IEICE TRANS. COMMUN., VOL.E86–B, NO.2 FEBRUARY 2003

PAPER *Special Issue on Internet Technology III*

A New QoS Routing Framework for Solving MCP

Gang CHENG[†], **Ye TIAN**[†], ***Nonmembers, and*** **Nirwan ANSARI**[†], ***Regular Member***

SUMMARY One purpose of Quality-of-Service (QoS) routing is to develop polynomial-time heuristic algorithms to tackle the MCP (multi-constrained-path) problem, which is NP-complete. In this paper, we introduce a new QoS routing heuristic framework, which focuses on how to increase the success ratio for finding a feasible path subject to multiple additive constraints. The key issue of this framework is to transform the single source single destination QoS routing problem to a single source multi-destination problem by expanding the destination vertex to its neighboring vertices. After that, the modified problem can be solved by existing source routing heuristic algorithms. The analysis and simulation results demonstrate that the framework can achieve a higher success ratio of finding a feasible path without increasing the computational complexity by setting the expansion operation properly.

key words: *QoS, source routing, MCP*

1. Introduction

Nowadays, a large number of multimedia applications are delivered through networks, in particular, the Internet. One of the critical issues is noted as the quality-of-service (QoS) provisioning. Manifold QoS requirements of multimedia applications make this issue extremely challenging, not to mention the inherent NP-completeness [1] of a multi-constrained-path (MCP) routing problem.

QoS requirements are diverse, subject to demands of different applications. Bandwidth, delay, delay jitter, and loss ratio are the commonly required QoS metrics. These requirements can be classified into three types: concave, additive, and multiplicative [2]. Since the concave type can be easily pruned by selecting the bottleneck, and the multiplicative type can be converted into the additive constraints by the logarithmic operation, the constraints considered in this paper are additive, unless otherwise mentioned.

The purpose of QoS routing is to find a path (or tree for multicast, which is beyond the scope of this paper) in a given network that meets all the end-to-end requirements while utilizing network resources efficiently [3]. The network can be represented by a directed graph $G(V, E)$ where V is the set of vertices and E is the set of edges. It is assumed that the link state information is well maintained and updated. When the number of constraints is one, the problem is simply a shortest path problem, which can be solved by existing shortest path algorithms with bounded computational complexity, e.g., Dijkstra and Bellman-Ford shortest path algorithms. When the number of constraints is more than one, the problem is NP-complete. Many polynomial-time heuristic algorithms have been proposed. Most of them use either source or distributed routing strategies.

Source routing assumes that the source vertex has a global view of the link state information of the network and is able to compute the path locally. We classify existing source routing heuristic algorithms that have polynomial-time performance into three types: the limited granularity heuristic, the limited path heuristic, and the constraint aggregation heuristic. Yuan [4] has given a comprehensive analysis of the first two types, focusing on their computational and space complexities.

The limited granularity heuristic [5] proposed by Chen and Nahrstedt transforms one of two constraints from a positive real number to a positive integer by the mapping function $w' = \lceil \frac{wx}{c} \rceil$, where w is a function of the link weight, c is the path constraint, and x is a user-defined value. This mapping offers a "coarser resolution" of the original constraints, so that the computational complexity is reduced to polynomial-time, $O(x^2|V|^2)$ for two constraints, and the performance of the algorithm can be improved by increasing x.

The limited path heuristic [4, 6] proposed by Yuan maintains a limited number of candidate paths, say x, at each hop. The computational complexity is $O(x^2|V||E|)$ for the Extended Bellman-Ford algorithm for two constraints. The drawback is that candidate paths may be eliminated from the search space due to decimation of paths.

Both the granularity heuristic and limited path heuristic adopt similar methodology. To reduce the computational complexity, which is essential for solving this NP-complete problem, the limited granularity heuristic quantizes the link metrics while the limited path heuristic reduces the search space by eliminating some candidate paths. According to Yuan's paper [4], for more than two constraints, the limited path heuristic has a better performance than the limited granular-

Manuscript received June 24, 2002.
Manuscript revised August 12, 2002.

[†]The authors are with the Advanced Networking Laboratory, Department of Electrical and Computer Engineering, NJIT, Newark 07102, U.S.A. This work has been supported in part by the New Jersey Commission on High Education via the NJI-TOWER project, and the New Jersey Commission on Science and Technology via the NJCWT.

ity heuristic in terms of space complexity.

The constraint aggregation heuristic [7]-[8] combines multiple constraints into one cost function. The problem can then be solved by using an extended shortest path algorithm for a single constraint. For instance, Lagrange Relaxation Based Aggregated Cost (*LARLAC*) was proposed in [7] for the Delay Constrained Least Cost path problem (*DCUR*). 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 that is iteratively updated according to the previous search. It was shown that the computational complexity of this algorithm was $O(|E|^2 \log^4 |E|)$. Typically, a linear cost function that combines multiple constraints linearly is chosen. This technique will be adopted in our proposed heuristic framework. Feng et al. [9] investigated the delay-constrained least-cost QoS routing problem based on both linear and non-linear cost functions.

Source routing is relatively simple to realize, but it suffers from a high computational overhead and poor scalability. Distributed routing may also depend on the global state information maintained by each vertex [2], [10]. The path computing decision is, however, made on a hop-by-hop basis.

This paper proposes a new source routing framework, source routing destination expansion (*SRDE*), to solve the MCP problem. It incorporates existing source routing heuristics, resulting in a higher success ratio in finding a feasible path without increasing the overall computational complexity. The rest of the paper is organized as follows. In Sect. 2, we discuss the motivation behind our proposed heuristics. The computational complexity analysis and related issues are discussed in Sect. 3. The proposed source routing framework is presented in Sect. 4, and simulation results are evaluated in Sect. 5. Sect. 6 summarizes the significance of the proposed framework. Conclusions are drawn in Sect. 7.

2. Motivation

Most of the heuristic algorithms for solving the MCP problem try to find a path subject to relaxed constraints or conditions in order to reduce the complexity. Thus, some feasible paths may become irretrievable. For example, given a network graph and the link state metrics as shown in Fig. 1, we want to find a path that satisfies the given end-to-end constraints $(x, y) = (1, 1)$ from Vertex 1 to Vertex 6.

Method 1: Using the aforementioned constraint aggregation approach, we define a new cost function $w = x + y$. The shortest path algorithm will select the path 1→2→3→6 with an aggregated weight of 1.7. Since the actual weights of the path are $(0.6, 1.1)$, the selected path violates the original end-to-end constraints, and is thus not a feasible path for the problem.

Method 2: (our proposed method): Before applying the shortest path selection algorithm, we expand the destination from Vertex 6 to its neighboring vertices, i.e., Vertices 3, 4, and 5. Then, we revise the constraints for Vertices 3, 4, and 5 by subtracting the link state metrics $(0.2, 0.5)$, $(0.3, 0.3)$, and $(0.6, 0.8)$ from the original constraints $(1, 1)$, accordingly. The network graph is updated as shown in Fig. 2. The dotted lines represent edges that have been eliminated and the hollow vertex represents the removed vertex.

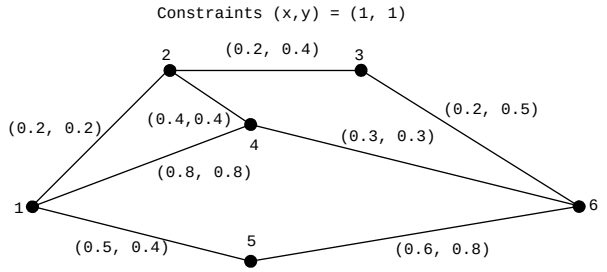


Fig. 1 A network graph with constraints.

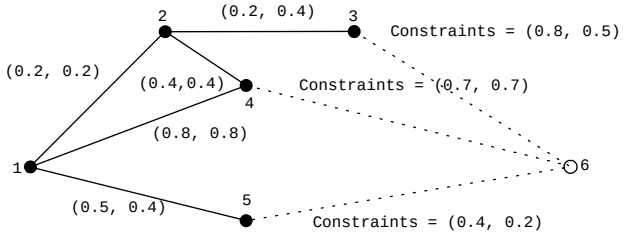


Fig. 2 A network graph with 1-hop expansion.

Our modified problem is now to find a feasible path that satisfies the revised constraints from Vertex 1 to any of Vertices 3, 4, or 5. Note that the feasible paths from Vertex 1 to Vertices 3, 4, or 5 are also the feasible paths of the original problem. By applying the same strategy in Method 1, the path 1→2→4 is selected. Vertex 4 is known to have been expanded from Vertex 6. Thus, 1→2→4→6 is the completed path from Vertex 1 to Vertex 6 and it turns out to be a feasible path.

As compared to Method 1, it is observed that the modified problem has multiple destinations instead of one destination in the original problem.

Although the expansion introduces a computational overhead to the routing, in return, it increases the probability of finding a feasible path by transforming the original problem from single source single destination to single source multi-destination. As a matter of fact, both Dijkstra and Bellman-Ford algorithms explore a shortest path tree rooted from the source [11]. The complexity of the algorithm for solving single source multi-destination shortest path problem is not necessarily higher than that of the single source single destination algorithms, which are considered in most literatures. Furthermore, due to the expansion,

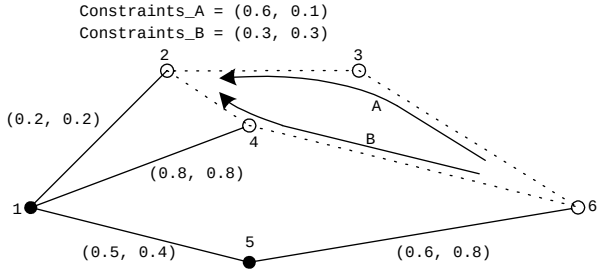


Fig. 3 A network graph with 2-hop expansion.

we keep more accurate link state information than that of the case without expansion. If we can tolerate the complexity introduced by the expansion, we may have a better chance to find a feasible path. It also enhances scalability of the source routing to some extent.

A case worth mentioning here is illustrated in Fig. 3. Note that multiple paths may exist between the destination and a vertex resulted from the expansion operation. Suppose we expand two hops from Vertex 6 along Path A (6→3→2) and Path B (6→4→2) respectively, and revise the constraints. The revised constraints are $(0.6, 0.1)$ along Path A (from Vertex 6 to Vertex 2 via Vertex 3) and $(0.3, 0.3)$ along Path B (from Vertex 6 to Vertex 2 via Vertex 4), respectively. Both of them need to be kept at Vertex 2, as shown in Fig. 3 since they both are candidates for a feasible path. On the other hand, if both constraints in one set are tighter than those in the other set, $(0.2, 0.2)$ and $(0.3, 0.3)$ for instance, we can simply prune $(0.2, 0.2)$ and keep $(0.3, 0.3)$.

Those candidate paths introduced by the expansion operation incur some computational overhead. However, this overhead can be completely mitigated by restricting the number of hops to expand. The detailed analysis is given in the following section.

3. Computational Complexity Analysis

UUsually, the computational complexity of the shortest path search in source routing depends on the number of edges and/or vertices, e.g., $O(|V||E|)$ for the Bellman-Ford algorithm. We can see that the expansion operation in Method 2 will eliminate some edges and vertices but possibly introduce more sets of constraints on the same vertex. Hence, the computational complexity of Method 2 is the reduced complexity of the shortest path search due to the reduced number of vertices and edges, plus the complexity introduced by the expansion operation.

In order to achieve the similar level of complexity as that of Method 1, the number of hops allowed to expand from the destination is critical in Method 2. This expansion operation is similar to the Extended Bellman-Ford algorithm *EBFA* in Yuan's work [4], per se. It expands the destination hop by hop and records all candidate paths. The computational complexity will increase with the number of hops. In Yuan's work [4], he limited this complexity by recording only a limited number of candidate paths at each vertex. In our approach, we restrict the computational complexity by limiting the number of hops to expand.

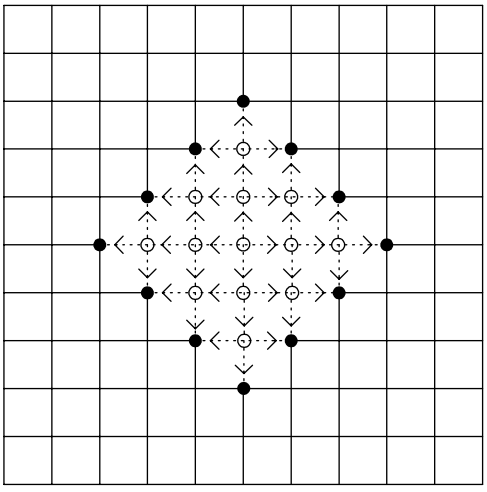


Fig. 4 A mesh network for illustrating the expansion operation.

3.1 Complexity Analysis of a Mesh Network

We shall use an $N \times N$ mesh network (see Fig. 4) to illustrate the effect of the expansion operation on the performance of the overall source routing. We expand the destination vertex to its neighboring vertices that are h hops away and observe how the number of network vertices and edges changes according to Method 2. Assume the network is large enough so that the edge effect is negligible. Figure 4 shows a vertex (center) has been expanded 3 hops. Dotted lines represent eliminated edges and hollow nodes represent removed vertices. Arrows illustrate the direction of propagation. Solid nodes reveal the boundary of the expansion operation.

Lemma 1: For a large $N \times N$ mesh network, expanding h hops from the destination vertex eliminates $2h^2 - 2h + 1$ vertices and $4h^2$ edges.

Lemma 2: For a large $N \times N$ mesh network, the worst-case scenario of expanding h hops from the destination vertex introduces $4 \times 3^{h-1}$ extra sets of constraints.

Lemmas 1 and 2 can be readily derived from the geometrical properties of the graph.

Theorem 1: For a large $N \times N$ mesh network, the complexity of the standard Bellman-Ford algorithm (Method 1) is $O(2N^3(N-1))$ and the worst-case complexity of an h-hop-expansion based Bellman-Ford al-

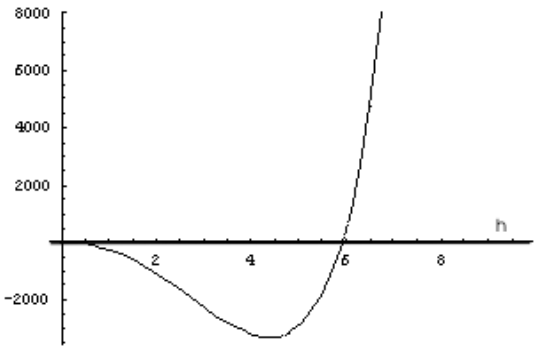


Fig. 5 The difference in the run time between two methods.

gorithm (Method 2) is $O(N^4 - N^2h^2 + 3^h)$ for 2 constraints, where h is a small integer representing the number of hops to expand.

Proof: In a mesh network, the number of vertices is $|V| = N^2$ and the number of edges is $|E| = 2N(N-1)$. Therefore, the overall complexity of the Bellman-Ford algorithm is: $O(|E||V|) = O(2N^3(N-1))$. (1)

The overall complexity after h-hop expansion is the reduced complexity of the Bellman-Ford algorithm due to the reduced number of vertices and edges, plus the complexity introduced by the expansion operation, i.e., $O((|V|-|V|_{removed})(|E|-|E|_{removed}))+O(expansion)$.

$|V|_{removed}$ and $|E|_{removed}$ are provided by Lemma 1, and $O(expansion)$ is given in Lemma 2. Hence, the overall complexity is:

$$\begin{aligned} &O((|V|-|V|_{removed})(|E|-|E|_{removed}))+O(expansion) \\ &= O((N^2-(2h^2-2h+1))(2N(N-1)-4h^2)+4\times4\times3^{h-1}) \\ &= O((N^2-2h^2+2h-1)(2N^2-2N-4h^2)+16\times3^{h-1}) \end{aligned} \quad (2)$$

Keeping only the higher order terms, the complexity is: $O(N^4 - N^2h^2 + 3^h)$. □

Based on Theorem 1, we can mathematically observe the complexity difference between Method 1 and Method 2. Figure 5 plots the difference in the algorithm run time between Method 2 and Method 1 (Eq. (2) - Eq. (1)) versus the number of hops to expand in a 7×7 mesh network.

The zero-crossing point is around $h = 6$, implying that Method 2 has less computational complexity than Method 1 as long as $h < 6$. Thus, Method 2 is able to achieve similar complexity as Method 1 if h is set properly.

3.2 Complexity Analysis of a Randomized Network

The impact of Method 2 on the computational complexity has been observed in a mesh network. Now we want to apply this method to a randomized network. A question that arises is how similar the randomized network is to the mesh network.

This is a relatively complex issue when considering a randomized network since the randomized network is less predictable, as compared to a mesh network. There are two parameters, the average degree of the vertex $\bar{D}$ and number of hops to expand h, related to the complexity. The introduced complexity due to the expansion operation is approximately equivalent to $O((\bar{D}-1)^h)$†. The base $(\bar{D}-1)$ of the exponential function reflects the fact that the intermediate vertex passes the constraint sets along all its edges to its neighbors except the one from which it receives. For an h-hop expansion, the complexity is thus approximately exponential, with the base $(\bar{D}-1)$ and the power of h. For the mesh network case, the introduced complexity due to the expansion is $O(3^h)$ since $\bar{D} = 4$. As long as we limit h to a small integer, the number of candidate paths at each vertex is also small, i.e., the linear region of the exponential curve. The introduced complexity of Method 2 can be balanced by the reduced complexity due to eliminated vertices and edges. Therefore, applying Method 2 to a randomized network should not increase the overall complexity by setting properly.

4. The Proposed Source Routing Framework

The MCP problem contains multiple additive constraints. For illustrative purposes, we consider cost and delay constraints, and formulate the cost-delay-constrained problem as follows:

Definition: Given a directed graph $G(V, E)$, a source vertex s, a destination vertex t, a cost function c, a delay function d, a cost bound C and a delay bound D, the $MCP(G, s, t, c, d, C, D)$ problem is to find a path p, from s to t such that the total cost $c(p) \leq C$ and the total delay $d(p) \leq D$.

The proposed source routing framework, Source Routing Destination Expansion ($SRDE$), can be described as a shortest path routing algorithm based on an h-hop expansion from the destination. The routing procedure is performed in three steps, as illustrated by Fig. 6.

- Step 1. The algorithm expands n hops from the destination t. It records all vertices t', which are h hops away from t, and paths p', which are sets of edges from t to t'. C' and D' are the cost and delay of p', i.e., $C' = c(p')$ and $D' = d(p')$. The edges and vertices along p' are eliminated from the network graph.
- Step 2. The algorithm solves the modified problem $MCP(G, s, t', c, d, C - C', D - D')$ that is to find a feasible path p'' from the single source s to any of the multiple destinations t' with revised cost and delay constraints $C - C'$ and $D - D'$.

†Note that this is the worst-case complexity. For a more general case, the complexity is far less than that.

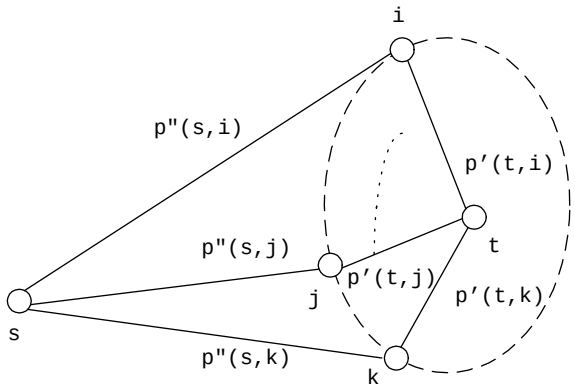


Fig. 6 Graphic illustration of SRDE.

- Step 3. Combine $p^{'}$ and $p^{''}$ to form a path p.

Theorem 2: Path $p = p^{'} + p^{''}$ is a solution to the original $MCP(G, s, t, c, d, C, D)$ problem.

Proof: We have $c(p^{''}) \leq C - C^{'}$ and $d(p^{''}) \leq D - D^{'}$ since $p^{''}$ is a solution to $MCP(G, s, t^{'}, c, d, C - C^{'}, D - D^{'})$. Also we have $C^{'} = c(p^{'})$ and $D^{'} = d(p^{'})$ from Step 2. Thus:

$$c(p) = c(p^{'}) + c(p^{''}) \leq C - C^{'} + C^{'} = C \quad (3)$$

$$d(p) = d(p^{'}) + d(p^{''}) \leq D - D^{'} + D^{'} = D \quad (4)$$

Hence, path satisfies both constraints of the original problem. □

Theorem 2 validates the proposed source routing framework *SRDE*. The pseudo-code of the algorithm is given in Fig. 7.

Procedure Expand(G,t,h) performs a "flood and prune" operation. It computes paths, which contain h hops starting from t. It prunes a path if both of the constraints are larger than the previously recorded values. Otherwise, it inserts the path as a new entry. It removes vertices and edges along those paths from the network graph G. It also returns a set of new destination vertices Dst and their corresponding constraints $con(p)$.

Procedure SRDE(G,s,t,$Constraints$,h) can employ any ***shortest path algorithm***(G,s,Dst,$con(p)$) (line 27) that solves the MCP problem for a given network G, a source vertex s, a destination Dst, and a set of constraints $con(p)$. It returns "success" if a feasible path is selected.

```
(1)  Initialize(G, t, constraints)
(2)  for each vertex in G
(3)    PATH(v)={0}
(4)  end for
(5)  PATH(t)={t}
(6)  set the constraint of path p={t} as (C,D)
(7)  Dst={t}

(8)  Expand(G, t, h)
(9)  for i from 1 to h
(10)   for each vertex v in Dst
(11)     for each path p in PATH(v) and its
         corresponding constraints con(p)
(12)       if(con(p)-(c(u,v),d(u,v))>(0,0))
(13)         Dst=Dst ⋃ {u}
(14)         p=p ⋃ {u}
(15)         PATH(v)=PATH(v) ⋃ p
(16)         con(p)=con(p)-(c(u,v),d(u,v))
(17)       end if
(18)       remove edge (u,v) from G
(19)     end for
(20)   end for
(21) end for
(22) remove dissociating vertices from G
(23) return PATH(v), Dst, con(p)

(24) SRDE(G, s, t, constraints, h)
(25) Initialize(G, t, constraints)
(26) Expand(G, t, h)
(27) shortest path algorithm(G, s, Dst, con(p))
(28) for each vertex v in Dst
(29)   for each path p in PATH(v)
(30)     if the weights of the least cost path
         from s to v less than con(p)
(31)       return "success"
(32)     end if
(33)   end for
(34) end for
(35) return "fail"
```

Fig. 7 The pseudo-code for the SRDE algorithm.

5. Simulation Results

In this section, we simulate our framework in the mesh network and randomized network of different sizes respectively. The size of the mesh network used by the simulation is 7×7. We use a 50-node and 32-node randomized network for the simulation. The 32-node network (see Fig. 8) is also used in Chen et al.'s work [5]. There are two equal QoS constraints, of which the range is from 0.5 (tight) to 5.5 (loose) with an increment of 0.2, in the simulation. The weights of the link state are uniformly distributed from 0 to 1. Each simulation invokes 100,000 experiments.

The performance parameter of concern is the success ratio, which is defined as:

$$\text{SR} = \frac{\text{The number of success requests of the algorithm}}{\text{The number of success requests of the optimal algorithm}}\ ^{\dagger}$$

The *SRDE* framework incorporates the Bellman-Ford algorithm using an aggregated linear cost function $w = c/C + d/D$ with h-hop expansion. We compare the results with the standard Bellman-Ford algorithm subject to the aggregated linear cost function $w = c/C + d/D$. It is expected that the former one has a better success

†The algorithm that can always find a feasible path as long as it exists is refereed to as the optimal algorithm. Simulation-wise, it is achieved simply by flooding which is rather exhaustive.

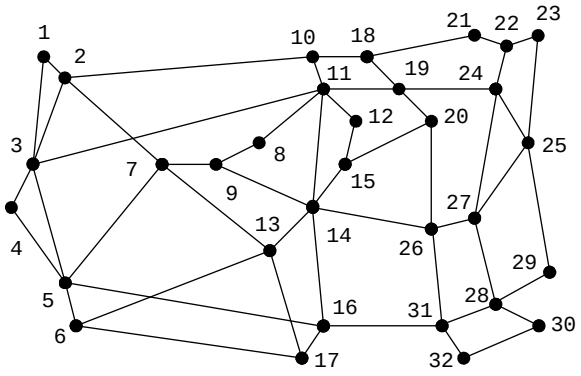


Fig. 8 The 32-node randomized network.

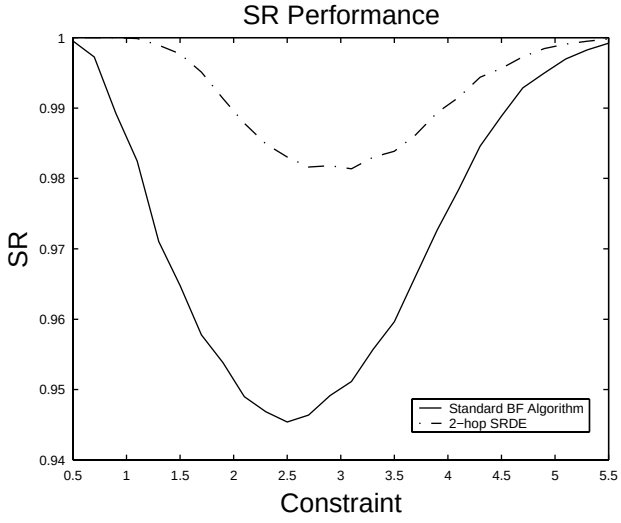


Fig. 9 Simulation results for a 7x7 mesh network.

ratio than the later one.

Figure 9 shows the simulation results of the 2-hop *SRDE* in a 7×7 mesh network as compared to the standard Bellman-Ford algorithm. Along the x-axis is the value of the constraints (either C or D, since $C = D$) and along the y-axis is the success ratio. The result shows that the 2-hop *SRDE* outperforms the standard Bellman-Ford Algorithm.

We have already shown in Sect. 3 that in a 7×7 (49-node) mesh network, the computational complexity of our proposed framework incorporated with the extended Bellman-Ford algorithm is no more than that of the standard Bellman-Ford algorithm as long as the number of hops allowed to expand $h < 6$. To ensure the complexity of our algorithm is still less than that of the standard Bellman-Ford algorithm in a randomized network, we choose an even smaller number of hops to expand. Based on our conservative estimation, we ran the simulation for the 2-hop and 3-hop *SRDE*, respectively, for the randomized network topology. The simulation results of the 50-node randomized network

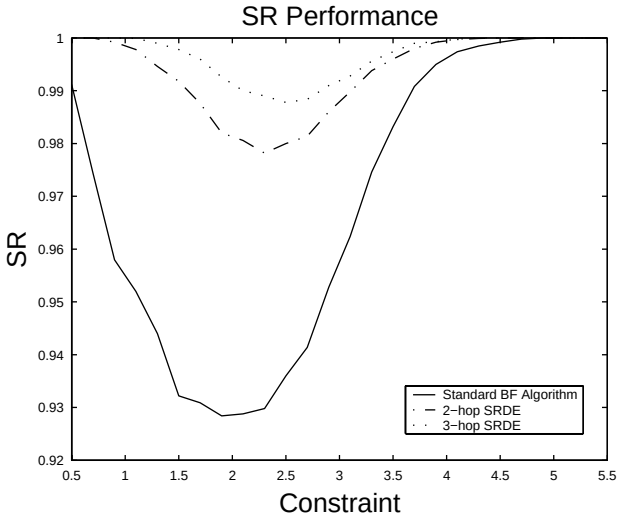


Fig. 10 Simulation results for a 50-node randomized network.

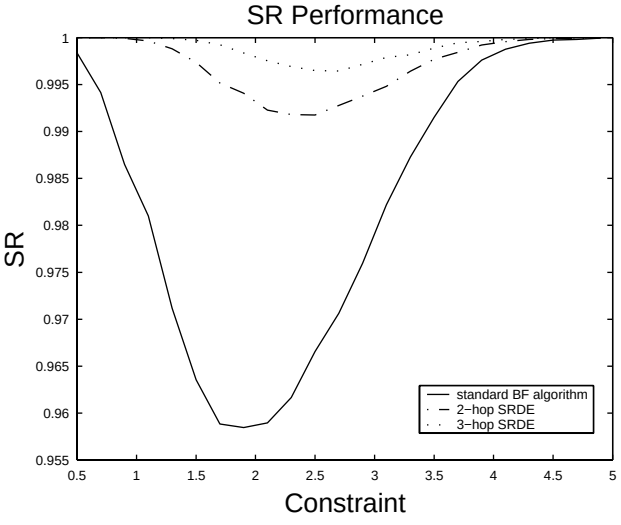


Fig. 11 Simulation results for a 50-node randomized network.

are shown in Fig. 10.

The plot shows that our algorithm achieves a higher success ratio for both cases of $h = 2$ and $h = 3$. Note the dramatic improvement even with $h = 2$. It implies that the larger the number h is, the higher the success ratio the algorithm may achieve.

The simulation is also performed in a 32-node randomized network (see Fig. 8). The simulation results in Fig. 11 show the improvement of *SRDE* with 2 and 3 hop expansion over the standard Bellman-Ford algorithm, which agrees with our analysis.

We observe that the phenomena of the success ratio improvement are similar for both the mesh network and the randomized network. We also observe that the success ratio is higher in a 32-node network than in a 50-node network. That is to say, a higher success ratio can be achieved in a smaller randomized network

topology.

6. Discussion

In this paper, we incorporate the extended Bellman-Ford algorithm (EBF) based on the aggregated cost function approach, which is typical for solving such problems, in our proposed framework. Practically, the EBF algorithm can be substituted by other source routing algorithms. The significance of the proposed framework is that it allows the adopted source routing algorithm to search for a feasible path via the expansion operation. This increases the probability of finding a feasible path. One path, which might be missed by the original algorithm, may be extracted by the algorithm after being incorporated by the proposed framework.

The proposed framework performs two operations, the destination expansion and the shortest path search. For illustrative purpose, we have used the 2-constraint paradigm in previous sections to solve the cost-delay-constrained problem, but the proposed framework is applicable to multiple constraints. For the multi-constraint case, the computational complexity given by Theorem 1 is still valid. The complexity introduced by the hop-by-hop expansion is due to the existence of multiple paths between the destination and its neighboring vertices, and bounded by $O(3^h)$ for a mesh network and $O((\bar{D}-1)^h)$for a randomized network. After the expansion, the performance of the shortest path search depends on the adopted multi-constrained routing algorithms, in our case, EBF.

7. Conclusions

We have introduced a source routing framework that can effectively solve the MCP. By applying this method, the original single source single destination problem becomes a single source multi-destination problem. A feasible path is searched from the source to any of the destinations. Moreover, during the expansion operation, no link state information is lost. Hence, a higher success ratio is expected. The advantage of this framework is that various source routing heuristics can be integrated into this framework to perform the task of the shortest path search. A better performance is expected over the original source routing heuristics without additional complexity as long as the number of hops allowed in the expansion operation is small enough. In practice, 2 or 3 hop expansion will suffice.

References

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

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

[3] S. Chen and K. Nahsted, "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, Dec. 1998.

[4] X. Yuan, "Heuristic algorithms for multiconstrained quality-of-service routing," IEEE/ACM Transactions on Networking, vol.10 issue.2, pp.244–256, April 2002.

[5] S. Chen and K. Nahrsted, "On finding multi-constrained paths," Proc. IEEE ICC'98, vol.2, pp.874–899, 1998.

[6] X. Yuan, "On the extended Bellman-Ford algorithm to solve two-constrained quality of service routing problems," Proc. Eighth International Conference on Computer Communications and Networks, pp.304–310, 1999.

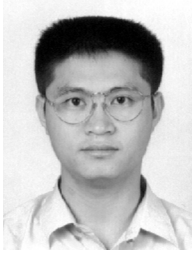
[7] A. Juttner, B. Szyiatovszki, I. Mecs, and Z. Rajko, "Lagrange relaxation based method for the QoS routing problem," Proc. IEEE INFOCOM 2001, vol.2, pp.859–868, 2001.

[8] T. Korkmaz , M. Krunz, and S. Tragoudas, "An efficient algorithms for finding a path subject to two additive constraints," ACM SIGCOMM 2000, pp.318–327, June 2000.

[9] G. Feng, C. Douligeris, K. Makki, N. Pissinou, "Performance evaluation of delay-constrained least-cost QoS routing algorithms based on linear and nonlinear lagrange relaxation," IEEE International Conference on Communications, vol.4, pp.2273–2278, 2002.

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

[11] T.H. Cormen, C.E. Leiserson, and R.L. Rivet. Introduction to Algorithms, Massachusetts: MIT Press, (Chapter 25, pp.514–556), 2000.



Gang Cheng received the B.E. and M.E. in Information Engineering from the Beijing University of Posts and Telecommunications, Beijing, P.R.China, in 1997 and 2000, respectively. His research interests include Quality of Service (QoS), network protocol design and performance analysis. He is currently pursuing the Ph.D. degree in electrical engineering at the New Jersey Institute of Technology, Newark, U.S.A.

Ye Tian received the B.E. (Honors) degree in Electrical and Electronic Engineering from University of Canterbury, Christchurch, New Zealand in 1999, and M.S.E.E. degree in Electrical and Computer Engineering from NJIT, Newark, U.S.A. in 2002, respectively. He is a Ph.D. candidate in Electrical and Computer Engineering, NJIT. His current research interests include QoS routing, WLAN and IPSec.

Nirwan Ansari received the B.S.E.E. (summa cum laude), M.S.E.E., and Ph.D. from NJIT, University of Michigan, and Purdue University in 1982, 1983, and 1988, respectively. He joined the Department of Electrical and Computer Engineering, NJIT, in 1988, and has been Professor since 1997. He is a technical editor of the IEEE Communications Magazine, was instrumental, while serving as its Chapter Chair, in rejuvenating the North Jersey Chapter of the IEEE Communications Society which received the 1996 Chapter of the Year Award, currently serves as the Chair of the IEEE North Jersey Section, and also serves in the IEEE Region 1 Board of Directors and various IEEE committees. He was the 1998 recipient of the NJIT Excellence Teaching Award in Graduate Instruction, and a 1999 IEEE Region 1 Award. His current research focuses on various aspects of high-speed networks. He authored with E.S.H. Hou Computational Intelligence for Optimization (1997, and translated into Chinese! in 2000), and edited with B. Yuha's Neural Networks in Telecommunications (1994), both published by Kluwer Academic Publishers.