Approximation Algorithm 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

Fred Baker
Cisco Research Center
San Jose, CA 95134, USA

***Abstract*—In this paper, we study the problem of computing 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 approximation algorithm called *multi-dimensional relaxation algorithm*. We formally prove that our algorithm produces smaller approximation error than the existing algorithms. We further verify the performance by extensive simulations.**[1]

I. INTRODUCTION

Precomputing the supported QoS from a source to a destination has several merits. Based on the precomputation of the supported QoS, the source can immediately respond an incoming connection request and find a feasible path for it. Moreover, precomputation can be carried out when the network element is idle or underutilized, so that the network resources can be effectively utilized [8]. Besides, precomputation facilitates scalability. In the Internet, nodes are grouped into multiple domains. Each border node in a domain must advertise the QoS of routing across its own domain to the border nodes in other domains. Therefore, each border node needs to precompute the supported QoS from itself to the other border nodes in the same domain [5].

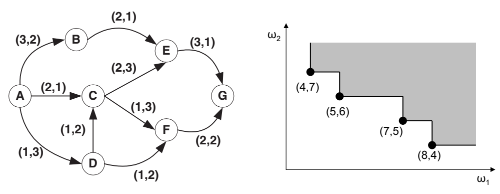
In a traditional routing protocol, for each destination, a node keeps only one path, or the next hop, and this path is the "best" path among all possible routes. However, when there are more than one constraint, "best" is not well-defined. For example, in Fig. 1(a), each link is associated with two additive metrics (ω_1, ω_2), where ω_1 is the first metric and ω_2 is the second metric. There are totally six paths from A to G. Since the metrics of each link are additive, each metric of a path is the sum of the metric of each link on this path. 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 ω_1 while p_3 is better in terms of metric ω_2. If there is no path that can dominate p_1, we call p_1 a *non-dominated path*. Different non-dominated paths define different supported QoSes. For example, given a request with the QoS requirements $(4, 8)$, p_3 is feasible but p_1 is not. On the other hand, p_1 can support the request with the QoS requirements $(7, 5)$ but p_3 cannot. In order to select a path for any incoming request, a node has to maintain all the non-dominated paths from itself to a certain destination. In other words, the supported QoS is defined by all the non-dominated paths. One exhaustive method to compute the supported QoS is to find the QoS parameters of all the paths, and then select the non-dominated paths. However, the time complexity of this method is exponential to the number of the nodes in the network.

[1]This work is supported in part by the Cisco Research Initiative Award.

There have been many works on QoS routing with multiple additive constraints [10]. Most of the existing works study the problem of finding a feasible path for a given request with the QoS requirements. Such works cannot be applied for precomputing the supported QoS. Several heuristic methods [3], [4], [13] were proposed for precomputing the supported QoS between any two nodes. In [4], [13], the problem with two additive constraints was considered. The problem with multiple additive constraints was studied in [3]. Unfortunately, the heuristic methods cannot provide any performance guarantee. In [8], [11], the approximation algorithms were proposed for the problem. Both apply in essence the same approximation method, which is interval partition [9]. The only difference is that the algorithm in [8] applies logarithmic sampling, while that in [11] applies uniform sampling. In [5], the performance of these two different sampling methods for computing the supported QoS with two additive constraints was studied. A new approximation algorithm, known as the *two-dimensional sampling mechanism*, was proposed. The two-dimensional sampling mechanism can use either uniform sampling or logarithmic sampling. We show, in this paper, that the two-dimensional sampling scheme can be extended to a multi-dimensional sampling method for computing the supported QoS with multiple additive constraints. Unfortunately, the performance enhancement of the multi-dimensional sampling method over one-dimensional sampling decreases as the number of constraints increases. Accordingly, in this paper, we propose a new approach, called *multi-dimensional relaxation*, which improves the accuracy of the approximation algorithm further. We investigate the performance of the proposed approach by extensive simulations.

II. NETWORK MODEL AND PROBLEM FORMULATION

This section formulates the general model addressed in this paper. A network is modeled as a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$.



(a) Network topology. (b) ω_1-ω_2 plane.

Fig. 1. Example for illustrating the supported QoS.

$\mathcal{V}$ 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 a neighbor of v_i. 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^{th} weight for link e. $\vec{\boldsymbol{\omega}}(e) = (\omega_1(e), \omega_2(e), \ldots, \omega_{\mathcal{K}}(e))$ is the weight vector for link e, and is also called the QoS parameter for link e. For example, in Fig. 1(a), link $e = (A, B)$ is associated with two weights such that $\omega_1(e) = 3$, $\omega_2(e) = 2$, and $\vec{\boldsymbol{\omega}}(e) = (3, 2)$. Let p be a path in $\mathcal{G}$. The k^{th} weight of p, denoted as $\omega_k(p)$, is the sum of the k^{th} weight for all links along p. $\vec{\boldsymbol{\omega}}(p) = (\omega_1(p), \omega_2(p), \ldots, \omega_{\mathcal{K}}(p))$ is the weight vector for path p. Given the path $p = \{A, B, E, G\}$ in Fig. 1(a), we have $\omega_1(p) = 8$, $\omega_2(p) = 4$, and hence $\vec{\boldsymbol{\omega}}(p) = (8, 4)$. In the previous section, we mentioned that the supported QoS from s to d is defined by all the non-dominated paths from s to d. We call the QoS parameters of the non-dominated paths the *representative vectors*.

Each request is associated with $\mathcal{K}$ constraints. We denote the requirement on the j^{th} constraint as c_j. That is, a request is represented as $(c_1, c_2, \ldots, c_{\mathcal{K}})$. A path p can support request $(c_1, c_2, \ldots, c_{\mathcal{K}})$ if $\omega_j(p) \leq c_j$ for all $j = 1, \ldots, \mathcal{K}$. Not all requests have constraints on every weight. When there is no constraint on the i^{th} weight, we put c_i to be ∞, where $i = 1, \ldots, \mathcal{K}$. There may be more than one path that satisfies a request. Among the paths that satisfy request $(c_1, \ldots, c_{i-1}, c_i = \infty, c_{i+1}, \ldots, c_{\mathcal{K}})$, we let $\mathbf{W}_{s,d}^{\text{opt},i}(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{\mathcal{K}})$ be $\omega_i(p)$ where p is the path with the minimum $\omega_i(p)$ among all the paths from s to d that satisfy the request. For example, let each link e in the network be associated with the QoS parameter pair $(\omega_1(e), \omega_2(e), \omega_3(e))$. 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. $\mathbf{W}_{s,d}^{\text{opt},i}(\bullet)$[2] denotes the minimum i^{th} weights with all possible constraint tuples, and is also called the minimum i^{th} weight function, where $i = 1, \ldots, \mathcal{K}$. For example, in Fig. 1(a), We can obtain $\mathbf{W}_{A,G}^{\text{opt},1}(\bullet)$. For instance, $\mathbf{W}_{A,G}^{\text{opt},1}(c_2) = 4, 5, 7$, and 8 under the condition that $c_2 \geq 7$, $6 \leq c_2 < 7$, $5 \leq c_2 < 6$, and $4 \leq c_2 < 5$, respectively.

If we plot this function on the $\omega_1 - \omega_2$ plane, the function is defined by a staircase, as illustrated in Fig. 1(b). This staircase is defined by four points, which actually are all the representative vectors. The shaded area defines the feasible region. All the requests falling in this region can be supported by at least one path from A to G. We can see that the shaded area is actually defined by the staircase or the representative vectors. Therefore, computing the supported QoS is actually to compute the i^{th} minimum weight function $\mathbf{W}_{s,d}^{\text{opt},i}(\bullet)$ for any given $i = 1, \ldots, \mathcal{K}$, or to find the QoS parameters of all the non-dominated paths from s to d. Note that for different i and j, $\mathbf{W}^{\text{opt},i}(\bullet)$ and $\mathbf{W}^{\text{opt},j}(\bullet)$ define the same supported QoS, since they are defined by the same representative vectors [14]. It has been proved that the problem of precomputing the supported QoS is NP-Complete. In the following section, we will discuss how to compute the approximated i^{th} weight function $\mathbf{W}_{s,d}^{i}(\bullet)$, where $i = 1, \ldots, \mathcal{K}$. For the ease of discussion, each link weight is bounded above by B, i.e., $\omega_i(e) \leq B$ for all $i = 1, \ldots, \mathcal{K}$ 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 $\mathcal{UB}$.

[2]When the context is clear, we drop the subscripts s and d from the weight function and simply denote the function as $\mathbf{W}^{\text{opt},i}$.

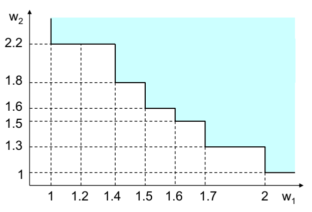
III. APPROXIMATION ALGORITHMS

A. Algorithm for links with integral weights

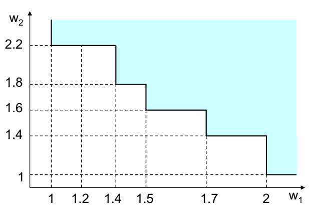
Without loss of generality, we are going to discuss how to compute $\mathbf{W}_{s,d}^{1}(\bullet)$. We first assume that the weights of each link are integers. The algorithm of computing the function is as follows.

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

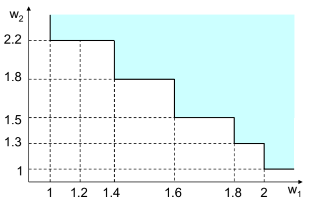
To compute $\mathbf{W}_{u,d}^{1}(\bullet)$ for all $u \in \mathcal{V}$, we keep a table of $|\mathcal{V}|$ rows and $(\mathcal{UB} + 1)^{\mathcal{K}-1}$ columns, where one row for each node and one column for each possible constraint tuple $(\infty, c_2, \ldots, c_{\mathcal{K}})$. Since there are $\mathcal{UB} + 1$ different possible values for each c_j, where $j = 1, \ldots, \mathcal{K}$, 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 $1, 2, \ldots, |\mathcal{V}|$ and each column as $(\infty, c_2, \ldots, c_{\mathcal{K}})$. The entry on row u and column $(\infty, c_2, \ldots, c_{\mathcal{K}})$ represents the estimated minimum 1st weight from u to d with the constraint c_k for all $k = 2, \ldots, \mathcal{K}$. 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 u of d sets each entry on row u and column $(\infty, c_2, \ldots, c_{\mathcal{K}})$ to $\omega_1((u, d))$, where $c_j \geq \omega_j((u, d))$ for all $j = 2, \ldots, \mathcal{K}$. In step n, we update the entries on each row u 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. It is worth noting that as the link weights are integers and we iterate each possible link weight, the algorithm gives the *precise* QoS information.



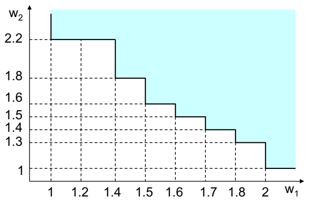
(a) The real supported QoS.



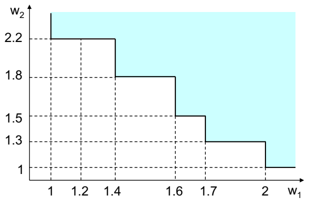
(b) QoS defined by $\mathcal{RP}^1$.



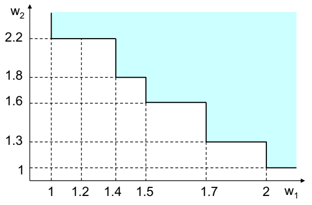
(c) QoS defined by $\mathcal{RP}^2$.



(d) QoS defined by $\mathcal{RP}^1$ and $\mathcal{RP}^2$.



(e) QoS defined by $\mathcal{RP}^{1,e}$.



(f) QoS defined by $\mathcal{RP}^{2,e}$.

Fig. 2. The construction of the approximated functions.

B. The existing approximation algorithms

When link weights are not integers, we can sample the j^{th} weight values in the range $(0, \mathcal{UB}]$ for all $j = 2, \ldots, \mathcal{K}$. For example, if we use uniform sampling [5], the sampling sequence is $\mathbf{S} = \{0, \delta, 2\delta, \ldots, m\delta, \mathcal{UB}\}$, where δ is called the *sampling parameter* and $m = \max\{t\delta < \mathcal{UB} | t \in \mathbb{Z}^+\}$. If we use logarithmic sampling [5], the sampling sequence is $\mathbf{S} = \{1, 1+\delta, (1+\delta)^2, \ldots, (1+\delta)^n, \mathcal{UB}\}$, where $n = \max\{(1+\delta)^t < \mathcal{UB} | t \in \mathbb{Z}^+\}$. The process of estimating the minimum weight function is similar to (1). The only difference is that c_k in (1) should be chosen from the sampling sequence, where $k = 2, \ldots, \mathcal{K}$. As we can only estimate the actual supported QoS by sampling, the source may not find a feasible path for a request, even when such a feasible path does exist. However, the QoS metric estimated by (1) must be an overestimation of the real one of a certain path in the network. That is, $\omega_i(p)$ must not be greater than the estimated one computed by (1) for all possible i and p. Due to space limitation, we refer readers to [14] for the formal proof. Suppose that s finds $w_1 = \mathbf{W}^1_{s,d}(w_2, \ldots, w_\mathcal{K})$ using (1). Then, s must find a path p satisfying the request $(w_1, w_2, \ldots, w_\mathcal{K})$. We call $(w_1, w_2, \ldots, w_\mathcal{K})$ *corresponds* to p.

(1) can be easily extended to compute $\mathbf{W}^i_{s,d}(\bullet)$, where $i = 1, \ldots, \mathcal{K}$. The approximated function $\mathbf{W}^i_{s,d}(\bullet)$ can be defined by a set of approximated representative vectors, $\mathcal{RP}^i_{s,d}$. 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 supported QoSes. For example, Fig. 2(a) illustrates a real supported QoS. Let $\delta = 0.2$. The sample sequence is $\{1, 1.2, \ldots\}$. The values of w_1 when $w_2 =$ 1, 1.2, 1.4, 1.6, 1.8, 2, and 2.2 are 2, 2, 1.7, 1.5, 1.4, 1.4, and 1, respectively. Then, $\mathcal{RP}^1$ becomes a staircase as illustrated in Fig. 2(b). Similarly, Fig. 2(c) illustrates the approximated supported QoS defined by $\mathcal{RP}^2$. We can see that $\mathbf{W}^1_{A,G}(\bullet)$ and $\mathbf{W}^2_{A,G}(\bullet)$ define different supported QoSes.

If the approximation algorithm just estimates the 1st minimum weight function, we call it the one-dimensional sampling algorithm, which is the algorithm in [11]. In [5], 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. The two-dimensional sampling mechanism can be easily extended to consider more than two additive constraints. Due to space limitation, we just give the overview of the *multi-dimensional sampling mechanism* as follows. Interested readers can refer to [14] for details.

In each step, each node u computes $\mathbf{W}^i_{u,d}(\bullet)$ for all $i = 1, \ldots, \mathcal{K}$. It then finds the set of representative vectors $\mathcal{RP}^i_{u,d}$ corresponding to $\mathbf{W}^i_{u,d}(\bullet)$ for all $i = 1, \ldots, \mathcal{K}$. Node u will obtain totally $\mathcal{K}$ different sets of approximated representative vectors, which may define different supported QoSes to d. The supported QoS from u to d is thus defined by $\bigcup_{i=1}^{\mathcal{K}} \mathcal{RP}^i_{u,d}$. Fig. 2(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 computational complexity of one-dimensional sampling and multi-dimensional sampling. Denote n as the number of samples in the sample sequence $\mathbf{S}$, where $n = \mathcal{O}(\frac{\mathcal{UB}}{\delta})$ by using uniform sampling. As mentioned in Section III-A, there are totally $n^{\mathcal{K}-1}$ entries in the routing table of each node. Therefore, in each step for each node computing the supported QoS from itself to a destination, the computational overhead of one-dimensional sampling is $\mathcal{O}(n^{\mathcal{K}-1})$, and the computational complexity of multi-dimensional sampling is $\mathcal{O}(\mathcal{K}n^{\mathcal{K}-1})$. We can see that by setting the same sampling parameter for both one-dimensional sampling and multi-dimensional sampling, the additional overhead incurred by multi-dimensional sampling increases exponentially with the number of weights. This implies that the performance enhancement of multi-dimensional sampling 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 for multi-dimensional sampling, 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, known as the *multi-dimensional relaxation mechanism*. We first introduce the following lemma.

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

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

■

In our approach, node s first computes $\mathbf{W}^i_{s,d}(\bullet)$ and the set of the corresponding vectors $\mathcal{RP}^i_{s,d}$ for all $i = 1, \ldots, \mathcal{K}$. Let a vector $(w_1, c_2, \ldots, c_{\mathcal{K}})$ be in $\mathcal{RP}^1_{s,d}$. We have $w_1 = \mathbf{W}^1(c_2, \ldots, c_{\mathcal{K}})$. We then compute $w_2 = \mathbf{W}^2_{s,d}(w_1, c_3, \ldots, c_{\mathcal{K}})$. Lemma 1 shows that $w_2 \leq c_2$. We can refine our estimation by iteratively computing $w_i = \mathbf{W}^i(w_1, \ldots, w_{i-1}, c_{i+1} \ldots, c_{\mathcal{K}})$. Lemma 1 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 $(w_1, \ldots, w_{\mathcal{K}})$ which defines a larger feasible region than $(w_1, c_2, \ldots, c_{\mathcal{K}})$. The similar operation is performed for each vector in each $\mathcal{RP}^i_{s,d}$. We call such operation *multi-dimensional relaxation*. Let $\mathcal{RP}^{i,e}_{s,d}$ denote the vector found by doing multi-dimensional relaxation for $\mathcal{RP}^i_{s,d}$. Finally, $\mathcal{RP}_{s,d} = \bigcup_{i=1}^{\mathcal{K}} \mathcal{RP}^{i,e}_{s,d}$ defines the estimated supported QoS. For example, for approximating the supported QoS illustrated in Fig. 2(a), our approach first computes $\mathcal{RP}^1$ and $\mathcal{RP}^2$ as shown in Fig. 2(b) and Fig. 2(c), respectively. There are five representative vectors in $\mathcal{RP}^1$. 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 $\mathcal{RP}^{1,e}$, which is illustrated in Fig. 2(e). Similarly, we also obtain the supported QoS defined by $\mathcal{RP}^{2,e}$, which is illustrated in Fig. 2(f). Combining $\mathcal{RP}^{1,e}$ and $\mathcal{RP}^{2,e}$, the supported QoS estimated by our approach is the same as the real supported QoS in Fig. 2(a).

Theorem 1: If $rp = (w_1, \ldots, w_{\mathcal{K}})$ is in $\mathcal{RP}_{s,d}$, it must be the real QoS parameter of a certain path $p_{s,d}$ from s to d.

Proof: We first show that w_i must be the i^{th} weight of a certain path, say p_i, in the network for all $i = 1, \ldots, \mathcal{K}$. We prove by induction.

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

For the inductive step, assume that the i^{th} value of each vector in $\mathcal{RP}$ computed by each node $h - 1$ hop away from d must be the i^{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^{th} weight w_i is the sum of the i^{th} weights of a $(h-1)$-hop path and a link. Based on the inductive assumption, w_i is the real i^{th} weight of this h-hop path. In other words, let $c_i = \mathbf{W}^i_{s,d}(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{\mathcal{K}})$. If $c_i < \infty$, it must be the i^{th} weight of a path satisfying the request $(c_1, \ldots, c_{i-1}, \infty, c_{i+1}, \ldots, c_{\mathcal{K}})$.

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

■

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. We would like to use the simulation experiments to further evaluate the performance of our algorithm.

IV. 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 [11], and SMAA (single-dimensional relaxation multi-dimensional sampling approximation algorithm).

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 [6]. 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 $\mathbf{C} = (0, \mathcal{C}_i] \times \ldots \times (0, \mathcal{C}_{\mathcal{K}}]$ as the request space. All the request requirements fall in $\mathbf{C}$. For each real representative vector $(w_1, \ldots, w_{\mathcal{K}})$, the supported feasible region is $[w_1, \mathcal{C}_1] \times \ldots \times [w_{\mathcal{K}}, \mathcal{C}_{\mathcal{K}}]$. 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^{th} weight value among all the real representative vectors, for all $i = 1, 2, \ldots, \mathcal{K}$. 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.

We use *the number of samples* to evaluate the computational overhead of the algorithms. In our approach, for each vector in $\mathcal{RP}^i$, where $i = 1, \ldots, \mathcal{K}$, we need to do an additional $\mathcal{K} - 1$ relaxations in order to obtain a more representative vector in $\mathcal{RP}^{i,e}$. Since the computational overhead induced by doing one relaxation for a vector is the same as that by computing the minimum i^{th} weight for a given constraint tuple, we consider that each relaxation takes one sample. It is obvious that, by setting the same sampling parameter δ, the number of samples taken by our approach is the most, and that of one-dimensional sampling is the least.

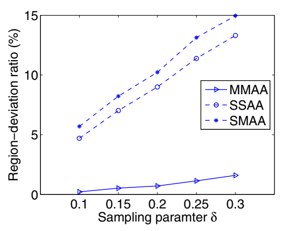
C. Simulation results

As we know, the performance of the approximation algorithms depends on the sampling parameter δ. A smaller δ reduces the approximation error at the expense of increasing the computational overhead. In order to clearly represent the performance differences among MMAA, SSAA, and SMAA, we set different sampling parameters for them, so that both the computational overheads of SSAA and SMAA are higher than that of MMAA. In this case, if the approximation error of MMAA is the lowest, we can say that MMAA outperforms the existing algorithms. Since logarithmic sampling produces the same trends as uniform sampling [5], we just show the simulation results by using logarithmic sampling.

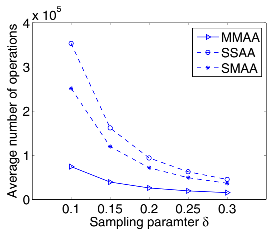
Fig. 3 just presents the simulation results in 100-node networks, and interested readers can refer to [14] the simulation results in 50-node networks. From Fig. 3, where the x-coordinate denotes the sampling parameter for MMAA, both the approximation error and the computational overhead of MMAA are the lowest. Therefore, MMAA outperforms SMAA and SSAA. Interested readers can refer to [14] for other simulation results when two and four constraints are considered. We can also observe that, by comparing with one-dimensional sampling, the performance enhancement of multi-dimensional sampling without multi-dimensional relaxation is small.

V. CONCLUSION

In this paper, we propose a new approach, called *multi-dimensional relaxation*, to precompute the supported QoS between any two nodes. We have proved that our algorithm produces smaller approximation error than the existing algorithms. We consider flat networks in this paper. However,



(a) Region-deviation probability with 100-node domains.



(b) Computational overhead with 100-node domains.

Fig. 3. The performance of the approximation algorithms with three weights.

due to the scalability issue, networks are widely structured hierarchically, 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 and K. Nahrstedt, “On finding multi-constrained paths,” *in Proceedings of ICC’98*, vol. 2, pp. 874-879, June 1998.

[3] 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.

[4] 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, May 2003.

[5] R. Hou, K.-S. Lui, K.-C. Leung, and F. Baker, “An approximation algorithm for QoS routing with two additive constraints,” *IEEE ICNP 2008*, pp. 328-337, October 2008.

[6] 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.

[7] P. W. Mieghem and F. A. Kuipers, “On the complexity of QoS routing,” *Computer communications*, vol. 26, no. 4, pp. 376-387, March 2003.

[8] A. Orda and A. Sprintson, “Precomputation schemes for QoS routing,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 578-591, August 2003.

[9] S. Sahni, “General techniques for combinatorial approximation,” *Operational Research*, vol. 25, no. 6, pp. 920-936, November/December 1977.

[10] 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.

[11] X. Yuan, “Heuristic algorithm for multiconstrained quality-of-service routing,” *IEEE/ACM Trasanctions on Networking*, vol. 10, no. 2, pp. 244-256, April 2002.

[12] 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.

[13] 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.

[14] R. Hou, K.-S. Lui, K.-C. Leung, and F. Baker, “An approximation QoS routing algorithm with multiple additive constraints,” *Technical Report, http://www.eee.hku.hk/research/research_reports.htm,* September 2008.