Title	Performance Analysis of Quantization-Based Approximation Algorithms for Precomputing the Supported QoS
Author(s)	Hou, R; Wong Lui, KS; Leung, KC; Baker, F
Citation	Journal of Network and Computer Applications, 2014, v. 40, p. 244-254
Issued Date	2014
URL	http://hdl.handle.net/10722/200615
Rights	Creative Commons: Attribution 3.0 Hong Kong License

Journal of Network and Computer Applications 40 (2014) 244–254

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Performance analysis of quantization-based approximation algorithms for precomputing the supported QoS

Ronghui Hou [a,*], King-Shan Lui [b], Ka-Cheong Leung [b], Fred Baker [c]

[a] The State Key Lab of Integrated Service Networks, Xidian University, China
[b] Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong
[c] Cisco Research Center, San Jose, CA 95134, USA

ARTICLE INFO

Article history:
Received 11 January 2013
Received in revised form
6 September 2013
Accepted 19 September 2013
Available online 28 September 2013

Keywords:
Precomputation
QoS routing
Approximation algorithms
Additive constraints

ABSTRACT

Precomputation of the supported QoS is very important for internet routing. By constructing routing tables before a request arrives, a packet can be forwarded with a simple table lookup. When the QoS information is provided, a node can immediately know whether a certain request can be supported without launching the path finding process. Unfortunately, as the problem of finding a route satisfying two additive constraints is NP-complete, the supported QoS information can only be approximated using a polynomial time mechanism. A good approximation scheme should reduce the error in estimating the actual supported QoS. Nevertheless, existing approaches which determine this error may not truly reflect the performance on admission control, meaning whether a request can be correctly classified as feasible or infeasible. In this paper, we propose using a novel metric, known as *distortion area*, to evaluate the performance of precomputing the supported QoS. We then analyze the performance of the class of algorithms that approximate the supported QoS through discretizing link metrics. We demonstrate how the performance of these schemes can be enhanced without increasing complexity. Our results serve as a guideline on developing discretization-based approximation algorithms.

1. Introduction

As the demand for deploying real-time and multimedia applications over the internet is increasing, providing guaranteed quality-of-service (QoS) for these applications becomes more and more important. In general, the QoS requirements can be divided into two categories: *bottleneck* metric and *additive* metric. The additive metric of a path is the sum of the metrics of the links along the path, while the bottleneck metric of a path is the minimum value of the metrics of the links along the path. For example, bandwidth is a *bottleneck* QoS metric, while delay and delay jitter are *additive* QoS metrics. In this work, we consider connection requests that have two additive QoS requirements or constraints, such as in delay and cost. To simplify our discussion, we assume that delay and cost are the two additive metrics under consideration, although our analysis and method can be applied to any additive metrics.

Many existing works study how to identify a feasible path for a request with two additive constraints, which is an NP-complete problem. These works usually assume either the cost or the delay requirement which is known. Nevertheless, such reactive routing mechanism, which finds a path after the requirement is known, cannot provide enough information to support efficient admission control. When a request is received, a node cannot immediately tell whether a possible feasible path exists until a path finding process is launched based on the requested cost/delay. On the other hand, by precomputing the supported QoS information, a source can immediately determine whether the connection request can be supported by the network. Moreover, accepting a new connection will not violate the service guarantees for the existing traffics, and also the transmission route satisfies the QoS requirement of the new connection.

The problem of computing the supported QoS between two nodes is more complicated than the extensively studied *multi-constrained path (MCP)* problem or the *delay-constrained least cost (DCLC)* path problem. The DCLC problem is also called the *restricted shortest path (RSP)* problem. The RSP problem aims at finding the minimum delay path among the paths that satisfy a certain cost constraint. The MCP problem studies finding a path satisfying both specified cost and delay constraints. Both problems focus on finding a single path between two nodes with a given (cost) constraint. Our problem, also known as the *all-costs optimal path (ACOP)* problem (Orda and Sprintson, 2003), finds, for each cost c, a c-cost constrained path from a source to a destination with the minimum delay. In other words, instead of finding a

* Corresponding author. Tel.: +86 15091671869.
E-mail address: hrhnpu@gmail.com (R. Hou).

1084-8045
http://dx.doi.org/10.1016/j.jnca.2013.09.010

single path given a cost constraint, the ACOP problem aims at finding a set of paths representing the supported QoS.

Due to the NP-complete nature of the problem, some approximation mechanisms have been developed (Garroppo et al., 2010). They usually identify a path with a cost (or delay) within a certain deviation from the optimal one. Denote c as the estimated optimal cost of all the paths satisfying a given delay constraint d_0, which is computed by an algorithm, and c_{opt} as the optimal cost of all the paths satisfying the delay constraint d_0 in the network. $c-c_{opt}$ is thus called the *cost deviation* at the delay constraint of d_0. An algorithm is "better" if the deviation is smaller. While deviation is appropriate for measuring the performance of the DCLC solutions, we believe that it does not directly reflect the performance of the ACOP solutions in supporting admission control.

A good pre-computation mechanism should approximate the supported QoS as precisely as possible. In other words, the error in estimation should be minimized. Since any possible delay constraint is considered, this "error" is *not* a single cost deviation, but an *area* on the Cartesian plane. To illustrate, consider that there are three paths connecting a source to a destination. The QoS parameters of the paths are $(1,10)$, $(2,2)$, and $(10,1)$, where the first element in the tuple reflects the cost of the path while the second element represents the path delay. In this paper, we write the QoS parameter of a path and the constraints of a request as (*cost*,*delay*). Request (c,d) can be supported by a path with the QoS parameter (c',d'), where $c'\le c$ and $d'\le d$. Request $(5,5)$ is feasible because it can be supported by the path with the QoS parameter $(2,2)$. Request $(1,15)$ is also feasible because it can be supported by the path with the QoS parameter $(1,10)$. However, Request $(1,1)$ is not feasible because no path can support it. If we plot the QoS parameters of the path on the cost-delay plane, any request that can be supported by any of the paths can be easily identified. Refer to Fig. 1(a), the shaded area is the optimal supported QoS, in which any request that falls in the region is feasible. Thus, a good pre-computation scheme should approximate this area as precisely as possible. The "error" in approximation is the difference in terms of the area between the region of the optimal supported QoS and that of the approximated supported QoS.

While cost deviation is related to the difference in area, it is not sufficient. For example, the shaded areas in Fig. 1(a) and (b) represent the optimal and approximate supported QoS regions, respectively. According to Fig. 1(b), when delay is two, the approximate best cost is three. The cost deviation is $3-2=1$, as the optimal cost is two. The area of $\{[2,3]\times[2,10]\}$ is the "error" in estimating the supported QoS. Any request with the QoS requirements falling in this area is considered as infeasible but actually they are supported by the network. For example, Request $(2,5)$ is in fact feasible but can be rejected by any approximation algorithms based on the approximate QoS in Fig. 1(b). On the other hand, in Fig. 1(d), the cost deviation with the delay constraint of two is also one. However, we can observe that the "error" in Fig. 1(d) is much smaller than that in Fig. 1(b). Request $(2,5)$ would be correctly classified as feasible. Thus, the approximate supported QoS in Fig. 1(d) provides a better network QoS providence than that in Fig. 1(b). The above example illustrates that cost deviation cannot sufficiently reflect the admission control ability of the algorithm, while the "area" does. In this work, we propose a new metric, known as *distortion area*, which is defined as the difference between the approximate supported QoS region and the optimal supported QoS region, to evaluate the accuracy performance of the algorithm for estimating the supported QoS. We first analyze the distortion area of the representative algorithms described in Orda and Sprintson (2003) and Xue et al. (2007). Then, we illustrate how to improve the algorithm to reduce the error.

2. Related work

The MCP problem and RSP (DCLC) problem have been studied extensively (Garroppo et al., 2010). The work in Hassin (1992) focuses on the RSP problem and presents two polynomial algorithms. The author first presented an ε-approximation algorithm by using the basic technique of rounding and scaling with the time complexity $O(\log\log(\frac{UB}{LB})(|E|(n/\varepsilon)+\log\log(\frac{UB}{LB})))$, where UB and LB are the cost metrics of the minimum delay path and the minimum cost path, respectively, $|E|$ is the number of links, and n is the number of nodes. Then, the author applied the basic technique of *interval partitioning* in Sahni (1977) to design a second $\varepsilon-$approximation algorithm with the time complexity of $O(|E|(n^2/\varepsilon)\log(n/\varepsilon))$. Since the time complexity of the first algorithm depends on the upper bound of delay metric of each link, this algorithm is classified in Orda and Sprintson (2003) as a *pseudo-polynomial algorithm*. To the best of our knowledge, the algorithm in Xue et al. (2007) is currently fastest one for precomputing the supported QoS with two additive constraints, which has the complexity of $O(|E|n(\log\log n+1/\varepsilon))$, where ε is the small positive constant. The smaller ε, the higher accuracy but the higher computational overhead. Chen et al. (2008) design heuristic techniques to improve the performance computational overhead of the algorithm in Goel et al. (2001). As we know, heuristic algorithm cannot provide performance guarantees and the introduced approximation error is difficult to be bounded.

Xue et al. (2008) study the decision version of the problem. Given a connection request with two additive constraints, the proposed algorithm either finds a feasible solution or confirms that there does not exist a source-destination path whose first metric is bounded by the first constraint and whose second weight is bounded by $(1-\varepsilon)$ times the second constraint. Afterwards, Huang et al. (2012) enhance the algorithm proposed by Xue et al. (2008) by reducing the computational complexity. Li and Zhang (2010) study MCP problem in smart grid and propose a simple heuristic algorithm. Avallone and Ventre (2012) develop a routing algorithm which searches for a feasible path for a given flow request with multiple additive constraints that requires the least number of nodes and links to be turned on. Lu and Zhu (2013)

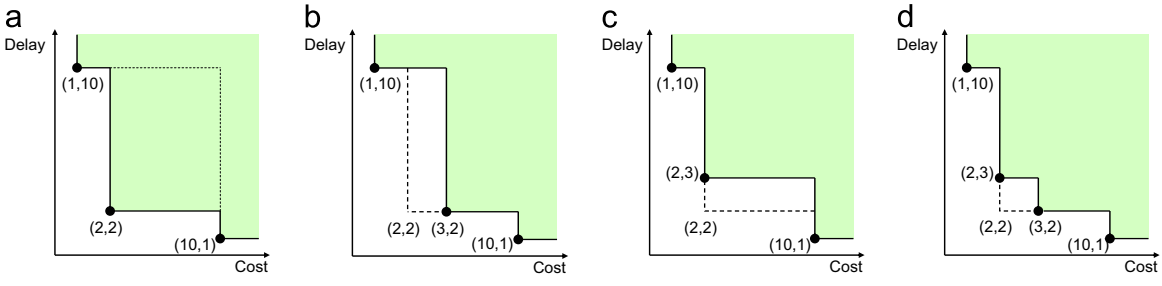

Fig. 1. An illustration for the different approximate solutions. (a) The optimal solution, (b) with cost quantization, (c) with delay quantization, (d) cost and delay quantization.

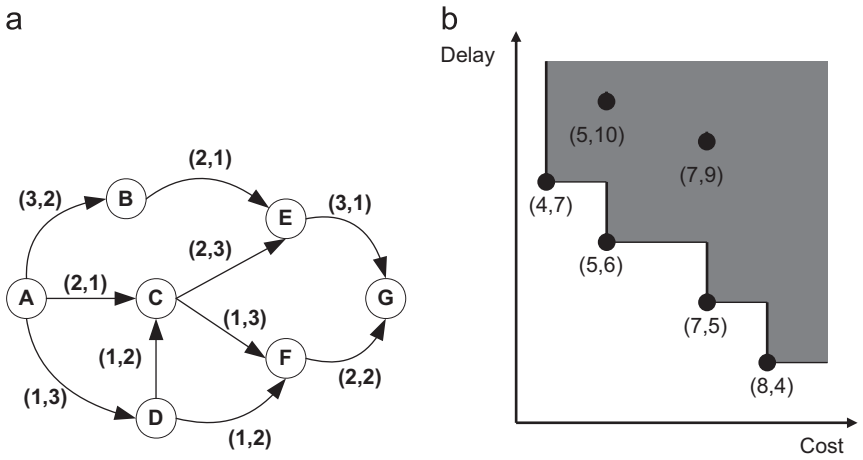

Fig. 2. An illustration for the supported QoS. (a) A simple network and (b) cost-delay plane.

present a genetic algorithm to find a path satisfying a connection request with two additive constraints. With the supported QoS region, the decision can be made immediately based on the QoS requirements.

In fact, the common technique used for designing approximation algorithms (for solving both RSP and MCP) is to map the cost (or delay) value of each link to an integer. Then, the cost (or delay) of a path becomes a value in a finite set of integers instead of the continuous real number line. We refer this technique as *quantization*. For example, in the *uniform scaling* quantization method, a link cost is multiplied by a constant and then round (up or down) to an integer. Moreover, there is another quantization method called *logarithmic scaling* (Orda and Sprintson, 2003), and we shall describe it in more detail later.

After quantizing the link costs (delays), we can apply the approximation algorithm to solve RSP for each possible cost (or delay) constraint to get the ACOP solution (Orda and Sprintson, 2003). Unfortunately, the complexity will be very high, and thus the authors in Orda and Sprintson (2003) developed a less computationally expensive approximation algorithm. Nevertheless, the performance analysis presented in Orda and Sprintson (2003) is based on cost deviation, but not on distortion area. In this paper, we shall analyze the upper bound of the distortion area produced by the approximation algorithm in Orda and Sprintson (2003). Nevertheless, the analysis can be generalized for other approximation algorithms based on link metric quantization.

Cui et al. (2003, 2005) also study the problem of precomputing the supported QoS. The proposed mechanisms are heuristic and there is no performance guarantee. There are two kinds of errors for admission control in these algorithms. The first one is when an algorithm rejects a feasible connection request, and the second one is whenever an algorithm accepts an infeasible connection request. The "error" defined in Cui et al. (2003, 2005) includes both kinds of errors. Since the quantization-based algorithms presented in our work will not accept an infeasible connection request, they will not introduce the second kind of error. Therefore, the *distortion area* presented by our work is suitable to reflect the admission control performance of the algorithms.

3. Network model and problem formulation

We model a computer network by a directed graph $G=(\mathcal{V},\mathcal{E})$, where $\mathcal{V}$ is the set of n vertices, and $\mathcal{E}$ is the set of m edges. Each edge $l=(u,v)$ is associated with two additive metrics, namely, cost and delay. Let (c_l,d_l) be the QoS parameter of Edge l, where c_l and d_l are the cost and delay of l, respectively. We assume that all parameters (cost and delay) are positive, bounded, and independent. Let $\mathcal{A}(v)$ be the node neighbor set of v. We define the optimal delay function of Link l, $\mathcal{D}_l^{\mathrm{opt}}(c)$, to specify the minimum delay value provided by link l at the cost constraint of c. Thus, $\mathcal{D}_l^{\mathrm{opt}}(c)$ is equal to ∞ if $c<c_l$, otherwise, it is equal to d_l if $c\geq c_l$.

Given a path $\mathcal{P}$ from s to g, the optimal delay function of Path $\mathcal{P}$, $\mathcal{D}_{\mathcal{P}}^{\mathrm{opt}}(c)$, is the minimum delay value provided by this path with a cost constraint of c. Thus, $\mathcal{D}_{\mathcal{P}}^{\mathrm{opt}}(c)$ is equal to ∞ if $c<\sum_{l\in\mathcal{P}}c_l$, otherwise, it is equal to $\sum_{l\in\mathcal{P}}d_l$.

Let $\mathbb{P}_{s\to g}$ be the set of paths from s to g. We define the optimal delay function from s to g, $\mathcal{D}_{s,g}^{\mathrm{opt}}(c)$, which is the minimum delay value provided by all the paths from s to g with the cost constraint of c, as $\mathcal{D}_{s,g}^{\mathrm{opt}}(c)=\min_{\mathcal{P}\in\mathbb{P}_{s\to g}}\{\mathcal{D}_{\mathcal{P}}^{\mathrm{opt}}(c)\}$.

Consider a simple network in Fig. 2(a). $\mathbb{P}_{A\to G}$ has six paths: Path $A\to D\to F\to G$ with the QoS parameter $(4,7)$, Path $A\to C\to F\to G$ with the QoS parameter $(5,6)$, Path $A\to D\to C\to F\to G$ with the QoS parameter $(5,10)$, Path $A\to C\to E\to G$ with the QoS parameter $(7,5)$, Path $A\to D\to C\to E\to G$ with the QoS parameter $(7,9)$, and Path $A\to B\to E\to G$ with the QoS parameter $(8,4)$. For the given path $\mathcal{P}_1=A\to B\to E\to G$, the corresponding optimal delay function is $\mathcal{D}_{\mathcal{P}_1}^{\mathrm{opt}}(c)=\infty(\text{if } c<8)$ or $4(\text{if} c\geq 8)$.

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

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

As illustrated in Fig. 2(b), $\mathcal{D}_{A,G}^{\mathrm{opt}}(c)$ is a staircase on the cost-delay plane, which is also called the *efficient frontier* in Bauer et al. (2000). We would like to introduce several definitions.

Definition 1. A point (x,y) is more representative than another point (x',y'), denoted by $(x,y)\prec(x',y')$, if $x\neq x'$ or $y\neq y'$, moreover, $x\leq x'$ and $y\leq y'$.

Definition 2. Given a set $\mathcal{S}$ of the QoS parameters, $(x,y)\in\mathcal{S}$ is a representative point of $\mathcal{S}$ if there does not exist any other point $(x',y')\in\mathcal{S}$ such that $(x',y')\prec(x,y)$.

For example, in Fig. 2(b), there are totally six QoS parameters, but four of them, namely, $(4,7)$, $(5,6)$, $(7,5)$, and $(8,4)$, are the

optimal *representative points*.[1] Denote $PF_{s,g}^{opt}$ as the set of representative points on the efficient frontier from s to g. In Fig. 2(b), $PF_{A,G}^{opt}=\{(4,7),(5,6),(7,5),(8,4)\}$.

Definition 3. Given a set of the optimal representative points PF^{opt}, define $\mathbb{R}=\{(c,d)|(c',d')\prec(c,d),(c',d')\in PF^{opt}\}$. The *feasible area* is defined as $\mathbf{A}_{feasible}^{opt}=\mathbb{R}\cup PF^{opt}$.

Definition 3 was also described in Bauer et al. (2000) and Cui et al. (2003, 2005). Any request that falls in the feasible area must be supported by at least one path. We call this request a *feasible request*. The problem of precomputing the supported QoS aims at finding the feasible area $\mathbf{A}_{feasible}^{opt}$ so that a routing table can tell whether a request is feasible upon a request arrives. This problem is NP-complete.

The feasible area can be uniquely defined by the set of the optimal representative points PF^{opt}. In Fig. 2(b), the shaded area is the feasible area which is on the upper right hand side of the efficient frontier. For instance, Request $(10,6)$ falls in the shaded area. The paths $(8,4)$ and $(7,5)$ can serve this request. Thus, this request is a feasible request.

Finding the optimal feasible area is NP-Complete (Orda and Sprintson, 2003). Some existing works propose the approximation algorithms to estimate the feasible area. However, no work analyzes the upper bound of the approximation error caused by the existing works, which is called *distortion area* in this work. In this work, we first analyze the upper bound of the approximation error produced by the existing quantized algorithms. We then propose a new method to estimate the feasible area, and also give the theoretical comparison between the existing schemes and our proposed method.

4. Analyzing the distortion area of the existing algorithms

In this section, we first present an exact pseudo-polynomial algorithm for computing the supported QoS with an integer cost metric. In fact, this algorithm was mentioned in many existing literatures (Hassin, 1992; Orda and Sprintson, 2003). We then present the existing quantization-based methods applied for precomputing the supported QoS. Finally, we analyze the performance of the existing quantization-based algorithms based on the proposed metric, *distortion area*. As mentioned in Section 1, distortion area is defined as the difference between the approximate supported QoS region calculated by an approximation algorithm and the optimal supported QoS region. For instance, the shaded area in Fig. 3 shows the optimal supported QoS region while Fig. 4(a) shows an approximate supported QoS region. We calculate the shaded area in Fig. 3 as 0.44 while that in Fig. 4(a) as 0.4. The distortion area caused by the approximation algorithm is 0.04. If we assume that the requirements of the connection request are uniformly distributed in the supported QoS region, we can consider that almost 10% of all the requests would be rejected by the approximation algorithm, while these requests actually can be supported by the network. In the following, we would give the upper bound of the distortion area introduced by different algorithms.

Suppose that we arrange the points in $PF_{s,g}^{opt}$ in cost-ascending order. The first representative point $r_c=(L_c,U_d)$ corresponds to the minimum cost path, and the last representative point $r_d=(U_c,L_d)$ corresponds to the minimum delay path. All other representative points must have a cost falling between L_c and U_c, and a delay falling between L_d and U_d (Bauer et al., 2000). If we obtain all the representative points, we can derive the optimal delay function $\mathcal{D}_{s,g}^{opt}(c)$, and vice versa. For example, in Fig. 2(b), the QoS parameter of the minimum cost path is $(4,7)$, and the QoS parameter of the minimum delay path is $(8,4)$. In order to compute the optimal delay function $\mathcal{D}_{A,G}^{opt}(c)$, we just need to find the representative points $(5,6)$ and $(7,5)$, which have the costs falling between 4 and 8. To make the problem tractable, we first consider that the cost metric associated with each link is integer. For the ease of the subsequent discussion, we assume that U_c and U_d are the same, denoted by UB, and L_c and L_d are the same, denoted by $LB=1$.

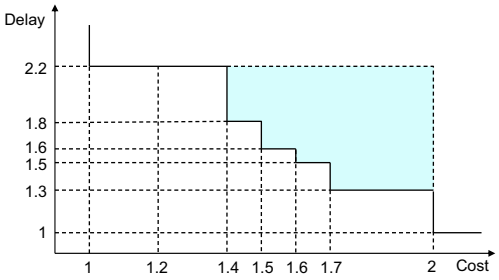

Fig. 3. The optimal delay function.

4.1. Exact algorithm

Similar to Hassin (1992) and Orda and Sprintson (2003), we assume that, for simplicity, network can be represented by a directed acyclic graph (DAG). The extension of the algorithm for a general graph is straightforward. In DAG, the network nodes are numbered in a way such that $(i,j)\in\mathcal{E}$ implies $i<j$. If the cost metric of each link is an integer, we can develop a pseudo-polynomial algorithm for computing the supported QoS as follows:

$$\begin{aligned}&\mathcal{D}_{g,g}(c)\leftarrow 0,\quad c\geq 0;\\&\mathcal{D}_{i,g}(c)\leftarrow\infty,\quad c\geq 0, i\in\mathcal{V}\backslash\{g\};\\&\mathcal{D}_{i,g}(c)=\min_{k\in\mathcal{A}(i)}\{\mathcal{D}_{k,g}(c-c_{(i,k)})+d_{(i,k)},\mathcal{D}_{i,g}(c-1)\}\\&\quad c=0,1,2,3,\ldots,\mathrm{UB},\ i\in\mathcal{V}.\end{aligned}\tag{1}$$

Orda and Sprintson (2003) and Hassin (1992) give the pseudo-polynomial algorithm description as the same as (1). To compute $\{\mathcal{D}_{i,g}(c),c\geq 0\}$ for all $i\in\mathcal{V}$, we keep a table of $|\mathcal{V}|$ rows and UB columns, where one row for each node and one column for each integer cost value. To ease our discussion, we label the nodes as $1,2,\ldots,|\mathcal{V}|$. The entry on Row i and Column j represents the estimated delay from Node i to Node g at Cost j. Initially, $\mathcal{D}_{g,g}(j)$, for all $j=0,1,\ldots,\mathrm{UB}$, are all set to be zero while $\mathcal{D}_{i,g}(j)$, for all $i\neq g$ and $j=0,1,\ldots,\mathrm{UB}$, are all set to be infinity. In the first step, each neighbor u of g sets $\mathcal{D}_{u,g}(c)$ to be $d_{u,g}$, where $c=c_{u,g},c_{u,g}+1,\ldots,\mathrm{UB}$. In step k, we update $\{\mathcal{D}_{i,g}(c),c\geq 0\}$ for those nodes i that can be k hops away from g. After $|\mathcal{V}|-1$ steps, the algorithm terminates since no path can have more than $(|\mathcal{V}|-1)$ hops. As referred to Orda and Sprintson (2003), the computational complexity of this pseudo-polynomial algorithm is $\mathcal{O}(|\mathcal{E}|\mathrm{UB})$.

4.2. Existing quantization-based algorithms

In general, cost values are not necessarily integers. The cost value of each link is quantized such that it is selected from a set of possible values, instead of the continuous real number line. Given the lower bound and the upper bound of the cost values L_c and U_c, we obtain a set of possible quantized cost values, denoted by $\{s_1,s_2,\ldots,s_n\}$. We then assume that each link or path cost is one of the quantized values. For instance, if the cost value of a link c_l falls between s_j and s_{j+1}, where $j=1,\ldots,n-1$, we assume $c_l\approx s_{j+1}$. Hence, we can use (1) to compute the delay function at the

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

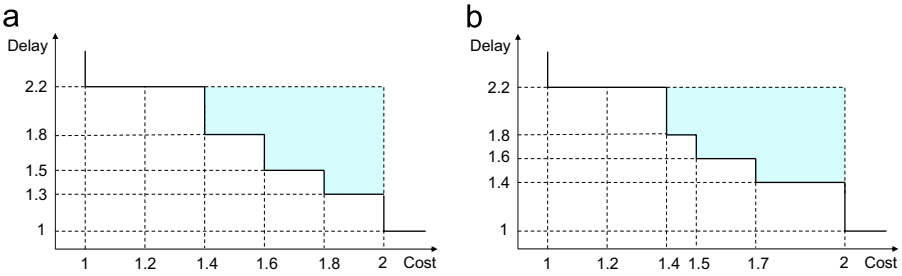

Fig. 4. The approximated functions with uniform scaling. (a) Uniformly sampled cost values and (b) uniformly sampled delay values.

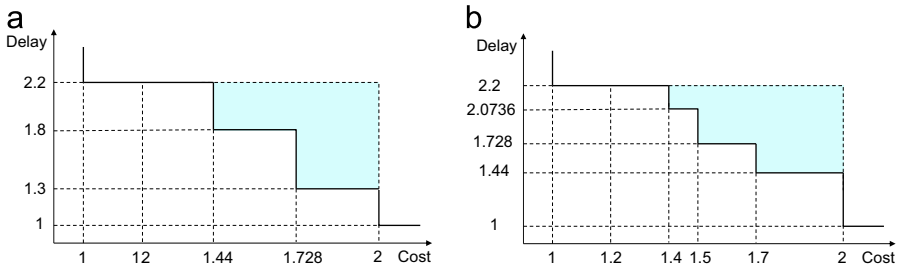

Fig. 5. The approximated functions with logarithmic scaling. (a) Logarithmically sampled cost values and (b) logarithmically sampled delay values.

samples. We thus can obtain the approximated representative points based on the quantized cost values. We call this method the *quantization-based approximation method*.

As mentioned in Section 2, the existing approximation algorithms generally apply this quantization method to solve the NP-complete problem. Different algorithms can indeed apply different quantization methods (Orda and Sprintson, 2003; Hassin, 1992; Lorenz and Raz, 2001; Goel et al., 2001; Chen et al., 2008; Sahni, 1977; Xue et al., 2007). Most of them apply uniform scaling, and the work in Orda and Sprintson (2003) applies logarithmic scaling. Generally speaking, when applying uniform scaling, the set of possible cost values is denoted by $\{1, x\delta, (x+1)\delta, \ldots, (x+m)\delta, \mathrm{UB}\}$, where $x = \min\{k | (k\delta > 1)(k \in \mathbb{Z}^+)\}$, and $m = \max\{t | ((x+t)\delta < \mathrm{UB})(t \in \mathbb{Z}^+)\}$. δ is the scaling parameter. Different methods may select different values for δ. We shall discuss how to calculate the distortion area produced by uniform scaling with the given scaling parameter δ. In logarithmic scaling, the cost metric of each path is selected from $\{1, (1+\delta), (1+\delta)^2, \ldots, (1+\delta)^n, \mathrm{UB}\}$, where $n = \max\{j | ((1+\delta)^j < \mathrm{UB})(j \in \mathbb{Z}^+)\}$.

Consider the optimal delay function $\mathcal{D}^{\mathrm{opt}}_{s,g}(c)$ as depicted in Fig. 3. Let $\mathrm{UB} = 2$ and $\delta = 0.2$. By using logarithmic scaling, there are three samples for cost, namely, 1.2, 1.44, and 1.728. From Fig. 3, the minimum delays at these cost values are 2.2, 1.8, and 1.3. We thus obtain the resulting approximated delay function as exhibited in Fig. 5(a). On the other hand, when uniform scaling is used, 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. 4(a). It can be observed that different scaling schemes lead to different approximated delay functions.

Define $\mathcal{D}_{s,g}(c)$ (without the *opt* superscript) as the approximated delay function. Without loss of generality, assume that the set of the possible cost values is $\mathbb{K} = \{1, k_1, k_2, \ldots, k_m, \mathrm{UB}\}$, which is generated by either uniform scaling or logarithmic scaling or any other quantization method. Let $k_0 = 1$ and $k_{m+1} = \mathrm{UB}$, and we compute the value of $\mathcal{D}_{s,g}(c)$ at the quantized values within $[1, \mathrm{UB}]$ as follows:

$$\begin{aligned}
&\mathcal{D}_{g,g}(c) = 0, \quad c \geq 0;\\
&\mathcal{D}_{i,g}(c) \leftarrow \infty, \quad c \geq 0, i \in \mathcal{V}\backslash\{g\};\\
&\mathcal{D}_{i,g}(c) = \min_{k \in \mathcal{A}(i)} \{\mathcal{D}_{k,g}(c - c_{(i,k)}) + d_{(i,k)}, \mathcal{D}_{i,g}(c')\}\\
&\quad c = k_j,\ c' = k_{j-1} \text{for } j = 0, \ldots, m+1. \qquad (2)
\end{aligned}$$

If $c = k_0$, then $c' = 0$. Eq. (2) is similar to (1), except that the set of the cost values is different. In (1), the QoS metric of each link is an integer. We just compute the minimum delay value at the integer cost value. On the other hand, in (2), the cost value is selected from a set of real numbers. We compute the minimum delay values at the discrete cost values, which may not necessarily be integers. By applying the quantization method, the representative points in $\mathcal{D}_{s,g}(c)$ must be at the selected cost values. The pseudo-code of the approximation algorithm can be referred to http://www.eee.hku.hk/research/technical_reports.htm.

As we only consider a limited number of the cost values, some representative points may be missed out under the estimation. A drop in the minimum delay may happen at a later quantized cost value. For example, the representative point $(1.5, 1.6)$ appears in Fig. 3 but not in Fig. 5(a). Besides, the optimal delay function in Fig. 3 drops to 1.3 at cost 1.7 but the drop occurs at 1.8 in Fig. 4(a). In other words, the scaling method can overestimate the delay at some costs. The following lemma describes this phenomena.

Lemma 1. *For any $c > 0$, if $\mathcal{D}_{s,g}(c) < \infty$, $\mathcal{D}_{s,g}(c) \geq \mathcal{D}^{\mathrm{opt}}_{s,g}(c)$.*

Proof. $\mathcal{D}_{s,g}(c)$ is initially set to infinity. According to (2), if $\mathcal{D}_{s,g}(c) < \infty$, there exists a physical path satisfying Request $(c, \mathcal{D}_{s,g}(c))$. If $\mathcal{D}_{s,g}(c) < \mathcal{D}^{\mathrm{opt}}_{s,g}(c)$, the optimal minimum delay from s to g with the cost constraint of c becomes $\mathcal{D}_{s,g}(c)$, which is less than $\mathcal{D}^{\mathrm{opt}}_{s,g}(c)$. In this case, $\mathcal{D}^{\mathrm{opt}}_{s,g}(c)$ is no more the optimal delay function, which contradicts our assumption. □

For ease of discussion, we drop the subscripts s and g in the delay functions and simply use $\mathcal{D}(c)$ and $\mathcal{D}^{\mathrm{opt}}(c)$ instead when the context is clear.

4.3. Distortion area analysis

Let PF be the approximated representative points found by the algorithm. For instance, in Fig. 4(a), PF has (1,2.2), (1.4,1.8), (1.6,1.5), (1.8,1.3), and (2,1). $\mathrm{PF}^{\mathrm{opt}}$ has (1,2.2), (1.4,1.8), (1.5,1.6), (1.6,1.5), (1.7,1.3), and (2,1), as illustrated in Fig. 3. For each point x on the cost-delay plane, denote $x \cdot c$ and $x \cdot d$ as the cost metric and delay metric of this point, respectively. For each representative point $r' \in \mathrm{PF}$, we have $r' \cdot d = \mathcal{D}(r' \cdot c)$. By Lemma 1, $\mathcal{D}^{\mathrm{opt}}(r' \cdot c) \leq \mathcal{D}(r' \cdot c) = r' \cdot d$. This implies that there exists a representative point

$r \in \mathrm{PF}^{\mathrm{opt}}$ such that $r \prec r'$ or $r = r'$. Let $\mathbf{A}_{\mathrm{feasible}}$ be the feasible area estimated by an algorithm, which is defined by PF. We thus have $\mathbf{A}_{\mathrm{feasible}} \subseteq \mathbf{A}^{\mathrm{opt}}_{\mathrm{feasible}}$. As mentioned in Section 1, the distortion area is defined as $\mathbf{A}_{\mathrm{error}} = \mathbf{A}^{\mathrm{opt}}_{\mathrm{feasible}} \backslash \mathbf{A}_{\mathrm{feasible}}$. Although $\mathbf{A}^{\mathrm{opt}}_{\mathrm{feasible}}$ and $\mathbf{A}_{\mathrm{feasible}}$ are both infinite, $\mathbf{A}_{\mathrm{error}}$ must be finite since both $\mathbf{A}^{\mathrm{opt}}_{\mathrm{feasible}}$ and $\mathbf{A}_{\mathrm{feasible}}$ contain the bounded minimum cost and minimum delay representative points.

As discussed in Section 1, the distortion area reflects the admission control performance of the network. We apply the *distortion area* as the metric to evaluate the performance of the quantization-based algorithms for estimating the supported QoS.

Lemma 1 implies that if $\mathcal{D}(c) = \mathcal{D}^{\mathrm{opt}}(c')$, $c \geq c'$. For example, in Figs. 3 and 5(a), the corresponding costs for delay=2 are 1.4 and 1.44, respectively, where the one on the approximation function is larger. The *cost deviation* captures the difference in the cost values of the optimal and approximated delay functions, as referred to Definition 4.

Definition 4. If c_0 is the cost value of a representative point, we call c_0 a representative cost. The *cost deviation* at the representative cost c_0 on $\mathcal{D}^{\mathrm{opt}}(c)$, denoted as $\mathcal{CD}(c_0)$, is $\min\{c | \mathcal{D}(c) \leq \mathcal{D}^{\mathrm{opt}}(c_0)\} - c_0$.

The existing algorithms also give the upper bound of the cost deviation by using different quantization methods. For instance, it has been shown (Hassin, 1992) that the cost-deviation at any cost value is no more than $\mathcal{H}\delta$ if we apply the scaling set $\mathbb{K} = \{1, \delta, 2\delta, \ldots, m\delta, \mathrm{UB}\}$, where $\mathcal{H}$ is the maximum number of hops in the network. The work in Orda and Sprintson (2003) also shows that the cost-deviation at cost value c_0 is no more than $((1+\delta)^{\mathcal{H}} - 1)c_0$ by using logarithmic scaling.

Since $\mathbf{A}_{\mathrm{feasible}} \subseteq \mathbf{A}^{\mathrm{opt}}_{\mathrm{feasible}}$, the cost deviation at any cost value cannot be negative by using any kind of quantization method. Generally, denote $\mathbb{C}_{\max}$ as the maximum cost-deviation at all the cost values. It is the time to discuss how to calculate the distortion area based on $\mathrm{PF}^{\mathrm{opt}} = \{r_0, r_1, \ldots, r_{n-1}, r_n\}$ and $\mathrm{PF} = \{r'_0, r'_1, \ldots, r'_m\}$, where $r_0 = r'_0 = r_c$, $r_n = r'_m = r_d$.

Suppose that the points in $\mathrm{PF}^{\mathrm{opt}}$ and PF are sorted in cost-ascending order. Note that the first and the last representative points of $\mathrm{PF}^{\mathrm{opt}}$ and PF are the same because these two points are the minimum cost and minimum delay representative points, respectively. They can be easily found out by Dijkstra's algorithm.

We first study how to calculate the distortion area between the delay values of two consecutive representative points r_i and r_{i-1} in $\mathrm{PF}^{\mathrm{opt}}$. By Definition 4, the point $(r_i.c + \mathcal{CD}(r_i.c), r_i.d)$ is on the efficient frontier defined by $\{\mathcal{D}(c), c \geq 0\}$. If there is no representative point in PF located in the area $[0, \mathrm{UB}] \times [r_i \cdot d, r_{i-1} \cdot d]$, the area $[r_i \cdot c, r_i \cdot c + \mathcal{CD}(r_i \cdot c)] \times [r_i \cdot d, r_{i-1} \cdot d]$ is NOT included in the feasible area found by the scaling mechanism. The distortion area between the delay values of $r_1 \cdot d$ and $r_2 \cdot d$ is $(r_1 \cdot d - r_2 \cdot d) \cdot \mathcal{CD}(r_2 \cdot c)$, where $\mathcal{CD}(r_2 \cdot c) = r'_3 \cdot c - r_2.c$.

If some representative points in PF, denoted by $\{r'_k, \ldots, r'_{k+j}\}$, are located in the region spanned by $[0, \mathrm{UB}] \times (r_i \cdot d, r_{i-1} \cdot d]$, $r'_{k+l} \cdot d > r_i.d$, $r'_{k+l} \cdot c < r_i \cdot c + \mathcal{CD}(r_i \cdot c)$ for all $l = 0, \ldots j$. This means that all these representative points are located in the area $[r_i \cdot c, r_i \cdot c + \mathcal{CD}(r_i \cdot c)] \times (r_i \cdot d, r_{i-1} \cdot d]$. By Definition 4, we have $\mathcal{CD}(r_1 \cdot c) = r'_3 \cdot c$, and we can observe that both points r'_1 and r'_2 are located in the area $(r_1 \cdot d, r_c \cdot d] \times [r_1 \cdot c, r_1 \cdot c + \mathcal{CD}(r_1 \cdot c)]$.

Therefore, the distortion area between the delay values of $r_i \cdot d$ and $r_{i-1} \cdot d$ is no more than $(r_{i-1} \cdot d - r_i \cdot d) \cdot \mathcal{CD}(r_i \cdot c)$.

The total distortion area is calculated by adding the distortion areas between the delay values of two consecutive optimal representative points as follows:

$$\begin{aligned}\mathbf{A}_{\mathrm{error}} &= \sum_{i=1}^{n} \mathcal{CD}(r_i \cdot c) \cdot (r_{i-1} \cdot d - r_i \cdot d) \\ &\leq \mathbb{C}_{\max} \cdot \left(\sum_{i=1}^{n} (r_{i-1} \cdot d - r_i \cdot d) \right) \\ &\leq \mathbb{C}_{\max} \cdot \mathrm{UB} \end{aligned} \tag{3}$$

For uniform scaling, we have

$$\mathbf{A}^{\mathrm{uni}}_{\mathrm{error}} \leq \mathcal{H} \cdot \delta \cdot \mathrm{UB} = \varepsilon \mathrm{UB}, \tag{4}$$

since $\mathbb{C}_{\max} \leq \mathcal{H}\delta$ and $\delta = \varepsilon/\mathcal{H}$.

Similarly, for logarithmic scaling, we have

$$\begin{aligned}\mathbf{A}^{\mathrm{log}}_{\mathrm{error}} &\leq ((1+\delta)^{\mathcal{H}} - 1)\mathrm{UB} \cdot \mathrm{UB} \\ &< \varepsilon \mathrm{UB}^2 \left(\delta = \frac{\varepsilon}{2\mathcal{H}} \right), \end{aligned} \tag{5}$$

since $\mathbb{C}_{\max} \leq ((1+\delta)^{\mathcal{H}} - 1)\mathrm{UB}$ and $\delta = \varepsilon/2\mathcal{H}$.

5. Further reducing the error

In this section, we propose a strategy to further reduce the distortion area, which is called two-dimensional scaling. Two-dimensional scaling works with any quantization method, such as uniform scaling or logarithmic scaling. We then calculate the upper bound of the distortion area produced by two-dimensional scaling. Finally, we consider both quantization schemes, namely, uniform scaling and logarithmic scaling, and compare the error of two-dimensional scaling with that of the existing quantization-based algorithms.

5.1. Two-dimensional scaling

In the existing approximation algorithms, the cost metric is quantized, so that we can get the approximate delay function $\mathcal{D}(c)$. If we indeed quantize the delay metric by using the same scaling mechanism (logarithmic scaling or uniform scaling), we would get a different approximate cost function, denoted by $\mathcal{C}(d)$. $\mathcal{D}(c)$ and $\mathcal{C}(d)$ may represent the different approximate supported QoS regions. If we combine the two supported QoS regions represented by $\mathcal{D}(c)$ and $\mathcal{C}(d)$, we can get a more accurate supported QoS region than the one denoted by either one. For example, consider the optimal delay function depicted in Fig. 3. If we use the logarithmic scaling method on cost, we get the approximated function in Fig. 5(a). On the other hand, if we quantize on delay, the function becomes the one shown in Fig. 5(b). The sets of the approximate representative points defined by the two functions are different. If we combine both sets, another set of seven representative points, i.e. {(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)} is obtained, as shown in Fig. 6(b). The supported QoS region of Fig. 6(b) is larger than that of Fig. 5(a) or (b).

Our approximation algorithm is as follows. Each node keeps two sets of the approximate representative points, $\mathrm{PF}^{d}_{u,g}$ and $\mathrm{PF}^{c}_{u,g}$. $\mathrm{PF}^{d}_{u,g}$ is obtained by computing the minimum delay with all possible quantized cost values, while $\mathrm{PF}^{c}_{u,g}$ is obtained by computing the minimum cost with all possible quantized delay values. Both $\mathrm{PF}^{d}_{u,g}$ and $\mathrm{PF}^{c}_{u,g}$ may define different supported QoS regions. We combine both sets to define the supported QoS region from u to g. The pseudo-code of our algorithm is referred to http://www.eee.hku.hk/research/technical_reports.htm. The computational complexity of the quantization-based approximation algorithm depends on the number of quantized values (Orda and Sprintson, 2003). The computational overhead of our algorithm is twice that of the existing quantization-based algorithm, since our algorithm conducts the quantization twice while the existing algorithm does once.

For clarity, the existing quantization-based algorithms use *cost-scaling*, while our approach employs *two-dimensional scaling*. Either kind of quantization schemes, such as uniform scaling or logarithmic scaling, can be applied through cost-scaling or two-dimensional

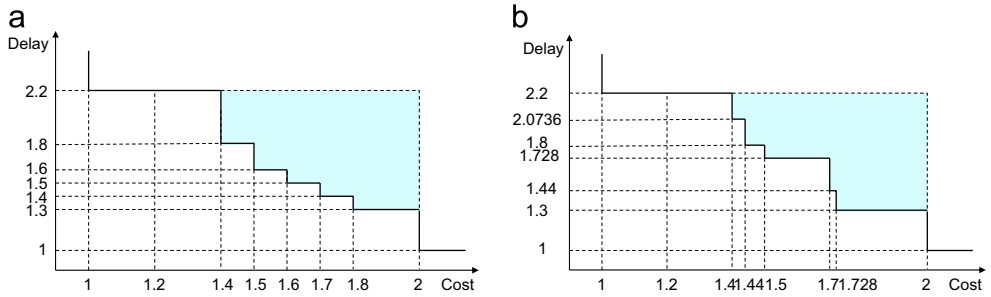

Fig. 6. The approximated functions by using the proposed scaling scheme. (a) Uniform scaling and (b) logarithmic scaling.

scaling. *Note that intuitively, two-dimensional scaling produces the smaller distortion area but the larger computational overhead if it uses the same scaling parameter as cost-scaling. In the following, we will show that with the condition that both cost-scaling and two-dimensional scaling produce the same computational overhead, two-dimensional scaling yields the smaller upper bound of the distortion area.*

5.2. Distortion area of two-dimensional scaling

In this subsection, we describe how to calculate the distortion area produced by our mechanism. Let $\mathrm{PF}^{\mathrm{opt}}=\{r_0,r_1,\ldots,r_n\}$, where $r_0=r_c$, $r_n=r_d$, and $r_i\cdot c<r_{i+1}\cdot c$ for all $i=0,\ldots,n-1$. With the similar method described in Section 4.3, we first calculate the distortion area between two consecutive delay values $r_i\cdot d$ and $r_{i-1}\cdot d$, where $i=1,\ldots,n-1$.

In two-dimensional scaling, the delay metric is also quantized. Let $\mathcal{C}^{\mathrm{opt}}(d)$ and $\mathcal{C}(d)$ be the optimal cost function and the approximated cost function, respectively. Similar to Definition 4, we define delay deviation as follows.

Definition 5. The *delay deviation* at the representative delay d_0 on $\mathcal{C}^{\mathrm{opt}}(d)$, denoted as $\mathcal{DD}(d_0)$, is $\min\{d|\mathcal{C}(d)\le\mathcal{C}^{\mathrm{opt}}(d_0)\}-d_0$.

In two-dimensional scaling, the cost metric and delay metric are quantized by the same method. Thus, the calculation of the delay deviation at a certain delay value is similar to that of the cost deviation. In other words, by using uniform scaling, the delay deviation at a certain delay value d_0 is no more than $\mathcal{H}\delta$, and by using logarithmic scaling, the delay deviation is no more than $((1+\delta)^{\mathcal{H}}-1)d_0$. By setting the same upper bound for both cost metric and delay metric as UB, the upper bound of the delay deviation with logarithmic scaling is the same as that of cost deviation. Thus, $\mathbb{C}_{\max}$ denotes the maximum cost deviation at any cost value as well as the maximum delay deviation at any delay value produced by two-dimensional scaling.

By Definitions 4 and 5, the points $(r_i\cdot c+\mathcal{CD}(r_i\cdot c),r_i\cdot d)$ and $(r_i\cdot c,r_i\cdot d+\mathcal{DD}(r_i\cdot d))$ are in the feasible area found by our mechanism. This implies that the distortion area must be no more than $\mathcal{CD}(r_i\cdot c)\cdot\mathcal{DD}(r_i\cdot d)$. The total distortion area produced by two-dimensional scaling is no more than $\sum_{i=1}^{n-1}\mathcal{CD}(r_i\cdot c)\cdot\mathcal{DD}(r_i\cdot d)$.

We can see that the total distortion area of two-dimensional scaling depends on the number of the optimal representative points n. To gain the insight into n, we assume that the QoS metrics of all paths are independent and identically distributed (i.i.d.) random variables (Van Mieghem and Kuipers, 2003). The following lemma is established in Van Mieghem and Kuipers (2003).

Lemma 2. *The expected number of the representative points among a set of T i.i.d. points in $\mathcal{K}-$dimensional space is bounded above by* $(\ln T)^{\mathcal{K}-1}$.

Given a source s, a destination g, and other $h-1$ nodes in the network, there are at most $(h-1)!$ different h-hop paths from s to g. Therefore, there are at most $(h-1)!\binom{|\mathcal{V}|-2}{h-1}$ h-hop paths but $\binom{|\mathcal{V}|-2}{h-1}$ different QoS parameters for all h-hop paths from s to g. Therefore, the maximum number of the QoS parameters of the paths from s to g is $\sum_{j=0}^{|\mathcal{V}|-2}\binom{|\mathcal{V}|-2}{h-1}=2^{|\mathcal{V}|-2}$. For the two-dimensional case, it holds that $(\ln T)^{\mathcal{K}-1}\le\ln 2^{|\mathcal{V}|-2}\le|\mathcal{V}|-2$. We thus have $n\le|\mathcal{V}|-2$. Therefore, the distortion area introduced due to two-dimensional scaling can be calculated as

$$\begin{aligned}\mathbf{A}_{\mathrm{error}}^{2-D}&=\sum_{i=1}^{n-1}\mathcal{CD}(r_i\cdot c)\cdot\mathcal{DD}(r_i.d)\\&\le(|\mathcal{V}|-2)\cdot\mathbb{C}_{\max}^2\end{aligned}\tag{6}$$

If two-dimensional uniform scaling is applied, we have $\mathcal{DD}(r_i\cdot d)\le(|\mathcal{V}|-1)\delta$ and $\mathcal{CD}(r_i\cdot d)\le(|\mathcal{V}|-1)\delta$. Therefore, the distortion area is

$$\begin{aligned}\mathbf{A}_{\mathrm{error}}^{2-D,\mathrm{uni}}&\le(|\mathcal{V}|-2)\cdot((|\mathcal{V}|-1)\delta)^2\\&=(|\mathcal{V}|-2)\varepsilon^2\end{aligned}\tag{7}$$

where $\delta=\varepsilon/(|\mathcal{V}|-1)$.

Similarly, if two-dimensional logarithmic scaling is employed, we have $\mathcal{DD}(r_i\cdot d)\le((1+\delta)^{|\mathcal{V}|-1}-1)\cdot r_i\cdot d$ and $\mathcal{CD}(r_i\cdot c)\le((1+\delta)^{|\mathcal{V}|-1}-1)\cdot r_i\cdot c$. Therefore, the distortion area is

$$\begin{aligned}\mathbf{A}_{\mathrm{error}}^{2-D,\log}&\le(|\mathcal{V}|-2)\cdot((1+\delta)^{|\mathcal{V}|-1}-1)^2r_c\cdot d\cdot r_d\cdot c\\&<(|\mathcal{V}|-2)((1+\delta)^{|\mathcal{V}|-1}-1)^2\mathrm{UB}^2\\&\le(|\mathcal{V}|-2)\varepsilon^2\mathrm{UB}^2\end{aligned}\tag{8}$$

where $\delta=\varepsilon/2(|\mathcal{V}|-1)$.

5.3. Performance comparison

According to the discussion in Sections 4.3 and 5.2, we know that the computational overhead of a quantization-based algorithm depends on the quantization scheme employed. In this section, we consider both quantization schemes, namely, uniform scaling and logarithmic scaling, and compare the error produced by our approach and cost-scaling under the condition that they produce almost the same computational overhead.

If the same scaling parameter δ is used, it is obvious that two-dimensional scaling generally outperforms cost-scaling by producing a smaller distortion area. For two-dimensional scaling, the supported QoS region is defined by both PF^d and PF^c. However, for cost-scaling, the supported QoS region is defined by PF^d. On the other hand, two-dimensional scaling yields a larger computational overhead if both techniques apply the same scaling parameter. According to Section 4, we know that the computational overheads of both our proposed approach and cost-scaling heavily depend on the number of possible cost values (and delay values for our approach) caused by the quantization scheme. We believe that if both approaches produce the same number of the possible quantized cost values (and delay values for our approach), they produce almost the same computational overhead. Thus, we are

going to discuss how to set the scaling parameter, such that the total number of the possible quantized values is the same.

Given a scaling parameter δ, the number of possible cost values by uniform scaling is UB/δ, and that by logarithmic scaling is $\log_{1+\delta} UB$, as referred to Section 4.2. For ease of discussion, denote δ_t and δ_s as the scaling parameters adopted in two-dimensional scaling and cost-scaling, respectively.

If we use uniform scaling, by setting $\delta_t = 2\delta_s$, we can consider that the total number of quantized cost and delay values for two-dimensional scaling is the same as that for cost-scaling. We thus consider both algorithms incur a comparable computational overhead. If we use logarithmic scaling, by setting $\delta_t = 2\delta_s$, the total number of possible cost values and delay values by two-dimensional scaling is $2\log_{1+2\delta_s} UB$, and the number of cost values by cost-scaling is $\log_{1+\delta_s} UB$. When δ_s is very small, say $\delta_s \le 0.1$, we have $1+2\delta_s \simeq (1+\delta_s)^2$. This means that $2\log_{1+2\delta_s} UB \simeq \log_{1+\delta_s} UB$. Therefore, if we set $\delta_t = 2\delta_s$, we can consider that the computational overheads produced by two-dimensional scaling and cost-scaling are comparable.

Denote δ as the scaling parameter used in cost-scaling. For two-dimensional scaling, the scaling parameter of 2δ is used. If we apply uniform scaling, $\delta = \varepsilon/|\mathcal{V}| - 1$. By (4) and (7), the distortion area for cost-scaling and two-dimensional scaling is upper bounded by $\varepsilon \cdot UB$ and $4(|\mathcal{V}|-2)\varepsilon^2$, respectively. For logarithmic scaling, $\delta = \varepsilon/2(|\mathcal{V}|-1)$. By (5) and (8), the distortion area for cost-scaling and two-dimensional scaling is upper bounded by $\varepsilon \cdot UB^2$ and $4(|\mathcal{V}|-2)\cdot\varepsilon^2 \cdot UB^2$, respectively.

Since UB is the maximum cost value for a path in the network, it is upper bounded by $(|\mathcal{V}|-1)W$, where W is the maximum cost value of each link. For uniform scaling with $\varepsilon < W/4$, two-dimensional scaling yields a smaller upper bound on the distortion area than cost-scaling. For logarithmic scaling with $4(|\mathcal{V}|-2)\varepsilon < 1$, two-dimensional scaling provides a better error guarantee.

For the worst case, the distortion area incurred equals to its upper bound. This means that the distortion area for two-dimensional scaling is smaller than that for cost-scaling in the worst case analysis.

We notice that the distortion area for two-dimensional scaling is proportional to the square of ε, while that for cost-scaling is proportional to ε. Generally speaking, $\varepsilon < 1$. As ε increases, the distortion area of two-dimensional scaling grows much slower than that of cost-scaling. On the other hand, the total number of the cost and delay values taken by two-dimensional scaling reduces with the same rate as that of cost-scaling. This implies that two-dimensional scaling can effectively reduce the computational overhead with a smaller increase in the distortion area when compared with cost-scaling.

As the actual improvement depends on the network topology, we have conducted extensive simulations to study the performance, which will be discussed in Section 6. We also notice that by setting the same scaling parameter δ, uniform scaling yields a smaller approximation error but a larger computational overhead than logarithmic scaling.

6. Performance evaluation

In this section, we present our simulation results. We compare our proposed method, two-dimensional scaling, with cost-scaling. We evaluate the performance of the algorithms from the perspectives of the approximation error and the computational overhead. As discussed in Section 1, we use the distortion area as the evaluation metric for the accuracy performance of the quantization-based approximation algorithms. We use the exhaustive method to compute the optimal feasible area A^{opt}. Denote A as the estimated feasible area computed by an algorithm. The distortion area is thus $A^{opt} - A \cdot (A^{opt} - A)/A^{opt}$ is called the *region-deviation ratio* which is proportional to the distortion area. We compare the region-deviation ratio for different quantization-based approximation algorithms.

It is obvious that the region-deviation ratio depends on the scaling parameter δ. A smaller δ gives a smaller region-deviation ratio, but a larger computational overhead. In our simulation experiments, we use the running time of an algorithm as the metric for evaluating the efficiency performance. In order to fairly test the computational overheads of the different algorithms, all the algorithms run in turn under the same machine configuration and the same operation system (Fedora 10). Moreover, during the running of each algorithm, the machine did not perform any other task. We use the function "gettimeofday" provided by the system to obtain the exact starting time and the end time, in order to calculate the running time delivered by different algorithms.

Similar to Xue et al. (2007), we used BRITE (Huang et al., 2010), a well-known Internet topology generator, to generate network topologies using the Waxman model. We apply the default parameters provided in BRITE. The details can be referred to Xue et al. (2007) and Huang et al. (2010). The physical links in the networks are asymmetric, and the link metrics of both directions are independently generated. Cost metrics are selected uniformly from $[1, 100]$, while the delay values are selected uniformly from $[1, 300]$. We consider five different network sizes with 100, 200, 300, 400, and 500 nodes. We generate ten different instances for each network size. In each instance, we randomly select four nodes and compute the supported QoS regions from each selected node to all the other nodes in the network. There are totally $99 \cdot 4$ different supported QoSes between two nodes in a network instance. For each network size, the average region-deviation ratio is thus an average value of the region-deviation ratios among these $99 \cdot 4 \cdot 10$ configurations.

6.1. Significance of distortion area

In Section 1, we mention that the distortion area reflects the admission control performance of the network. In this section, we have conducted the simulation experiments to demonstrate the relationship between the admission control performance and the size of the distortion area.

After a source receives a connection request, it first determines whether there is a path satisfying the QoS requirement of a request. The network will accept the request if it is feasible or reject it if the source cannot find a feasible path according to its precomputed supported QoS region. We define the ratio of the number of the requests accepted to the total number of the incoming requests as the *acceptance ratio*. Denote S_{opt} as the acceptance ratio corresponding to the optimal feasible area $\mathbf{A}^{opt}$ and S as the acceptance ratio for an approximation algorithm with the approximate feasible area $\mathbf{A}$. Similar to Korkmaz and Krunz (2001), we define S/S_{opt} as the relative acceptance ratio which implies how well an algorithm works for the provision of the QoS guarantees. S/S_{opt} thus reflects the admission control performance of the network. A larger S/S_{opt} implies a better network performance.

We generate the connection requests as follows. Given a source and a destination, let p_1 and p_2 be the minimum cost and minimum delay paths, respectively. Denote the QoS parameters of p_1 and p_2 as (c_{min}, d_{max}) and (c_{max}, d_{min}), respectively. We generated 1000 requests from each node pair. Each request has the cost requirement c_{req} and the delay requirement d_{req}. As similar to Korkmaz and Krunz (2001), we take $c_{req} \sim \text{uniform}[f_l \cdot c_{min}, f_u \cdot c_{max}]$ and $d_{req} \sim \text{uniform}[f_l \cdot d_{min}, f_u \cdot d_{max}]$, where $f_l = 1$. In our simulation experiments, the value of the scaling parameter for our proposed approach is twice of that for cost-scaling.

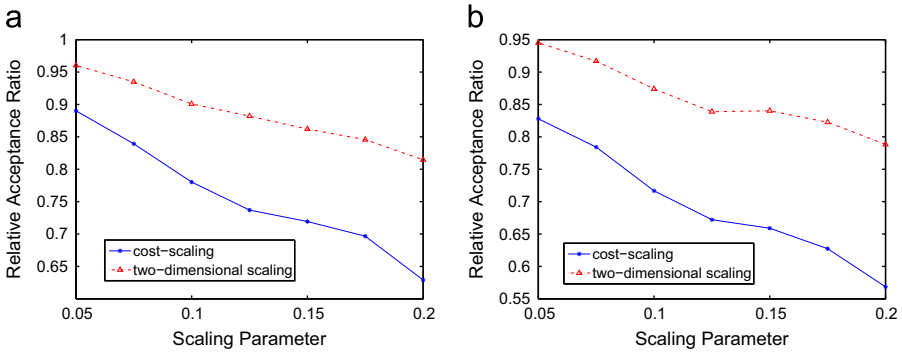

Fig. 7. Relative acceptance ratio against scaling parameter. (a) $f_u=1$ and (b) $f_u=0.9$.

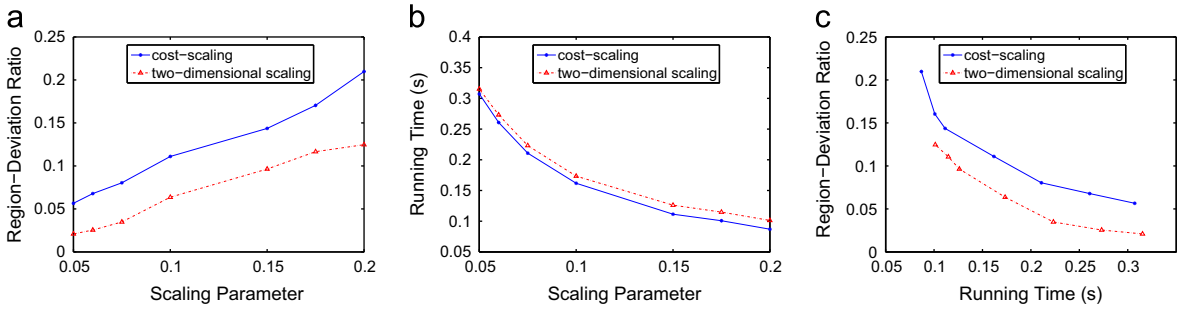

Fig. 8. Performance plots with uniform scaling. (a) Region-deviation ratio against scaling parameter, (b) running time against scaling parameter and (c) region-deviation ratio against running time.

Figure 7(a) and (b) shows the relative acceptance ratio against the scaling parameter for cost-scaling when $f_u=1$ and $f_u=0.9$, respectively. For example, when $\delta=0.05$, the scaling parameter for two-dimensional scaling becomes $2\delta=0.1$. The simulation results show that the relative acceptance ratio decreases as the scaling parameter increases. Since the distortion area becomes larger as the scaling parameter increases, the distortion area varies with the relative acceptance ratio. That is, the distortion area reflects the admission control performance of the network. We also observe that the relative acceptance ratio for our approach is higher than that for cost-scaling. In the following, we are going to show that the computational overheads of our approach and cost-scaling are almost the same. Therefore, our approach can provide a better admission control service than cost-scaling without inducing additional overhead. Comparing Fig. 7(a) and (b), we observe that the relative acceptance ratio with $f_u=0.9$ is less than that with $f_u=1$. The larger f_u implies that more requests fall in the feasible region, and so, the relative acceptance ratio with larger f_u is greater.

6.2. Two-dimensional scaling against cost-scaling

The relative acceptance ratio does not only depend on the distortion area, but also it relates to the specific QoS requirements of the connection requests. We would like to use the region-deviation ratio as the metric to evaluate the accuracy performance of an approximation algorithm.

Figure 8 shows the simulation results with uniform scaling. Figure 8(a) exhibits the relationship between the region-deviation ratios and the scaling parameter δ. We can see that the region-deviation ratio for cost-scaling is the higher than that for two-dimensional scaling. For each running time, there is a corresponding region-deviation ratio produced by algorithm. With the same running time, the lower the region-deviation ratio, the better the algorithm. Figure 8(c) shows the region-deviation ratio of two-dimensional scaling is lower than that of cost-scaling, and so the proposed method outperforms cost-scaling.

We then test the performance of the algorithms applying logarithmic scaling. Figure 9 shows the simulation results. In Fig. 9(b), the scaling parameter for two-dimensional scaling is twice of that for cost-scaling. The computational overheads for both algorithms are comparable. This accords with our theoretical analysis in Section 5.3. That is, when the scaling parameter is small enough, $(1+2\delta)$ is approximately the same as $(1+\delta)^2$. Figure 9 (c) shows the region-deviation ratio of the algorithms against the running time. We observe that the performance improvement for two-dimensional scaling over cost-scaling becomes greater with a smaller scaling parameter, which leads to a larger running time.

Finally, we would like to evaluate the performance of the algorithms with different network sizes. When applying uniform scaling, we set the scaling parameters as 0.1 and 0.2 for cost-scaling and two-dimensional scaling, respectively. When applying logarithmic scaling, the scaling parameters for cost-scaling and two-dimensional scaling are 0.05 and 0.1, respectively. The simulation results in Figs. 10(b) and 11(b) show that the computational overheads incurred by the algorithms are comparable. Figures 10 (a) and 11(a) show that the region-deviation ratio for two-dimensional scaling is less than half of that for cost-scaling. These results show that our approach outperforms over cost-scaling in the general network topology.

7. Conclusion

In this paper, we investigated the problem of precomputing the supported QoS with two additive constraints, which is NP-complete. We proposed a new metric, *distortion area*, to evaluate the performance of the approximation algorithms for estimating the supported QoS. We gave the theoretical analysis for the upper bound of the distortion area produced by the existing quantization-based approximation algorithms, and then we presented a new

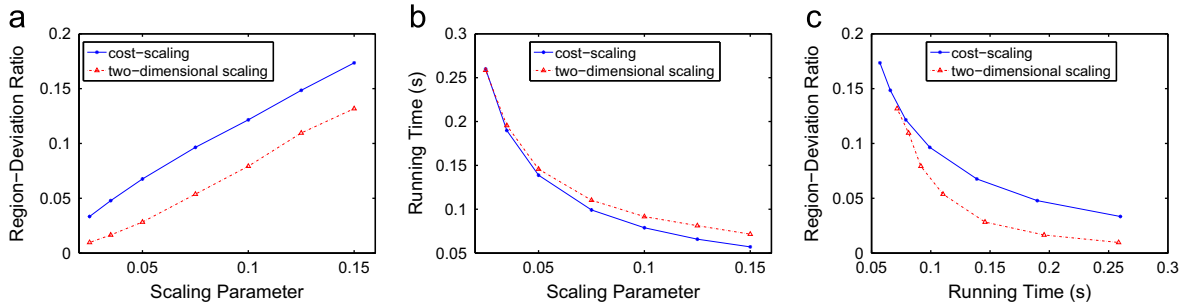

Fig. 9. Performance plots with logarithmic scaling. (a) Region-deviation ratio against scaling parameter, (b) running time against scaling parameter and (c) region-deviation against running time.

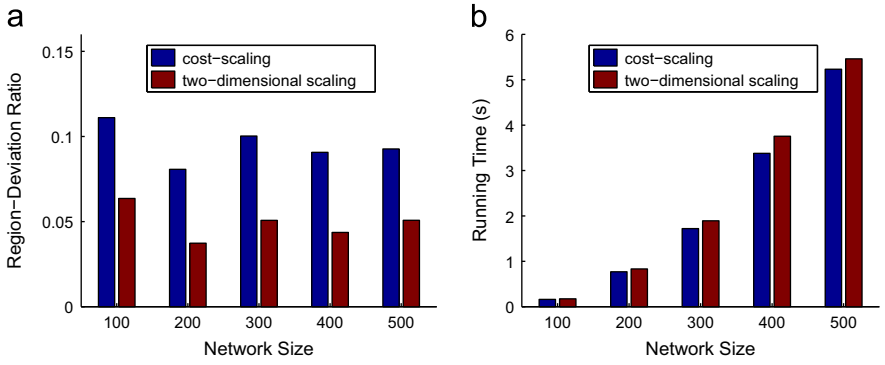

Fig. 10. Performance plots for various network sizes with uniform scaling. (a) Accuracy performance and (b) efficiency performance.

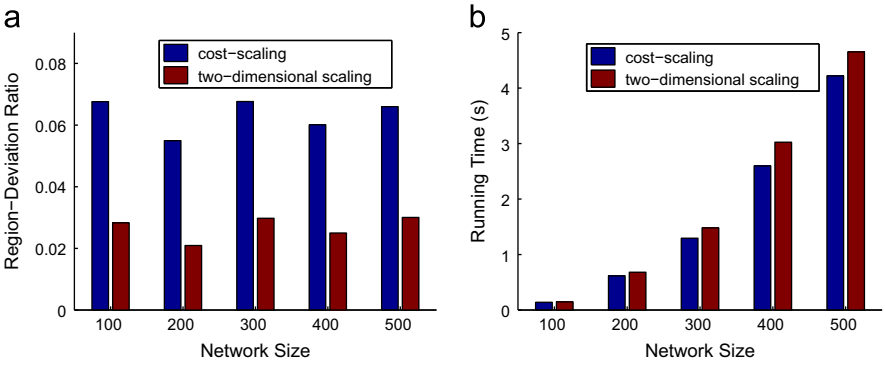

Fig. 11. Performance plots for various network sizes with logarithmic scaling. (a) Accuracy performance and (b) efficiency performance.

method to further improve the accuracy performance, which is called *two-dimensional scaling*. We also formally show that two-dimensional scaling produces the smaller approximation error than the existing methods. Finally, we demonstrated the performance of our method and compared it with the existing methods by conducting the extensive simulation experiments. Our method can be extended for the case of routing with multiple additive constraints.

Acknowledgments

This work was supported in part by the Cisco Research Initiative Award, the National Natural Science Foundation of China (Grant nos. 61101143 and 61231008), the University of Hong Kong Small Project Fundings, the Fundamental Research Funds for the Central Universities K50511010006, and the 111 Project under Grant B08038.

References

Avallone S, Ventre G. Energy efficient online routing of flows with additive constraints. Computer Networks 2012;56:2368–83.
Bauer D, Daigle JN, Iliadis I, Scotton P. Efficient frontier formulation for additive and restrictive metrics in hierarchical routing. In: IEEE ICC, June 2000, vol. 3. p. 1353–9.
Chen S, Song M, Sahni S. Two techniques for fast computation of constrained shortest paths. IEEE/ACM Transactions on Networking 2008;16(February (1)):105–14.
Cui Y, Xu K, Wu J. Precomputation for multi-constrained qos routing in high speed networks. In: IEEE INFOCOM, April 2003, vol. 2. p. 1414–24.
Cui Y, Xu K, Wu J. Precomputation for multi-constrained QoS routing in high speed networks. Journal of Computer Networks 2005;47(April (6)):923–37.
Garroppo RG, Giordano S, Tavanti L. A survey on multi-constrained optimal path computation: exact and approximation algorithms. Computer Networks 2010;54:3081–107.
Goel A, Ramakrishnan KG, Kataria D, Logothetis D. Efficient computation of delay-sensitive routes from one source to all destinations. In: Proceedings of IEEE INFOCOM 2001, 22–26 April 2001, vol. 2. p. 854–8.
Hassin R. Approximation schemes for the restricted shortest path problem. Mathematics of Operations Research 1992;17(February (1)):36–42.

Hou R, Lui K-S, Leung K-C, Baker F. Performance analysis of quantization-based approximation algorithms for precomputing the supported QoS (Technical Report), 2011. ⟨http://www.eee.hku.hk/research/technical_reports.htm⟩.
Huang J, Huang X, Ma Y. An effective approximation scheme for multiconstrained quality-of-service routing. IEEE Globecom, 2010.
Huang J, Huang X, Ma Y. Routing with multiple quality-of-services constraints: an approximation perspective. Journal of Network and Computer Application 2012;35:469–79.
Korkmaz T, Krunz M. Multi-constrained optimal path selection. IEEE INFOCOM 2001;2(April):834–43.
Li H, Zhang W. QoS routing in smart grid. In: IEEE Globecom, 2010.
Lorenz DH, Raz D. A simple efficient approximation scheme for the restricted shortest path problem. Operations Research Letters 2001;28(March):213–9.
Lu T, Zhu J. A genetic algorithm for finding a path subject to two constraints. Applied Soft Computing 2013;13:891–8.
Orda Ariel, Sprintson Alexander. Precomputation schemes for QoS routing. IEEE/ACM Transactions on Networking 2003;11(August (4)):578–91.
Sahni S. General techniques for combinatorial approximation. Operational Research 1977;25(November/December (6)):920–36.
Van Mieghem P, Kuipers FA. On the complexity of QoS routing. Computer Communications 2003;26(March (4)):376–87.
Xue GL, Sen A, Zhang W, Tang J, Thulasiraman K. Finding a path subject to many additive QoS constraints. IEEE/ACM Transactions on networking 2007;15(February (1)):201–11.
Xue G, Zhang W, Tang J, Thulasiraman K. Polynomial time approximation algorithms for multi-constained QoS routing. IEEE/ACM Transactions on Networking 2008;16(3):656–69.