Achieving 100% Success Ratio In Finding The Delay Constrained Least Cost Path

Gang Cheng and Nirwan Ansari

Abstract— **In this paper, we introduce an Iterative All Hops k-shortest Paths (IAHKP) algorithm that is capable of iteratively computing all hops k-shortest path (AHKP) from a source to a destination. Based on IAHKP, a high performance algorithm, Dual Iterative All Hops k-shortest Paths (DIAHKP) algorithm, that can achieve 100% success ratio in finding the Delay Constrained Least Cost (DCLC) path with very low average computational complexity is proposed. The underlining concept is that since DIAHKP is a *k*-shortest-paths-based solution to DCLC, implying that its computational complexity increases with *k*, we can minimize its computational complexity by adaptively minimizing *k*, while achiving 100% success ratio in finding the optimal feasible path. Through extensive analysis and simulations, we show that DIAHKP is highly effective and flexible. By setting a very small upper bound to k (k=1,2), DIAHKP still can achieve very satisfactory performance. With only an average computational complexity of twice that of the standard Bellman-Ford algorithm, DIAHKP achieves 100% success ratio in finding the optimal feasible path in the typical 32-node network.**

Index Terms— **Delay Constrained Least Cost (DCLC), All Hops *k*-shortest Paths Selection (AHKP), NP-complete.**

I. Introduction

Selecting feasible paths that satisfy various (QoS) requirements of applications in a network 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 routing strategy, especially finding the delay constrained least cost path, and assume that accurate network state information is available to each node.

It has been proved that the delay constrained least cost path selection is NP-complete [3]. Hence, tackling this problem requires heuristics. The limited path heuristic proposed by Yuan [4] maintains a limited number of candidate paths, say x, at each hop. The computational complexity is $O(x^2nm)$ for the Extended Bellman-Ford algorithm for two constraints, where n and m are the number of links and nodes, respectively. For the purpose of improving the response time and reducing the computation load on the network, precomputation-based methods [5] have been proposed. Korkmaz and Krunz [6] provided a heuristic with the computational complexity compatible to that of the Dijkstra algorithm to find the least cost path subject to multiple constraints. An algorithm [7], called A*Prune, is capable of locating multiple shortest feasible paths from the maintained heap in which all candidate paths are stored. For the case that only inaccurate link state information is available to nodes, approximate solutions [8] have been proposed for the Most Probable Bandwidth Delay Constrained Path (MP-BDCP) selection problem by decomposing it into two sub-problems: the Most Probable Delay Constrained Path (MP-DCP) and the Most Probable Bandwidth Constrained Path (MP-BCP). A LAgrange Relaxation based Aggregated Cost (LARAC) was proposed in [9] 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 was shown that the computational complexity of this algorithm is $\mathrm{O}(m^2 \log^4 m)$. Many researchers have posed the QoS routing problem as the k-shortest path problem [10]. The authors in [11] proposed an algorithm, called TAMCRA, for Multiple Constrained Path selection (MCP) by using a non-linear cost function and a k-shortest path algorithm. The computational complexity of TAMCRA is $O(kn\log(kn) + k^3mM)$, where k is the number of shortest paths and M is the number of constraints.

Many ε-approximation algorithms (the solution has a cost within a factor of $(1+\varepsilon)$ of the optimal one) subject to DCLC have been proposed in the literature. Lorenz et al. [12] presented several ε-approximation solutions for both the DCLC and the multicast tree. Among them, the algorithm subject to DCLC possesses the best-known computational complexity of $O\left(nm\log n \log(\log n) + \frac{nm}{\varepsilon}\right)$. Hassin [13] presented two ε-approximations algorithms for the Restricted Shortest Path problem (RSP) with complexities of $O((\frac{nm}{\varepsilon})\log\log U)$ and $O(m\frac{n^2}{\varepsilon}\log(\frac{n}{\varepsilon}))$, where U is the upper bound of the cost of the path computed. Raz and Shavitt [14] proposed an efficient dynamic programming solution for the case in which the QoS parameters are integers, and a sub-linear algorithm for the case in which all link costs use the (same) function of their corresponding delays.

Existing algorithms reviewed above may have the following drawbacks.

1) Although the algorithms such as the ε-approximation approaches [12], [13] can achieve 100% or near 100% success ratio, their worst-case computational complexities are too high to be practical (assume ε is very small in ε-approximation algorithms so that their success ratios are close to 1).
2) The algorithms such as [11] have the advantage of having low computational complexities. However, they

Authors are with the Advanced Networking Laboratory, ECE Dept., NJIT, Newark, NJ 07012, U.S.A. (corresponding author to provide phone/fax: 973-596-3670; e-mail: ansari@njit.edu).

This work has been supported in part by the New Jersey Commission on Higher Education via the NJ I-TOWER project, and the New Jersey Commission on Science and Technology via NJWINS.

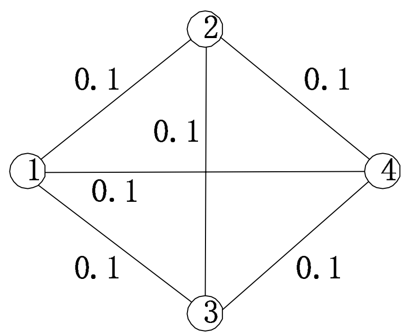
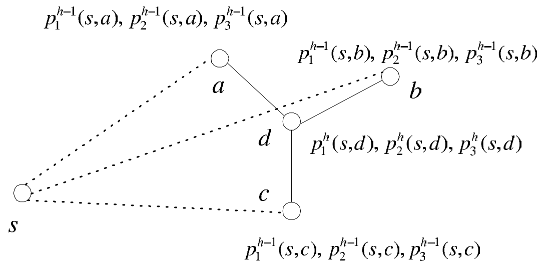


Fig. 1. A 4-node network.



$p_1^{h-1}(s,a)$, $p_2^{h-1}(s,a)$, $p_3^{h-1}(s,a)$
$p_1^{h-1}(s,b)$, $p_2^{h-1}(s,b)$, $p_3^{h-1}(s,b)$
a
b
d
$p_1^{h}(s,d)$, $p_2^{h}(s,d)$, $p_3^{h}(s,d)$
c
s
$p_1^{h-1}(s,c)$, $p_2^{h-1}(s,c)$, $p_3^{h-1}(s,c)$

Fig. 2. Node d has three neighboring nodes, a, b, and c.

cannot guarantee in finding a feasible path when it exists. Moreover, their success ratios in finding a feasible path may decrease sharply with the network size. In order to increase the success ratio, many proposed algorithms deploy k-shortest paths selection solutions (the number of shortest paths are generally fixed in these algorithms), instead of the shortest path selection solutions. However, the computational complexities of the algorithms increase unnecessarily in the case in which a feasible path can be found with only one shortest path searching algorithm.

In this paper, we propose an algorithm to solve the DCLC problem (defined below) that can overcome the above drawbacks. We denote $p_1 + p_2$ as the concatenation of two paths p_1 and p_2, and $\min\{p_1, p_2\}$ as the shortest path between p_1 and p_2. The least weight path is also referred to as the shortest path in this paper.

Definition 1: Delay Constrained Least Cost Path Selection (DCLC) [9]: Assume a network is modeled as a directed graph $G(N, E)$, where N is the set of nodes and E the set of links. Each link connected from node u to v, denoted by $e(u, v)$, is associated with a cost $c(u, v)$ and a delay $d(u, v)$. Given a delay constraint and a pair of nodes, s and t, the objective of DCLC is to find a path p that has the least cost among the paths from s to t subject to $D(p) = \sum_{e(u,v)\in p} d(u, v) < d$.

II. An Efficient Solution to AHKP

In this section, we propose an efficient solution, the Iterative All Hops *k*-shortest Paths (IAHKP) algorithm, to All Hops k-Shortest Paths (AHKP) selection problem defined below:

Definition 2: All Hops k-shortest Paths (AHKP) Selection Problem: Given a network $G(N, E)$ in which each link $e(u, v)$ is associated with an additive weight $w(u, v)$. For a given source node $s \in N$ and maximal hop count H, $H < n$, find, for each hop count value h, $1 \le h \le H$, and a destination node $u \in N$, the k shortest h-hop constrained paths from s to u.

Given a network shown as Fig. 1, $k = 1$, and $H = 3$, by the above defintion, we can compute the shortest h-hop paths from 1 to 4 (the solutions to AHKP) as (1,4), (1,2,4), and (1,2,3,4), respectively, when h equals to 1, 2, and 3, where $(\alpha_1, \alpha_2, ..., \alpha_h)$ represents an $(h-1)$-hop path from node α_1 to node α_h sequentially traversing nodes $\alpha_1, \alpha_2, ..., \alpha_h$. Intuitively, given any path, the predecessor node of the destination must be one of its neighboring node(s). Hence, for any hop count value h, $2 \le h \le H$, the k shortest h-hop paths from s to a node must be in the set of the paths constructed by concatenating the k least weight $(h-1)$-hop paths from s to its neighboring nodes and the corresponding links. For instance, as shown in Fig. 2, assume the neighboring nodes of node d are nodes a, b, and c, the 3 shortest h-hop paths from s to d, $p_1^h(s, d)$, $p_2^h(s, d)$, and $p_3^h(s, d)$, must be included in the set of paths $\{p_g^{h-1}(s, u) + e(d, u),\ u = a, b, c \text{ and } g = 1, 2, 3\}$, where $p_g^h(s, i)$ represents the gth shortest h-hop path from s to i. Denote d_i as the degree of node i and $i_1, i_2, \ldots, i_{d_i}$ as its neighboring nodes. Assume there exists a virtual link $\widehat{e}(s, i)$ between the source node s and any other node i, whose cost is infinity. We can compute the least cost h-hop paths from s to i, $p_1^h(s, i), p_2^h(s, i), \ldots, p_k^h(s, i)$, as follows:

1) $\forall i \in N$, $p_1^1(s, i) = e(s, i)$ and $p_g^1(s, i) = \widehat{e}(s, i)$, $g = 2, 3, ..., k$, (if, in reality, no link between the source s and node i exists, $p_1^1(s, i) = \widehat{e}(s, i)$).
2) $p_1^h(s, i)$, $p_2^h(s, i)$, $\ldots$, $p_k^h(s, i)$ are computed by selecting the k least weight h-hop paths from the paths $p_g^{h-1}(s, i_d) + e(i_d, i), d = 1, 2, ..., d_i$, $g = 1, 2, ..., k$. If, in reality, the total number of h-hop paths from s to i is less than k, we assume that there exist virtual h-hop paths whose costs are infinity.

We can apply the above two steps to the previous example to verify its correctness. The theoretical proof that $p_1^h(s, i)$, $p_2^h(s, i), \ldots, p_k^h(s, i)$ computed as above are the k shortest h-hop paths from s to i was detailed in [15]. We do not present it here due to the page limit. For the purpose of avoiding loops, we adopt a simple method: associating paths with indicators. For example, we associate a path traversing nodes s, 1, 5, and 7 with an integer array of size n, in which 1st, 5th, and 7th array elements are set to 1, and the rest to 0. Hence, we can easily find out if a node is in the path by only checking the corresponding array element's value. By this method, we can prevent loops without increasing the worst-case computational complexity.

Namely, IAHKP is capable of iteratively computing the all hops k-shortest path, i.e., it computes the all hops shortest paths from the source to all other nodes in the first iteration, the all hops second shortest paths in the second iteration, and so on. We first intuitively introduce how IAHKP works, i.e.,

```
Relax(i, j, k, h)
1  if $w(p_k^{h+1}(s,i)) > w(p^h_{v(i,h,k,j)}(s,j)+e(i,j))$ then
2      $p_k^h(s,i) = p^{h-1}_{v(i,h,k,j)}(s,j)+e(i,j)$
3      $\pi(i) = j$
4  end if
5  return
```

Fig. 3. The relaxation procedure of IAHKP

how IAHKP computes the all hops kth shortest paths when the all hops $(k-1)$ shortest path have been computed. As shown in Fig. 2, assume the three paths on any node are in the increasing order of their costs, and $p_1^h(s,d) = p_1^{h-1}(s,a) + e(d,a)$, $p_2^h(s,d) = p_2^{h-1}(s,a) + e(d,a)$, and $p_3^h(s,d) = p_1^{h-1}(s,c) + e(d,c)$. Hence, since $p_4^h(s,d)$ is selected as the 4th shortest path of $\{p_g^{h-1}(s,u) + e(d,u)$, $u = a,b,c$ and $g = 1,2,3,4\}$, and on any node, the paths are in the increasing order of their costs, $p_4^h(s,d) = \min\{p_3^{h-1}(s,a)+e(d,a),\ p_1^{h-1}(s,b)+e(d,b),\ p_2^{h-1}(s,c)+e(d,c)\}$, i.e., $p_4^h(s,d)$ is the shortest path of $p_3^{h-1}(s,a)+e(d,a)$, $p_1^{h-1}(s,b)+e(d,b)$, and $p_2^{h-1}(s,c)+e(d,c)$. Therefore, denote $pre(i,h,g)$ as the predecessor node of node i on $p_g^h(s,i)$ and $count(i,h,g)$ as the number satisfying that $p_g^h(s,i) = p^{h-1}_{count(i,h,g)}(s,pre(i,h,g)) + e(i,pre(i,h,g))$, i.e., $p_g^h(s,i)$ is the path constructed by concatenating the $count(i,h,g)$th shortest $(h-1)$-hop path from s to $pre(i,h,g)$ and the link $e(i,pre(i,h,g))$. Denote $v(i,h,k,\alpha)$ as the maximum number among all the $count(i,h,g)$, $g = 1,2,...,k$, that satisfy $pre(i,h,g) = \alpha$ (for the neighboring node α of i satisfying that for all $g = 1,2,...,k$, $pre(i,h,g) \neq \alpha$, $v(i,h,k,\alpha) = 0$). For instance, in the previous example, $v(d,h,3,a) = 2$, $v(d,h,3,b) = 0$, and $v(d,h,3,c) = 1$. Hence, it can be observed that $p_4^h(s,d) = \min\{p^{h-1}_{v(d,h,3,a)+1}(s,a) + e(d,a),\ p^{h-1}_{v(d,h,3,b)+1}(s,b) + e(d,b),\ p^{h-1}_{v(d,h,3,c)+1}(s,c) + e(d,c)\}$. Therefore, if all hops k-shortest paths have been computed, IAHKP computes the all hops $(k+1)$th shortest paths as follows:

1) $\forall i \in N$, $p^1_{k+1}(s,i) = \widehat{e}(s,i)$.
2) $p^h_{k+1}(s,i) = \min\{p^{h-1}_{v(i,h,k,\alpha)+1}(s,\alpha) + e(i,\alpha), \alpha = i_1, i_2, \ldots, i_{d_i}\}$.

Note that since IAHKP iteratively computes the all hops k-shortest paths, for any node and hop count h, the paths computed sequentially are in increasing order of their costs.

The relaxation procedure and pseudo codes of IAHKP are shown in Figs. 3 and 4, respectively, where $w(p)$ denotes the weight of path p and $\pi(i)$ the predecessor node of node i.

Computational Complexity: As mentioned above, $p_g^h(s,i)$ is computed by selecting the shortest path among the set of paths $\{p^{h-1}_{v(d,h,g,a)+1}(s,\alpha)+e(i,a), \alpha = i_1, i_2, \ldots, i_{d_i}\}$, whose computational complexity is $O(d_i)$. Hence, the computational complexity of computing the gth shortest h-hop paths for all nodes is $\sum_{i=1}^{N} O(d_i) = O(m)$. Since there are H hops and k shortest paths, the computational complexity of IAHKP is

$$O(kHm). \qquad (1)$$

```
IAHKP algorithm(G(N,E), s, t, d)
1  Initialize v(i, 2, 1, j) = 0 for any pair of $i, j \in N$
2  $\forall i \in N$, intialize $p_g^1(s,i)$, $g = 1,2,...,k$
3  for $u = 2$ to $k$
4    for $h = 2$ to $H$
5      for all $i \in N$
6        for all $e(i,j) \in E$
7          relax(i, j, u, h)
8        end for
9        $v(i,h,u,\pi(i)) = v(i,h,u,\pi(i)) + 1$
10     end for
11   end for
12 end for
```

Fig. 4. The pseudo code of IAHKP

Memory Complexity: We can divide memory complexity into two parts: the memory used to record the paths and the memory used in the process of computing. For any given node i, hop count h, and $1 \leq g \leq k$, define

- $n_0 = i$ and $g_0 = g$.
- $n_j = pre(n_{j-1}, h-j, g_{j-1})$ and $g_j = count(n_{j-1}, h-j, g_{j-1})$, $j \leq h$.

Hence, it can be observed that $p_g^h(s,i) = (s, n_{h-1}, n_{h-2}, ..., n_1, i)$, i.e., $p_g^h(s,i)$ sequentially traverses nodes $s, n_{h-1}, n_{h-2}, ..., n_1$, and i. Therefore, all paths can be backward reconstructed as long as for any node i, hop count h, and $1 \leq g \leq k$, $pre(i,h,g)$ and $count(i,h,g)$ are available. Hence, for a single node, the memory cost to record all k shortest h-hop paths is $O(k)$. Since there are H hops and n nodes, the first part of the memory cost is $O(kHn)$. As mentioned before, we use indicator arrays (of size n) to avoid loops, which contributes to the memory cost of the second part. The total memory cost used by indicator arrays is $O(Hn^2)$ for all nodes and the all hops gth shortest paths. Since there are k shortest path, the second part of the memory cost resulting from the indicators is $O(kHn^2)$. Combining memory costs mentioned above together, the memory complexity of our proposed algorithm is $O(kHn^2)$.

III. Proposed Routing Algorithm

In this section, based on IAHKP, a high performance QoS routing algorithm, which can achieve 100% success ratio in finding the delay constrained least cost path is proposed. Note that IAHKP is only applicable to the case in which there is only one weight associated with each link. Therefore, we first map the cost and delay of each link into a single weight with a weight function $f(d,c)$, where d denotes the delay of a link and c its cost, and then find the delay constrained least cost path by IAHKP. In this paper, we define the weight of a path as the sum of its link weights, i.e., given a path p and a weight function $f(d,c)$, the weight of p is $\sum_{e(u,v)\in p} f(d(u,v), c(u,v))$. The least cost path that satisfies the delay constraint is also referred to as the least cost feasible path and the optimal path. As a by-product of IAHKP,

$p_k^h(s,i) + e(i,t)$ is checked if it is a feasible path and has a weight less than that of the least cost feasible path computed so far. Hence, the computational complexity of IAHKP is not increased while the success ratio in finding the least weight feasible path is increased.

We divide our routing algorithm into two parts: forward IAHKP and backward IAHKP. We search for the feasible path from the source to the destination using the forward IAHKP, and reverse the search by the backward IAHKP. Therefore, the worst-case computational complexity of our proposed algorithm is only twice that of IAHKP. The weight functions used in both searches are different. Since we try to find the delay constrained least cost path, we first need to find out if there exists a path satisfying the delay constraint. Hence, the cost function used in the forward IAHKP is $f(d,c) = d$. Since IAHKP is capable of iteratively computing all hops k shortest path(s) between a source and a destination, no feasible path exists if no feasible path is found by the forward IAHKP, implying that even the least delay path has a delay larger than the delay constraint. Therefore, we terminate the search if we fail to find a feasible path in the first iteration of the forward IAHKP. If a feasible path is found in the forward IAHKP, we will try to minimize the cost in the second search. Therefore, the weight function adopted in the backward IAHKP is $f(d,c) = c$. If the least cost path in the backward IAHKP is a feasible path, it is definitely the least cost feasible path. Note that we can guarantee that the least cost delay constrained path has been found if the following two cases occur:

i If in the gth iteration of the forward IAHKP, all computed paths (the all hops gth shortest paths) are infeasible paths, implying that all feasible paths have been exhausted, the least cost delay constrained path must be the least cost feasible path among the all hops k shortest paths of the forward IAHKP, where $k < g$.

ii If in the gth iteration of the backward IAHKP, all computed paths have the costs larger than the cost, say c, of the least cost feasible path computed so far, implying that all the remaning paths having not been computed also have the costs larger than c, we are sure that the cost of the least cost feasible path is also c .

Since there are two IAHKP algorithms in our proposed routing algorithm, we refer to it as the Dual Iterative All Hops k-shortest Paths (DIAHKP) algorithm.

IV. Simulations

We propose two performance indices for the purpose of performance comparison. The first performance index, success ratio (SR), is proposed to evaluate the probability of an algorithm to locate the optimal feasible path.

$$SR = \frac{\text{The number of optimal paths of the algorithm}}{\text{The number of optimal paths of the optimal algorithm}}. \quad (2)$$

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

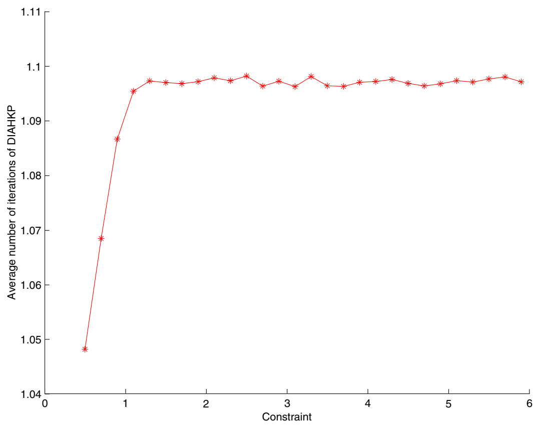


Fig. 5. Average number of iterations of DIAHKP in the 32-node network.

which is rather exhaustive. Many proposed algorithms can find a feasible path if a feasible path exists, but cannot guarantee that it is the optimal feasible path. Hence, we adopt another performance index, the cost ratio (CR), to evaluate how close the average cost of the solutions of an algorithm is to that of the optimal paths, which is defined below:

$$CR = \frac{\text{Average cost of the solutions}}{\text{Average cost of the optimal solutions}}. \quad (3)$$

Our simulations are divided into two parts. In both simulations, data are obtained by running 1,000,000 requests. The delay and cost of each link are independent and uniformly distributed from 0 to 1. The network topology is the 32-node network [16]. In our first simulation, we show that DIAHKP achieves 100% success ratio in finding the optimal feasible path in the 32-node network with the average computational complexity of only twice that of the standard Bellman-Ford algorithm. In our second simulation, by setting a small upper bound on the number of shortest paths ($k = 1, 2$), DIAHKP is compared with LARAC [9] and H_MCOP [6]. It was shown that our proposed algorithm outperforms LARAC [9] and H_MCOP [6] in terms of SR and CR.

A. Simulation 1:

Without setting an upper bound on k, implying DIAHKP is always capable of locating the optimal feasible path as long as feasible paths exit, we illustrate its average computational complexity in the 32-node network as shown in Fig. 5. It can be observed that it is very low: with the average computational complexity of only twice that of the IAHKP algorithm, we can achieve 100% success ratio in finding the optimal feasible path.

B. Simulations 2:

Setting a small upper bound on the number of iterations, we compare the performance of DIAHKP with that of LARAC [9] and H_MCOP [6]. Simulation results are shown in Figs.

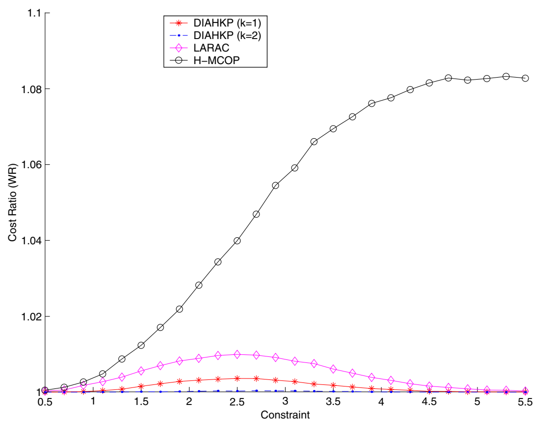


Fig. 6. The cost ratios of algorithms in the 32-node network.

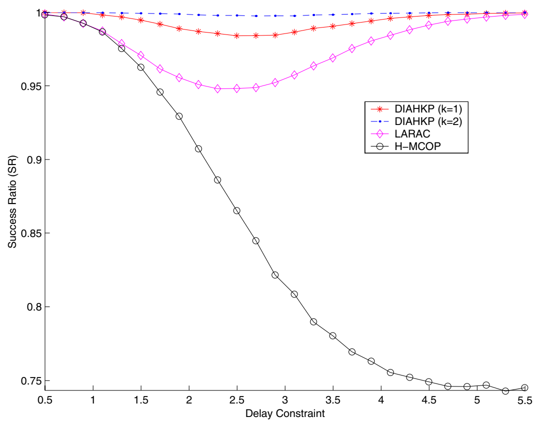


Fig. 7. The success ratios of algorithms in the 32-node network.

6 and 7. Note that the worst-case computational complexity of LARAC is $O(m^4 \log^4 m)$, which is definitely higher than that of DIAHKP when the number of shortest paths (k) of DIAHKP is only 2. Even with only one iteration, DIAHKP outperforms LARAC on both the average costs of the solutions, success ratio in finding the optimal feasible paths, and the worst-case computational complexity. When the number of shortest paths is 2, our proposed algorithm can achieve near 100% success ratio in finding the optimal feasible path. Note that although H_MCOP [6] has lower worst-case computational complexity than that of DIAHKP, its performance is not satisfactory. Moreover, it is generally believed that the standard Bellman-Ford algorithm has better performance than Disjktra algorithm in sparse networks, into which most communication networks can be classified [5]. Hence, since the average number of iterations of our proposed algorithm is only around 1, our proposed algorithm may achieve better performance than H_MCOP [6] on the success ratio in finding the optimal feasible path, the average cost of paths, and the average computational complexity in real communication networks.

V. Conclusions

Observing that the number of shortest paths is fixed in most proposed k-shortest-path-based routing algorithms, which inevitably unnecessarily waste the running time, we have proposed a high performance routing algorithm (Dual Iterative All Hops k-shortest Path selection (DIAHKP) algorithm) in this paper. DIAHKP can achieve 100% success ratio in finding the delay constrained least cost path. DIAHKP is based on IAHKP, a solution to the All Hops k-shortest Path selection (AHKP) problem. Extensive simulations show that DIAHKP is a practical, highly efficient, and flexible QoS routing algorithm, i.e., DIAHKP possesses the advantage of very low average computational complexity and 100% success ratio in finding the optimal feasible path. Even by setting a small upper bound on the number of iterations, DIAHKP still has very satisfactory performance. In the typical 32-node network, DIAHKP achieves 100% success ratio in finding the optimal feasible paths with the average computational complexity of only twice that of the standard Bellman-Ford algorithm.

References

[1] S. Chen and K. Nahsted, “An overview of quality of service routing for next-generation high-speed network: problems and solutions,” *IEEE Networks*, 1998, 12, (6), pp. 64-79.
[2] A. Shaikh, J. Rexford, and K. G. Shin, “Evaluating the impact of stale link state on quality-of-service routing,” *IEEE/ACM Transactions on Networking*, 2001, 9, (2), pp. 162-176.
[3] Z. Wang and J. Crowcroft, ”Quality of Service routing for supporting multimedia applications,” *IEEE Journal on Selected Areas on Communications*, 1996, 14, (7), vol. 14, pp. 1228-1234.
[4] X. Yuan, “Heuristic algorithm for multiconstrained quality-of-service routing, ” *IEEE/ACM Transactions on Networking*, 2002, 10, (2), pp. 244-256.
[5] A. Orda and A. Sprintson, “Precomputation schemes for QoS routing,” *IEEE/ACM Transactions on Networking*, 2003, 11, (4), pp. 578-591.
[6] T. Korkmaz and M. Krunz, “Routing multimedia traffic with QoS guarantees,” *IEEE Transactions on Multimedia*, 2003, 5, (3), pp. 429-443.
[7] G. Liu and K. G. Ramakrishnan, “A*Prune: an algorithm for finding K shortest paths subject to multiple constraints,” Proceedings of IEEE INFOCOM 2001, vol. 2, pp. 743-749, 2001.
[8] T. Korkmaz and M. Krunz, “Bandwidth-delay constrained path selection under inaccurate state information,” *IEEE/ACM Transactions on Networking*, 2003, 11, (3), pp. 384-398.
[9] A. Juttner, B. Szyiatovszki, I. Mecs, and Rajko, ”Lagrange releaxation based method for the QoS routing problem,” *Proceedings of IEEE INFOCOM 2001*, vol. 2, pp. 859-868, 2001.
[10] D. Eppstein, ”Finding the k shortest path,” *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, pp. 154-165, 1994.
[11] H. De Neve and P. Van Mieghem, ”A multiple quality of service routing algorithm for PNNI,” *Proceedings of 1998 IEEE ATM workshop*, pp. 324-328, 1998.
[12] D. H. Lorenz and A. Orda, ”Efficient QoS partition and routing of unicast and multicast,” *Proceedings of 8th International Workshop on Quality of Service*, pp. 75-83, 2001.
[13] R. Hassin, ”Approximation schemes for the restricted shortest path problem,” *Mathematics of Operations Research*, 1992, 2, (2), pp. 36-42.
[14] D. Raz, and Y. Shavitt, “Optimal partition of QoS requirements with discrete cost functions,” *IEEE Journal on Selected Areas in Communications*, 2000, vol. 12, (18), pp. 2593-2602.
[15] G. Cheng and N. Ansari, ”Finding All Hops k-shortest Paths,” *Proceedings of IEEE PACRIM’03*, vol. 1, pp. 474-477, 2003.
[16] S. Chen and K. Nahrsted, ”On finding multi-constrained path,” *Proceedings of IEEE ICC’98*, vol. 2, pp. 874-899, 1998.