Multiple additively constrained path selection

G. Cheng and N. Ansari

Abstract: Finding a feasible path subject to multiple constraints in a network is an NP-complete problem and has been extensively studied. However, current algorithms suffer either high computational complexity or low success ratio in finding feasible paths. The authors propose a novel extended Bellman–Ford algorithm (EB), from which they present a high-performance algorithm with low computational complexity in finding feasible paths with multiple additive constraints. Through analysis and simulations, it is shown that the algorithm outperforms its contenders in the success rate of finding a feasible path as well as in terms of scalability; the proposed algorithm can achieve almost 100% success ratio as long as a feasible path exists. Furthermore, the worst case computational complexity is only twice that of the Bellman–Ford algorithm.

1 Introduction

One of the challenging issues for high-speed packet switching networks to facilitate various applications is to select feasible paths that satisfy different quality-of-service (QoS) requirements. This problem is known as QoS routing. In general, two issues are related to QoS routing: state distribution and routing strategy [1]. State distribution addresses the issue of exchanging the state information throughout the network [2]. Routing strategy is used to find a feasible path that meets the QoS requirements. In this paper, we focus on the latter task and assume that accurate network state information is available to each node. A number of research works have also addressed inaccurate information [3–7], which is, however, beyond the scope of this paper.

QoS constraints can be categorised into three types: concave, additive and multiplicative. Since concave parameters set the upper limits of all the links along a path, such as bandwidth, we can simply prune all the links and nodes that do not satisfy the QoS constraints. We can also convert multiplicative parameters into additive parameters by using the logarithm function. For instance, we can take $-\log(1-\alpha)$ as the replacement for loss rate α. Thus, we focus only on additive constraints in this paper. It has been proved that multiple additively constrained QoS routing is NP-complete [8]. Hence, tackling this problem requires heuristics. In [9], a heuristic algorithm was proposed based on a linear cost function for two additive constraints; this is a MCP (multiple constrained path selection) [1] problem with two additive constraints. A binary search strategy for finding the appropriate value of k in the linear cost function $w_1(p)+kw_2(p)$ or $kw_1(p)+w_2(p)$, where $w_i(p)$ $(i=1,2)$ are two respective weights of the path p, was proposed, and a hierarchical Dijkstra algorithm was introduced to find the path. It was shown that the worst-case complexity of the algorithm is $O(\log B(m+n\log n))$, where B is the upper bound of the parameter k, m is the number of links and n is the number of nodes. The authors in [10] simplified the multiple constrained QoS routing problem into the shortest path selection problem, in which the weighted fair queuing (WFQ) service discipline is assumed. Hence, this routing algorithm cannot be applied to networks where other service disciplines are employed. Similar to [9], Lagrange relaxation based aggregated cost (LARAC) was proposed in [11] for the delay constrained least-cost path problem (DCLC). This algorithm is based on a linear cost function $c_\lambda = c+\lambda d$, where c denotes the cost, d the delay and λ an adjustable parameter. It differs from [9] on how λ is defined; λ is computed by Lagrange relaxation instead of the binary search. It was shown that the computational complexity of this algorithm is $O(m^2\log^4 m)$. However, in [12], for the same problem (DCLC), a nonlinear cost function was proposed. Many researchers have posed the QoS routing problem as the k-shortest path problem, but the computational complexity is generally very high [13, 14]. To solve the delay-cost-constrained routing problem, Chen and Nahrstedt proposed an algorithm [15], which maps each constraint from a positive real number to a positive integer. By doing so, the mapping offers a 'coarser resolution' of the original problem, and the positive integer is used as an index in the algorithm. The computational complexity is reduced to pseudo-polynomial time, and the performance of the algorithm can be improved by adjusting a parameter, but with a larger overhead.

As reviewed above, existing routing algorithms have one or more of the following drawbacks:

(i) High computational complexity

(ii) Lack of scalability

(iii) Poor success ratio

(iv) Loss of generality; focus only on one special case of multiconstrained routing, i.e. two constraints.

Here, we propose an algorithm to solve the QoS routing problem with multiple additive constraints, by which we are able to overcome the above drawbacks.

© IEE, 2002

IEE Proceedings online no. 20020672

doi:10.1049/ip-com:20020672

Paper first received 23rd January and in revised form 24th July 2002

The authors are with the Advanced Networking Lab, Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA

2 Problem formulation

Most works reported in the literature approach the QoS routing problem as a special case of the multiconstrained QoS routing problem, i.e. mostly considering two constraints only. We will propose an algorithm in which there is no limitation on the number of QoS constraints. Since concave constraints can be easily addressed by pruning, and multiplicative constraints can be generally converted into additive constraints, without loss of generality, we only consider additive constraints and formulate the problem as follows:

Definition 1: *Multiple additively constrained path selection* (MACP): Assume a network is modelled as a directed graph $G(N,E)$, where N is the set of all nodes and E is the set of all links. Each link connected from node u to v, denoted by $e_{u,v} = (u, v) \in E$, is associated with M additive parameters: $w_i(u, v) \geq 0$, $i = 1, 2, \ldots, M$. Given a set of constraints $c_i > 0$, $i = 1, 2, \ldots, M$ and a pair of nodes s and t, the objective of MACP is to find a path p from s to t subject to $W_i(p) = \sum_{e_{u,v} \in p} w_i(u, v) < c_i$, $i = 1, 2, \ldots, M$.

Definition 2: Any path selected by MACP is a feasible path; that is, any path p from s to t that meets the requirement $W_i(p) = \sum_{e_{u,v} \in p} w_i(u, v) < c_i$, $i = 1, 2, \ldots, M$ is a feasible path.

Notations:

- $f(\boldsymbol{x})$ cost function, where $\boldsymbol{x} = (x_1, x_2, \ldots, x_M)$
- $\boldsymbol{c}$ vector representation of the QoS constraints $(c_1, c_2, \ldots, c_M)$
- $\boldsymbol{W}(p)$ weight vector of path p, i.e. $\boldsymbol{W}(p) = (W_1(p), W_2(p), \ldots, W_M(p))$, where $W_i(p) = \sum_{e_{u,v} \in p} w_i(u, v)$
- $C(p)$ cost of path p, $C(p) = \sum_{e_{u,v} \in p} f(w_1(e_{u,v}), w_2(e_{u,v}), \ldots, w_M(e_{u,v}))$, where $f(\cdot)$ is the cost function.

Note that $C(p)$ is generally not $f(\boldsymbol{W}(p))$ because

$$f(\boldsymbol{W}(p)) = f\left(\sum_{e_{u,v} \in p} w_1(u, v), \sum_{e_{u,v} \in p} w_2(u, v), \ldots, \sum_{e_{u,v} \in p} w_M(u, v)\right) \tag{1}$$

However, if $f(\boldsymbol{x})$ is linear, i.e. $f(\boldsymbol{x}) = \sum_{i=1}^{M} \beta_i x_i$

$$\begin{aligned} f(\boldsymbol{W}(p)) &= f\left(\sum_{e_{u,v} \in p} w_1(u, v), \sum_{e_{u,v} \in p} w_2(u, v), \ldots, \sum_{e_{u,v} \in p} w_M(u, v)\right) \\ &= \sum_{i=1}^{M} \beta_i \left(\sum_{e_{u,v} \in p} w_i(u, v)\right) \\ &= \sum_{i=1}^{M} \left(\sum_{e_{u,v} \in p} \beta_i w_i(u, v))\right) \\ &= \sum_{e_{u,v} \in p} \left(\sum_{i=1}^{M} \beta_i w_i(u, v)\right) \\ &= \sum_{e_{u,v} \in p} f(w_1(e_{u,v}), w_2(e_{u,v}), \ldots, w_M(e_{u,v})) = C(p) \end{aligned} \tag{2}$$

3 Proposed QoS routing algorithm

Assume the link weights are randomly distributed, and define $P_r\{W_1(p) \leq c_1, W_2(p) \leq c_2, \ldots, W_M(p) \leq c_M | C(p) = k, H(p) = n\}$ as the probability that a path p is a feasible path with $C(p) = k$, and its hop count $H(p) = n$. Thus, the probability of the least-cost path of being a feasible path may not be the largest in all possible paths. Note that the probability that a path is a feasible path is not only related to the cost of the path but also the hop count.

For simplicity, the following linear normalised cost function is adopted:

$$f(\boldsymbol{x}) = \sum_{i=1}^{M} \frac{x_i}{c_i} \tag{3}$$

With the above setting, the following theorem can be readily established.

Theorem 1: A feasible path does not exist if the least-cost path has a cost greater than or equal to $f(\boldsymbol{c})$.

Proof: By contradiction. Assume path $\hat{p}$ satisfies the constraint $\boldsymbol{c}$ and the least cost among all paths is larger than or equal to $f(\boldsymbol{c})$; that is

$$C(p) \geq f(\boldsymbol{c}), \forall p \Rightarrow C(\hat{p}) \geq f(\boldsymbol{c}) \tag{4}$$

Also, since $f(\boldsymbol{x})$ is linear, from (2)

$$f(\boldsymbol{W}(\hat{p})) = C(\hat{p}) \tag{5}$$

Thus,

$$f(\boldsymbol{W}(\hat{p})) \geq f(\boldsymbol{c}) \tag{6}$$

However, since $\partial f(\boldsymbol{x})/\partial x_i \geq 0$ and path $\hat{p}$ satisfies the constraint $\boldsymbol{c}$

$$W_i(\hat{p}) < c_i, \forall i \in \{1, 2, \ldots, M\} \Rightarrow f(\boldsymbol{W}(\hat{p})) < f(\boldsymbol{c}) \tag{7}$$

which contradicts (6*a*), and thus theorem 1 is proved. □

Lemma 1: If path p is a feasible path

$$f(\boldsymbol{W}(p)) < f(\boldsymbol{c}) \tag{8}$$

Proof: The proof is similar to that of theorem 1. □

Lemma 2: If $f(\boldsymbol{W}(p)) < f(\boldsymbol{c})$, path p can be an infeasible path.

Proof: The proof is similar to that of theorem 1. □

Lemmas 1 and 2 motivate us to propose the following operation in executing our proposed algorithm: whenever we have a path p that satisfies $C(p) \leq f(\boldsymbol{c})$ (it may not be the least-cost path), we check if it is a feasible path. This operation is included in lines 1–8 of the pseudocode of the relaxation procedure of our proposed extended Bellman–Ford (EB) algorithm shown in Fig. 1. Here, $p^n(s, i)$ represents an n-hop path from source s to i computed by the EB algorithm (we will prove later that $p^n(s, i)$ is a least-cost path among all the n-hop paths from source s to i). D_i^n is the cost of this path, $\pi^n(i)$ represents the predecessor node of i along the path and $c(i, j)$ is the cost of link (i, j) (if there is no link between i and j, $c(i, j) = \infty$).

```
Relax(j, i)
1    if i is the destination node t
2        if D_j^n + c(j, t) ≤ f(c)  /*check if this path is feasible path*/
4            if p^n(s, j) + e(j, t) is feasible, then
5                return SUCCESS /*feasible path is found*/
6            end if
7        end if
8    end if
9    if D_i^{n+1} ≥ D_j^n + c(i, j) then
10       D_i^{n+1} = D_j^n + c(i, j)
11       p^{n+1}(s, i) = p^n(s, j) + e(i, j)
12       π^{n+1}(i) = j
13   end if
```

Fig. 1 *Relaxation procedure of EB*

Note that, from Fig 1, the checking procedure (lines 1–8) is required only when node i is the destination, and hence its computational complexity is negligible when compared to the whole relaxation procedure. Since there is no difference between the relaxation procedure of the EB algorithm and that of the standard Bellman–Ford algorithm except the checking procedure, the computational complexity and memory cost of the EB algorithm are compatible with those of the standard Bellman–Ford algorithm. However, it should be noted that there are two distinct differences between our algorithm and the Bellman–Ford algorithm, that are not shown in Fig. 1:

(i) D_i^{n+1} is defined as $\min_j[c(i,j)+D_j^n]$, $n=1,2,\ldots$, in our algorithm, while D_i^{n+1} is defined as $\min\{\min_j[c(i,j)+D_j^n], D_i^n\}$ in the Bellman–Ford algorithm. Here, $D_i^1=c(s,i)$. D_i^{n+1} can be obtained iteratively through the relaxation procedure by simply setting the initial value of D_i^{n+1} as infinity, instead of D_i^n in the Bellman–Ford algorithm.

(ii) In our algorithm, $D_s^n=\infty$, $n=1,2,\ldots$, while $D_s^n=0$, $n=1,2,\ldots$, in the Bellman–Ford algorithm.

By the above two modifications, we can compute $p^n(s,i)$ which is a least-cost path among all n-hop paths from s to i. Note that the computational complexity of the EB algorithm is just compatible with that of the Bellman–Ford algorithm. However, as compared to the Bellman–Ford algorithm which only computes the least-cost path from node s to any other node i subject to the constraint that the path contains at most n hops, our algorithm finds more possible paths and thus increases the success ratio in finding a feasible path.

Proposition: For any node $i\in\{1,2,\ldots,N\}$, if at least one n-hop path from s to i physically exists, the path $p^n(s,i)$ generated by the EB algorithm must be a least-cost path among all n-hop paths from s to i.

Proof: When $n=1$, from the definition of the initial value of D_i^1 ($i\neq s$), $p^1(s,i)$ is the one hop least-cost path from s to i, $i\in\{1,2,\ldots,N\}$.

We assume that the proposition is correct for $n=k$. We want to prove by deduction that it is true for $n=k+1$.

Assume when $n=k+1$, if $\exists j\neq s$, $p^{k+1}(s,j)$ is not a least-cost path in all $(k+1)$-hop paths from s to j (D_j^{k+1} is larger than the cost of the $(k+1)$-hop least-cost path from s to j). Further assume path $\hat{p}^{k+1}(s,j)$ is a least-cost path in all $(k+1)$-hop paths from s to j, then the predecessor node of node j in $\hat{p}^{k+1}(s,j)$ is d, the path from s to d in $\hat{p}^{k+1}(s,j)$ is $\hat{p}^k(s,d)$ (note that $\hat{p}^k(s,d)$ may not be a least-cost k-hop path from s to d, and by the earlier assumption, since a k-hop path exists, $p^k(s,d)$ generated by the EB algorithm is a least-cost path from s to d), the cost of $\hat{p}^{k+1}(s,j)$ is c and the cost of $\hat{p}^k(s,d)$ is c'. Thus

$$c<D_i^{k+1} \tag{9}$$

if $\hat{p}^k(s,d)=p^k(s,d)$ or $c'=D_d^k$, $\hat{p}^{k+1}(s,j)$ is obtained by concatenating $\hat{p}^k(s,d)$ with link $e(j,d)$. Thus

$$c=D_d^k+c(j,d)\geq\min_l[c(j,l)+D_l^k]=D_j^{k+1} \tag{10}$$

which contradicts (9). Hence,

$$\hat{p}^k(s,d)\neq p^k(s,d) \tag{11}$$

and

$$c'>D_d^k \tag{12}$$

So, the cost of $\hat{p}^{k+1}(s,j)$ is

$$c=c'+c(j,d)\geq D_d^k+c(j,d)\geq\min_l[c(j,l)+D_l^k]=D_i^{k+1} \tag{13}$$

which contradicts (9). So, when $n=k+1$, $p^{k+1}(s,i)$, $i\in\{1,2,\ldots,N\}$ is a least-cost path among all $(k+1)$-hop paths from s to i.

Thus, for any node $i\in\{1,2,\ldots,N\}$, if at least one n-hop path from s to i physically exists, the path $p^n(s,i)$ generated by the EB algorithm must be a least cost path among all n-hop paths from s to i. □

As shown in Fig. 2, the EB algorithm first computes $p^n(s,j)$, where j is a neighbour node of destination t, adds the link (j,t) to the path $p^n(s,j)$ and then checks if the resulting new path is a feasible path.

Note that the EB algorithm is asymmetric with respect to the source and destination. Intuitively, if no feasible path is found in the first execution of the EB algorithm, we may search in the reverse direction if the cost of the least-cost path is less than $f(\boldsymbol{c})$ (otherwise, by theorem 1, we already know that there does not exist a feasible path), that is, from the destination to the source node, with a well designed new cost function $f'(\boldsymbol{x})$. We thus propose to incorporate this bidirectional search (both forward and reverse) in our proposed QoS routing algorithm, and call it the BEB (bidirectional extended Bellman–Ford) algorithm. We shall next address the issue of choosing the new cost function for the backward EB.

It is desirable to either have found a feasible path or ascertain that a feasible path does not exist (invoking theorem 1) by the second execution of the EB algorithm (backward EB). However, if the same cost function is deployed, the least-cost path of the first search (forward EB) that is not a feasible path is still a least-cost path in the second search (backward EB). Since we already know that the least-cost path p of the first search (forward EB) is not a feasible path, the cost function should be adjusted for the second search such that

(i) If a feasible path does exist, p should not be the least-cost path computed by the second search

(ii) If p is the least-cost path of the second search, $f'(W(p))\geq f'(\boldsymbol{c})$ so that theorem 1 can be invoked.

Since p is not a feasible path, $\exists w_i(p)$ for some i such that $w_i(p)>c_i$. (It is also possible that $w_i(p)=c_i$, $i=1,2,\ldots,M$, for p to be an infeasible path. In this case, $f(\boldsymbol{c})=C(p)$, and by theorem 1, a feasible path does not exist and it is unnecessary to invoke the second search). Thus, we may simply adjust this component in the cost

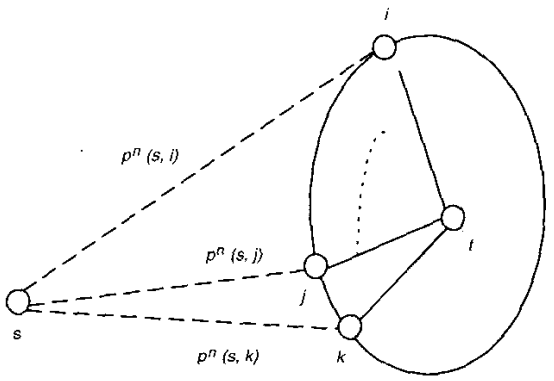

Fig. 2 *EB algorithm*

```
Algorithm BEB(G,s,t,c)
1 if EB (G,s,t, c, LeastCost) = SUCCESS
2    return SUCCESS
3 else
4    if LeastCost ≥ f (c)
5       return No Feasible Path Exists  /*no feasible path existing*/
6    else
7       Compute New Cost Function f (x)
8       if EB (G,t,s,c, LeastCost)= SUCCESS
9          return SUCCESS
10      else
11         if LeastCost ≥ f (c)
12      return No Feasible Path Exists  /*no feasible path existing*/
13         end if
14      end if
15   end if
16 else
17   return FAIL   /*fail to find a feasible path*/
18 end if
```

Fig. 3 *Pseudocode of the BEB algorithm*

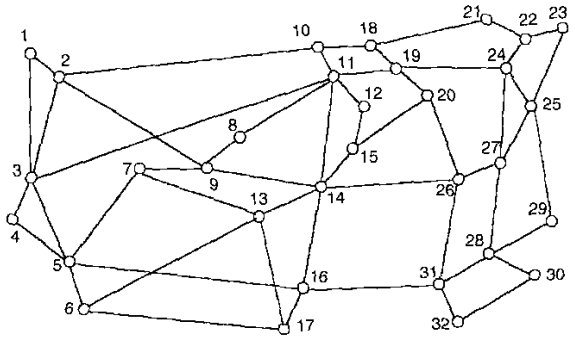

Fig. 4 *32-node network topology*

function such that $f'(\boldsymbol{W}(p)) = f'(\boldsymbol{c})$ as follows:

$$f'(x_1, x_2, \ldots, x_M) = \sum_{\substack{j=1 \\ j \neq i}}^{M} \frac{x_j}{c_j} + \left(\frac{f(\boldsymbol{c}) - C(p)}{w_i(p) - c_i} + \frac{1}{c_i} \right) x_i$$

$$= \sum_{j=1}^{M} \frac{x_j}{c_j} + \left(\frac{f(\boldsymbol{c}) - C(p)}{w_i(p) - c_i} \right) x_i \tag{14}$$

since

$$f(\boldsymbol{c}) > C(p), \frac{f(\boldsymbol{c}) - C(p)}{w_i(p) - c_i} > 0 \text{ and } \frac{\partial f'(\boldsymbol{x})}{\partial x_j} \geq 0,$$

$$j = 1, 2, \ldots, M$$

(theorem 1 is valid only when $\partial f'(\boldsymbol{x})/\partial x_i \geq 0$, $i = 1, 2, \ldots, M$). It should be noted that this procedure is derived under the condition that the cost function is linear because, only in this case, $f(\boldsymbol{W}(p)) = C(p)$ holds for any path p.

Fig. 3 shows the pseudocode of our proposed QoS routing algorithm.

4 Simulations

Our simulations are divided into two parts. The first part involves evaluating the performance of our proposal (BEB) and comparing that with the Korkmaz *et al.* algorithm [9]. The network topology presented in [9, 15] is adopted for comparison purposes. The second part is a demonstration of the scalability of our proposed algorithm, BEB. Two larger networks with 50 and 100 nodes are generated using Doar's model [16]. In all simulations, the link weights are independent and uniformly distributed from 0 to 1, and all data are obtained by running 1 000 000 requests.

To evaluate the performance, we do not adopt the success ratio defined in [9, 15] that is defined as

$$SR = \frac{\textit{total no. of successful requests}}{\textit{total no. of requests}} \tag{15}$$

Since there may not exist a feasible path if the given constraints are tight, in which case, this success ratio (15) cannot truly reflect the algorithm's capability in finding a feasible path. Therefore, we propose the following more appropriate success ratio definition as our performance index:

$$SR = \frac{\textit{total no. of successful requests of the algorithm}}{\textit{total no. of successful requests of optimal algorithm}} \tag{16}$$

The algorithm that can always locate a feasible path as long as it exists is referred to as the optimal algorithm. Here, it is achieved simply by flooding which is rather exhaustive.

Simulation 1

In simulation 1, our proposed algorithm is compared with the Korkmaz *et al.* algorithm [9] ($B = 1000$). The network topology is shown in Fig. 4. Two QoS constraints are set to be equal and increase from 0.5 to 4.9 with an increment of 0.2.

From Fig. 5, it can be observed that the lower bound of our algorithm in this simulation is 99.9%, while that of Korkmaz *et al.* in this simulation is about 99.1%. However, note that the worst-case computational complexity of the BEB algorithm is only twice that of EB, i.e. twice that of the Bellman–Ford algorithm.

Simulation 2

Intuitively, the larger the network, the harder it is to find a feasible path. Thus, the performance of an algorithm may degrade quickly with the network size. However, by deploying the EB algorithm, our algorithm can essentially overcome this problem, i.e. our algorithm is scalable, as shown in Figs. 5–7.

It can be observed that the lower bound of the SR of our algorithm is about 99.5% in both networks (50 and 100 nodes), while the worst-case SRs of the Korkmaz *et al.* algorithm are 97.5% and 97%, respectively. Thus, as compared to the Korkmaz *et al.* algorithm, our algorithm is more scalable.

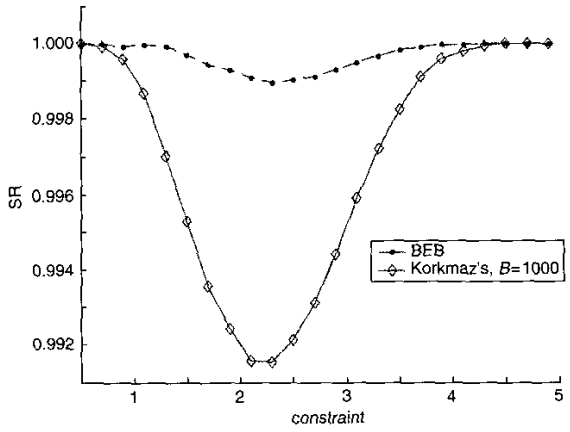

Fig. 5 *SR for the 32-node network*

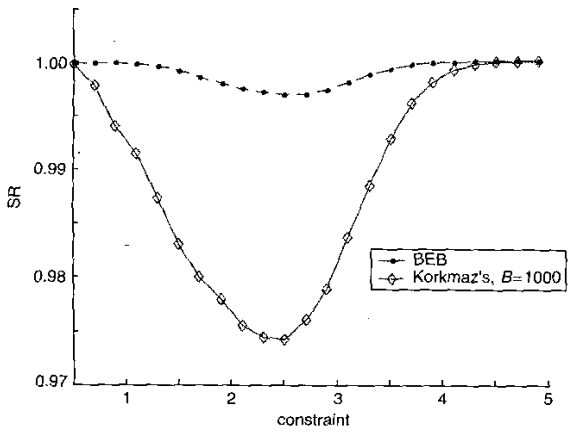

Fig. 6 *SR for the 50-node network*

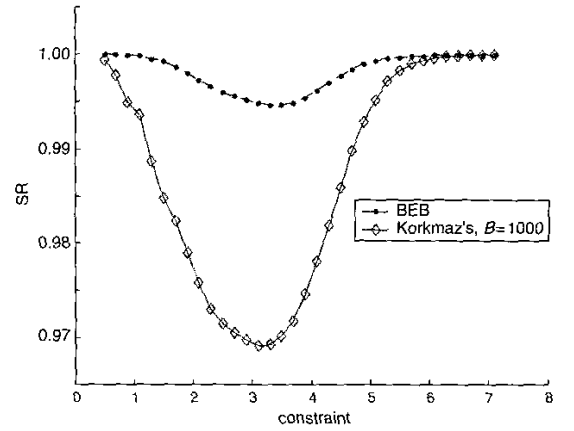

Fig. 7 *SR for the 100-node network*

5 Conclusions

We have proposed an efficient algorithm (BEB), which possesses the properties of low computational complexity and good scalability for multiple additively constrained routing. The worst-case computational complexity of our algorithm is $O(2NE)$, where N is the number of nodes and E is the number of links. Simulations show that our algorithm outperforms its contender. With a slight modification, our algorithm can also be employed for many other problems, such as the DCLC problem.

6 Acknowledgments

This work has been supported in part by OpenCon Systems Inc., the New Jersey Commission on Science and Technology via the NJ Center for Wireless Telecommunications, and the New Jersey Commission on Higher Education via the NJ I-TOWER project.

7 References

1 CHEN, S., and NAHSTED, K.: 'An overview of quality of service routing for next-generation high-speed network: problems and solutions', *IEEE Netw.*, 1998, **12**, (6), pp. 64–79
2 SHAIKH, A., REXFORD, J., and SHIN, K.G.: 'Evaluating the impact of stale link state on quality-of-service routing', *IEEE/ACM Trans. Netw.*, 2001, **9**, (2), pp. 162–176
3 GUERIN, R., and ORDA, A.: 'QoS based routing in networks with inaccurate information: theory and algorithms'. Proceedings of INFOCOM'97, Kobe, Japan, 1997, pp. 75–83
4 WANG, J., WANG, W., CHEN, J., and CHEN, S.: 'A randomized QoS routing algorithm on networks with inaccurate link-state information'. Proceedings of WCC-ICCT 2000, Beijing, China, 2002, Vol. 2, pp. 1617–1622
5 LORENZ, D.H., and ORDA, A.: 'QoS routing in networks with uncertain parameters'. Proceedings of INFOCOM'98, San Francisco, CA, USA, 1998, Vol. 1, pp. 3–10
6 LORENZ, D.H., and ORDA, A.: 'QoS routing in networks with uncertain parameters', *IEEE/ACM Trans. Netw.*, 1998, **6**, (6), pp. 768–778
7 CHEN, S., and NAHRSTEDT, K.: 'Distributed QoS routing with imprecise state information'. Proceedings of 7th International Conference on Computer communications and networks, 1998, pp. 614–621
8 WANG, Z., and CROWCROFT, J.: 'Quality of service routing for supporting multimedia applications', *IEEE J. Sel. Areas Commun.*, 1996, **14**, (7), pp. 1228–1234
9 KORKMAZ, T., KRUNZ, M., and TRAGOUDAS, S.: 'An efficient algorithms for finding a path subject to two additive constraints'. Proceedings of ACM SIGMETRICS'2000, 2000, pp. 318–327
10 POMAVALZI, C., CHAKRABORTY, G., and SHIRATORI, N.: 'QoS based routing algorithm in integrated services packet networks'. Proceedings of IEEE 1997 Conference on Network protocols, Atlanta, GA, USA, 1997, pp. 167–174
11 JUTTNER, A., SZYIATOVSZKI, B., MECS, and RAJKO, I.: 'Lagrange releaxation based method for the QoS routing problem'. Proceedings of IEEE INFOCOM 2001, Anchorage, AK, USA, 2001, Vol. 2, pp. 859–868
12 GUO, L., and MATTA, I.: 'Search space reduction in QoS routing'. Proceedings of 19th IEEE International Conference on Distributed computing systems, 1999, pp. 142–149
13 GANG, L., and RAMAKRISHNAN, K.G.: 'A* Prune: an algorithm for finding k shortest paths subject to multiple constraints'. Proceedings of IEEE INFOCOM 2001, Anchorage, AK, USA, 2001, Vol. 2, pp. 743–749
14 EPPSTEIN, D.: 'Finding the k shortest path'. Proceedings of 35th Annual Symposium on Foundations of computer science, 1994, pp. 154–165
15 CHEN, S., and NAHRSTEDT, K.: 'On finding multi-constrained path'. Proceedings of IEEE ICC'98, 1998, Vol. 2, pp. 874–899
16 DOAR, M.B.: 'A better model for generating test networks'. Proceedings of GLOBECOM'96, 1996, pp. 86–93