An Approximation Algorithm for QoS Routing with Two 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*—The problem of finding a path that satisfies two additive constraints, such as delay and cost, has been proved to be NP-complete. Many heuristic and approximation algorithms have been developed to identify a path given a certain QoS request. Unfortunately, these algorithms cannot be applied directly in the Internet because routing in the Internet is based on table lookups and routing tables are computed before a request arrives. In this paper, we develop an approximation algorithm for computing the supported QoS going across a domain. We analyze the approximation error of our algorithm and formally prove that the approximation error of our proposed algorithm is smaller than those of the existing approaches. We further verify our performance using extensive simulations.**[1]

***Index Terms*—Hierarchical networks, multiple constraints, precomputation, QoS routing.**

I. INTRODUCTION

Various real-time services, such as webcasting, audio/videoconferencing and telemedicine, are deployed over the Internet. This requires the network to provide the quality-of-service (QoS) guarantees to the customers. QoS routing, aiming at establishing a connection that satisfies multiple constraints, is one of the major building blocks for supporting QoS and a necessary component for future networks. Because of different QoS requirements (such as real-time, reliability, and cost) imposed by the multimedia applications, routing algorithms are designed with respect to different metrics, such as *additive* metrics and *bottleneck* metrics. In this paper, we study the routing problem with multiple additive constraints, that is NP-complete [18].

As the network size grows, it is impossible for each node to have the knowledge of the global network. In order to cope with the scalability, large networks are structured hierarchically by grouping the nodes into different domains. In the Internet, each domain is called an *Autonomous System* (AS). For example, in Fig. 1, there are three domains (ASes) in the network. In Domain $AS1$, A is a border node. In Domain $AS2$, C and D are border nodes, and E and F are internal nodes. In the domain $AS3$, B is a border node. The border nodes in AS are called BGP borders [11]. The tuple (x, y) associated with each edge represents the QoS metrics of cost and delay values, which are collectively called the QoS parameter pair.

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

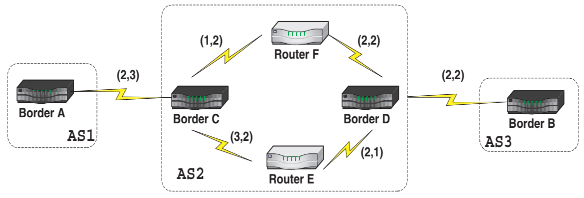


Fig. 1. A simple two-layer hierarchical network topology.

For example, the QoS parameter pair of link (F, D) is $(2, 2)$, and the QoS parameter pair of path $C{\rightarrow}F{\rightarrow}D$ is $(3, 4)$, which corresponds to the sum of the cost values and the sum of the delay values of the links on this path.

In Fig. 1, suppose that border A receives the connection request to B with the requirement $(9, 11)$. This means that the cost and delay of a qualifying connection from A to B are no more than 9 and 11, respectively. Before establishing the connection for this request, A needs to determine whether there exists a physical path satisfying the requirements. In the BGP operation, the neighbor of A, C, has to advertise to A the supported QoS from C to B. C first computes the QoS from itself to D, a neighboring BGP node. As both C and D are in the same domain, C can calculate the QoS of the two paths leading to D and they are $C \rightarrow F \rightarrow D$ and $C \rightarrow E \rightarrow D$. Their QoS parameter pairs are $(3, 4)$ and $(5, 3)$, respectively. On the other hand, D advertises its QoS to B, which is $(2, 2)$. Then, C can compute the supported QoS from itself to B and they are $(5, 6)$ and $(7, 5)$. C sends this information to A and finally A knows the supported QoS to B are $(7, 9)$ and $(9, 8)$. Based on the precomputed supported QoS, A can determine that the connection request from A to B with the requirements $(9, 11)$ is feasible. We can see that the whole process involves the problem of computing the supported QoS from a border to another border in the same domain.

The problem of finding a path that satisfies both delay and cost constraints has been proved to be NP-complete. In order to cope with the NP-completeness, heuristic algorithms were proposed in [6], [16], [17] to compute the QoS of the paths from a source to a destination. Unfortunately, these algorithms do not provide any performance guarantee. In the approximation method, cost values are sampled and the corresponding delay of each cost sample is found. Two sampling schemes have been developed: *uniform sampling* and *logarithmic sampling*. The rounding method [13] applied in [1], [5], [7] and the interval partition method [13] applied in [12], [15] are basically uniform sampling methods. On the other hand, the works in [9], [10] adopt the logarithmic sampling technique. They all consider finding the minimum delay path at a certain cost. Unfortunately, our problem is more complicated in the sense that a border router has to advertise the QoS to go across a domain instead of the minimum delay of a certain cost. That is, we need to know the minimum delay of every possible cost to go between two border routers. [2], [3], [9], [14] are the works that consider the problem of precomputing the supported QoS. However, the works in [2], [3], [14] propose the heuristic algorithms for computing the supported QoS, which cannot provide any performance guarantee. To the best of our knowledge, the work in [9] is the only work considering the approximation algorithm for computing the supported QoS. However, this work did not give the analysis for the performance of the proposed approximation algorithm. We will show later that this algorithm produces a larger approximation error compared with that of our proposed algorithm.

Accordingly, the purpose of this study is to provide a more efficient approximation algorithm to compute the supported QoS. First, we analyze theoretically the performance of the general approximation techniques, namely, uniform sampling and logarithmic sampling, from the perspective of the approximation error and the computational complexity for computing the supported QoS. We have found that uniform sampling produces smaller approximation error but larger computational complexity, while logarithmic sampling produces greater approximation error but lower computational complexity. Second, we propose an approximation algorithm which produces smaller approximation error and computational complexity.

The major contribution of this study is a novel technique, known as *the two-dimensional sampling scheme*, which reduces the approximation error without increasing the computational complexity. In addition, we formally prove that the approximation error of the proposed algorithm is smaller than those of the traditional approximation algorithms, regardless of which sampling scheme, uniform or logarithmic, is being used.

The rest of the paper is organized as follows. Section II introduces the network model and problem formulation. Section III presents the proposed polynomial-time approximation algorithm to compute the supported QoS from a source to a destination. Our simulation results are discussed in Section IV. Finally, we conclude our work in Section V.

II. NETWORK MODEL AND PROBLEM FORMULATION

This section formulates the general model addressed in this paper. A network is modeled as a directed graph $\mathcal{G}(\mathcal{V},\mathcal{E})$. $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of directed links among the nodes in $\mathcal{V}$. For any two nodes $v_i, v_j \in \mathcal{V}$, if $(v_j, v_i) \in \mathcal{E}$, v_j is a neighbor of v_i. Let $\mathcal{LA}(v_i)$ be the node neighbor set of v_i.

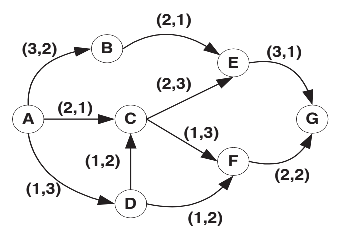


Fig. 2. A simple network where (x, y) represents two additive QoS metrics, cost and delay, respectively.

We assume that all parameters (cost and delay) are positive and independent. Let (c_l, d_l) be the QoS parameter pair of link l. We define the optimal delay function of link l, $\mathcal{D}_l^{opt}(c)$, to specify the minimum delay value provided by link l at the cost constraint of c. We have:

$$\mathcal{D}_l^{opt}(c) = \begin{cases} \infty & \text{if } c < c_l \\ d_l & \text{if } c \geq c_l \end{cases}$$

Given a path $\mathcal{P}$ from src to dst, where $src, dst \in \mathcal{V}$ are a source and a destination. The optimal delay function of the path $\mathcal{P}$, $\mathcal{D}_{\mathcal{P}}^{opt}(c)$, is the minimum delay value provided by this path with a cost constraint of c. We have:

$$\mathcal{D}_{\mathcal{P}}^{opt}(c) = \begin{cases} \infty & \text{if } c < \sum_{l\in\mathcal{P}} c_l \\ \sum_{l\in\mathcal{P}} d_l & \text{if } c \geq \sum_{l\in\mathcal{P}} c_l \end{cases}$$

Let $\mathbb{P}_{src\rightarrow dst}$ be the path set from src to dst. We define the optimal delay function from src to dst, $\mathcal{D}_{src,dst}^{opt}(c)$, which is the minimum delay value provided by all the paths from src to dst with the cost constraint of c as follows:

$$\mathcal{D}_{src,dst}^{opt}(c) = \min_{\mathcal{P}\in\mathbb{P}_{src\rightarrow dst}} \{\mathcal{D}_{\mathcal{P}}^{opt}(c)\} \tag{1}$$

Consider the simple network in Fig. 2. $\mathbb{P}_{A\rightarrow G}$ has six paths: path $A\rightarrow D\rightarrow F\rightarrow G$ with the QoS parameter pair $(4, 7)$, path $A\rightarrow C\rightarrow F\rightarrow G$ with the QoS parameter pair $(5, 6)$, path $A\rightarrow D\rightarrow C\rightarrow F\rightarrow G$ with the QoS parameter pair $(5, 10)$, path $A\rightarrow C\rightarrow E\rightarrow G$ with the QoS parameter pair $(7, 5)$, path $A\rightarrow D\rightarrow C\rightarrow E\rightarrow G$ with the QoS parameter pair $(7, 9)$, and path $A\rightarrow B\rightarrow E\rightarrow G$ with the QoS parameter pair $(8, 4)$. For the given path $\mathcal{P}_1 = \{A, B, E, G\}$, the corresponding optimal delay function is:

$$\mathcal{D}_{\mathcal{P}_1}^{opt}(c) = \begin{cases} \infty & \text{if } c < 8 \\ 4 & \text{if } c \geq 8 \end{cases}$$

We can compute the optimal delay function $\mathcal{D}_{A,G}^{opt}(c)$ from A to G based on $\{\mathcal{D}_{\mathcal{P}}^{opt}(c)|\mathcal{P} \in \mathbb{P}_{A\rightarrow G}\}$, which is:

$$\mathcal{D}_{A,G}^{opt}(c) = \begin{cases} 4 & \text{if } c \geq 8 \\ 5 & \text{if } 7 \leq c < 8 \\ 6 & \text{if } 5 \leq c < 7 \\ 7 & \text{if } 4 \leq c < 5 \\ \infty & \text{if } c < 4 \end{cases}$$

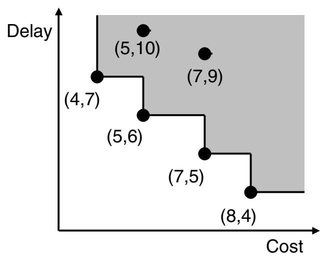


Fig. 3. The QoS parameter pairs on the cost-delay plane.

The optimal delay function from src to dst is a staircase on the cost-delay plane, as illustrated in Fig. 3. We can see that there are totally six QoS parameter pairs, but four of them, $\{(4, 7), (5, 6), (7, 5), (8, 4)\}$, can define the optimal delay function, which are called the *representative points*[2]. These representative points (or the optimal delay function) can define the *region of the supported service*, which is the shaded area in Fig. 3. At least one physical path can be found for serving the requests falling in this area. For example, the requirement $(10, 5)$ falls in the shaded area and both $(8, 4)$ and $(7, 5)$ can serve a request with that requirement. Before we describe how we can find the supported region, we first describe several definitions.

Definition 1: A point (x, y) is more representative than another point (x', y') if:

1) $x \neq x'$ or $y \neq y'$, and;
2) $x \leq x'$ and $y \leq y'$.

Definition 2: Given a QoS parameter set $\mathcal{S}$, $(x, y) \in \mathcal{S}$ is a representative point of $\mathcal{S}$ if there does not exist any other point $(x', y') \in \mathcal{S}$ which is more representative than (x, y).

Refer to our example in Fig. 3. $(4, 7)$ is more representative than $(5, 10)$ and $(4, 7)$ is a representative point since there is no other point with a cost smaller than 4 and a delay smaller than 7.

The set of representative points among the QoS of all paths going from src to dst is denoted as $\mathcal{SR}^{opt}_{src,dst}$. In Fig. 3, $\mathcal{SR}^{opt}_{A,G} = \{(4, 7), (5, 6), (7, 5), (8, 4)\}$.

In order to be a representative point, the QoS of a path must be better than the others in either cost or delay. We thus have the following preposition:

Proposition 1: Given any two cost values c_1 and c_2, if $c_1 \leq c_2$, it holds that $\mathcal{D}^{opt}_{src,dst}(c_1) \geq \mathcal{D}^{opt}_{src,dst}(c_2)$.

For convenience, Table I summarizes some important notations frequently used in this paper, some of them will be introduced in the following sections.

III. Supported QoS Across a Single Domain

In this section, we discuss how to estimate $\mathcal{SR}^{opt}_{src,dst}$, where both src and dst are in the same domain. Theoretically, if we can compute $\mathcal{D}^{opt}_{src,dst}(c)$ at all possible cost values, we can find $\mathcal{SR}^{opt}_{src,dst}$. However, this approach is intractable. Suppose that

[2]The set of representative points is called the nondominated front. The corresponding paths of the representative points are called the non-dominated paths or Pareto optimal paths in the literature.

TABLE I
Notations for our model

$\lvert\mathcal{V}\rvert$	The total number of the nodes in each domain
$\mathcal{LA}(v_i)$	The node neighbor set of v_i
$\mathcal{D}^{opt}_{src,dst}(c)$	The optimal delay function from src to dst
$\mathcal{SR}^{opt}_{src,dst}$	The optimal representative points
rp_{mc}	The minimum-cost representative point
rp_{md}	The minimum-delay representative point
$\mathcal{D}(c)$	The approximated function by sampled cost values
$\mathcal{C}(d)$	The approximated function by sampled delay values
$\mathcal{CD}(c_0)$	The cost-deviation at c_0 induced by $\mathcal{D}(c)$
$\mathcal{DD}(d_0)$	The delay-deviation at d_0 induced by $\mathcal{C}(d)$
$\mathcal{MD}(c)$	The approximated function computed by our algorithm
δ	Sampling parameter

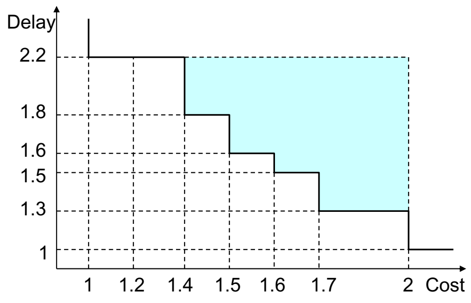


Fig. 4. The optimal delay function.

we arrange the points in $\mathcal{SR}^{opt}_{src,dst}$ in cost-ascending order. The first representative point $rp_{mc} = (\mathcal{LC}, \mathcal{UD})$ corresponds to the minimum cost path and the last representative point $rp_{md} = (\mathcal{UC}, \mathcal{LD})$ corresponds to the minimum delay path. All other representative points must have a cost falls between $\mathcal{LC}$ and $\mathcal{UC}$. We can say that the problem of computing the optimal delay function $\mathcal{D}^{opt}_{src,dst}(c)$ is to find all the representative points with the cost values between $\mathcal{LC}$ and $\mathcal{UC}$. For example, in Fig. 3, the QoS parameter pair of the minimum cost path is $(4, 7)$, and the QoS parameter pair of the minimum delay path is $(8, 4)$. In order to compute the optimal delay function $\mathcal{D}^{opt}_{A,G}(c)$, we just need to find the representative points $(5, 6)$ and $(7, 5)$, which have the costs falling between 4 and 8. If all the cost values are integers, we can traverse every integer in the range and find the corresponding delay. However, the range may be very large and cost values are not necessarily integers. To make the problem tractable and reduce the computational complexity, we sample the cost values between $\mathcal{LC}$ and $\mathcal{UC}$ and estimate the minimum delay values at these samples. We thus can get the approximated representative points based on the computed samples.

A. The approximation methods

As mentioned before, there are two sampling schemes available: *uniform sampling* and *logarithmic sampling*. Let δ be the *sampling parameter*. The sampling sequence set for uniform sampling is $\{1, x\delta, (x+1)\delta, \ldots, (x+m)\delta, \mathcal{UB}\}$, where $x = \min\{k | k\delta > 1, k \in \mathbb{Z}^+\}$, $m = \max\{t | (x+t)\delta < \mathcal{UB}, t \in \mathbb{Z}^+\}$, and $\mathcal{UB}$ is the maximum sample value. The sampling sequence set for logarithmic sampling is $\{1, (1+\delta), (1+\delta)^2, \ldots, (1+\delta)^n, \mathcal{UB}\}$, where $n = \max\{j | (1+\delta)^j < \mathcal{UB}, j \in \mathbb{Z}^+\}$. Generally speaking, δ falls between 0 and 1.

To apply the sampling schemes to our problem, we first have to normalize the cost values by dividing every cost with $\mathcal{LC}_{src,dst}$. Consider the optimal delay function $\mathcal{D}^{opt}_{src,dst}(c)$ as depicted in Fig. 4. Define $\delta = 0.2$. By using logarithmic sampling, there are three samples for delay when the normalized cost is between 1 and 2 which are 1.2, 1.44, and 1.728. The resulting approximated delay function is exhibited in Fig. 5(c). By using the uniform sampling method, there are four samples between 1 and 2 which are 1.2, 1.4, 1.6, and 1.8. The resulting approximated delay function is exhibited in Fig. 5(a).

Define $\mathcal{D}_{src,dst}(c)$ (without the *opt* superscript) as the approximated delay function. $\mathcal{D}_{src,dst}(c)$ represents the approximated minimum delay value supported by the paths from src to dst with the cost constraint of c. If uniform sampling is used, we can compute the value of $\mathcal{D}_{src,dst}(c)$ at the samples within $[1, \mathcal{UB}]$ as follows:

$$\begin{aligned}
&\mathcal{D}_{dst,dst}(c) = 0, \quad c \geq 0; \\
&\mathcal{D}_{i,dst}(c) \to \infty, \quad c \leq 0, \quad i \in \mathcal{V}; \\
&\mathcal{D}_{i,dst}(c) = \min_{k \in \mathcal{LA}(i)} \{\mathcal{D}_{k,dst}(c - c_{(i,k)}) + d_{(i,k)}, \mathcal{D}_{i,dst}(c)\} \\
&\qquad c = 1, x\delta, (x+1)\delta, (x+2)\delta, \ldots, (x+m)\delta, \mathcal{UB}, \quad i \in \mathcal{V}
\end{aligned} \tag{2}$$

To find $\mathcal{D}_{i,dst}(\bullet)$ for all $i \in \mathcal{V}$ and for all sampled costs, we keep a table of $|\mathcal{V}|$ rows and m columns where one row for each node and one column for each sampled cost. To ease our discussion, we label the nodes as $1, 2, ..., |\mathcal{V}|$, and the costs (in increasing order) as $c_1, c_2, ..., c_m$. The entry on row i and column j represents the estimated delay from node i to dst at the cost c_j. Initially, $\mathcal{D}_{dst,dst}(c_j)$ for all $1 \leq j \leq m$ are all set to be zero while $\mathcal{D}_{i,dst}(c_j)$ for all $i \neq dst$ and $1 \leq j \leq m$ are all set to be infinity. In step k, we update $\mathcal{D}_{i,dst}(\bullet)$ for those nodes i that can be k hops away from dst. After $|\mathcal{V}| - 1$ steps, the algorithm terminates since no path can have more than $(|\mathcal{V}| - 1)$ hops. That is, in the first step, we update $\mathcal{D}_{i,dst}(\bullet)$ for the nodes that have a direct edge to dst. Let the cost and delay of the edge from i to dst be c and d, respectively. If $c_k < c \leq c_{k+1}$, we take $\mathcal{D}_{i,dst}(c_j) = d$ for all $k < j \leq m$ while $\mathcal{D}_{i,dst}(c_j)$ remains to be infinity for those $1 \leq j \leq k$. If $c \leq c_1$, $\mathcal{D}_{i,dst}(c_j) = d$ for all $1 \leq j \leq m$. The neighbors of neighbors of dst are two hops away from dst and their estimated delays can be computed according to (2). We can see that for the computations for the values of $\mathcal{D}_{src,dst}(c)$ at the samples within $[1, \mathcal{UB}]$ takes $\mathcal{O}(|\mathcal{V}|^2 \frac{\mathcal{UB}}{\delta})$ time. As $\mathcal{D}_{src,dst}(c)$ approximates for $\mathcal{D}^{opt}_{src,dst}(c)$, we have the following proposition:

Proposition 2: For any two cost values c_1 and c_2, if $c_1 \leq c_2$, it holds that $\mathcal{D}_{src,dst}(c_1) \geq \mathcal{D}_{src,dst}(c_2)$.

By sampling, the representative points in $\mathcal{D}_{src,dst}(c)$ must be at the sampled costs. We thus have the following properties:

Property 1: In uniform sampling, it holds that $\mathcal{D}_{src,dst}(c_0) = \mathcal{D}_{src,dst}(\mathcal{M}\delta)$, if $\mathcal{M}\delta \leq c_0 < (\mathcal{M}+1)\delta$, where $\mathcal{M}$ is an integer.

Property 2: In logarithmic sampling, it holds that $\mathcal{D}_{src,dst}(c_0) = \mathcal{D}_{src,dst}((1+\delta)^{\mathcal{M}})$, if $(1+\delta)^{\mathcal{M}} \leq c_0 < (1+\delta)^{\mathcal{M}+1}$, where $\mathcal{M}$ is an integer.

It is easy to understand Properties 1 and 2. For example, in Property 2, there exists a path satisfying the requirements $((1+\delta)^{\mathcal{M}}, d_0)$, if we have $\mathcal{D}_{src,dst}((1+\delta)^{\mathcal{M}}) = d_0$, where $d_0 \neq \infty$. Of course, this path satisfies the requirements (c_0, d_0), where $c_0 > (1+\delta)^{\mathcal{M}}$.

From the above discussion as illustrated in Fig. 5, some representative points may be missed out under the estimation so that a drop in the minimum delay is known at a later sample. For example, the representative point $(1.5, 1.6)$ appears in Fig. 4 but not in Fig. 5(c). Besides, the optimal delay function in Fig. 4 drops to 1.3 at the cost of 1.7 but the drop occurs at 1.8 in Fig. 5(a). In other words, the sampling methods can overestimate the delay at some costs. The following lemma describes this phenomena.

Lemma 1: For any $c > 0$, if $\mathcal{D}_{src,dst}(c) < \infty$, we have $\mathcal{D}_{src,dst}(c) \geq \mathcal{D}^{opt}_{src,dst}(c)$.

Proof: $\mathcal{D}_{src,dst}(c)$ is initially set to infinity. According to (2), if $\mathcal{D}_{src,dst}(c) < \infty$, there must exist a physical path satisfying the constraint $(c, \mathcal{D}_{src,dst}(c))$. If $\mathcal{D}_{src,dst}(c) < \mathcal{D}^{opt}_{src,dst}(c)$, the optimal minimum delay from src to dst with the cost constraint of c should be $\mathcal{D}_{src,dst}(c)$ which is less than $\mathcal{D}^{opt}_{src,dst}(c)$. In this case, $\mathcal{D}^{opt}_{src,dst}(c)$ is not the optimal delay function, which contradicts our assumption. ■

B. Approximation error

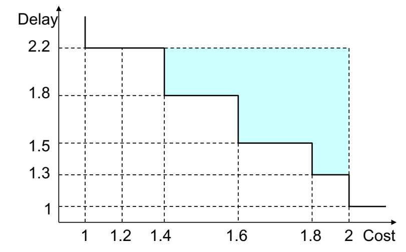
Lemma 1 suggests that the sampling methods never underestimate delay. In other words, if $\mathcal{D}_{src,dst}(c) = \mathcal{D}^{opt}_{src,dst}(c')$, $c \geq c'$. For example, in Fig. 5(a) and (d), the corresponding costs for delay = 2 are 1.4 and 1.44, where the one on the approximation function is larger. We define *cost deviation* to capture the difference in the cost values of the optimal and approximated delay functions, so as to calculate the approximation error. For ease of discussion, when the context is clear, we drop the subscripts *src* and *dst* in the delay functions and simply use $\mathcal{D}(c)$ and $\mathcal{D}^{opt}(c)$ instead.

Definition 3: If c_0 be the cost value of a representative point, we call c_0 a representative cost. The *cost deviation* at the representative cost c_0 on $\mathcal{D}^{opt}(c)$, denoted as $\mathcal{CD}(c_0)$, is $\min\{c | \mathcal{D}(c) \leq \mathcal{D}^{opt}(c_0)\} - c_0$.

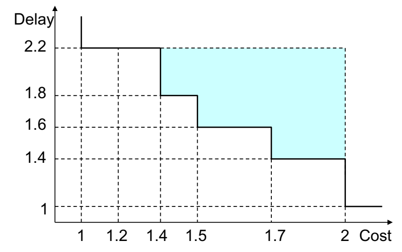
Refer to the example in Fig. 5. When $c_0 = 1.4$, $\mathcal{CD}(1.4) = 0$ for uniform sampling in Fig. 5(b) and $\mathcal{CD}(1.4) = 1.44 - 1.4 = 0.04$ for logarithmic sampling in Fig. 5(d). The following lemmas give the upper bounds on the cost deviation for the two sampling methods. In both lemmas, $\mathcal{H}$ is the number of hops for the optimal path from src to dst with the cost constraint c_0.

Lemma 2: If $\mathcal{D}(c)$ is produced by uniform sampling, $\mathcal{CD}(c_0) \leq \mathcal{H}\delta$.

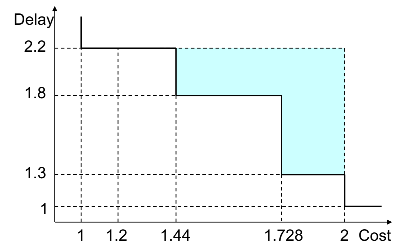
Proof: We prove this lemma by induction on $\mathcal{H}$. When $\mathcal{H} = 1$, let $(\mathcal{M}-1)\delta < c_0 \leq \mathcal{M}\delta$. According to (2),



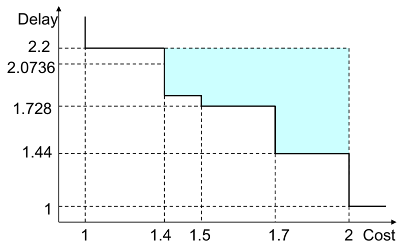
(a) The approximated function constructed by uniformly sampled cost values.



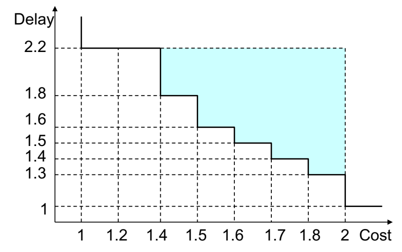
(b) The approximated function constructed by uniformly sampled delay values.



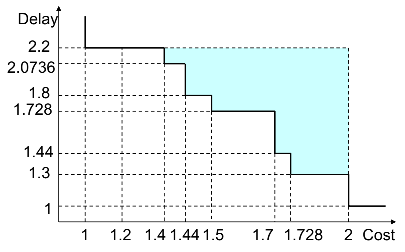
(c) The approximated function constructed by logarithmically sampled cost values.



(d) The approximated function constructed by logarithmically sampled delay values.



(e) The approximated function constructed by the proposed approximated algorithm using uniform sampling.



(f) The approximated function constructed by the proposed approximated algorithm using logarithmic sampling.

Fig. 5. An example illustrating the construction of the approximated functions for the approximation algorithm.

$\mathcal{D}(\mathcal{M}\delta) = \mathcal{D}^{opt}(c_0)$. Therefore, $\min\{c|\mathcal{D}(c) \le \mathcal{D}^{opt}(c_0)\} \le \mathcal{M}\delta$. Then, $\mathcal{CD}(c_0) = min\{c|\mathcal{D}(c) \le \mathcal{D}^{opt}(c_0)\} - c_0 \le \mathcal{M}\delta - (\mathcal{M}-1)\delta = \delta$.

Suppose the lemma holds for $\mathcal{H} \le k$. When the optimal path consists of $k+1$ hops, there must exist a node j on the path such that

$$\mathcal{D}^{opt}_{src,dst}(c_0) = \mathcal{D}^{opt}_{src,j}(c_1) + \mathcal{D}^{opt}_{j,dst}(c_2), c_1 + c_2 \le c_0 \quad (3)$$

We further assume that there are $\mathcal{H}_1$ hops and $\mathcal{H}_2$ hops on the optimal paths from src to j and j to dst, respectively. Denote $c_1' = \min\{c|\mathcal{D}_{src,j}(c) \le \mathcal{D}^{opt}_{src,j}(c_1)\}$ and $c_2' = \min\{c|\mathcal{D}_{j,dst}(c) \le \mathcal{D}^{opt}_{j,dst}(c_2)\}$. Since $\mathcal{H}_1, \mathcal{H}_2 \le k$,

$$\begin{cases} c_1' \le c_1 + \mathcal{H}_1\delta \\ c_2' \le c_2 + \mathcal{H}_2\delta \end{cases} \quad (4)$$

We also have:

$$\begin{aligned} \mathcal{D}_{src,dst}(c_1' + c_2') &\le \mathcal{D}_{src,j}(c_1') + \mathcal{D}_{j,dst}(c_2') \\ &\le \mathcal{D}^{opt}_{src,j}(c_1) + \mathcal{D}^{opt}_{j,dst}(c_2) \\ &= \mathcal{D}^{opt}_{src,dst}(c_0) \end{aligned} \quad (5)$$

Since $\mathcal{D}_{src,dst}(c_1' + c_2') \le \mathcal{D}^{opt}_{src,dst}(c_0)$, $\min\{c|\mathcal{D}(c) = \mathcal{D}^{opt}(c_0)\} \le c_1' + c_2' \le c_1 + c_2 + \mathcal{H}_1\delta + \mathcal{H}_2\delta = c_0 + (k+1)\delta$. Hence, the lemma is proved. ■

Lemma 3: If $\mathcal{D}(c)$ is produced by logarithmic sampling, $\mathcal{CD}(c_0) \le ((1+\delta)^{\mathcal{H}} - 1)c_0$.

Proof: We proof this lemma by induction on $\mathcal{H}$. When $\mathcal{H} = 1$, the lemma becomes $\min\{c|\mathcal{D}(c) \le \mathcal{D}^{opt}(c_0)\} \le (1+\delta)c_0$. Suppose $(1+\delta)^{\mathcal{M}-1} < c_0 \le (1+\delta)^{\mathcal{M}}$. According to (2), $\mathcal{D}((1+\delta)^{\mathcal{M}}) = \mathcal{D}^{opt}(c_0)$. Therefore, $\min\{c|\mathcal{D}(c) \le \mathcal{D}^{opt}(c_0)\} \le (1+\delta)^{\mathcal{M}} = (1+\delta)(1+\delta)^{\mathcal{M}-1} \le (1+\delta)c_0$.

Suppose the lemma holds for $\mathcal{H} \le k$. When the optimal path consists of $k+1$ hops, there must exist a node j on the path such that

$$\mathcal{D}^{opt}_{src,dst}(c_0) = \mathcal{D}^{opt}_{src,j}(c_1) + \mathcal{D}^{opt}_{j,dst}(c_2), c_1 + c_2 \le c_0 \quad (6)$$

We further assume that there are $\mathcal{H}_1$ hops and $\mathcal{H}_2$ hops on the optimal paths from src to j and j to dst, respectively. Denote $c_1' = \min\{c|\mathcal{D}_{src,j}(c) \le \mathcal{D}^{opt}_{src,j}(c_1)\}$ and $c_2' = \min\{c|\mathcal{D}_{j,dst}(c) \le \mathcal{D}^{opt}_{j,dst}(c_2)\}$. Since $\mathcal{H}_1, \mathcal{H}_2 \le k$,

$$\begin{cases} c_1' \le (1+\delta)^{\mathcal{H}_1} c_1 \\ c_2' \le (1+\delta)^{\mathcal{H}_2} c_2 \end{cases} \quad (7)$$

We also have:

$$\begin{aligned} &\mathcal{D}_{src,dst}((1+\delta)(c_1' + c_2')) \\ &\le \mathcal{D}_{src,j}((1+\delta)c_1') + \mathcal{D}_{j,dst}((1+\delta)c_2') \\ &\le \mathcal{D}_{src,j}(c_1') + \mathcal{D}_{j,dst}(c_2') \\ &\le \mathcal{D}^{opt}_{src,j}(c_1) + \mathcal{D}^{opt}_{src,j}(c_2) \\ &= \mathcal{D}^{opt}_{src,dst}(c_0) \end{aligned} \quad (8)$$

Since $\mathcal{D}_{src,dst}((1+\delta)(c_1' + c_2')) \le \mathcal{D}^{opt}_{src,dst}(c_0)$, $\min\{c|\mathcal{D}(c) = \mathcal{D}^{opt}(c_0)\} \le (1+\delta)(c_1' + c_2') \le (1+\delta)((1+\delta)^{\mathcal{H}_1}c_1 + (1+\delta)^{\mathcal{H}_2}c_2) \le (1+\delta)^{k+1}(c_1+c_2)\}$. The lemma is thus proved. ■

Both $\mathcal{D}^{opt}$ and $\mathcal{D}$ define the feasible regions for serving requests. Therefore, we define the *approximation error* to be the difference in terms of areas between the regions. As $\mathcal{D}$ never underestimates the actual delay constraint at all costs, the area corresponding to $\mathcal{D}$ is always a subset of that to $\mathcal{D}^{opt}$ as shown in Figure 6. In the figure, the representative points on $\mathcal{D}^{opt}$ are $\{rp_0 = rp_{mc}, rp_1, \ldots, rp_{n-1}, rp_{md} = rp_n\}$ while the representative points on $\mathcal{D}$ are $\{rp_{mc}, rp_1', \ldots, rp_m', rp_{md}\}$.

Note that the first and the last representative points on $\mathcal{D}^{opt}$ and $\mathcal{D}$ are the same because these two points can be found out easily by the Dijkstra's algorithm as described earlier in this section.

To see why there may be more representative points in $\mathcal{D}(\bullet)$ than $\mathcal{D}^{opt}(\bullet)$, consider the following example. Assume that src has two neighbors i and j. There is a path $p_{i,dst}$ from i to dst with the QoS parameter pair $(3.3, 3.3)$. There is also a path $p_{j,dst}$ from j to dst with the QoS parameter pair $(1.2, 1.2)$. The QoS parameter pair of the link (src, i) is $(1, 1)$ and that of the link (src, j) is $(3.2, 3.2)$. Let p_1 be the path from src to dst via i and p_2 be the path via j. We can see that the QoS parameter pair of p_1 is $(4.3, 4.3)$ and that of p_2 is $(4.4, 4.4)$. If we use uniform sampling on cost with the sampling parameter $\delta = 0.4$, i will compute the QoS parameter pair of the path $p_{i,dst}$ as $(3.6, 3.3)$, and j will compute the QoS parameter pair of the path $p_{j,dst}$ as $(1.2, 1.2)$. Accordingly, src will compute that the QoS parameter pair of the path p_1 is $(4.8, 4.3)$ and that of p_2 is $(4.4, 4.4)$. Therefore, the approximated representative points defined by $\mathcal{D}(\bullet)$ may be more than the optimal representative points. For any c, $\mathcal{D}(c)$ may not be a delay value of a real representative point, as illustrated in Fig. 6.

In Fig. 6, the difference in areas can be partitioned into several rectangles according to different delay ranges. We now study how to compute the approximation error between two consecutive representative points, rp_i and rp_{i+1}, on $\mathcal{D}^{opt}$ in the following two cases: (1) There is no representative point rp' on $\mathcal{D}$ such that $rp_{i+1}.d < rp'.d < rp_i.d$. (2) There is at least one point rp' on $\mathcal{D}$ such that $rp_{i+1}.d < rp'.d < rp_i.d$.

Case I: In Figure 6, there is no rp' falling in the delay range between rp_1 and rp_2 as well as rp_2 and rp_3. The area of the rectangle formed between $rp_{i+1}.d$ and $rp_i.d$ is $(rp_i.d - rp_{i+1}.d) * \mathcal{CD}(rp_{i+1}.c)$.

Case II: In this case, we divide the range $[rp_{i+1}.d, rp_i.d]$ according to the representative points on $\mathcal{D}$ falling in this range (rp'_1 and rp'_2 in Figure 6). Let $rp_i.d > rp'_k.d > rp'_{k+1}.d > ... > rp'_{k+j}.d > rp_{i+1}.d$. The total area not covered by $\mathcal{D}$ in $[rp_{i+1}.d, rp_i.d]$ is

$$\begin{aligned}&(rp_i.d - rp'_k.d) * (rp'_k.c - rp_{i+1}.c) + \\ &\textstyle\sum_{l=1}^{j} (rp'_{k+l-1}.d - rp'_{k+l}.d) * (rp'_{k+l}.c - rp_{i+1}.c) \\ &+ (rp_{k+j}.d - rp_{i+1}.d) * \mathcal{CD}(rp_{i+1}.c)\end{aligned}$$

As $rp'_{k+l}.c - rp_{i+1}.c \leq \mathcal{CD}(rp_{i+1}.c)$ for $0 \leq l \leq j$, the error in this case is less than or equal to $(rp_{i+1}.d - rp_i.d) * \mathcal{CD}(rp_{i+1}.c)$ as in Case I.

By Lemmas 2 and 3, the total approximation error induced by uniform sampling is no more than

$$\begin{aligned}&(|\mathcal{V}| - 1)\delta \cdot \left(\textstyle\sum_{i=1}^{n} (rp_{i-1}.d - rp_i.d) \right) \\ &= (|\mathcal{V}| - 1)\delta(rp_{mc}.d - rp_{md}.d)\end{aligned} \tag{9}$$

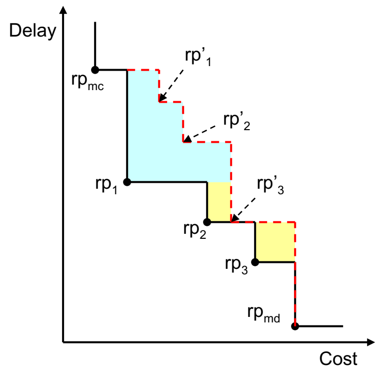


Fig. 6. An illustration for computing the approximation error.

and the approximation error induced by logarithmic sampling is bounded above by

$$\begin{aligned}&((1+\delta)^{|\mathcal{V}|-1} - 1) \cdot \left(\textstyle\sum_{i=1}^{n} \{(rp_{i-1}.d - rp_i.d) \cdot rp_i.c\} \right) \\ &= ((1+\delta)^{|\mathcal{V}|-1} - 1)(rp_{mc}.d - rp_{md}.d) \cdot rp_{md}.c\end{aligned} \tag{10}$$

We can see that logarithmic sampling induces a larger approximation error than uniform sampling with the same δ. In the following section, we describe an enhancement to the sampling method to further reduce the approximation error.

C. Two-dimensional sampling

Existing works sample only according to cost. In fact, we can also sample according to delay and the results can be different. For example, for the optimal delay function depicted in Fig. 4, if we use the logarithmic sampling method on cost, we get the approximated function in Fig. 5(c). On the other hand, if we sample on delay, the function becomes the one shown in Fig. 5(d). The representative points on the two functions are different. If we combine the two, we get seven representative points and they are {(1,2.2), (1.4,2.0736), (1.44,1.8), (1.5,1.728), (1.7,1.44), (1.728,1.3), (2,1)} as shown in Fig. 5(f). The approximation error of the one in Fig. 5(f) is smaller than those in both Fig. 5(c) and Fig. 5(d). That is, by sampling on both cost and delay, we can further reduce the approximation error.

We let $\mathcal{C}^{opt}(d)$ be the optimal cost function and the approximated cost function be $\mathcal{C}(d)$. Following Definition 3, we define delay deviation as follows:

Definition 4: The *delay deviation* at representative delay d_0 on $\mathcal{C}^{opt}(d)$, denoted as $\mathcal{DD}(d_0)$, is $\min\{d | \mathcal{C}(d) \leq \mathcal{C}^{opt}(d_0)\} - d_0$.

Following similar arguments in Lemmas 2 and 3, when uniform sampling is used, $\mathcal{DD}(d_0) \leq \mathcal{H}\delta$ while when logarithmic sampling is used, $\mathcal{DD}(d_0) \leq ((1+\delta)^{\mathcal{H}} - 1)d_0$, where d_0 is a representative delay and $\mathcal{H}$ is the number of hops for the optimal path satisfying the delay constraint d_0.

Denote $\mathcal{MD}_{src,dst}(c)$ as the approximated delay function computed by our algorithm. Define $\mathcal{C}^{-1}_{j,dst}(c)$ as the inverse function of $\mathcal{C}_{j,dst}(d)$, which represents the approximated minimum delay at the cost constraint of c defined by $\mathcal{C}_{j,dst}(d)$.

Define $\mathcal{MD}^{-1}_{j,dst}(d)$ as the inverse function of $\mathcal{MD}_{j,dst}(c)$, which represents the approximated minimum cost at the delay constraint of d defined by $\mathcal{MD}_{j,dst}(c)$. Assume that our algorithm uses uniform sampling, the procedure of computing $\mathcal{MD}_{j,dst}(c)$, where $j \in \mathcal{V}$, can be found in Algorithm 1. Of course, our algorithm can apply logarithmic sampling. In this case, the procedure is the same as Algorithm 1 except that the sample sequence set is different. That is to say, Line 13 of Algorithm 1 should be $\forall c = 1, (1+\delta), (1+\delta)^2, \ldots, \mathcal{UB}$, and Line 21 of Algorithm 1 should be $\forall d = 1, (1+\delta), (1+\delta)^2, \ldots, \mathcal{UB}$. Note that in Line 29, in order to compute $\mathcal{MD}_{j,dst}(c)$, we just need to find all the representative points on $\mathcal{D}_{j,dst}(c)$ and $\mathcal{C}_{j,dst}(d)$, and $\mathcal{MD}_{j,dst}(c)$ is defined by all these representative points.

Algorithm 1 Computing $\mathcal{MD}(c)$

```
$c_{(j,k)}$ is the cost metric of link $(j,k)$
$d_{(j,k)}$ is the delay metric of link $(j,k)$
 1: for each $j \in \mathcal{V}$ do
 2:     $\mathcal{MD}_{j,dst}(c) \leftarrow \infty$
 3:     $\mathcal{D}_{j,dst}(c) \leftarrow \infty$
 4:     $\mathcal{C}_{j,dst}(d) \leftarrow \infty$
 5: end for
 6: $\mathcal{MD}_{dst,dst}(c) \leftarrow 0, c \geq 0$
 7: $\mathcal{D}_{dst,dst}(c) \leftarrow 0, c \geq 0$
 8: $\mathcal{C}_{dst,dst}(d) \leftarrow 0, d \geq 0$
 9: $h \leftarrow 0$
10: while $h < |\mathcal{V}| - 1$ do
11:     for each $j \in \mathcal{V} \setminus \{dst\}$ do
12:         $x \leftarrow \lceil \frac{1}{\delta} + 0.5 \rceil$
13:         for each $c = 1, x\delta, (x+1)\delta, \ldots, \mathcal{UB}$ do
14:             for each $k$ in $\mathcal{LA}(j)$ do
15:                 $d \leftarrow \mathcal{MD}_{k,dst}(c - c_{(j,k)}) + d_{(j,k)}$
16:                 if $d < \mathcal{D}_{j,dst}(c)$ then
17:                     $\mathcal{D}_{j,dst}(c) \leftarrow d$
18:                 end if
19:             end for
20:         end for
21:         for each $d = 1, x\delta, (x+1)\delta, \ldots, \mathcal{UB}$ do
22:             for each $k$ in $\mathcal{LA}(j)$ do
23:                 $c \leftarrow \mathcal{MD}^{-1}_{k,dst}(d - d_{(j,k)}) + c_{(j,k)}$
24:                 if $c < \mathcal{C}_{j,dst}(d)$ then
25:                     $\mathcal{C}_{j,dst}(d) \leftarrow c$
26:                 end if
27:             end for
28:         end for
29:         $\mathcal{MD}_{j,dst}(c) \leftarrow \min\{\mathcal{D}_{j,dst}(c), \mathcal{C}^{-1}_{j,dst}(c)\}$
30:     end for
31:     $h \leftarrow h + 1$
32: end while
```

Now, we are going to discuss how to set the sampling parameter δ in order that our approximation solution satisfies the upper bound of the deviation factor ϵ_0. This means that $\mathcal{CD}(c) \leq \epsilon_0 c$ and $\mathcal{DD}(d) \leq \epsilon_0 d$. Since a smaller δ causes a larger computational overhead, we will give the upper bound value such that the computational overhead is minimized.

Lemma 4: Let $\delta = \frac{\epsilon_0}{2\mathcal{H}}$, where $\mathcal{H}$ is the number of hops for the optimal path from src to dst satisfying the cost constraint of c. Given the upper bound of the approximation error ϵ_0, it holds that $\mathcal{CD}(c) \leq \epsilon_0 c$ and $\mathcal{DD}(d) \leq \epsilon_0 d$.

Proof: For the uniform sampling method, by Lemma 2, we have $\mathcal{CD}(c) \leq \mathcal{H}\delta = \mathcal{H}\frac{\epsilon_0}{2\mathcal{H}} \leq \epsilon_0$.

For the logarithmic sampling method, by Lemma 3, we have $\mathcal{CD}(c) \leq ((1+\delta)^{\mathcal{H}} - 1)c$. Now, we need to prove that $(1+\delta)^{\mathcal{H}} \leq \epsilon_0 + 1$. Since $0 < \delta < 1$, we have the following inequality.

$$\begin{aligned} & \ln(1+\delta)^{\mathcal{H}} \\ = \; & \mathcal{H}\ln(1+\delta) \\ \leq \; & \mathcal{H}\delta \quad (\because \ln(1+x) \leq x, 0 < x < 1) \\ = \; & \tfrac{\epsilon_0}{2} \\ \leq \; & \ln(\epsilon_0 + 1) \quad (\because \tfrac{x}{2} \leq \ln(1+x), 0 < x < 1) \end{aligned} \tag{11}$$

For the same reason, we have $\mathcal{DD}(d) \leq \epsilon_0 d$. ■

D. The approximation error of our algorithm

In this section, we analyze the approximation error caused by our approximated delay function. Suppose that the representative points on $\mathcal{D}^{opt}$ are $\{rp_0 = rp_{mc}, rp_1, \ldots, rp_{n-1}, rp_{md} = rp_n\}$, the representative points on $\mathcal{D}$ are $\{rp_{mc}, rp'_1, \ldots, rp'_m, rp_{md}\}$, and the representative points on $\mathcal{C}$ are $\{rp_{mc}, rp''_1, \ldots, rp''_l, rp_{md}\}$. Note that the first and the last representative points on $\mathcal{D}^{opt}$, $\mathcal{D}$, and $\mathcal{C}$ are the same. In Section III-B, we show that if we just do the cost sampling, the feasible region in the area $\mathbb{R} = [rp_{i-1}.d, rp_i.d] \times [rp_i.c, rp_i.c + \mathcal{CD}(rp_i.c)]$ may not be included by $\mathcal{D}$. For example, in Fig. 6, the shaded area is the feasible region not included by $\mathcal{D}$, while some of them is included by $\mathcal{C}$, as illustrated in Fig. 7. Now, we compute the region in $\mathbb{R}_i$ which is also not included by $\mathcal{C}$. We also need to consider two cases.

Case I: There is no rp'' on $\mathcal{C}$ falling in $\mathbb{R}_i$. The area not covered by $\mathcal{C}$ is $\mathcal{DD}(rp_i.d) * \mathcal{CD}(rp_i.c)$. For example, in Fig. 6, there is no rp'' falling in $\mathbb{R}_3$ and $\mathbb{R}_2$.

Case II: In this case, we divide the range $[rp_i.c, rp_i.c + \mathcal{CD}(rp_i.c)]$ according to the representative points on $\mathcal{C}$ falling in $\mathbb{R}_i$ (rp''_2 and rp''_3 falling within $\mathbb{R}_1$ in Fig. 7). Let $rp_i.c < rp''_k.c < rp''_{k+1}.c < \ldots < rp''_{k+j}.c < rp_i.c + \mathcal{CD}(rp_i.c)$. The total area not covered by $\mathcal{C}$ is

$$\begin{aligned} & (rp''_k.c - rp_i.c) * \mathcal{DD}(rp_i.d) + \\ & \textstyle\sum_{l=1}^{j} (rp''_{k+l}.c - rp''_{k+l-1}.c) * (rp''_{k+l}.d - rp_i.d) + \\ & (rp_i.c + \mathcal{CD}(rp_i.c) - rp_{k+j}.c) * (rp_{k+j}.d - rp_i.d) \end{aligned}$$

As $rp''_{k+l}.d - rp_i.d \leq \mathcal{DD}(rp_i.d)$ for $0 \leq l \leq j$, the error in this case is less than or equal to $\mathcal{DD}(rp_i.d) * \mathcal{CD}(rp_i.c)$ as in Case I.

If uniform sampling is used, by Lemma 2, $\mathcal{DD}(rp_i.d) \leq ((|\mathcal{V}| - 1)\delta)$ and $\mathcal{CD}(rp_i.c) \leq ((|\mathcal{V}| - 1)\delta)$. So, the total approximation error is less than:

$$n \cdot ((|\mathcal{V}| - 1)\delta)^2 \tag{12}$$

If our algorithm uses logarithmic sampling, by Lemma 3, $\mathcal{DD}(rp_i.d) \leq ((1+\delta)^{|\mathcal{V}|-1} - 1) \cdot rp_i.d$ and $\mathcal{CD}(rp_i.c) \leq ((1+\delta)^{|\mathcal{V}|-1} - 1) \cdot rp_i.c$. So, the total approximation error is less than:

$$\begin{aligned} & ((1+\delta)^{|\mathcal{V}|-1} - 1)^2 \textstyle\sum_{i=1}^{n} \{rp_i.c * rp_i.d\} \\ & \leq n \cdot ((1+\delta)^{|\mathcal{V}|-1} - 1)^2 rp_{mc}.d \cdot rp_{md}.c \end{aligned} \tag{13}$$

In [8], the expectation number of the representative points depends on the number of the nodes and links. If we apply uniform sampling, the approximation error produced by our

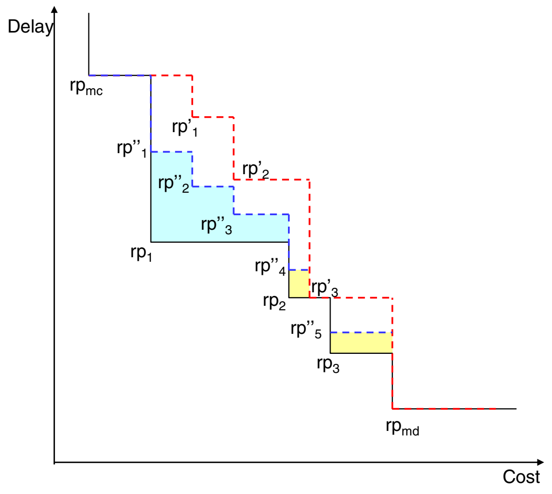


Fig. 7. An illustration for the approximation error of our algorithm.

approach just depends on the network size and the sampling parameter, while that by the cost-sampling mechanism also depends on the maximum metric of each link and each path. We thus say that the approximation error of our approach can be easily controlled.

E. Complexities of the proposed approximation algorithm

In this section, we focus on analyzing the time and space complexities of our approximation algorithm. We assume that our algorithm uses logarithmic sampling method. The analysis of our proposed algorithm using the uniform sampling method can be obtained similar to the one described as follows.

Define $\mathcal{B}$ as the maximum cost value and the maximum delay value of each link, so $\mathcal{UB} \leq |\mathcal{V}|\mathcal{B}$. We assume that $\mathcal{B}$ is constant. The number of the samples taken is no more than $\log_{1+\delta}\mathcal{UB}$. Since $0 < \delta < 1$, we have $\mathcal{O}(\log(1+\delta)) = \mathcal{O}(\delta)$. So, $\log_{1+\delta}\mathcal{UB} = \mathcal{O}(\frac{\log|\mathcal{V}|}{\delta})$. Since $\delta = \frac{\epsilon}{2\mathcal{H}}$, $\delta \geq \frac{\epsilon}{2(|\mathcal{V}|-1)}$. We have $\frac{1}{\delta} \leq \frac{2(|\mathcal{V}|-1)}{\epsilon}$. Therefore, the maximum number of the samples is $\mathcal{O}(\frac{|\mathcal{V}|\log|\mathcal{V}|}{\epsilon})$.

Lemma 5: The space complexity of our algorithm is $\mathcal{O}(|\mathcal{LA}(src)|\frac{|\mathcal{V}|\log|\mathcal{V}|}{\epsilon})$, and the time complexity is $\mathcal{O}(\frac{|\mathcal{V}|\log|\mathcal{V}|}{\epsilon})$.

Proof: Since the maximum number of the representative points on $\mathcal{D}$ and $\mathcal{C}$ are no more than the number of samples for these two functions, it is $\mathcal{O}(\frac{|\mathcal{V}|\log(|\mathcal{V}|)}{\epsilon})$. So, the time complexity is $\mathcal{O}(\frac{|\mathcal{V}|\log|\mathcal{V}|}{\epsilon})$. To compute $\mathcal{MD}_{src,dst}(c)$, src needs to record $\mathcal{O}(|\mathcal{LA}(src)|\frac{|\mathcal{V}|\log(|\mathcal{V}|)}{\epsilon})$ representative points corresponding to $\mathcal{MD}_{j,dst}(c)$, where $j \in \mathcal{LA}(src)$. In addition, src also needs to record an additional $\mathcal{O}(\frac{|\mathcal{V}|\log|\mathcal{V}|}{\epsilon})$ representative points corresponding to $\mathcal{D}_{src,dst}(c)$ and $\mathcal{C}_{src,dst}(d)$. Therefore, the space complexity of our algorithm is $\mathcal{O}(|\mathcal{LA}(src)|\frac{|\mathcal{V}|\log|\mathcal{V}|}{\epsilon})$. ■

IV. PERFORMANCE EVALUATION

In this section, we present our simulation results and compare our algorithm with the existing algorithms. As we know, all the existing approximation algorithms compute the supported QoS just by sampling cost values, and they can be divided into two categories: those applying uniform sampling and those applying logarithmic sampling. We call the traditional approximation algorithm [15] using uniform sampling SUAA, and that using logarithmic [9] sampling SLAA. We call our algorithm applying uniform sampling MUAA, and that applying logarithmic sampling MLAA. 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 the Waxman's model [4]. Four nodes are selected randomly, and we compute the supported QoS from each one to the others. All physical link metrics are asymmetric. Each link (i,j) is associated with two randomly generated metrics, delay and cost. These metrics are uniformly distributed, such that $d_{i,j} \sim \text{uniform}[1,100]$, $c_{i,j} \sim \text{uniform}[1,100]$. We have generated 100 sets of domains with 50 nodes in each domain and another 100 sets of domains with 100 nodes in each domain. Each data point reported in the simulation results is an average value. The 95% confidence interval is also shown for each data point.

B. Evaluation metrics

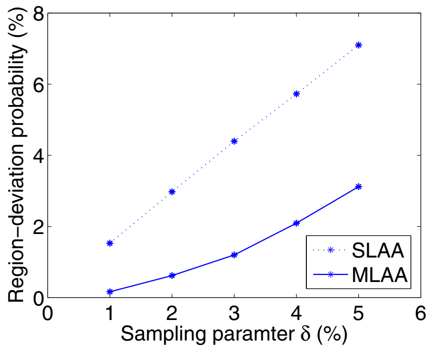
Two metrics are used in our evaluating the performance of the approximation algorithms: *the region-deviation probability* and *the average samples of each delay function*.

The region-deviation probability is used for evaluating the performance due to the approximation error. Since the feasible region is open, we just consider the feasible region which is included in the region $[\mathcal{LC}_{src,dst}, \mathcal{UC}_{src,dst}] \times [\mathcal{LD}_{src,dst}, \mathcal{UD}_{src,dst}]$. We know that the feasible region is composed by multiple squares, and the sum of the area of these squares is computed to find the size of the feasible region. Let V_{opt} be the area of the optimal feasible region, and V be the area of the approximated feasible region. We define the region-deviation probability as $\frac{V_{opt}-V}{V_{opt}}$. In order to compute the optimal feasible region, we find the QoS parameters of all the paths from a source to a destination. We can also find the optimal representative points defining the optimal feasible region.

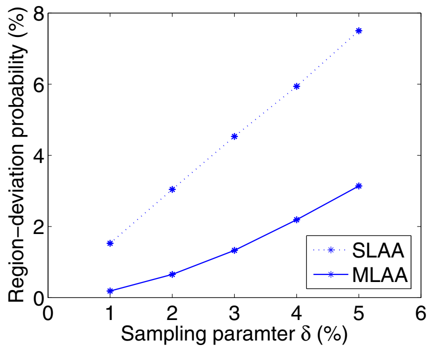
The average samples of each delay function is used for evaluating the performance due to the computation overhead.

C. Simulation results

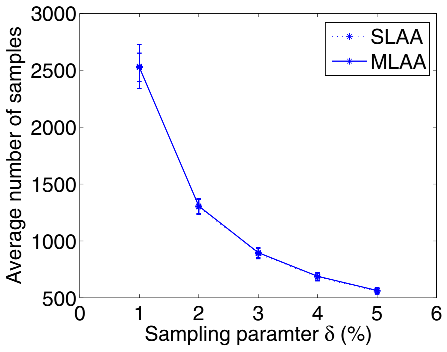
Since our approximation algorithm adopts the two-dimensional sampling scheme which induces some extra computational overhead. In order to evaluate the benefit of the two-dimensional sampling scheme, we compute the approximation error of our algorithm and the existing algorithm when the computational overheads of them are comparable. Therefore, we set the sampling parameter of our algorithm two times than that of the existing algorithm. For instance,



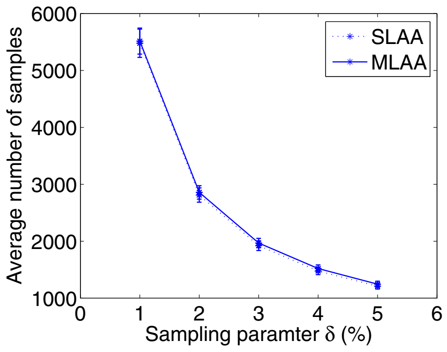
(a) Region-deviation probability with 50-node domains.



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

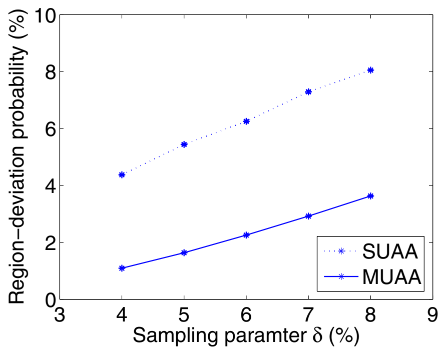


(c) Computational overhead with 50-node domains.

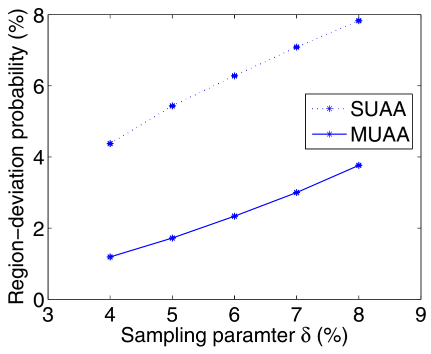


(d) Computational overhead with 100-node domains.

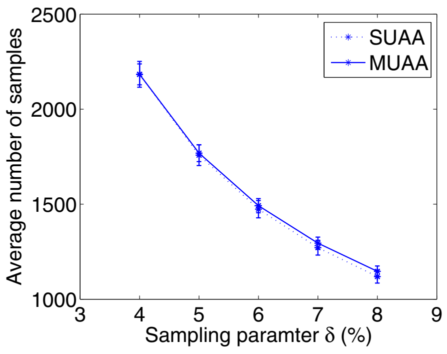
Fig. 8. The performance of the algorithms applying logarithmic sampling.



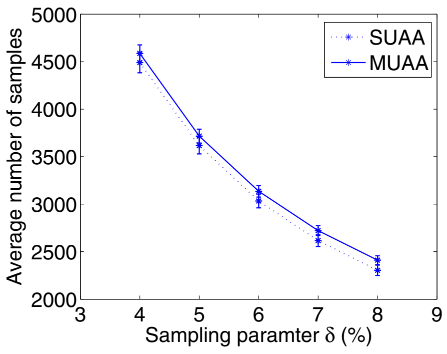
(a) Region-deviation probability with 50-node domains.



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



(c) Computational overhead with 50-node domains.



(d) Computational overhead with 100-node domains.

Fig. 9. The performance of the algorithms applying uniform sampling.

if we compare MLAA and SLAA, the sampling sequence of SLAA is $\{1, (1+\delta), \ldots, (1+\delta)^n, \mathcal{UC}_{src,dst}\}$, and that of MLAA is $\{1, (1+2\delta), \ldots, (1+2\delta)^n, \mathcal{UC}_{src,dst}\}$. If we compare MUAA and SUAA, the samples sequence of SUAA is $\{1, x\delta, (x+1)\delta, (x+2)\delta, \ldots, \mathcal{UC}_{src,dst}\}$, and that of MUAA is $\{1, 2y\delta, 2(y+1)\delta, 2(y+2)\delta, \ldots, \mathcal{UC}_{src,dst}\}$, where x and y are integers.

The simulation results are shown in Figs. 8 and 9. The computational overhead of MLAA and SLAA are almost same, and that of MUAA and SLAA are almost same. We observe that the approximation error produced by MLAA is smaller than that by SLAA and that by MUAA is smaller than that by SUAA. Therefore, our approach (two-dimensional sampling mechanism) outperforms the existing method (cost-sampling mechanism). We would like to compare the performance of the uniform sampling and the logarithmic sampling. For instance, we let δ be 0.04. In Fig. 8(a), the approximation error produced by SLAA is about 6% and that of MLAA is about 2%. The approximation error produced by SUAA is about 4% and that of MUAA is less than 1%, as exhibited in Fig. 9(a). That means the approximation error produced by uniform sampling is lower than that of logarithmic sampling. In Fig. 8(c), the computational overhead produced by SLAA and MLAA is less than 1000 samples, but that of SUAA and MUAA is more than about 2200 samples, as illustrated in Fig. 9(c). We can see that the computational overhead produced by uniform sampling is much higher than that of logarithmic sampling.

Now, we analyze the effect of the sampling scheme on the performance of our algorithm. We use MLAA(4) to denote that MLAA uses the sampling parameter $\delta = 4\%$. In Figs. 8(b) and 9(b), the approximation error of MLAA(1) is comparable with that of MUAA(4), but the computational overhead of MLAA(1) is higher than that of MUAA(4), as illustrated in Figs. 8(d) and 9(d). In this case, MUAA(4) outperforms MLAA(1). However, in Figs. 8(c) and 9(c), the computational overhead of MUAA(6) is comparable with that of MLAA(2), but the approximation error of MUAA(6) is higher than that of MLAA(2), as illustrated in Figs. 8(a) and 9(a). In this case, MLAA(2) outperforms MUAA(6). Therefore, we cannot outweigh a sampling mechanism over the other.

V. CONCLUSION

Precomputing the supported QoS across a domain is a fundamental problem of supporting QoS in the Internet. As the routing with two additive constraints is NP-complete, we focus on the design of the approximation algorithm. The existing approximation algorithms sample the cost metric and compute the minimum delay path with each sampled cost value, so that the region of the supported service can be obtained. The accuracy of the existing algorithms only can be improved with the expense of increasing the computational complexity. Accordingly, we propose a new approach, *the two-dimensional sampling scheme*, to improve the accuracy performance without increasing the computational complexity. We show that our algorithm gives a smaller approximation error than the existing approximation algorithms. The simulation results demonstrate that our algorithm outperforms the existing algorithms. More generally, the two dimensional sampling scheme can be easily extended, so that we can improve the performance of the approximation algorithm for computing the supported QoS with multiple additive constraints.

REFERENCES

[1] S. Chen and K. Nahrstedt, "On finding multi-constrained paths," *in Proceedings of ICC'98*, vol. 2, pp. 874-879, June 1998.

[2] 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.

[3] Y. Cui, K. Xu, J. P. Wu, and M. W. Xu, "Precomputation for finding paths with two additive weights," *ICC'03*, vol. 1, pp. 636-640, 11-15 May 2003.

[4] K. I. Calvert, M. B. Doar, and E. W. Zegura, "Modeling internet topology," *IEEE Communication Magazine*, vol. 35, no. 6, pp. 160-163, 1997.

[5] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Mathematics of Operations Research*, vol. 17, no. 1, pp. 36-42, February 1992.

[6] T. Korkmaz, M. Krunz, "Multi-constrained optimal path selection," *in Proceedings of the INFOCOM 2001 Conference*, vol. 2, pp. 834-843, Apr. 2001.

[7] 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.

[8] P. Van Meighem and F.A. Kuipers, "On the complexity of QoS routing," *Computer Communications*, vol. 26, no. 4, pp. 376-387, March 2003.

[9] Ariel Orda and Alexander Sprintson, "Precomputation schemes for QoS routing," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 578-591, Aug. 2003.

[10] A. Orda and A. Sprintson, "A scalable approach to the partition of QoS requirements in unicast and multicast," *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 1146-1159, Oct. 2005.

[11] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," *Network Working Group*, RFC 1771, Mar. 1995.

[12] 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.

[13] S. Sahni, "General techniques for combinatorial approximation," *Operational Research*, vol. 25, no. 6, pp. 920-936, Nov./Dec. 1977.

[14] Y. Zheng, T. Korkmaz, and W. Dou, "Pareto Optimal based partition framework for two additive constrained path selection", *IEEE International Conference on Networking (ICN'05)*, pp. 1-8, April 2005.

[15] Y. Xin, "Heuristic algorithm for multiconstrained quality-of-service routing," *IEEE/ACM Trasanctions on Networking*, vol. 10, no. 2, pp. 844-853, April 2002.

[16] 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.

[17] G. L. Xue, A. Sen, W. Zhang, J. Tang, and K. Thulasiraman, "Finding a path subject to many additive QoS constraints," *IEEE/ACM Transactions on networking,* vol. 15, no. 1, pp. 201-211, February 2007.

[18] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications" *IEEE J. Select. Areas Comm.*, vol. 14, pp. 1228-1234, Sept. 1996.