Precomputation for Multi-constrained QoS Routing in High-speed Networks

Yong Cui, Ke Xu, Jianping Wu

Department of Computer Science, Tsinghua University, Beijing, P.R.China, 100084

{cy, xuke}@csnet1.cs.tsinghua.edu.cn; jianping@cernet.edu.cn

Abstract—**As one of the most challenging problems of the next-generation high-speed networks, quality-of- service routing (QoSR) with multiple (*k*) constraints is an NP-complete problem. In this paper, we propose a multi-constrained energy function-based precomputation algorithm, MEFPA. It cares each QoS weight to *b* degrees, and computes a number ($B= C_{b+k-2}^{k-1}$) of coefficient vectors uniformly distributed in the k-dimensional QoS metric space to construct *B* linear energy functions. Using each LEF, it then converts *k* QoS constraints to a single energy value. At last, it uses Dijkstra's algorithm to create *B* least energy trees, based on which the QoS routing table is created. We first analyze the performance of energy functions with *k* constraints, and give the method to determine the feasible and unfeasible areas for QoS requests in the k-dimensional QoS metric space. We then introduce our MEFPA for k-constrained routing with the computation complexity of *O*(*B*(*m*+*n*+*n*log*n*)). Extensive simulations show that, with few coefficient vectors, this algorithm performs well in both absolute performance and competitive performance. In conclusion, for its high scalability, high performance and simplicity, MEFPA is a promising QoSR algorithm in the next-generation high-speed networks.**

Keywords—**QoS routing, precomputation, scalability, linear energy function, performance evaluation**

I. Introduction

Providing different quality-of-services (QoS) support for different applications in the Internet is a challenging issue [1], of which QoS Routing (QoSR) is one of the most pivotal problems [2]. The main function of QoSR is to find a feasible path that satisfies multiple constraints for QoS applications. Although QoSR was initially proposed for the IntServ model, it could also be used in the DiffServ model [3], [4], [5]. The QoS constraints can be divided into link constraints and path constraints. The link constraint of a path can be converted to the constraint of the bottleneck link in the path, such as bandwidth. It can be easily dealt with in a preprocessing step by pruning all links that do not satisfy the constraint and computing a path from the rest sub-graph. The path constraint is the restriction of each link along the path, such as delay. Hence, in this paper we will focus on the path constraint problem.

Many heuristics have been proposed for the multi-constrained QoSR problem because of its NP-completeness [6], [7]. However, these algorithms have some or all of the following limitations [2]: (1) High computation complexity prevents their practical applications; (2) Low performance means that these algorithms sometimes cannot find a feasible path even when it does exist. (3) Some algorithms work only for a specific network. Furthermore, most of the on-line routing algorithms, which calculate the path when QoS request arrives, will not be able to afford the high computational load in high-speed networks. In most cases, a considerable reduction in the overall computational load could be achieved by precomputation, especially when the rate of QoS requests is much higher than that of (significant) changes in the network state [8].

This paper proposes a novel algorithm MEFPA (Multi-constrained Energy Function based Precomputation Algorithm) for multi-constrained QoSR problem based on the analysis of linear energy functions (LEF). We assume that each node *s* in the network maintains a consistent copy of the global network state information. This algorithm cares each QoS metric to *b* degrees. It then computes *B* ($B= C_{b+k-2}^{k-1}$) coefficient vectors that are uniformly distributed in the k-dimensional QoS metric space, and constructs one LEF for each coefficient vector. Then based on each LEF, node *s* use Dijkstra's algorithm to calculate a least energy tree rooted by s and a part of QoS routing table. At last, *s* combines the *B* parts of the routing table to form the complete QoS routing table it maintains. For distributed routing, for a path from *s* to *t*, in addition to the destination *t* and the *k* weights, the QoS routing table only needs to save the next hop of each path. For source routing, the end-to-end path from *s* to *t* along the least energy tree should be saved in the routing table. Therefore, when a QoS connection request arrives, it can be routed by looking up a feasible path satisfying the QoS constraints in the routing table.

Both theoretical analysis and experimental results show that our MEFPA can be easily implemented with high performance and high scalability. There are two major contributions in this paper. (1) We give a mathematical model that decides, in k-dimensional constraint space, the area that cannot be determined by the continuous change of k-dimensional LEFs. (2) We propose the precomputation algorithm MEFPA for multi-constrained QoSR problem.

The rest of this paper is organized as follows. Related work is discussed in Section II. We analyze the relation of LEFs and the constraint space in Section III, and then propose MEFPA in Section IV. In Section V MEFPA is evaluated by extensive simulations. Finally, conclusions appear in Section VI.

* Supported by: (1) the National Natural Science Foundation of China (No. 90104002; No. 69725003); (2) the National High Technology Research and Development Plan of China (No. 2002AA103067).

II. Related Work

Finding a feasible path that satisfies multiple constraints is a NP-complete problem, for which many heuristic algorithms have been proposed. If some scheduling schemes [9], [10] (e.g. weighted fair queuing) are used, based on the dependencies among QoS parameters, the constraints of queuing delay, jitter, and loss can be formulated as a function of bandwidth. Then the original NP-complete problem can be reduced to the standard shortest path problem [11], [12]. Based on this approach, Orda did an extensive study [13]. However, this is not the case for propagation delay, which needs to be taken into account for QoS routing in high-speed networks [14]. Furthermore, such algorithms can only be applied in networks with specific scheduling schemes.

In order to increase the applicability, pseudo-polynomial-time and approximate algorithms have been proposed. For 2-constrained problems, Jaffe proposed a distributed algorithm with the complexity $O(n^5 b \log nb)$, where b is the upper bound of the weight [15]. Because its complexity depends on the values of weights (e.g., the maximum link weight) in addition to the size of network, it is called a pseudo-polynomial-time algorithm. Approximate algorithms have been proposed for the DCLC problem [16], [17]. For an arbitrary $\varepsilon > 0$, these algorithms can find a path in polynomial time. Not only is the delay constraint satisfied in the path, but also its cost is less than $(1+\varepsilon)$ times the optimal cost. For example, Lorenz proposed the algorithm with the computation complexity $O(nm \log n \log\log n + (nm/\varepsilon))$ [17]. In order to improve the performance, the complexity of such algorithms will increase heavily.

In addition to the above algorithms, heuristics based on the convergence of multiple weights have been proposed. Jaffe proposed the linear function $g(p) = a_1 w_1(p) + a_2 w_2(p)$ to solve 2-constrained problems first [15]. As his conclusion, for a given constraint vector (c_1, c_2) of a QoS request, when $a_2/a_1 = \sqrt{c_1/c_2}$, the path found by Dijkstra's algorithm with minimizing $g(p)$ can be feasible with maximum probability. Based on the nonlinear function $g_\lambda(p) = \sum_{l=1}^{k} (w_1(p)/c_1)^\lambda$, Neve proposed the TAMCRA [18] for multi-constrained problems for PNNI protocol, and its reformation SAMCRA [19] for IP networks. The algorithm tries to find the path with minimum $g_\lambda(p)$ by heuristics. Because of the nonlinear characteristics, such a path cannot be found in polynomial time. The algorithm uses the variant of Dijkstra's algorithm to seek to find and store K undominated paths on each node with the complexity of $O(Kn\log(Kn) + K^3 km)$. Based on marking labels reversely, Korkmaz proposed H_MCOP for multi-constrained optimal-path problems [20]. This algorithm marks each node by running Dijkstra's algorithm reversely with $g_1(p)$. Then when it runs Dijkstra's algorithm forward with $g_{\lambda>1}(p)$, it considers both the labels marked reversely and the tree partly constructed forwardly. However, such a forward process cannot guarantee to find the path with minimum $g_{\lambda>1}(p)$.

An extensive survey on QoSR can be found in [2], [21]. Among the proposed QoSR algorithms, most of them use an on-line scheme. That is to say, when the QoS request arrives, the algorithms have to compute the path based on the network-state information for each request, respectively. The next-generation networks have a high speed, so that such routing schemes are difficult to be used. On the other hand, QoS requests arrive faster than network state changes. Thus, the precomputation scheme is fitter for high-speed networks than the on-line scheme [8]. However, current precomputation algorithms are often based on distance vectors and use the extended Bellman-Ford algorithm. Yuan presented a limited granularity heuristic and a limited path heuristic [22]. The former limits the weight of each link and the latter limits the size of the routing table directly. The latter has a less computation complexity $O(n^3 m \log n)$ with the routing table size $O(n^2 \log n)$. Pointing to the multi-constrained optimal problem, Orda proposed a precomputation algorithm by mapping the cost to a discrete space [8]. Its complexity is $O(\frac{1}{\varepsilon} Hm \log C)$, where H is the longest hop number and C is the upper bound of the cost. Some other early algorithms have larger complexity [23]. Additionally, distance-vector algorithms have some inherent problems, e.g. the count-to-infinity problem, inevitable routing loops and a large quantity of updating information that may overload the network.

There are three major differences between MEFPA proposed in this paper and other similar algorithms. (1) From the viewpoint of objectives and functions, MEFPA is to solve the general multi-constrained routing problem, while some other algorithms require some specific scheduling schemes [12], [13] or a limited set of QoS weights [15], [18]. (2) Viewed from the methods, MEFPA is an off-line algorithm based on network link states while some others are on-line algorithms based on link states [15], [18], [19], [20] or off-line ones based on distance vectors [22]. (3) MEFPA uses multiple energy functions that are independent of QoS requests, and the linear characteristic guarantees to find the least energy path easily. Some similar algorithms only use one energy function. They either seek for a particular linear energy function for a specific QoS request [15], or emphasize how to find the least energy path with non-linear energy function by heuristics [20].

III. Linear Energy Function Analysis

A. Problem Formulation

A directed graph $G(V, E)$ presents a network. V is the node set and the element $v \in V$ is called a node representing a router in the network. E is the set of edges representing links that connect the routers. The element $e_{ij} \in E$ represents the edge $e = v_i \to v_j$ in G. In QoSR, each link has a group of independent weights $(w_1(e), w_2(e), \cdots, w_k(e))$, which is also called QoS metric $w(e)$.

The path constraints can be divided into additive constraints (e.g. cost, delay) and multiplicative constraints (e.g. loss rate). Because either type can be transformed into the other, we only consider additive constraints in the paper. Accordingly, for a path $p = v_0 \to v_1 \to \cdots \to v_n$ and $1 \le l \le k$, the weight $w_l(e) \in R^+$ satisfies the additive characteristic if $w_l(p) = \sum_{i=1}^{n} w_l(v_{i-1} \to v_i)$.

Definition 1. *Multi-constrained path*

For a given graph $G(V,E)$, source node *s*, destination node *t*, $k \ge 2$ and a constraint vector $c = (c_1, c_2, \cdots, c_k)$, the path p from s to t is called multi-constrained path (MCP), if $w_l(p) \le c_l$ for any $1 \le l \le k$. We write $w(p) \le c$ in brief. □

Note: $w(e)$ and c are both k-dimensional vectors. For a given QoS request and its constraints *c*, QoSR seeks to find a feasible path *p* satisfying $w(p) \le c$ based on the current network-state information.

B. Linear Energy Function

Dijkstra gave the Shortest Path Tree (SPT) algorithm, which has a low computation complexity [24]. However, QoSR problem is related to multiple weights simultaneously. Thus the problem is changed to the one, in which the complexity is NP-complete, and the original Dijkstra's algorithm cannot be used to solve it. In this case, one feasible method is to convert the multiple weights to a single value, as follows:

Definition 2. *Linear Energy Function (LEF)* g_a [1]

The LEF of link *e* is defined as the linear function

$$g_a(e) = \sum_{l=1}^{k} a_l w_l \ , \qquad (1)$$

which represents the "cost" of e. Here, the coefficient $a_l \in [0,1]$ is independent of e for $l = 1,2,\cdots,k$, and it satisfies $\sum_{l=1}^{k} a_l = 1$. The vector $a = (a_1, a_2, \cdots, a_k)$ that satisfies the above conditions is called an energy coefficient. □

Based on LEF, we convert the original multi-constrained path problem to a least-energy path problem. Each coefficient of LEF represents the important degrees of different elements in the weight vector when the SPT is computed.

Theorem 1: For a given graph *G* and a vector *a*, Dijkstra's SPT algorithm with respect to $g_a(e)$ can create a least-energy tree T_a rooted by s. The path p_a along the tree T_a from s to an arbitrary node t satisfies

$$g_a(p_a) = \min_{p(s,t) \in G} g_a(p(s,t)) \ . \qquad (2)$$

Proof: The original Dijkstra's algorithm can guarantee that the path from s to an arbitrary node t along the tree has the least cost. Because $g_a(e)$ is a linear function, which satisfies $g_a(e_1 + e_2) = \sum_{l=1}^{k} a_l w_l(e_1 + e_2) = \sum_{l=1}^{k} a_l w_l(e_1) + \sum_{l=1}^{k} a_l w_l(e_2) = g_a(e_1) + g_a(e_2)$, we can calculate $g_a(e)$ for each link e first, and then run the Dijkstra's SPT algorithm with respect to $g_a(e)$ rather than the original cost. Thus, a least-energy tree can be created and p_a is the least-energy path from s to t, viz. $g_a(p_a) = \min_{p(s,t) \in G} g_a(p(s,t))$. □

If we take the energy coefficient a as the independent variable, the question is changed: For a given G and source-destination pair (s,t), when energy coefficient a reaches to all of the feasible values, what characteristics does the set $\{p_a \mid \forall a\}$ have? For example, how many elements does it have and how are they distributed? For convenience, we will first define the QoS metric space and then analyze it as followings.

C. QoS metric space

Definition 3. *QoS metric space*

$W^k = W_1 \times W_2 \times \cdots \times W_k$ is called the QoS metric space, if $w_l(p) \in W_l$ for any $p \in G$. □

For the common condition as $w_l(e) \in R^+$, we can take $W_l = R^+$, so $w_l(p) \in W_l$ for any path p. Thus, $w(p)$ is a point in the space W^k, viz. $w(p) \in W^k$, and $\{w(p_a) \mid \forall a\}$ is a set of points in W^k.

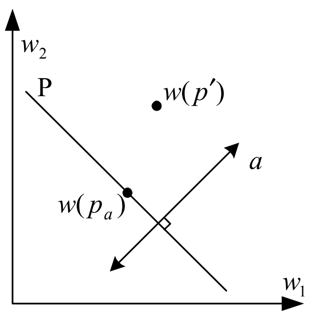
Theorem 2: For a given G, a source-destination pair (s,t), an energy coefficient a and taking $g_{opt} = \sum_{l=1}^{k} a_l w_l(p_a)$, $w(p)$ of an arbitrary path p from s to t must be on the upside of the hyperplane

$$P = \{w(p) \mid \sum_{l=1}^{k} a_l w_l(p) = g_{opt}\} \qquad (3)$$

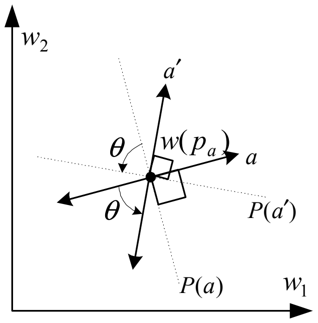
in the space W^k.

Proof: We use the reduction to absurdity. If there is a point $w(p')$ on the downside of hyperplane P, we have $g_{opt} = \sum_{l=1}^{k} a_l w_l(p_a) > \sum_{l=1}^{k} a_l w_l(p')$, which is contrary to $g(p_a) = \min_{p(s,t) \in G} g(p(s,t))$ in theorem 1. Thus, if there is any other path p' from s to t, $w(p')$ must be on the upside of the hyperplane P. □

[1] We use the term "energy" instead of the traditional name "cost", so that it is easy to distinguish the function value and link cost. A traditional cost configured for a link can be an element of the weight vector. LEF here converts multiple weights to a single value, which does not have a practical meaning.

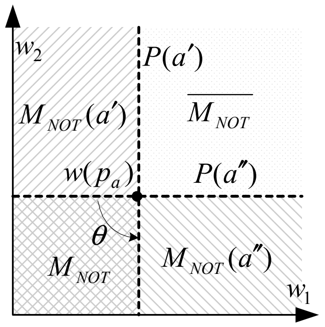


a. p' is in the upper side of P

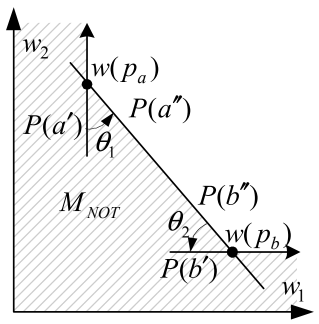


b. fixed p_a with different a

Figure 1. Vector a and hyperplane P



a. M_{NOT} with a unique p_a



b. Transition from p_a to p_b

Figure 2. Unfeasible area M_{NOT} with least energy paths p_a

For example, when $k=2$ as shown in Fig. 1, for the given vector a, we use Dijkstra's algorithm with respect to g_a and create the least-energy path p_a from s to t. Drawing the perpendicular P of vector a crossing the point $w(p_a)$, we get a partition of space W^2. All of the weight points $w(p')$ of paths p' from s to t must be on the upside of P as shown in Fig. 1.a. We should note that, because of the discreteness of network topology graph, the corresponding p_a is changed discretely with the continuous change of vector a. Thus, we can see that the mapping $a \mapsto p_a$ is not an injection. When the given vector changes from a' to a'', the least-energy path does not change, i.e. $p_b = p_a$ for $b \in [a', a'']$ in Fig. 1.b. Therefore, the single path p_a corresponds to the continuous set $\{P(b) \mid b \in [a', a'']\}$ of hyperplanes.

D. *Feasibility of the constraint in W^k*

We will give a partition of the QoS metric space W^k, including unfeasible area M_{NOT}, feasible area $M_{FEASIBLE}$ and unknown area $M_{UNKNOWN}$ in this section. Accordingly, for a given QoS request with a constraint in W^k, we can judge its feasibility and find a feasible path for a feasible request.

Definition 4. *Unfeasible area*

The point set $M(a) = \{ w \mid w \in W^k, w$ is in the lower side of $P(a)\}$ is called an unfeasible area determined by a given vector a.

$$M_{NOT} = \bigcup_{\sum_{l=1}^{k} a_l = 1, a \geq 0} M(a) \tag{4}$$

is called the unfeasible area. □

Theorem 3: For a constraint $c \in M_{NOT}$ of a request from *s* to *t*, there is NO feasible path p satisfying $w(p) \leq c$.

Proof: According to the definition of M_{NOT}, M_{NOT} is the union of all the $M(a)$ with the continuous change of vectors a. For a given constraint $c \in M_{NOT}$, there must be a hyperplane $P(a)$ corresponding to the vector a, so that c is on the downside of $P(a)$. According to theorem 2, any path p' from s to t must be on the upside of $P(a)$. Therefore, there is no feasible path p satisfying $w(p) \leq c$. □

Definition 5. *Available area*

The point set

$$M_{AVL} = \overline{M_{NOT}} = \overline{\bigcup_{\sum_{l=1}^{k} a_l = 1, a \geq 0} M(a)} \tag{5}$$

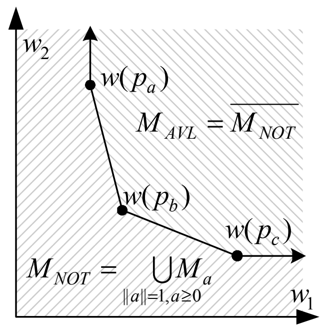
is called the available area in space W^k, where $\overline{M_{NOT}} = W^k \setminus M_{NOT}$ is the complement of M_{NOT}. □

Fig. 2 shows the relation between M_{AVL} and M_{NOT}. For example, if there is only one least-energy path, Fig. 2.a shows the unfeasible area and the available area, where $M_{NOT} = M_{NOT}(a') \cup M_{NOT}(a'')$. For the common sense where multiple least-energy paths exist, we should consider that (1) multiple vectors a may map to a single point $w(p_a)$ and (2) a single vector may also map to multiple points as shown in Fig. 2.b. When the vector changes from a' to a'' continuously, the least-energy point keeps the same value, i.e. $w(p_a)$. It is the same case of the least-energy point $w(p_a)$ when the vector changes from b' to b''. However, the vector a'', equal to b', has two least-energy paths, i.e. p_a and p_b. In this case, a discrete change of least-energy paths occurs. Such a discrete change introduces a good characteristic: although the least-energy points are discrete, the unfeasible area is continuous.

Theorem 4: The available area M_{AVL} is a convex set, the weight point of an arbitrary least-energy path p_a is on the border of M_{AVL}, and there must be a vector a mapping to a vertex of M_{AVL} so that the point $w(p_a)$ is the vertex.

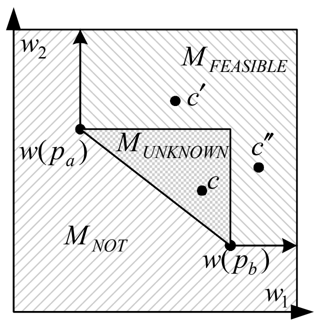
Proof: We prove the above three parts in turn.

(1) To prove that M_{AVL} is a convex set (See [25] for the



w_2
$w(p_a)$
$M_{AVL} = \overline{M_{NOT}}$
$w(p_b)$
$w(p_c)$
$M_{NOT} = \bigcup_{\|a\|=1, a\geq 0} M_a$
w_1

a. The convexity of M_{AVL}



b. Feasible constraints: c' and c''

Figure 3. Three partitions of W^k : unfeasible, unknown and feasible areas

definition and the characteristics of a convex set): For a given vector a, the corresponding hyperplane divides the space W^k into two parts and either of them is a half-space. Thus, the available area $M_{NOT}(a) = W^k \setminus M_{NOT}(a)$ is a convex set for each give vector a. On the other hand, since $M_{AVL} = \overline{M_{NOT}} = \overline{\bigcup_{\|a\|=1, a\geq 0} M_{NOT}(a)} = \bigcap_{\|a\|=1, a\geq 0} \overline{M_{NOT}(a)}$ and the intersection of convex sets is still a convex set, M_{AVL} is a convex set.

(2) To prove that an arbitrary $w(p_a)$ is on the border of M_{AVL}: For an arbitrary $w(p_a)$, drawing the hyperplane $P(a)$ crossing the point $w(p_a)$ with a being the normal vector, M is on one side of $P(a)$ according to theorem 2. Therefore, $w(p_a)$ is on the border of M_{AVL}.

(3) To prove that there must be a vector a mapping to a vertex of M_{AVL}: For any vertex $\forall x$ of M_{AVL}, because M_{AVL} is a convex set, there must be a hyperplane $P(a)$. Here, $P(a)$ crosses the point x and M_{AVL} is on one side of $P(a)$, where a is a normal vector of the hyperplane. Because M_{AVL} is an unlimited set, i.e. $(\infty, \infty, \cdots, \infty) \in M_{AVL}$, M_{AVL} must be on the upside of $P(a)$. Therefore, for a given vector a, the point $w(p_a) = x$ must be able to be calculated. □

For example, when there are multiple least-energy paths with the different energy functions, the available area M_{AVL} must be a convex set, and each vertex of M_{AVL} must be a least-energy path as shown in Fig. 3.a.

We further divide M_{AVL} into two parts: the feasible area $M_{FEASIBLE}$, and the unknown area M_{UNKNOW}.

Definition 6. *Feasible area* $M_{FEASIBLE}$

$$M_{FEASIBLE} = \{w(p) \mid \exists a, w(p) \geq w(p_a)\} \tag{6}$$

is called the feasible area. □

Theorem 5: For an arbitrary constraint c of a QoS request, if $c \in M_{FEASIBLE}$, there must be a feasible path.

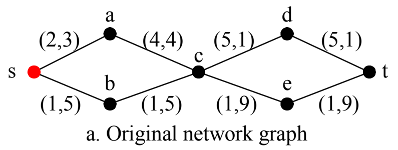
Proof: From the definition of $M_{FEASIBLE}$, $\exists a$ that satisfies $w(p_a) \leq c$, so p_a can be a feasible path for the constraint c. □

Definition 7. *Unknown area* $M_{UNKNOWN}$

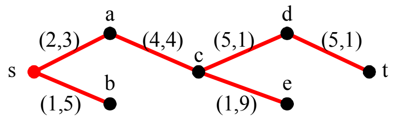
$$M_{UNKNOWN} = M_{AVL} - M_{FEASIBLE} \tag{7}$$

is called unknown area. □

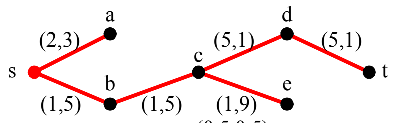
For example, in Fig. 3.b path p_a is feasible for constraint $c' \in M_{FEASIBLE}$ because $w(p_a) < c'$. We don't know the existence of a feasible path for any constraint $c \in M_{UNKNOWN}$. Thus, we divide the space W^k into three areas: an unfeasible area M_{NOT}, an unknown area $M_{UNKNOWN}$ and a feasible area $M_{FEASIBLE}$.



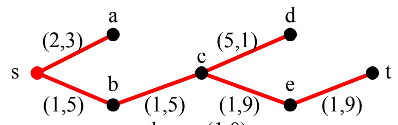
a. Original network graph



b. a=(0,1)



c. a=(0.5,0.5)



d. a=(1,0)

Figure 4. An example of proposed algorithm

IV. PROPOSED PRECOMPUTATING MEFPA

A. *The idea of MEFPA*

Based on the above theory, the set $\{p_a\}$ of least-energy paths can be pre-computed with different energy coefficients a, and then the QoS routing table can be constructed based on $\{p_a\}$. Fig. 4 shows an example, where node s is going to construct its QoS routing table. First, it uses a=(0,1) to compute the least energy tree as shown in Fig. 4.b. Then it uses a=(0.5,0.5) to construct another tree in Fig 4.c, and so on in Fig 4.d. At last, node s can get three different trees with these different vectors. All of these three trees compose the final QoS routing table maintained by s. When a QoS request arrives, s looks up a feasible path in the routing table and forwards the request.

According to the positions of QoS constraints c in the space W^k, there are three possibilities. (1) We know a feasible path for $c \in M_{FEASIBLE}$, and then we can find an element in the $\{p_a\}$ as the feasible path. (2) We know that there is NO feasible path for $c \in M_{NOT}$. Thus we can refuse the QoS request or start the QoS negotiation. (3) We don't know whether a feasible path exists for $c \in M_{UNKNOWN}$. We expect that the probability of the third case is small. In the performance evaluation (section V), extensive simulations will show that the area $M_{UNKNOWN}$ is really small and most (95%) QoS requests will be in the other two areas. Furthermore, we will also demonstrate that a QoS request in $M_{UNKNOWN}$ has a small opportunity to be feasible. Therefore, we take this area as the unfeasible area without affecting the performance significantly, so we refuse this kind of QoS requests.

A practical algorithm cannot use a continuous change of a, so we discuss how to construct multiple energy coefficient vectors a as followings. In order to make the discrete a independent of networks, we normalize the weights of each link first. That is to say, the potential maximum weight, $\max_{e \in E} w_l(e)$, is a constant, which is independent of l. Then, we select b uniform numbers in [0,1], i.e. D={0/(b-1), 1/(b-1), …,1} with totally b elements. Thus, we get the uniform vectors

$$A = \{a \mid a \in D^k, \sum_{l=1}^{k} a_l = 1\} \quad (8)$$

as the coefficients a in the subset $[0,1]^k$ of QoS metric space W^k. At last, the node, carrying out MEFPA, calculates the QoS routing table for all $a \in A$ based on the network link state it maintains.

Theorem 6: The number of the elements in the energy coefficient set $A = \{a \mid a \in D^k, \sum_{l=1}^{k} a_l = 1\}$ is

$$|A| = C_{b+k-2}^{k-1}. \quad (9)$$

Proof: There are b elements in set D={0/(b-1), 1/(b-1), …,1}, and they are uniform in [0,1]. We have $a \in D^k$, $\sum_{l=1}^{k} a_l = 1 = (b-1)/(b-1)$. Thus, if we take each 1/(b-1) as a ball, the set D can be considered as D={0 ball, 1 ball, 2 balls, …, (b-1) balls}. The total number of the balls represented by the vector a is (b-1). Additionally, the meaning of a_{l1} is different to that of a_{l2} for $l1 \neq l2$.

Therefore, we can equate $|A|$ with the number of the methods to put (b-1) same balls into k different boxes, and an arbitrary number of balls can be put into each boxes. According to the combinatorics, there are $C_{b+k-2}^{k-1} = (b+k-2)!/(b-1)!(k-1)!$ different methods to put the balls [26], i.e. $|A| = C_{b+k-2}^{k-1}$. □

```
MEFPA (G, s, k, b)
1)  IF (k==K-1)
2)      a[k]=b-1
        // we have got the coefficient a[K]
3)      FOR EACH edge  e  IN G
4)          $g_a(e) = \sum_{l=1}^{k} a_l w_l$
5)      dijkstra (G,s)
6)      FOR EACH node  t  IN G
7)          store  $p_a(s,t)$
8)  ELSE
9)      FOR(i=0; i<b; i++)
10)         a[k]=i
11)         MEFPA(G, s, k+1, b-i)
```

Figure 5. The proposed MEFPA

We denote B as the number of LEFs, i.e. $B = |A| = C_{b+k-2}^{k-1}$. For the k-constrained routing problem, node s first constructs B energy coefficients. Then s computes the least energy tree for each coefficient, respectively, and saves the path from s to each destination node t in the network along the tree to the QoS routing table. Thus, there are at most B different paths from s to each destination t. When a QoS request arrives at s, s only needs to look up a feasible path satisfying k constraints in the routing table.

B. Description of MEFPA

We propose the precomputation algorithm, MEFPA, for the k-constrained routing problem, as shown in Fig. 5. G is the network graph with K weights, s is the node running MEFPA, and (b-1) is the number of degrees to which each weight is cared. Our MEFPA, running on s, includes two parts. (a) A number B of energy coefficients $a = (a_1, a_2, \cdots, a_k)$ are constructed according to the configuration of b (Line 1, 2, 8-11). (b) A part of QoS routing table is calculated with respect to each coefficient a (Line 3-7). This includes the following steps: (i) For the given G and each vector a, calculate the energy $g_a(e)$ for each link (Line 3-4). (ii) Use Dijkstra's algorithm to compute a least-energy tree T_a rooted by node s with LEF g_a (Line 6). (iii) Save the least-energy path from s to each node along T_a to QoS routing table (Line 6-7).

Because the linear function $g_a(e)$ satisfies the isotonicity, there are no routing loops in the routing table calculated by node s [27]. Therefore, in the source routing scheme, MEFPA can avoid routing loops even when different nodes have the inconsistent copies of network state information. However, in the distributed routing scheme, we need the consistence of the network state information in different nodes. If all nodes use the routing table entries generated with same $g_a(e)$ to forward a specific QoS request hop by hop, routing loops are avoided.

C. Analysis of the proposed algorithm

We first analyze the computation complexity of MEFPA to calculate QoS routing table. In a network graph G with k QoS metrics, the node number is $n=|V|$ and the edge number is $m=|E|$. Step (i) has the complexity of $O(m)$. Step (ii) is $O(n\log n+m)$ with the improved Dijkstra's algorithm and step (iii) is $O(n)$. The number of coefficient vectors is $B=C_{b+k-2}^{k-1}$ (theorem 6). As a result, including the recursive part, the overall computation complexity of MEFPA is $O(C_{b+k-2}^{k-1}(m+n\log n+n))$, which is $B=C_{b+k-2}^{k-1}$ times the original Dijkstra's algorithm with a single weight. In a general situation, there are only a few kinds of path constraints, e.g. cost, delay, jitter, and loss rate. Therefore, k will not be very large and complexity of MEFPA is acceptable.

We now analyze the computation complexity to look up a feasible path in the QoS routing table. As we analyzed above, the QoS routing table is *B* times the original routing table with a single weight. Because the QoS routing table saves *k* weights, the current packet classification technique in multiple dimensions can achieve a constant time complexity [28]. That is to say, it only needs to access the memory for *d* times to find a route, where *d* is the reduction of ranges to prefix lookup. We can even obtain the computation complexity of O(1) if we use some special hardware, e.g. TCAM [29].

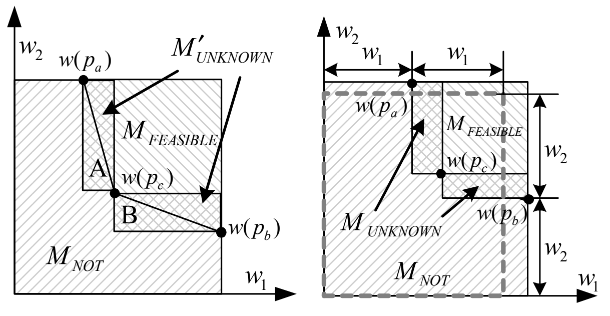
Because the value $B=C_{b+k-2}^{k-1}$ is important to MEFPA, we will illustrate the relation between *B* and the performance by extensive simulations as follows, which show that MEFPA performs well when *B* is small (e.g. when *k*=2, we let *B*=*b*=7).

V. Performance Evaluation

We first propose a method, the unknown-area proportion, to the absolute performance evaluation of MEFPA. Additionally, in order to compare it with others directly, we also simulate QoS requests to evaluate its performance. In each of these experiments with a node number N being 50, 100, 200 and 500 respectively, we generate 10 pure random network graphs [30], [31] with k weights for each link, where $w_l(e)\sim$ uniform[1,1000] for $l=1,2,\cdots,k$, and $w_l(e)$ have no correlation for different e or l. In each graph, we select source-destination node pair (*s*,*t*) 100 times (a particular node can be selected more than once), where we guarantee that the minimum hop is not less than two. Each source node *s* use our MEFPA to calculate least energy tree for $B=C_{b+k-2}^{k-1}$ times with *B* different energy coefficients a.

A. The absolute performance

QoSR algorithms are often evaluated by two methods. (1) Competitive ratio, which indicates how well a heuristic algorithm performs, is defined as the ratio of the number of requests satisfied using a heuristic algorithm and the number of requests satisfied using the exhaustive algorithm. (2) Success



a. increase of unknown area b. the area we consider

Figure 6. Unknown area proportion

ratio (SR) is defined as the ratio of the number of requests satisfied using a heuristic algorithm and the total number of requests generated.

The difference between the two methods is the feasibility of the requests being the denominator in theory. Both have shortcomings because the evaluation depends heavily on the generated constraints of the requests, e.g. the distribution of constraints. For a large-scale network, it is difficult to judge the feasibility of a request, so SR is used widely in most cases. Because different distributions are used in different papers, the absolute value of SR means nothing, and only the comparison with the same network data makes sense.

In order to evaluate the absolute performance of MEFPA and avoid the above problem, we propose the method of unknown-area proportion, which is independent of QoS requests. We take the unknown area in Fig. 3.b as the inefficiency of MEFPA and analyze the proportion of this area to the whole area. The unknown area is $M_{UNKNOWN}$ in theory, but for the limited number of discrete coefficients a, the unknown area will extend to $M'_{UNKNOWN}$ as shown in Fig. 6.a. The reason is that we cannot guarantee the nonexistence of least energy paths in triangle *A* and *B*. $M'_{UNKNOWN}$ is a limited space while M_{NOT} and $M_{FEASIBLE}$ are unlimited spaces. For a given (*s*,*t*) pair, we use Dijkstra's algorithm to construct the shortest path p_l with respect to w_l. The unknown-area proportion is defined as

$$\Pr = M'_{UNKNOWN} / (M_{NOT} + M_{FEASIBLE} + M'_{UNKNOWN}) \qquad (10)$$

in the subset $\{\mathrm{w}|\, w_l \le 2\times w_l(p_l)\}$ shown as the rectangle enclosed by the gray dashed line in Fig. 6.b.

Fig. 7 shows the average unknown-area proportion Pr with the 95% confidence interval for 10 random graphs. The X-coordinate is the number of the degrees to care each weight and the Y-coordinate is Pr. As shown in this figure, (1) the proportion of unknown area is small. (2) Pr decreases rapidly to a constant value when *b* increases. This shows that in practice, to ensure high performance, we only need few uniform coefficients a to find enough paths of different characteristics, e.g. when *k*=2, *B*=*b*=7. With larger *b*, most of

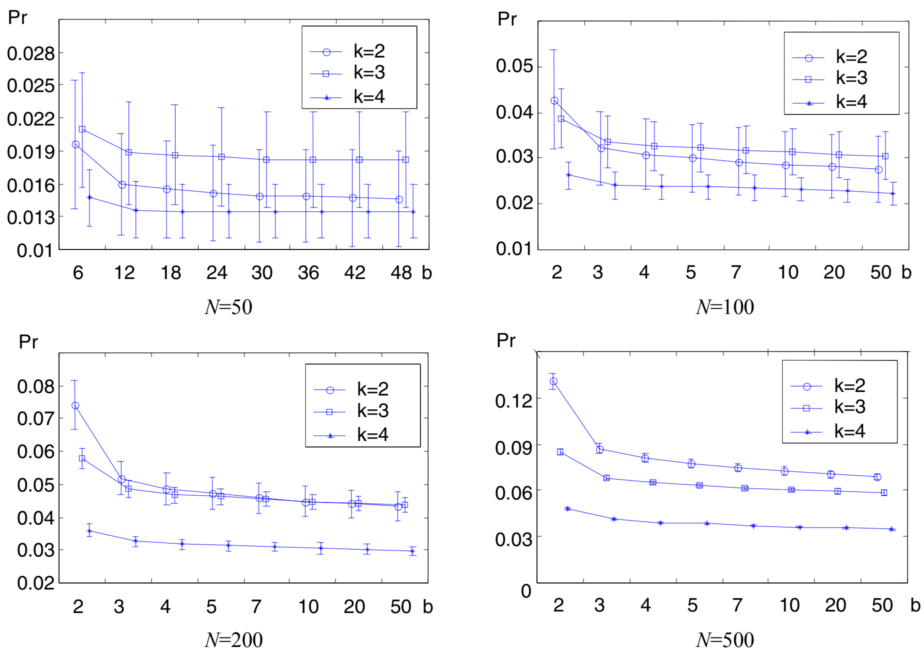


Figure 7. The absolute performance of LEPFA

the newly found paths are reduplications, which cannot decrease Pr furthermore. This is consistent with the conclusion in [22]. (3) In large-scale networks, Pr decreases when the number of weights k increases, and the larger the k, the smaller the change that Pr decreases with b increasing. This shows that in multiple dimensions, when b is small, we use a number ($B = C_{b+k-2}^{k-1}$) of coefficients. Therefore, MEFPA performs well with a small b in multiple dimensions. (4) With the increase of node number N, b plays a more significant role to the performance. The reason is that in larger networks, more paths are available between a particular (s,t) pair and the possibility to optimize a particular weight increases.

We have demonstrated that the unknown-area proportion Pr is small, and now we demonstrate that the feasibility for a request $c \in M_{UNKNOWN}$ is also small. First, we generate some constraints within the unknown area in Fig. 6 randomly. Then, we use H_MCOP [20] to seek feasible paths for each request. Fig. 8 shows the SR with k=2, i.e. 2-constrained routing. When b=7, the success ratio is less than 5% so that most requests may be inherently infeasible.

Having established that (1) the unknown-area proportion is small, and (2) a request within the unknown area has a low feasibility, MEFPA can refuse the requests within the unknown area with a small probability of misjudgment (refuse feasible requests). As a result, MEFPA achieves a high absolute performance. Since the probability of misjudgment is small, the misjudgment is no longer the major factor that decreases the performance. Instead, the inherent staleness of network-state information based on which QoSR operates may be the major factor in practice [32].

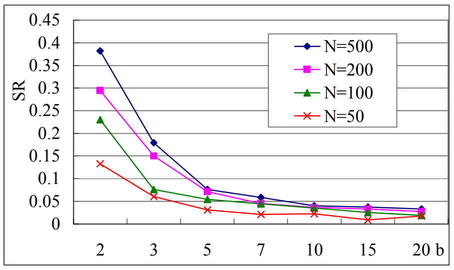


Figure 8. The unfeasibility of requests in the unknown area

B. *Performance comparison based on random constraints*

From the related work in section II, it is seen that the current precomputation algorithms for QoSR are not efficient enough. Some of them tend to have the prohibitive computation complexity or low performance, and some are based on distance vectors, so they are not fit for large-scale networks. In order to show the performance of MEFPA, we compare MEFPA with H_MCOP [20], which is also based on Dijkstra's algorithm.

We use the method of generating constraints for requests in [20] to compare the two algorithms first. For a given (s,t) pair, we use Dijkstra's algorithm to calculate the shortest path p_l with the keyword w_l, respectively. We then generate the constraints randomly for each (s,t) pair: $c_{l+1} \sim$ uniform $[0.8\, w_{l+1}(p_l),\ 1.2\, w_{l+1}(p_l)]$. The shadowed area in Fig. 9 shows the constraints generated in two dimensions.

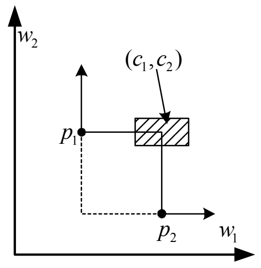


Figure 9. Random constraints

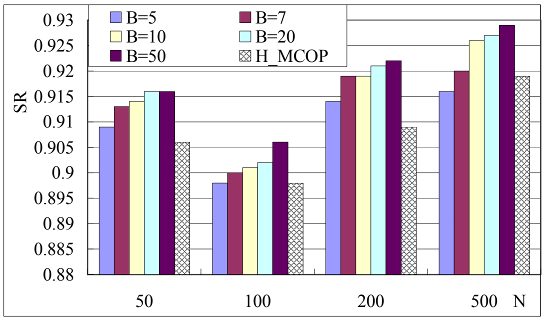


Figure 10. Performance eveluation with random constraints (two constraints)

Fig. 10 shows the routing success ratio in two dimensions, i.e. for 2-constrained routing. When *b*=7, the SR of MEFPA is higher than that of H_MCOP. Table I shows the cases in multiple dimensions, i.e. k-constrained routing. With the increase of weight number *k*, because more requests generated by this random method are inherently infeasible, the performance of both algorithms decreases rapidly. However, from the comparison, MEFPA performs well when *b* is small with a relatively large *k* (e.g., when *k*=5, we choose *b*=3). This further confirms the conclusion of the absolute performance evaluation.

C. *Performance comparison based on simulated constraints*

Because the performance depends on the distribution or scale that the generated QoS constraints obey, experimental results in different papers cannot be compared directly with others. However, the Internet does not have typical topologies or traffic models [34], and we are even more short of knowledge about the constraints of the upcoming QoS applications. Therefore, it is difficult to give a reasonable model and a distribution of the constraints. Nevertheless, we know that most QoS applications care different weights to different degrees. For example, file transfer applications may care loss rate to a much higher degree than delay, and multimedia applications may take delay as the most important parameter. The distribution of constraints in Fig. 9 does not consider this aspect, and it can generate constraints only round the middle between the w_1 axis and w_2 axis. For example, it cannot generate a constraint pair, in which w_1 is as important as three times w_2.

Based on the normalized weights in the whole graph, we use the method of weight ratio simulation to generate the

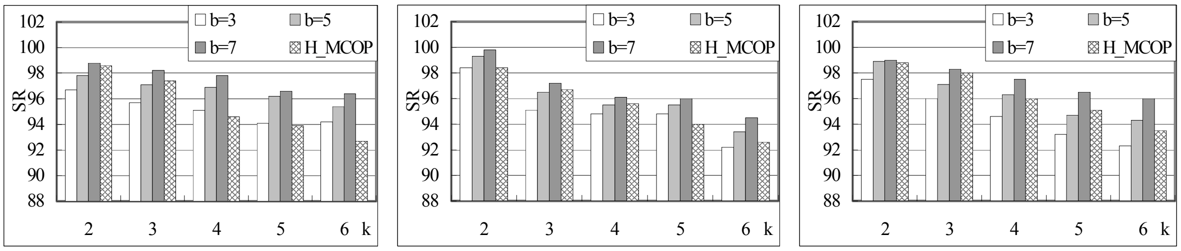
TABLE I
PERFORMANCE EVALUATION WITH RANDOM CONSTRAINTS (MULTIPLE CONSTRAINTS)

SR(%)		*k*=2	*k*=3	*k*=4	*k*=5
N=50	*b*=3	91.8	79.7	64.4	55.7
	b=7	92.8	80.5	64.7	55.8
	H_MCOP	92.5	80.0	64.2	55.6
N=100	*b*=3	91.0	73.0	61.8	51.6
	b=7	92.2	73.7	62.7	52.3
	H_MCOP	91.9	73.5	62.0	50.8
N=200	*b*=3	88.2	73.4	59.4	47.8
	b=7	90.0	75.0	60.9	49.9
	H_MCOP	89.9	73.8	59.9	48.2

constraints for a given pair (s,t). First, we assume that each QoS application has a coefficient vector a. The normalized $a_l/(a_1+\cdots+a_k)$ presents the degree, to which this application cares the weight w_l. Based on this assumption, we use the LEF g_a (Definition 2) to construct the constraints for a request. For a given (s,t), we use Dijkstra's algorithm to calculate the least energy path $p_a(s,t)$ and take the weights vector $w(p_a(s,t))$ as the QoS constraint of (s,t), i.e. $c(s,t)=w(p_a(s,t))$. Because the request with such simulated constraints must be feasible, SR can be extended to the absolute performance evaluation of algorithms. For each pair (s,t), we take $a_l \sim uniform(0,1)$ in practice. Fig. 11 shows SR of these QoS requests we simulate. The experiment shows that MEFPA overmatches H_MCOP, and MEFPA has a good scalability because it is insensitive not only to the network scale, but also to the constraint number *k*.

D. *Running time comparison*

Our extensive simulations show that MEFPA has a higher SR than H_MCOP, although H_MCOP has to calculate the feasible path for each individual request respectively. On the aspect of computation complexity, H_MCOP algorithm is $O(km\log kn + n\log n \ +(k^2+1)m)$ with k-shortest path algorithm [33], while MEFPA is $O(B(m+n\log n+n))$. For given $B=7$ when *k*=2, MEFPA is better than H_MCOP, not only on computation complexity but also on the running time in practical experiments. For example, the running time of MEFPA is 51.8 millisecond in a 500-node graph, while that of H_MCOP is 15.3 millisecond. Considering the difference between precomputation and on-line computation furthermore, MEFPA has much less complexity. For example, if on a given



N=50 *N*=100 *N*=200
Figure 11. Comparison with simulated constraints

source node, there are (N-1) requests to the other (N-1) nodes in the network, the running time of MEFPA keeps fixed while that of H_MCOP increases (N-1) times.

VI. Conclusion

For the NP-complete problem of multi-constrained QoS routing, we propose a novel precomputing algorithm, MEFPA, based on the theoretical analysis of linear energy functions. With this algorithm, a router constructs a number (B) of uniform coefficients to construct B linear energy functions. It then calculates B least energy trees to precompute the QoS routing table with the computation complexity $O(B(m + n\log n + n))$. Compared with the current Internet routing scheme, the router only needs to replace the current cost by the energy value in SPT computation. Thus, the precomputation complexity of MEFPA is only B times that of the current algorithm with a single cost.

When QoS requests arrive, the router can look up a feasible path in the QoS routing table. The size of the QoS routing table is less than or equal to B times that of the current routing table with a single cost, so that the present techniques of routing table look-up is competent. Because of the small complexity and the precomputation scheme, MEFPA has a good scalability in the number of QoS constraints, the network scale and also the high-speed arrival of packets in next-generation networks. It is also consistent with the routing architecture of the current Internet.

From the performance evaluation of routing success ratio, we find that the performances of heuristics are different with different distributions or scales that the generated QoS constraints obey. In order to evaluate the absolute performance, we propose a novel approach, the unknown-area proportion, which shows that our MEFPA achieves high absolute performance. Additionally, we generate the constraints of QoS requests by both of the random method and weight-ratio simulation, respectively. Extensive simulations also show that our MEFPA performs competitively great.

As a conclusion, following the current precomputation routing architecture, we believe that the MEFPA is a promising QoSR algorithm for next-generation high-speed networks because of its high scalability, performance and simplicity. Furthermore, MEFPA can be easily extended to resolve multi-constrained optimal cost problem. For example, taking the cost as the $(k+1)^{th}$ weight, this problem can be transferred to (k+1)-constrained problem and MEFPA can generate the QoS routing table with (k+1) weights. When QoS requests arrive, we can look up an optimal feasible path, which satisfies the k constraints and has a least cost in the routing table.

Acknowledgments

The authors would like to thank Ruibing Hao for his careful revision.

References

[1] X. Xiao and L. M. Ni, Internet QoS: a big picture, IEEE Network, vol. 13, no. 2, pp. 8–18, Mar.-Apr. 1999.

[2] Y. Cui, J. P. Wu, K. Xu, et al. Research on internetwork QoS routing algorithms: a survey. Chinese Journal of Software, vol. 13, no. 11, 2065-2076, 2002.

[3] J.C. Oliveira, C. Scoglio, I.F.Akyildiz, et al. A new preemption policy for diffServ-aware traffic engineering to minimize rerouting, IEEE INFOCOM'02, 2002.

[4] J. Wang and K. Nahrstedt. Hop-by-hop routing algorithms for premium-class traffic in diffServ networks. IEEE INFOCOM'02, 2002.

[5] IETF Integrated Services (diffserv) working group, http://www.ietf.org/html.charters/diffserv-charter.html

[6] M. S. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, New York, 1979.

[7] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. IEEE Journal on Selected Areas in Communications, vol. 14, no. 7, pp. 1228–1234, Sep. 1996.

[8] A. Orda and A. Sprintson, QoS routing: the precomputation perspective. IEEE INFOCOM'00., vol. 1, pp. 128–136, 2000.

[9] J. Bennett and H. Zhang, Hierarchical packet fair queueing algorithms. ACM SIGCOMM'96, Aug. 1996.

[10] L. Zhang, Virtual Clock: A new traffic control algorithm for packet switching networks. ACM SIGCOMM'90, pp. 19-29, Sep. 1990.

[11] Q. Ma and P. Steenkiste, Quality-of-service routing with performance guarantees. 4th International IFIP Workshop on Quality of Service, May 1997.

[12] C. Pornavalai, G. Chakraborty, and N. Shiratori, QoS based routing algorithm in integrated services packet networks, IEEE ICNP'97, Atlanta, GA, 1997.

[13] A. Orda, Routing with end-to-end QoS guarantees in broadband networks, IEEE/ACM Transactions on Networking, vol. 7, no. 3, pp. 365–374, 1999.

[14] H. F. Salama, D. S. Reeves, and Y. Viniotis, A distributed algorithm for delay-constrained unicast routing. IEEE INFOCOM'97, Japan, April 1997.

[15] J. M. Jaffe, Algorithms for finding paths with multiple constraints, IEEE Networks, vol. 14, pp. 95–116, 1984.

[16] D. Raz, Y. Shavitt, Danny Raz and Yuval Shavitt, Optimal partition of QoS requirements with discrete cost functions, IEEE INFOCOM'00, vol.2, pp. 613 –622, 2000.

[17] H. Lorenz, Ariel Orda, Danny Raz, and Yuval Shavitt, 'Efficient QoS partition and routing of unicast and multicast', IWQoS 2000, June 2000.

[18] H. de Neve and P. Van Mieghem. A multiple quality of service routing algorithm for PNNI, IEEE ATM Workshop Proceedings, 1998

[19] P. Van Mieghem, H. de Neve and F. Kuipers. Hop-by-hop quality of service routing, Computer Netwroks, vol. 37, pp. 407-423, 2001.

[20] T. Korkmaz, M. Krunz, Multi-Constrained Optimal Path Selection. IEEE INFOCOM'01, vol.2, pp. 834 -843, 2001.

[21] S. Chen and K. Nahrstedt, An overview of quality-of-service routing for next-generation high-speed networks: Problems and solutions, IEEE Network, vol. 12, no. 6, pp. 64–79, Nov. 1998.

[22] X. Yuan, X. Liu, Heuristic Algorithms for Multi-Constrained Quality of Service Routing, IEEE INFOCOM'01, vol.2, pp. 844 –853, 2001.

[23] A.Shaikh, J. Rexford and K.Shin. Efficient precomputation of quality-of-service routes. In Proceedings of Workshop on Network and Operating Systems Support for Audio and Video (NOSSDAV'98). Cambride, England, Jul. 1998.

[24] E. Dijkstra, A note on two problems in connexion with graphs. Numerische Mathematik, vol. 1, pp.269-271, 1959.

[25] E. Kreyszig. Introductory functional analysis with applications. New York : Wiley, 1978.

[26] R. A. Brualdi, Introductory Combinatorics. Third edition, Upper Saddle River, N.J.: Prentice Hall, 1999.

[27] J. L. Sobrinho, Algebra and Algorithms for QoS path computation and hop-by-hop routing in the Internet, IEEE INFOCOM'01, vol.2, pp. 727 –735, 2001.

[28] P. Gupta and N. Mckeown. Algorithms for packet classification. IEEE Network, pp.24-32, Mar.-Apr. 2001.

[29] M. Kobayashi, T. Murase and A. Kuriyama. A longest prefix match search engine for multi-gigabit IP, IEEE ICC'00, 2000.

[30] E. W. Zegura, K. L. Calvert, S. Bhattacharjee, How to model an internetwork. IEEE INFOCOM'96, vol.2, pp. 594 -602, 1996.

[31] E. W. Zegura, K. L. Calvert, M. J.Donahoo, A quantitative comparison of graph-based models for Internet topology. IEEE/ACM Transactions on Networking, vol. 5, no. 6, pp. 770 –783, Dec. 1997.

[32] A. Shaikh, J. Rexford, K. G. Shin, Evaluating the impact of stale link state on quality-of-service routing. IEEE/ACM Transactions on Networking, vol. 9, no. 2, pp. 162 –176, Apr. 2001

[33] E. I. Chong, S. R. Sanjeev Rao Maddila, and S. T. Morley, On finding single-source single-destination shortest paths, in the Seventh International Conference on Computing and Information (ICCI'95), pp. 40–47, Jul. 1995.

[34] S. Floyd, V. Paxson, Difficulties in simulating the Internet. IEEE/ACM Transactions on Networking, vol. 9, no. 4, pp. 392–403, Aug. 2001.