption,  $w_i$  is the real  $i^{\text{th}}$  weight of this  $h$ -hop path. In other words,

let  $c_i = \mathbf{W}_{s,d}^i(c_1, \dots, c_{i-1}, c_{i+1}, \dots, c_{\mathcal{K}})$ . If  $c_i < \infty$ , it must be the  $i^{\text{th}}$  weight of a path satisfying the request  $(c_1, \dots, c_{i-1}, \infty, c_{i+1}, \dots, c_{\mathcal{K}})$ . ■

*Theorem 1:* If  $rp = (w_1, \dots, w_{\mathcal{K}})$  in  $\mathcal{RP}_{s,d}$  corresponds to  $p_{s,d}$  from  $s$  to  $d$ ,  $rp$  is the real QoS parameter of  $p_{s,d}$ .

*Proof:* Lemma 7 shows that  $w_i$  must be the  $i^{\text{th}}$  weight of a path  $p_i$  in network for all  $i = 1, \dots, \mathcal{K}$ . We now show that  $p_i$  is  $p_{s,d}$  for all  $i = 1, \dots, \mathcal{K}$ . Without loss of generality, we assume that  $rp$  is found by doing the multi-dimensional relaxation for the vector  $ap$  in  $\mathcal{RP}_{s,d}^1$ . By the description of our approach, denote  $ap$  as  $(w_1, v_2, \dots, v_{\mathcal{K}})$ , where  $w_j \leq v_j$  for all  $j = 2, \dots, \mathcal{K}$ . Since  $p_{s,d}$  satisfies request  $(w_1, \dots, w_{\mathcal{K}})$ , it holds that  $\omega_i(p_{s,d}) \leq w_i$  for all  $i = 1, \dots, \mathcal{K}$ . We know that  $w_1 = \mathbf{W}_{s,d}^1(v_2, \dots, v_{\mathcal{K}})$ . Since  $p_{s,d}$  must satisfy the request  $(\infty, v_2, \dots, v_{\mathcal{K}})$ , we have  $w_1 \leq \omega_1(p_{s,d})$ . Therefore,  $w_1 = \omega_1(p_{s,d})$ . With the similar deduction, we can prove that  $w_i = \omega_i(p_{s,d})$  for all  $i = 1, \dots, \mathcal{K}$ . ■

Theorem 1 says that our multi-dimensional relaxation algorithm can find the real QoS of some (but not all) paths. Since existing algorithms cannot guarantee this feature, we can conclude that our approach achieves a better estimation. Note that Theorem 1 does not mean that each vector in  $\mathcal{RP}_{s,d}$  must be the QoS parameter of a non-dominated path. It is possible that a vector in  $\mathcal{RP}_{s,d}$  corresponds to a dominated path. A node may not be able to find all the non-dominated paths from itself to a destination. For instance, assume that each link is associated with two additive metrics. There exist three paths  $p_{1,v}$ ,  $p_{2,v}$ , and  $p_{3,v}$  from  $v$  to  $d$  with the QoS parameters  $(2.1, 1.8)$ ,  $(2, 2)$ , and  $(1.8, 2.1)$ , respectively. Assume that the QoS parameter of the link  $(s, v)$  is  $(1, 1)$ . There also exists another path from  $s$  to  $d$ , denoted by  $p'_s$ , with the QoS parameter  $(3.1, 3)$ . Assume that path  $p_{2,s}$  is composed by  $(s, v)$  and  $p_{2,v}$ . We have  $\vec{\omega}(p_{2,s}) = (3, 3)$ . We can see that  $p'_s$  is dominated by  $p_{2,s}$ . However, by setting  $\delta = 0.3$ ,  $v$  will not find the QoS parameter of  $p_{2,v}$ , so that  $s$  will not find  $p_{2,s}$ . Hence,  $s$  considers  $p'_s$  as a non-dominated path. We can see that each node may not find all the non-dominated paths from itself to a destination. Given a non-dominated path  $p_{s,d}$  from  $s$  to  $d$  in network, let  $p_{u,d}$  be the portion of  $p_{s,d}$  from  $u$  to  $d$ , where  $u \in \mathcal{A}(s)$ . Assume that  $u$  has obtained the QoS parameter of  $p_{u,d}$ , we would like to compute the probability that  $s$  can find the QoS parameter of  $p_{s,d}$ .

To know how often that happens, we would like to compute the probability that a node find the QoS of a non-dominated path. Theoretically, the larger the probability, the smaller the approximation error. In order to gain some insight into the probability, we assume that the weight values of the paths are i.i.d and follow the uniform distribution  $\mathbf{U}[0, 1]$  [15], [24]. In the following discussion, we assume that we apply uniform sampling. We first introduce the following lemmas and corollary. They help us to determine the condition that node  $s$  cannot find the QoS parameter of a non-dominated path which does exist in the network.

*Lemma 8:* By using uniform sampling, if there exists a vector  $\vec{r} = (v_1, v_2, \dots, v_{\mathcal{K}})$  corresponding to a non-

dominated path  $p_{s,d}$  in  $\mathcal{RP}_{s,d}^1$ , we have  $v_1 = \omega_1(p_{s,d})$  and  $v_j - \delta < \omega_j(p_{s,d}) \leq v_j$  for all  $j = 2, \dots, \mathcal{K}$ .

*Proof:* Since  $\mathcal{RP}_{s,d}^1$  is based on  $\mathbf{W}_{s,d}^1(\bullet)$  as discussed in Section IV-B, there exists a neighbor  $u$  of  $s$ , such that  $v_1 = \mathbf{W}_{s,d}^1(v_2, \dots, v_{\mathcal{K}}) = \mathbf{W}_{u,d}^1(v'_2, \dots, v'_{\mathcal{K}}) + \omega_1(s, u)$ , where  $v'_j = v_j - \omega_j(s, u)$  for all possible  $j$  such that  $j \neq i$ . In Algorithm 3,  $v'_i = \mathbf{W}_{u,d}^1(v'_2, \dots, v'_{\mathcal{K}})$  must be the  $i^{\text{th}}$  value of a vector in  $\mathcal{RP}_{u,d}$ . By Lemma 7,  $v'_i$  must be the  $i^{\text{th}}$  weight of a path from  $u$  to  $d$ , which is the portion of  $p_{s,d}$ . Therefore,  $v_1$  must be  $\omega_i(p_{s,d})$ .

Since  $v_j$  is a sample, it can be represented by  $x_j\delta$ ,  $x_j \in \mathbf{Z}^+$ , where  $j = 2, \dots, \mathcal{K}$ . If  $\omega_j(p_{s,d}) \leq (x_j - 1)\delta$ , since  $(x_j - 1)\delta$  also is a sample, node  $s$  will get a vector  $(v_1, \dots, v_{j-1}, (x_j - 1)\delta, v_{j+1}, \dots, v_{\mathcal{K}})$ , so that  $\vec{r}$  is not a representative vector. That contradicts our assumption. Thus, we have  $(x_j - 1)\delta < \omega_j(p_{s,d}) \leq x_j\delta$ , which can also be represented by  $v_j - \delta < \omega_j(p_{s,d}) \leq v_j$ . ■

Similarly, we also have the following lemma.

*Lemma 9:* By using logarithmic sampling, if there exists a vector  $\vec{r} = (v_1, v_2, \dots, v_{\mathcal{K}})$  corresponding to a non-dominated path  $p_{s,d}$  in  $\mathcal{RP}_{s,d}^1$ , we have  $v_1 = \omega_1(p_{s,d})$  and  $\frac{v_j}{1+\delta} < \omega_j(p_{s,d}) \leq v_j$  for all  $j = 2, \dots, \mathcal{K}$ .

Suppose that a vector  $(c_1, \dots, c_{i-1}, w_i, c_{i+1}, \dots, c_{\mathcal{K}})$  in  $\mathcal{RP}_{s,d}^i$  corresponds to a non-dominated path  $p_{s,d}$ . Since  $\mathcal{RP}_{s,d}^i$  is based on  $\mathbf{W}_{s,d}^i(\bullet)$  as discussed in Section IV-B,  $w_i = \omega_i(p_{s,d})$ . We assume that, given two different  $p_1$  and  $p_2$ ,  $\omega_i(p_1) \neq \omega_i(p_2)$  for all possible  $i$ . Theorem 1 also means that if we find a vector corresponding to a non-dominated path  $p_{s,d}$  in any  $\mathcal{RP}^i$ , where  $i = 1, \dots, \mathcal{K}$ , we can find the QoS parameter of  $p_{s,d}$ .

Assume that there exists a non-dominated path  $p_{s,d}$  in the network. Define a sample  $v_j$  chosen from the sample sequence  $\{0, \delta, 2\delta, \dots\}$ , such that  $v_j - \delta < \omega_j(p_{s,d}) \leq v_j$  for all  $j = 1, \dots, \mathcal{K}$ . Lemma 8 also means that if there is vector in  $\mathcal{RP}^1$ , which corresponds to  $p_{s,d}$ , this vector can be represented by  $(\omega_1(p_{s,d}), v_2, \dots, v_{\mathcal{K}})$ . In other words, if the vector  $(\omega_1(p_{s,d}), v_2, \dots, v_{\mathcal{K}})$  is not a representative vector, there is no vector corresponding to  $p_{s,d}$  in  $\mathcal{RP}^1$ . We thus give the following corollaries.

*Corollary 1:* Given a non-dominated  $p_{s,d}$  in the network, define a sample  $v_j$  chosen from the sample sequence  $\{0, \delta, 2\delta, \dots\}$  such that  $v_j - \delta < \omega_j(p_{s,d}) \leq v_j$  for all  $j = 1, \dots, \mathcal{K}$ . By using uniform sampling, if the vector  $(v_1, \dots, v_{i-1}, \omega_i(p_{s,d}), v_{i+1}, \dots, v_{\mathcal{K}})$  is not a representative vector for all  $i = 1, \dots, \mathcal{K}$ , there is no vector corresponding to  $p_{s,d}$  in  $\bigcup_{i=1}^{\mathcal{K}} \mathcal{RP}^i$ . We thus cannot find the QoS parameter of  $p_{s,d}$ .

*Corollary 2:* Given a non-dominated  $p_{s,d}$  in the network, define a sample  $v_j$  chosen from the sample sequence  $\{1, (1+\delta), (1+\delta)^2, \dots\}$  such that  $\frac{v_j}{1+\delta} < \omega_j(p_{s,d}) \leq v_j$  for all  $j = 1, \dots, \mathcal{K}$ . By using logarithmic sampling, if the vector  $(v_1, \dots, v_{i-1}, \omega_i(p_{s,d}), v_{i+1}, \dots, v_{\mathcal{K}})$  is not a representative vector for all  $i = 1, \dots, \mathcal{K}$ , there is no vector corresponding to  $p_{s,d}$  in  $\bigcup_{i=1}^{\mathcal{K}} \mathcal{RP}^i$ . We thus cannot find the QoS parameter of  $p_{s,d}$ .

*Theorem 2:* Suppose that there exists a non-dominated path  $p_{s,d}$  from  $s$  to  $d$  in network. Let  $w = \max_i \omega_i(p_{s,d})$  for all possible  $i$ . The probability that node  $s$  cannot find  $p_{s,d}$  is less than  $\delta^2 \lambda^{2(\mathcal{K}-1)}$ , where  $0 < \lambda < 1$  is a minimum sample in the sample sequence such that  $w \leq \lambda$  and  $\delta$  is the sampling parameter.

*Proof:* If we apply uniform sampling, define a sample  $v_j$  chosen from the sample sequence  $\{0, \delta, 2\delta, \dots\}$  such that  $v_j - \delta < \omega_j(p_{s,d}) \leq v_j$  for all  $j = 1, \dots, \mathcal{K}$ . If we apply uniform sampling, define a sample  $v_j$  chosen from the sample sequence  $\{1, (1 + \delta), (1 + \delta)^2, \dots\}$  such that  $\frac{v_j}{1+\delta} < \omega_j(p_{s,d}) \leq v_j$  for all  $j = 1, \dots, \mathcal{K}$ . Let  $\vec{r}_i$  be  $(v_1, \dots, v_{i-1}, \omega_i(p_{s,d}), v_{i+1}, \dots, v_{\mathcal{K}})$ . We first compute the probability that the vector  $\vec{r}_1$  is not a representative vector. If there exists another path  $p_1$  such that  $\omega_1(p_1) < \omega_1(p_{s,d})$ ,  $\omega_j(p_1) \leq v_j$  for all  $j = 2, \dots, \mathcal{K}$ , we should have  $\mathbf{W}_{s,d}^1(v_2, \dots, v_{\mathcal{K}}) \leq \omega_1(p_1) < \omega_1(p_{s,d})$ , so that the vector  $\vec{r}_1$  is not a representative vector. We now analyze the probability that  $p_1$  exists.

Note that there must exist some  $k$ ,  $k = 2, \dots, \mathcal{K}$ , such that  $\omega_k(p_{s,d}) < \omega_k(p_1)$ , otherwise,  $p_{s,d}$  is dominated by  $p_1$ , since each weight of  $p_1$  is smaller than that of  $p_{s,d}$ . Without loss of generality, we assume that  $\omega_k(p_{s,d}) < \omega_k(p_1) \leq v_k$  for all  $k = h + 1, \dots, \mathcal{K}$

We first consider uniform sampling. Given two independent variables  $x$  and  $y$  with the uniform distribution  $U[0, 1]$ , it holds that  $\Pr(x - y \leq z) = z - \frac{z^2}{2}$ , where  $z \geq 0$ . Therefore, we compute the probability of  $\omega_k(p_{s,d}) < \omega_k(p_1) \leq v_j$  as follows.

$$\begin{aligned}
& \Pr(\omega_j(p_{s,d}) < \omega_j(p_1) \leq v_j) \\
&= \Pr(0 < \omega_j(p_1) - \omega_j(p_{s,d}) \leq v_j - \omega_j(p_{s,d})) \\
&< \Pr(0 < \omega_j(p_1) - \omega_j(p_{s,d}) \leq \delta) \\
&= \delta - \frac{\delta^2}{2} \\
&< \delta
\end{aligned} \tag{5}$$

We then consider logarithmic sampling. We have

$$\begin{aligned}
& \Pr(\omega_j(p_{s,d}) < \omega_j(p_1) \leq v_j) \\
&= \Pr\left(1 < \frac{\omega_j(p_1)}{\omega_j(p_{s,d})} \leq \frac{v_j}{\omega_j(p_{s,d})}\right) \\
&< \Pr\left(1 < \frac{\omega_j(p_1)}{\omega_j(p_{s,d})} \leq 1 + \delta\right) \\
&= \delta \omega_j(p_1) \\
&< \delta
\end{aligned} \tag{6}$$

Denote  $\Pr(p_1)$  as the probability that  $p_1$  exists. Since the probability of  $\omega_j(p_1) \leq w$ , denoted by  $\Pr(\omega_j(p_1) \leq w)$  is  $w$ , where  $0 < w < 1$ , we have

$$\begin{aligned}
\Pr(p_1) &< \prod_{k=1}^h \Pr(\omega_j(p_1) \leq v_k) \\
&\quad \cdot \prod_{k=h+1}^{\mathcal{K}} \Pr(\omega_j(p_{s,d}) < \omega_j(p_1) \leq v_j) \\
&< \prod_{i=1}^h v_i \delta^{\mathcal{K}-h}
\end{aligned} \tag{7}$$

Denote  $\lambda < 1$  as the maximum value of  $v_j$  for all  $j = 1, \dots, h$ . We thus have  $\Pr(p_1) < \lambda^h \delta^{\mathcal{K}-h}$ . Generally speaking,  $\delta < \lambda$ . When  $h = \mathcal{K} - 1$ ,  $\Pr(p_1)$  is maximized.

Denote  $\mathbf{Pr}_k$  as the probability that all the vectors  $\{\vec{\mathbf{r}}_i, i = 1, \dots, k\}$  are not the representative vectors.  $\mathbf{Pr}_k$  denotes the probability that  $s$  cannot find the QoS parameter of  $p_{s,d}$ . First, we have  $\mathbf{Pr}_1 < \delta\lambda^{\mathcal{K}-1}$ .

Now, we analyze the condition that the vector  $\vec{\mathbf{r}}_2$  is also not a representative vector. Actually, if  $\omega_2(p_1) < \omega_2(p_{s,d})$ , we should have  $\mathbf{W}_{s,d}^2(v_1, v_3, \dots, v_{\mathcal{K}}) < \omega_2(p_1)$ , so that  $\vec{\mathbf{r}}_2$  is not a representative vector. Thus, we have  $\mathbf{Pr}_2 < \delta\lambda^{\mathcal{K}-1}$ . Based on the same deduction, we obtain that  $\mathbf{Pr}_{\mathcal{K}-1} < \delta\lambda^{\mathcal{K}-1}$ .

We then analyze the condition that the vector  $\vec{\mathbf{r}}_{\mathcal{K}}$  is also not a representative vector. Since  $\omega_{\mathcal{K}}(p_1)$  must be larger than  $\omega_{\mathcal{K}}(p_{s,d})$ ,  $p_{s,d}$  provides smaller  $\mathcal{K}^{\text{th}}$  weight than  $p_1$  for the request  $(v_1, \dots, v_{\mathcal{K}-1}, \infty)$ . In order for the vector  $(v_1, \dots, v_{\mathcal{K}-1}, \omega_{\mathcal{K}}(p_{s,d}))$  not being a representative vector, there must exist another path  $p'$  such that  $\omega_j(p') \leq v_j$  for all  $j = 1, \dots, \mathcal{K} - 1$  and  $\omega_{\mathcal{K}}(p') < \omega_{\mathcal{K}}(p_{s,d})$ . With the same deduction, the probability that  $p'$  exists is less than  $\delta\lambda^{\mathcal{K}-1}$ . Therefore, we have  $\mathbf{Pr}_{\mathcal{K}} < \delta^2\lambda^{2(\mathcal{K}-1)}$ .  $\blacksquare$

By Theorem 2, with the increase of the number of constraints, the probability of each node finding all the non-dominated paths increases. This means that the approximation error reduces.

#### D. Extensions

In Algorithm 5, each nodes computes  $\mathcal{RP}^i$  for all  $i = 1, \dots, \mathcal{K}$ . Assume that  $\mathcal{RP}^i$  and  $\mathcal{RP}^j$ , where  $i \neq j$ , corresponds to the same set of paths. By Theorem 1,  $\mathcal{RP}^i$ , which is operated by the multi-dimensional relaxation, is the same as  $\mathcal{RP}^j$ . In this case, we just need to compute either of them. Accordingly, given  $\mathcal{T} \leq \mathcal{K}$ , we just let each node computes  $\mathcal{RP}^i$  for all  $i = 1, \dots, \mathcal{T}$ . The supported QoS is defined by all the representative vectors in  $\bigcup_{i=1}^{\mathcal{T}} \mathcal{RP}_{s,d}^i$ , denoted by  $\mathcal{RP}_{s,d}$ . We call such mechanism as  $\mathcal{T}$ -dimensional sampling mechanism.

By Theorem 2, if  $\mathcal{T} < \mathcal{K}$ , the probability that each node cannot find a non-dominated path  $p_{s,d}$  is less than  $\delta\lambda^{\mathcal{K}-1}$ , which is larger than the case of  $\mathcal{T} = \mathcal{K}$ . However, when  $\mathcal{T} < \mathcal{K}$ , we can apply smaller  $\delta$  due to the smaller computational overhead. We will evaluate the performances of the proposed algorithms with different  $\mathcal{T}$  by simulation experiments.

#### E. The complexities of our algorithms

In the approximation algorithms, we use the number of samples to denotes the time and spacial complexities. If we use uniform sampling, the maximum number of one-dimensional sampling is  $\mathcal{O}(\left(\frac{\mathcal{U}B}{\delta}\right)^{\mathcal{K}-1}) = \mathcal{O}\left(\left(\frac{|\mathcal{V}|^2 B}{\epsilon}\right)^{\mathcal{K}-1}\right)$  and that of multi-dimensional sampling is  $\mathcal{O}\left(\mathcal{K}\left(\frac{|\mathcal{V}|^2 B}{\epsilon}\right)^{\mathcal{K}-1}\right)$ . In our approach, each representative vector has to be performed by the multi-dimensional relaxation, which can be considered to compute additional  $\mathcal{K} - 1$  samples. Therefore, the time complexities of our approach is  $\mathcal{O}\left(\mathcal{T}\mathcal{K}\left(\frac{|\mathcal{V}|^2 B}{\epsilon}\right)^{\mathcal{K}-1}\right)$ , and the spacial complexity of our approach is  $\mathcal{O}\left(\mathcal{T}\left(\frac{|\mathcal{V}|^2 B}{\epsilon}\right)^{\mathcal{K}-1}\right)$ . As mentioned earlier, with the increase of  $\mathcal{K}$ , the additional complexity produced by our approach also increases exponentially with the number of constraints. However, we also formally proved that with the increase of  $\mathcal{K}$ , the approximation error produced by our approach reduces. In the next section, we would

like to use simulation experiments to evaluate the performance of these three approximation algorithms from the perspectives of the computational overhead and the approximation error.

## V. PERFORMANCE EVALUATION

In this section, we present our simulation results and compare our algorithm, MMAA (multi-dimensional relaxation multi-dimensional sampling approximation algorithm), with SSAA (single-dimensional relaxation single-dimensional sampling approximation algorithm) in [23], and SMAA (single-dimensional relaxation multi-dimensional sampling approximation algorithm).  $\mathcal{T}$ -MMAA denotes the multi-dimensional relaxation  $\mathcal{T}$ -dimensional sampling mechanism. In the following subsections, we discuss our simulation testbed, performance metrics, and our simulation results.

### A. Simulation testbed

We evaluate the performance of the algorithms by using a self-written C++ network simulator. The network topology is generated based on Waxman model by using the BRITE software [14]. We evaluate the performance of the algorithms in asymmetric networks. We generated the networks with 50 and 100 nodes. Each link in the networks is associated with three metrics in order to evaluate the impact of the number of the QoS constraints on the performance of the algorithms. The QoS metrics of each link, which are real numbers, are independently and uniformly distributed within  $[1, 100]$ . In each graph, we randomly select a node as the destination, and compute the supported QoS from each node to the destination. In each experiment, we simulated ten network instances.

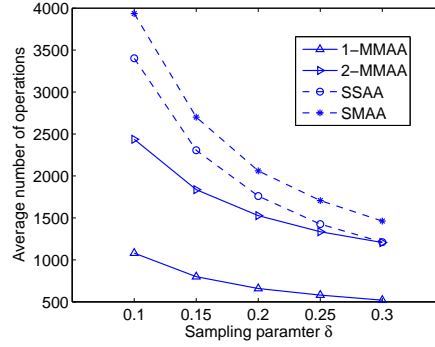
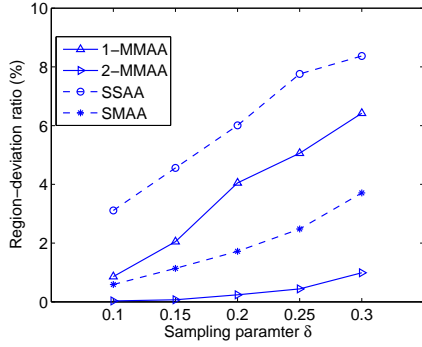
### B. Performance metrics

Define the region  $\mathbf{C} = (0, \mathcal{C}_1] \times \dots \times (0, \mathcal{C}_{\mathcal{K}}]$  as the request space. All the request requirements fall in  $\mathbf{C}$ . For each real representative vector  $(w_1, \dots, w_{\mathcal{K}})$ , the supported feasible region is  $[w_1, \mathcal{C}_1] \times \dots \times [w_{\mathcal{K}}, \mathcal{C}_{\mathcal{K}}]$ . The feasible region is thus composed by multiple hypercubes, and the total volume of these hypercubes, defined as  $\prod_{i=1}^{\mathcal{K}} (\mathcal{C}_i - w_i)$ , represents the size of the feasible region. Therefore, we use the sum of the volumes of these hypercubes to represent the size of the feasible region.

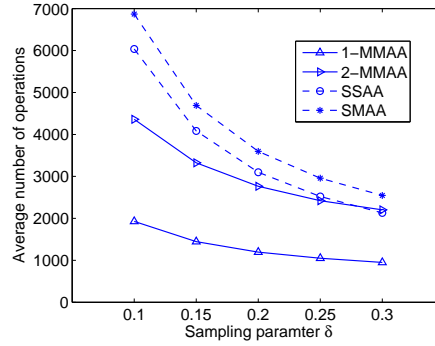
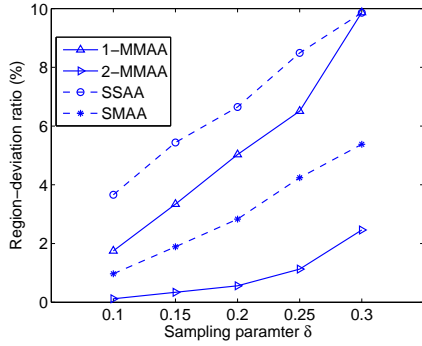
We use the exhaustive method to find the real representative vector set from  $s$  to  $d$ . We set  $\mathcal{C}_i$  to be the maximum  $i^{\text{th}}$  weight value among all the real representative vectors, for all  $i = 1, 2, \dots, \mathcal{K}$ . Denote the volume of the optimal feasible region spanned by the set of the representative vectors as  $V_r$  and the volume of the approximated feasible region computed by the approximation algorithm as  $V_a$ . We compute the *region-deviation ratio*, which is defined as  $\frac{V_r - V_a}{V_r}$ , to evaluate the accuracy performance of the algorithms.

From Algorithms 1, 4, and 5, we can see that the computational overheads are composed by two parts: computing  $\mathbf{W}^i(\bullet)$  and finding representative vectors. For testing the computational overhead, we ignore the part of finding representative vectors. This means that we do not consider the time consumed by Line 7-13 of Algorithm 2, that consumed by Line 10-25 of Algorithm 4, and that consumed by Line 10-26 of Algorithm 5. Since comparing with

the number of samples contained by  $\mathbf{W}_{s,d}^i(\bullet)$ , the relative number of representative vectors is very small. Therefore, the computational overhead produced by SMAA is approximately  $\mathcal{K}$  times of that by SSAA, and that produced by  $\mathcal{T}$ -MMAA is approximately  $\mathcal{T}$  times of that by SSAA.



(a) Region-deviation probability with 50-node domains. (b) Computational overhead with 50-node domains.

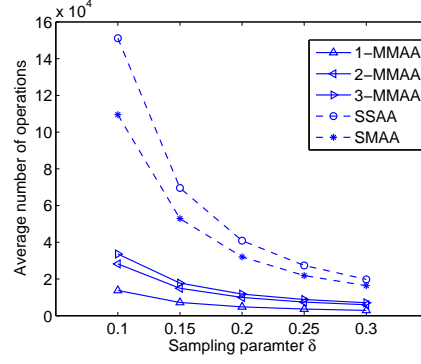
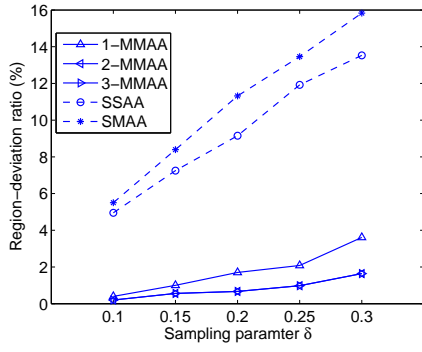


(c) Region-deviation probability with 100-node domains. (d) Computational overhead with 100-node domains.

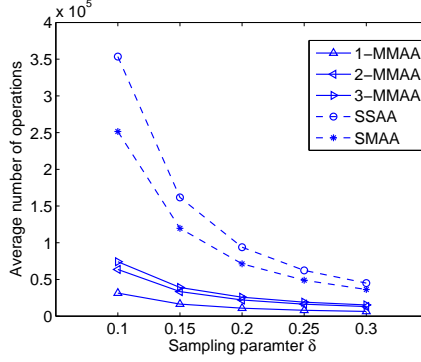
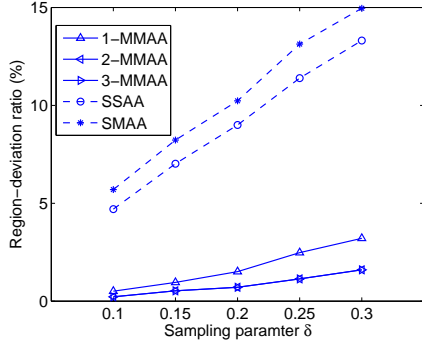
Fig. 5. The performance of the approximation algorithms with two weights.

### C. Simulation results

As we know, the performance of the approximation algorithms depend on the sampling parameter  $\delta$ . The smaller  $\delta$  reduces the approximation error at the expense of increasing the computational overhead. In order to represent the performance difference among  $\mathcal{T}$ -MMAA, SSAA, and SMAA clearly, we set different sampling parameters for them, so that both the computational overheads of SSAA and SMAA are higher than that of  $\mathcal{T}$ -MMAA. In this case, if the approximation error of  $\mathcal{T}$ -MMAA is the lowest, we can say that  $\mathcal{T}$ -MMAA outperforms the existing algorithms. In our simulation, the sampling parameter of MMAA is four times that of SSAA and two times that of SMAA. For the figures describing our simulation results, the x-coordinate denotes the sampling parameter of MMAA. In our simulations, we just show the results by using logarithmic sampling, since the trail of uniform sampling is same as logarithmic sampling [10].



(a) Region-deviation probability with 50-node domains. (b) Computational overhead with 50-node domains.



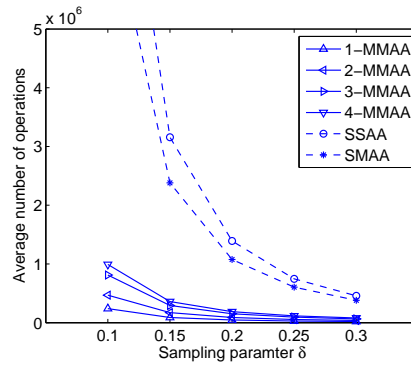
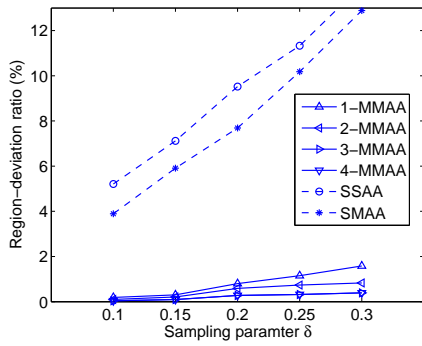
(c) Region-deviation probability with 100-node domains. (d) Computational overhead with 100-node domains.

Fig. 6. The performance of the approximation algorithms with three weights.

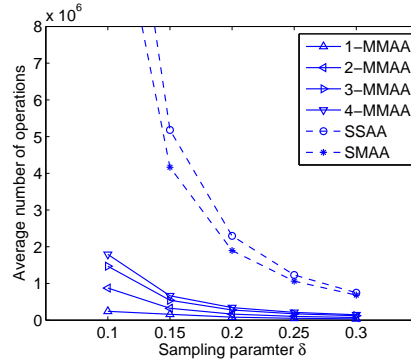
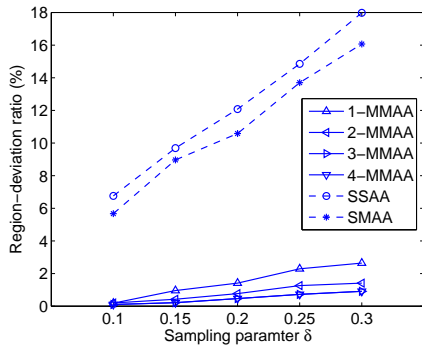
The simulation results are shown in Fig. 5(c), Fig. 6(c), and Fig. 7(c). Generally speaking, the approximation error and the computational overhead produced by our approach is the least. This means that our approach outperforms the existing algorithms. When the number of constraints is three or four, the approximation error and the computational overhead produced by one-dimensional sampling and multi-dimensional sampling are comparable. This means that the performance enhancement of multi-dimensional sampling becomes less with the increase of the number of constraints.

In Fig. 6(c), we can observe that multi-dimensional sampling mechanism outperforms 1-MMAA, so, when  $\mathcal{K} = 2$ , we should apply 2-MMAA. When  $\mathcal{K} = 3$  or 4, even 1-MMAA outperforms the existing algorithms. When  $\mathcal{K} = 3$ , the approximation error of 2-MMAA and 3-MMAA are almost the same, but the computational overhead of 2-MMAA is smaller. Similarly, when  $\mathcal{K} = 4$ , the approximation error of 3-MMAA and 4-MMAA are almost the same, but the computational overhead of 3-MMAA is smaller. We then conclude that  $\mathcal{K} - 1$ -MMAA outperforms  $\mathcal{K}$ -MMAA when  $\mathcal{K} > 2$ .

In Fig. 6, when  $\delta = 0.3$ , the computational overhead of 1-MMAA, 2-MMAA, and 3-MMAA are comparable,



(a) Region-deviation probability with 50-node domains. (b) Computational overhead with 50-node domains.



(c) Region-deviation probability with 100-node domains. (d) Computational overhead with 100-node domains.

Fig. 7. The performance of the approximation algorithms with four weights.

but the approximation error of 2-MMAA is lower than that of 1-MMAA. When  $\delta = 0.1$ , the approximation error of 1-MMAA, 2-MMAA, and 3-MMAA are comparable, but the computational overhead of 1-MMAA is the lowest. In Fig. 7, when  $\delta = 0.3$ , the performance of 3-MMAA is the highest. When  $\delta = 0.1$ , the performance of 1-MMAA is the highest. Therefore, we say that when setting larger sampling parameter, the performance of  $(\mathcal{K} - 1)$ -MMAA is the highest. when setting the smaller sampling parameter, the performance of 1-MMAA is the highest.

We formally proved that, with the increase of the number of constraints, the probability that our algorithm can find all the non-dominated paths increases. The simulation results shows that the approximation error produced by our algorithms changes slowly with the increase of the number of constraints. This means that the number of constraints cannot affect significantly the performance of our algorithms. However, we can observe that the approximation produced by the existing algorithms grows with the number of constraints.

## VI. CONCLUSION

In this paper, we studied the problem of QoS routing with multiple additive constraints, which is NP-Complete. A lot of works study the problem of finding a feasible path for a given request. While, we study a more challengeable

problem, which is precomputing the supported QoS between any two nodes. We apply the sampling approximation mechanism. For the sampling approximation method, smaller sampling parameter produces smaller approximation error and larger computational overhead. In order to be an  $\epsilon$ -approximation algorithm, we give the upper bound of the sampling parameter so as to minimize the computational overhead. Moreover, we propose a new approach, *multi-dimensional relaxation mechanism*, to improve the performance of the approximation algorithm. In our approach, the estimated QoS parameter of a path in the network is the real one, while the existing approximation algorithm cannot guarantee this feature. Therefore, our approach produces smaller approximation error than the existing algorithms. By using the multi-dimensional relaxation mechanism, we propose a set of approximation algorithms. Since our algorithms may not be able to find the QoS parameters of all the non-dominated paths, which define the real supported QoS. We thus give a theoretical analysis, and we show that the probability that our algorithms can find the real supported QoS increases with the increase of the number of constraints. By doing the extensive simulations, we show that our algorithms outperform the existing approximation algorithms. We also analyze the impact of the number of constraints on the performance of the approximation algorithms and evaluate the performance difference of the proposed approximation algorithms. In this work, we consider the computational of the supported QoS between any two nodes in flat networks. However, due to the scalability, the hierarchical networks are widely applied, such as the Internet. How to compute the supported QoS between any two nodes in the hierarchical networks is still an unsolved problem.

## REFERENCES

- [1] L. L. H. Andrew and A. A. N. A. Kusuma, "Generalised analysis of a QoS-Aware routing algorithms," *IEEE Globecom'98*, vol. 1, pp. 1-6, August 1998.
- [2] S. Chen, M. Song, and S. Sahni, "Two techniques for fast computation of constrained shortest paths," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 105-114, February 2008.
- [3] S. Chen and K. Nahrstedt, "On finding multi-constrained paths," in *Proceedings of ICC'98*, vol. 2, pp. 874-879, June 1998.
- [4] S. Chen and K. Nahrstedt, "An overview of quality-of-service routing for next-generation high-speed networks: problems and solutions," *IEEE Networks*, vol. 12, no. 6, pp. 64-79, November 1998.
- [5] Y. Cui, K. Xu, and J. Wu, "Precomputation for multi-constrained QoS routing in high speed networks," *Journal of Computer Networks*, vol. 47, no. 6, pp. 923-937, April 2005.
- [6] Y. Cui, K. Xu, J. P. Wu, and M. W. Xu, "Precomputation for finding paths with two additive weights," in *proceedings of ICC'03*, vol. 1, pp. 636-640, 11-15 May 2003.
- [7] A. Goel, K. G. Ramakrishnan, D. Kataria, and D. Logothetis, "Efficient computation of delay-sensitive routes for one source to all destinations," in *Proc. IEEE INFOCOM*, p. 854-858, April 2001.
- [8] R. Guerin and A. Orda, "QoS routing in networks with inaccurate information: theory and algorithms," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 350-364, April 1999.
- [9] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Mathematics of Operations Research*, vol. 17, no. 1, pp. 36-42, February 1992.
- [10] R. Hou, K.-S. Lui, K.-C. Leung, and F. Baker, "Quality-of-Service Routing with Two additive constraints in Hierarchical Networks," *to appear in IEEE ICNP2008*.

- [11] J. M. Jaffe, "Algorithms for finding paths with multiple constraints," *Networks*, vol. 14, no. 1, pp. 95-116, October 1984.
- [12] K.-S. Lui, K. Nahrstedt, and S. Chen, "Routing with topology aggregation in delay-bandwidth sensitive networks," *IEEE/ACM Transactions on Networking*, 12(1): 17-29, February 2004.
- [13] D. H. Lorenz and D. Raz, "A simple efficient approximation acheme for the restricted shortest path problem," *Operations Research Letters*, vol. 28, pp. 213-219, March 2001.
- [14] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: an approach to universal topology generation," in *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS'01*, pp. 346-353, August 2001.
- [15] P. W. Miegheem and F. A. Kuipers, "On the complexity of QoS routing," *Computer communications*, vol. 26, no. 4, pp. 376-387, March 2003.
- [16] A. Orda and A. Sprintson, "Precomputation schemes for QoS routing," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 578-591, August 2003.
- [17] S. Sahni, "General techniques for combinatorial approximation," *Operational Research*, vol. 25, no. 6, pp. 920-936, November/December, 1977.
- [18] A. Shaikh, J. Rexford, and K. G. Shin, "Evaluating the impact of stale link state on quality-of-service routing," *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, pp. 162-186, April 2001.
- [19] M. Song and S. Sahni, "Approximation algorithms for multiconstrained quality-of-service routing," *IEEE Transactions on computers*, vol. 55, no. 5, pp. 603-617, May 2006.
- [20] S. Uludag, K.-S. Lui, K. Nahrstedt, G. Brewster, "Analysis of Topology Aggregation Techniques for QoS Routing," *ACM Computing Surveys (CSUR)*, vol. 39, no. 3, Article No. 7, 2007.
- [21] G. L. Xue and S. K. Makki, "Multiconstrained QoS routing: a norm approach," *IEEE Transactions on Computers*, vol. 56, no. 6, pp. 859-863, June 2007.
- [22] G. Xue, W. Zhang, J. Tang, and K. Thulasiraman, "Polynomial time approximation algorithms for multi-constrained QoS routing," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 656-669, June, 2008.
- [23] X. Yuan, "Heuristic algorithm for multiconstrained quality-of-service routing," *IEEE/ACM Transactions on Networking*, vol. 10, no. 2, pp. 244-256, April 2002.
- [24] T. Zhang, Y. Cui, Y. Zhao, L. Fu, and T. Korkmaz, "Scalable BGP QoS Extension with Multiple Metrics," in *Proceedings of 2006 International Conference on Networking and Services (ICNS'06)*, pp. 80-85, 2006.
- [25] Y. Zheng, T. Korkmaz, and W. Dou, "Pareto Optimal based partition framework for two additive constrained path selection", in *proceedings of IEEE International Conference on Networking (ICN'05)*, pp. 1-8, April 2005.
- [26] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 7, pp. 1228-1234, September 1996.

## APPENDIX

---

**Procedure 1**  $\text{PMCOP\_I}(i)$ 

---

parameters

$q$ -an array with  $\mathcal{K}$  doubles

```
1:  $h \leftarrow 0$ 
2: while  $h < |\mathcal{V}| - 1$  do
3:   for each  $s \in \mathcal{V} \setminus d$  do
4:      $\mathcal{SR}_{s,d}^i \leftarrow \emptyset$ 
5:      $\text{APPWCF}(i, 0, q, \mathcal{SR}_{s,d}^i)$ 
6:   end for
7:    $h \leftarrow h + 1$ 
8: end while
```

---

---

**Procedure 2**  $\text{APPWCF}(i, k, q)$ 

---

```
1: if  $k = i$  then
2:    $\text{APPWCF}(i, k + 1, q, \mathcal{SR}_{s,d}^i)$ 
3: end if
4: if  $k = \mathcal{K}$  then
5:    $q[i] \leftarrow \text{RELAX}(i, (q[0], \dots, q[i - 1], q[i + 1], \dots, q[\mathcal{K}]))$ 
6:   for each  $\vec{w}$  in  $\mathcal{SR}_{s,d}^i$  do
7:     if  $\vec{w} \prec (q[0], \dots, q[\mathcal{K}])$  then
8:       return
9:     end if
10:    if  $(q[0], \dots, q[\mathcal{K}]) \prec \vec{w}$  then
11:      Remove  $\vec{w}$  from  $\mathcal{SR}_{s,d}^i$ 
12:    end if
13:  end for
14:  Add  $(q[0], \dots, q[\mathcal{K}])$  in  $\mathcal{SR}_{s,d}^i$ 
15:  return
16: end if
17: for each  $j = 0, 1, 2, \dots, \mathcal{UB}$  do
18:    $q[k] \leftarrow j$ 
19:    $\text{APPWCF}(i, k + 1, q)$ 
20: end for
```

---

---

**Procedure 3 RELAX** $(i, (q_1, \dots, q_{i-1}, q_{i+1}, \dots, q_{\mathcal{K}}))$ 

---

```
1:  $q_i \rightarrow \infty$ 
2: for each  $u \in \mathcal{A}(s)$  do
3:   for each  $j = 1, \dots, i-1, i+1, \dots, \mathcal{K}$  do
4:      $q'_j \leftarrow q_j - \omega_j(s, u)$ 
5:   end for
6:   if  $u = d$  then
7:     if  $q'_j \geq 0$  for all  $j = 1, \dots, \mathcal{K}$  and  $j \neq i$  then
8:       if  $\omega_i(s, u) < q_i$  then
9:          $q_i \leftarrow \omega_i(s, u)$ 
10:      end if
11:    end if
12:   else
13:     for each  $(w_1, \dots, w_{\mathcal{K}})$  in  $\mathcal{SR}_{u,d}^i$  do
14:       if  $(w_1, \dots, w_{\mathcal{K}}) \preceq (q'_1, \dots, q'_{i-1}, \infty, q'_{i+1}, \dots, q'_{\mathcal{K}})$  then
15:         if  $w_i + \omega_i(s, u) < q_i$  then
16:            $q_i \leftarrow w_i + \omega_i(s, u)$ 
17:         end if
18:       end if
19:     end for
20:   end if
21: end for
22: return  $q_i$ 
```

---

---

**Procedure 4** PMCOP\_SM

---

parameters

 $q$ -an array with  $\mathcal{K}$  doubles $b$ -boolean variable

```
1:  $h \leftarrow 0$ 
2: while  $h < |\mathcal{V}| - 1$  do
3:   for each  $s \in \mathcal{V} \setminus d$  do
4:      $\mathcal{SR}_{s,d} \leftarrow \emptyset$ 
5:     for each  $i = 1, \dots, \mathcal{K}$  do
6:        $\mathcal{SR}_{s,d}^i \leftarrow \emptyset$ 
7:       APPWCF( $i, 0, q, \mathcal{SR}_{s,d}^i$ )
8:     end for
9:      $\mathcal{SR}_{s,d} \leftarrow \mathcal{SR}_{s,d}^1$ 
10:    for each  $i = 2, \dots, \mathcal{K}$  do
11:      for each  $\vec{w}_1$  in  $\mathcal{SR}_{s,d}^i$  do
12:         $b \leftarrow \text{true}$ 
13:        for each  $\vec{w}_2$  in  $\mathcal{SR}_{s,d}$  do
14:          if  $\vec{w}_2 \preceq \vec{w}_1$  then
15:             $b \leftarrow \text{false}$ 
16:            break
17:          else
18:            Remove  $\vec{w}_2$  from  $\mathcal{SR}_{s,d}$ 
19:          end if
20:        end for
21:        if  $b$  then
22:          Add  $\vec{w}_1$  in  $\mathcal{SR}_{s,d}$ 
23:        end if
24:      end for
25:    end for
26:  end for
27:   $h \leftarrow h + 1$ 
28: end while
```

---

---

**Procedure 5** PMCOP\_MM

---

parameters

$q$ -an array with  $\mathcal{K}$  doubles

$b$ -boolean variable

```
1:  $h \leftarrow 0$ 
2: while  $h < |\mathcal{V}| - 1$  do
3:   for each  $s \in \mathcal{V} \setminus d$  do
4:      $\mathcal{SR}_{s,d} \leftarrow \emptyset$ 
5:     for each  $i = 1, \dots, \mathcal{K}$  do
6:        $\mathcal{SR}_{s,d}^i \leftarrow \emptyset$ 
7:       APPWCF( $i, 0, q, \mathcal{SR}_{s,d}^i$ )
8:       ERELAX( $i, \mathcal{SR}_{s,d}^i$ )
9:     end for
10:     $\mathcal{SR}_{s,d} \leftarrow \mathcal{SR}_{s,d}^1$ 
11:    for each  $i = 2, \dots, \mathcal{K}$  do
12:      for each  $\vec{w}_1$  in  $\mathcal{SR}_{s,d}^i$  do
13:         $b \leftarrow \text{true}$ 
14:        for each  $\vec{w}_2$  in  $\mathcal{SR}_{s,d}$  do
15:          if  $\vec{w}_2 \preceq \vec{w}_1$  then
16:             $b \leftarrow \text{false}$ 
17:            break
18:          else
19:            Remove  $\vec{w}_2$  from  $\mathcal{SR}_{s,d}$ 
20:          end if
21:        end for
22:        if  $b$  then
23:          Add  $\vec{w}_1$  in  $\mathcal{SR}_{s,d}$ 
24:        end if
25:      end for
26:    end for
27:  end for
28:   $h \leftarrow h + 1$ 
29: end while
```

---

---

**Procedure 6** ERELAX $\{i, \mathcal{SR}_{s,d}^i\}$ 

---

```
1: for each vector  $(w_1, \dots, w_{\mathcal{K}})$  in  $\mathcal{SR}_{s,d}^i$  do
2:   for  $j = 1, \dots, i - 1, i + 1, \dots, \mathcal{K}$  do
3:      $w_j \leftarrow \text{RELAX}(j, (w_1, \dots, w_{j-1}, w_{j+1}, \dots, w_{\mathcal{K}}))$ 
4:   end for
5: end for
```

---