Search Space Reduction in QoS Routing*

Liang Guo Ibrahim Matta
College of Computer Science
Northeastern University
Boston, MA 02115
{guol, matta}@ccs.neu.edu

Abstract

To provide real-time service, integrated networks require the underlying routing algorithm to be able to find low-cost paths that satisfy given Quality of Service (QoS) constraints. The problem of constrained shortest (least-cost) path routing is known to be NP-hard, and some heuristics have been proposed to find a near-optimal solution. However, these heuristics either impose relationships among the link metrics to reduce the complexity of the problem which may limit the general applicability of the heuristic, or are too costly in terms of execution time to be applicable to large networks. In this paper, we focus on solving the delay-constrained minimum-cost path problem, and present a fast algorithm to find a near-optimal solution. This algorithm, called DCCR (for Delay-Cost-Constrained Routing), is a variant of the k-shortest path algorithm. DCCR uses a new adaptive path weight function together with an additional constraint imposed on the path cost, to restrict the search space. Thus, DCCR can return a near-optimal solution in a very short time. Furthermore, we use the method proposed by Blokh and Gutin [1] to further reduce the search space by using a tighter bound on path cost. This makes our algorithm more accurate and even faster. We call this improved algorithm SSR+DCCR (for Search Space Reduction+DCCR). Through extensive simulations, we confirm that SSR+DCCR performs very well compared to the optimal but very expensive solution.

1 Introduction

The constrained shortest-path problem is encountered in many aspects of routing in integrated-services networks. For example, delay-sensitive applications, such as real-time voice and video, require traffic to be received at the destination within a given period of time. At the same time, it is highly desirable to reduce the path cost as much as possible; this could be monetary cost or the cost of utilizing network resources. However, this constrained shortest (least-cost) path problem, or in general the multi-constrained optimization path selection problem, is notoriously challenging and has been proved to be NP-hard [15, 9].

In this paper, we study the problem of finding a least-cost path subject to an end-to-end delay constraint in connection-oriented networks. This problem can be formulated as a *Delay-Constrained Least-Cost (DCLC)* unicast routing problem, or more generally, a constrained optimization problem. Widyono [17] proposed an optimal solution, namely the Constrained Bellman-Ford (CBF) algorithm, to solve this problem. The CBF algorithm performs a breadth-first search to find the optimal solution, thus its running time might grow exponentially in the worst case. In [1], Blokh and Gutin use a variant of the *Lagrangian relaxation method* to iteratively find a near-optimal solution. Although no analysis on the algorithm's complexity is given, this heuristic is found to practically find a near-optimal solution in a very short time. The algorithms in [12] and [14] try to compute the path distributively in order to alleviate the centralized computation overhead, however, paths returned by these algorithms may be costly, and the path setup time may be too long. The algorithm in [10] assumes that delay, delay-jitter and buffer space are functions of the available bandwidth, thus the routing algorithm can take advantage of these relationships to find a path in polynomial time. Some previous studies mainly focus on a related but possibly simpler problem — the *Multiple-Constraints Path (MCP)* problem. The difference between the MCP problem and the DCLC problem is that MCP doesn't optimize the value of any of the metrics, in-

*This work was supported in part by NSF grants CAREER ANIR-9701988 and MRI EIA-9871022, and NEU grant RSDF-377090.

stead, it only seeks a *feasible* path that satisfies all the constraints. Nevertheless, this problem is still NP-hard if more than one metric is additive and takes real values (or unbounded integer values) [15]. Jaffe [8] proposed a pseudo-polynomial heuristic and a polynomial-time heuristic for solving the MCP problem, given that the metrics have a small range of values. In [2], Chen and Nahrstedt try to reduce the problem's complexity by approximating the real values of link metrics by integer values and then use dynamic integer programming to solve it in polynomial time. However, to accurately find a near-optimal path, this algorithm has to resort to higher accuracy in approximating the values of link metrics, thus it becomes very costly in terms of space and time complexity. In [11], a non-linear function of link cost and delay is proposed to convert the problem into the much simpler single-metric routing problem, and so as to efficiently find a path that is far away from all the metric bounds.

Since the MCP problem seems to be easier than the DCLC problem and the heuristics to the former problem are generally more efficient in terms of execution time, it appears attractive to transform a DCLC problem to a MCP problem. Based on this premise, we propose a heuristic, called DCCR (for Delay-Cost-Constrained Routing), to rapidly generate a near-optimal delay-constrained path in large networks with asymmetric link metrics (delay and cost). This algorithm first *introduces* a cost bound according to the network state, then, it employs the k-shortest path algorithm proposed by Chong *et al.* [3] with a new non-linear weight function of path delay and cost to efficiently search for a path subject to *both* the requested delay constraint and the (introduced) cost constraint. Our weight function is designed to give more priority to lower cost paths. This algorithm is very similar to the TAMCRA algorithm proposed in [11], but we observe that our algorithm is more suitable for solving the DCLC problem since TAMCRA has a different objective, that of solving an MCP problem. Moreover, we also notice that using a tighter cost bound may help increase the accuracy and speed of the algorithm, thus, as an improvement, we further employ the algorithm proposed by Bolkh and Gutin [1] to refine our search space. We show by analysis that the complexity of this algorithm, called SSR+DCCR (for Search Space Reduction+DCCR), is asymptotically in the same order as a regular (unconstrained) single-metric shortest-path algorithm such as Dijkstra's algorithm [4]. Furthermore, through extensive simulations, we confirm that the cost of the path found by our SSR+DCCR algorithm is very close to that of the optimal path generated by the much more computationally expensive CBF algorithm.

The rest of this paper is organized as follows. Section 2 defines the DCLC problem. Section 3 describes our SSR+DCCR algorithm after motivating the design of our path weight function. We compare our SSR+DCCR algorithm with some other heuristics via simulations in Section 4. We conclude the paper in Section 5. Due to lack of space, we refer the reader to [6] for more details on the analysis and implementation of the algorithms.

2 Problem Description

The Delay-Constrained Least-Cost (DCLC) problem has been formulated in [12]. For completeness, we briefly restate it here. We represent the network by a directed graph $G = (V, E)$, where V is the set of all vertices (nodes), representing routers or switches, E is the set of edges (links) representing physical or logical connectivity between nodes. Each link is bidirectional, *i.e.*, the existence of a link $e = (u, v)$ from node u to node v implies the existence of another link $e' = (v, u)$ for any $u, v \in V$. Any link $e \in E$ has a cost $c(e) : E \mapsto \mathcal{R}^+$ and a delay $d(e) : E \mapsto \mathcal{R}^+$ associated with it, where $\mathcal{R}^+$ is the set of non-negative real numbers. The function $c(.)$ defines the measure we want to optimize (minimize). The function $d(.)$ defines the measure we want to constrain (bound). Due to the asymmetric nature of computer networks, it is possible that $c(e) \neq c(e')$ and $d(e) \neq d(e')$.

For a given source node $s \in V$ and destination node $d \in V$, $\mathcal{P}(s, d) = P_1, \cdots, P_m$ is the set of all possible paths from s to d. The cost and delay of a path P_i are defined as:

$$C(P_i) = \sum_{e \in P_i} c(e)$$

and

$$D(P_i) = \sum_{e \in P_i} d(e)$$

respectively. A delay constraint Δ_d is specified by the application as a performance guarantee. The DCLC problem thus can be formulated as follows:

Problem 1 (DCLC Path Problem) *Given a directed network G, a source node s, a destination node d, a non-negative link delay function $d(.)$, a non-negative link cost function $c(.)$ for each link $e \in E$ and a positive delay constraint Δ_d, the constrained minimization problem is to find a path that satisfies the following:*

$$\min_{P_i \in \mathcal{P}'(s,d)} C(P_i)$$

and

$$P_i \in \mathcal{P}'(s,d) \ \text{ iff } \ D(P_i) \leq \Delta_d$$

where $\mathcal{P}'(s,d) \subseteq \mathcal{P}(s,d)$ is the set of paths from s to d for which the end-to-end delay is bounded by Δ_d.

The DCLC problem can be more generally categorized as a *Constrained Optimization* problem, which involves optimizing one or more variables and imposing constraints on other variables. A variation of this problem, namely the Multi-Constraints Path (MCP) problem, only searches for a feasible solution for which all variables are bounded by the constraints. A special case of the MCP problem is the *Delay-Cost-Constrained (DCC)* problem which can be stated similarly as the DCLC problem except that the objective is to find a path $P_i \in \mathcal{P}'(s,d)$, where

$$P_i \in \mathcal{P}'(s,d) \ \text{ iff } \ D(P_i) \leq \Delta_d \ \text{ and } \ C(P_i) \leq \Delta_c$$

where Δ_c is the application specified cost bound.

Both the DCLC and DCC problems are NP-hard [5], however, since DCC does not involve optimization, it appears easier to find an efficient DCC heuristic. Thus, first converting a DCLC problem to a DCC problem may help to efficiently solve the original DCLC problem. This idea is applied in our DCCR algorithm.

3 Our SSR+DCCR Algorithm

3.1 Motivation

We convert the DCLC problem into a DCC problem by defining an appropriate cost bound for DCC so that the solution to the DCLC problem remains a feasible solution to the DCC problem. This could be easily achieved by using a sufficiently loose cost bound. In our algorithm, we solve a DCC problem, where we start with the least-delay path as a possible feasible solution. The cost of the least-delay path is selected as the cost bound. Indeed, if there is no feasible path with cost less than this, then the least-delay path itself must be the optimal path and this is what our algorithm returns. Thus, we can convert the DCLC problem into the problem of searching for a near-optimal path in the solution space of this new DCC problem.

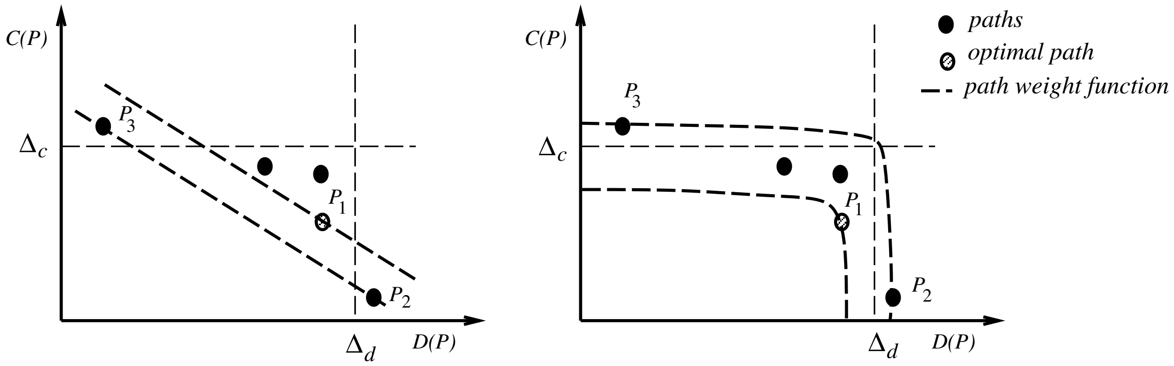
To search the solution space for the least-cost path, we need to examine the feasible paths for the DCC problem (i.e., those paths that satisfy both the requested delay bound and the introduced cost bound). For this purpose, we can use any well-known shortest-path algorithm (e.g. Dijkstra, Bellman-Ford [4]). But since these algorithms only deal with a single metric, we need to define a weight function which combines all features of the link metrics so that by optimizing (minimizing) the weight, we will finally find a solution that minimizes all link metrics simultaneously. A simple way to mix the metrics is to use a linear function, for example, $w(e) = \alpha c(e) + \beta d(e)$ as the new weight for each edge. This approach has the advantage that it is easy to implement since now the multiple constraints on path delay and cost become a single path weight constraint $\Delta = \alpha\Delta_c + \beta\Delta_d$. However, this linear weight function may not reflect the actual quality of a path, *i.e.*, an optimal path according to the new weight function may in fact violate the constraints while a suboptimal path satisfies them. Figure 1 shows why a linear function may not work.

We can see from Figure 1(a) that although the delay of path P_2 violates the delay bound, it still has the least weight since its cost is relatively low, thus, the search process may miss the actual optimal *and* feasible path P_1. We can also see from the illustration that for the DCLC problem, the feasible path P_3 may be returned by the linear weight based algorithm as it has a relatively low delay, and we miss the actual optimal (least-cost) path P_1.

Using a non-linear function may help to overcome this difficulty. Neve *et al.* propose to use the concave path weight function $\max(C(P)/\Delta_c, D(P)/\Delta_d)$ in their TAMCRA algorithm [11]. It is shown that with this function, the algorithm can find the shortest path (path whose *both* cost and delay are far from the bounds) with a relatively high successful rate. Figure 1(b) illustrates the advantage of using a non-linear weight function over a linear one. Now, the search process would not return P_2 nor P_3 since they have high delay and high cost respectively, resulting in a high weight value.

The problem with defining a non-linear weight function for a link is that now the weight of a path is no longer the sum of the weight of all links on this path, *i.e.*, $W(P) \neq \sum_{e \in P} w(e)$. But since it is easy to record the cumulative delay and cumulative cost of a path, we can easily solve this problem by computing the path weight as a function $\mathcal{F}(.)$ of the delay and cost of the path (rather than the sum of link weights), *i.e.*, $W(P) = \mathcal{F}(C(P), D(P))$.

A more serious problem is that a non-linear function does not have the *optimal-substructure property* [4], *i.e.* subsections of shortest paths are not necessarily shortest paths themselves. Therefore, a shortest-path algorithm like Dijkstra's will sometimes fail to find the shortest (least-weight) path. Consider the following example shown in Figure 2 assuming a concave (max) weight function is used.



(a) Linear Weight Function *(b) Non−linear Weight Function*

Figure 1. Why a linear function fails?

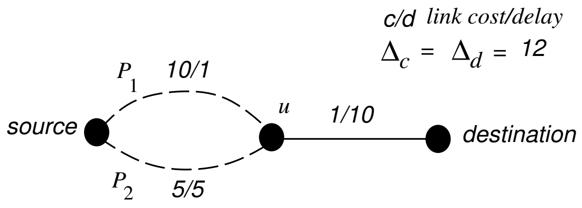


Figure 2. Problem with a non-linear function.

For intermediate node u, path P_2 will be chosen since it has a smaller weight[1], thus the actual feasible path to the destination through P_1, with feasible delay and cost of 11, will be missed. Neve *et al*. solves this problem by taking advantage of the k-shortest path algorithm proposed by Chong *et al.* [3], which can store k shortest paths in increasing weight order at each node. Thus with an appropriate value of k, the algorithm can almost always find the least-weight and feasible path.

The non-linear (max) weight function in TAMCRA works well so as to find a path that is far from all the bounds. It is not a goal of TAMCRA to optimize any of the metrics. However, since now our objective is to find a path with least cost, this function is no longer suitable since it treats all link measures equally. Instead, we should use a weight function that gives priority to low-cost paths. The weight function used in our algorithm is defined as:

$$W(P_i^u) = \begin{cases} \frac{D(P_i^u)}{1-C(P_i^u)/\Delta_c} & \text{if } D(P_i^u) \leq \Delta_d \text{ and } C(P_i^u) \leq \Delta_c \\ \infty & \text{otherwise} \end{cases}$$

where $P_i^u \in \mathcal{P}(s,u)$ is the ith path from source node s to node u found by the algorithm. With our definition, the path weight has an exponential growth with the path cost, and is only linearly proportional to the path delay.

[1] $W(P_1) = max(10/12, 1/12) = 10/12$, $W(P_2) = max(5/12, 5/12) = 5/12$.

3.2 Improvement to DCCR algorithm

Recall that our DCCR algorithm restricts the search space by only examining paths that satisfy the requested delay bound *as well as* a cost bound. The cost bound is taken to be that of the least-delay path. This is reasonable since if there is no path with lower cost than that of the least-delay path, then the least-delay path itself is the optimal path, and this is the path returned by DCCR. However, this cost bound may be too loose, especially when the relationship between cost and delay is negative, *i.e.*, the lower the delay, the higher the cost and vice versa. Since we set the weight of all infeasible paths to be infinity, it is easy to see that if we use a tighter cost bound, the number of possible feasible solutions decreases, and thus the opportunity that our algorithm finds the optimal (least-cost) solution increases. Another advantage of using a tighter cost bound is that since the success rate becomes higher, only a small value of k would work well, thus the speed of the algorithm is also enhanced.

To search for a tighter cost bound, we use another heuristic to the DCLC problem as a prelude to our DCCR algorithm. This heuristic, called the BG algorithm, was proposed by Blokh and Gutin [1]. It still uses a linear function of the link delay and cost to compute link weight. A salient feature of it is that it adjusts the weights given to cost and delay in the weight function according to the quality of the current path, thus it iteratively approaches the optimal (least-cost) solution.

The path found by the BG algorithm may still not be the optimal path due to the inherent weakness of the linear weight function (cf. Figure 1). But its cost is close enough to the optimal cost to be effectively used as a tight cost bound for DCCR. We denote this improved algorithm by SSR+DCCR since using a tighter cost bound is a mechanism for Search Space Reduction.

4 Simulation Model and Results

4.1 Simulation Model

We built a discrete-event simulator to investigate the performance of different algorithms in a realistic communication environment. We use the same graph generation process as in [13] where the positions of the nodes lie in a rectangular area. A random generator based on Waxman's generator [16] is used to create links interconnecting the nodes. Some modifications are added to ensure that the generated network is connected and the probability of existence of a short link is larger than that of a longer link. We fixed the position of the source node and the destination node such that the *Manhattan distance* between them is the longest possible distance in the graph. The average node degree is set to 4, which is approximately what the situation is in current networks.

The link delay function consists of the propagation delay T_p, the transmission delay T_t and the queuing delay T_q. Since we are considering high-speed links, transmission delay is assumed negligible. Denote by $\tau = T_q/T_p$ the ratio between the queuing delay and propagation delay; this parameter reflects how busy the communication link is. Thus, the link delay is defined as:

$$d(e) = (1 + \tau) * T_p$$

In our simulation model, we let τ be uniformly distributed in $[0, T]$, where T is a parameter that reflects the maximum queueing delay allowed at each switch. Also, the larger the value of T is, the more likely the generated network is asymmetric. We set T to be 10.0 in our experiments.

The way to generate the link cost can affect the difficulty in finding the optimal path. If there is a *positive correlation* between link cost and delay, *i.e.* the higher the link delay is, the more costly the link is, then it is enough to just use a single-metric shortest-path algorithm since faster paths are also likely to be cheaper. Thus, in our simulation model, we consider the most difficult situation where a *negative correlation* exists between cost and delay. Link cost is thus defined as:

$$c(e) = M/(c + d(e))$$

where M and c are parameters chosen so as to adjust the value of $c(e)$ within a reasonable range. We choose $M = 1000$ and $c = 1$ in our simulations, and $d(e)$ varies from 0.1 to 20.

Since the tightness of the delay bound might affect the performance of the algorithms under investigation, we choose the delay bound based on the configuration of the graph. Each time a new graph is generated, we first use Dijkstra's algorithm to find the least-delay path and least-cost path, then compute the delay of these two paths, denoted by $D(LDP)$ and $D(LCP)$ respectively. We then define the delay bound Δ_d to be:

$$\Delta_d = D(LDP) + \rho(D(LCP) - D(LDP))$$

where $\rho \in [0, 1]$ is called the *delay bound ratio* [7] and reflects the tightness of the delay bound. In most of our experiments, ρ is set to 0.5.

We assume a link-state type routing, where the routing nodes have complete knowledge of the state of the entire network, and the state information is accurate (up-to-date). The network size is set to 200, 500, 1000, and 2000. 500 executions on different networks are conducted for each experiment, and 95% confidence intervals were computed for all performance measures. We choose $k = 3$ and $m = 5$ for all network sizes, where k is the number of shortest (least-weight) paths maintained from the source to each node, and m is the number of BG iterations executed to compute a tight cost bound for DCCR. Note that k and m are much smaller than the network size, but we will see shortly that even such a small value is enough to get good performance.

4.2 Performance Metrics

Two performance metrics are used to measure the inefficiency (inaccuracy) and speed of the heuristics. As mentioned earlier, the CBF algorithm provides the optimal solution in terms of path cost. Thus we define the *inefficiency* of an algorithm $\mathcal{A}$ as the path cost difference relative to the cost of the CBF path:

$$inefficiency_{\mathcal{A}} = \frac{(C(P_{\mathcal{A}}) - C(P_{CBF}))}{C(P_{CBF})}$$

We also measure the *actual execution time* of each investigated algorithm. The experiments were conducted on a SUN Ultra 10 workstation, and the codes for all the simulated algorithms were not optimized for speed.

4.3 Simulation Results

Figure 3 shows the performance measures of different heuristics for different network sizes[2]. We can see

[2] We use log scale for the cost axis due to the huge difference between the algorithms under investigation.

that with negative correlation between link cost and delay, the least-delay path can cost as high as three times (200% more than) that of the optimal path, and the cost of the path returned by the TAMCRA algorithm is nearly 50% more than the optimal cost. The cost of the BG path and DCCR path is only about 5% higher than the optimal cost. The improved SSR+DCCR algorithm, as a combination of BG and DCCR, shows a very attractive cost performance; the relative excess cost always remains under 1%. We can also see that the relative order and the scale of cost difference does not change much with the network size. Considering that we set $k = 3$ for all network sizes, we can then argue that k **can be kept small even for a very large network**. As for the execution time, Figure 3(b) shows the data for all heuristics except the CBF algorithm. The LDP runs the fastest. Then comes the BG algorithm, which implies that practically, BG can converge very fast to the final solution even though an analytical bound does not exist. The speed of TAMCRA and DCCR is about the same because they take almost the same steps except in computing the weight function. What is surprising is that the improved SSR+DCCR algorithm, *i.e.* the combination of BG and DCCR, runs in almost the same speed as the original DCCR algorithm. The speed of all of the above 3 algorithms, TAMCRA, DCCR and SSR+DCCR, is about 4 times that of the LDP algorithm. The SSR+DCCR algorithm was found to be orders of magnitude faster than the optimal CBF algorithm [6].

To see the role k plays in the performance of the heuristics that use the k-shortest path method, we conduct another experiment whose results are shown in Figure 4. Increasing k's value results in more candidate paths being examined, thus both heuristics, namely DCCR and TAMCRA, return a cheaper path as k becomes larger. However, this performance improvement is offset by a large increase in execution time. Since in SSR+DCCR, we already use the BG heuristic to find a tighter cost bound, there aren't many feasible paths left in the solution space, thus, a small k is enough to find a good (low-cost) path.

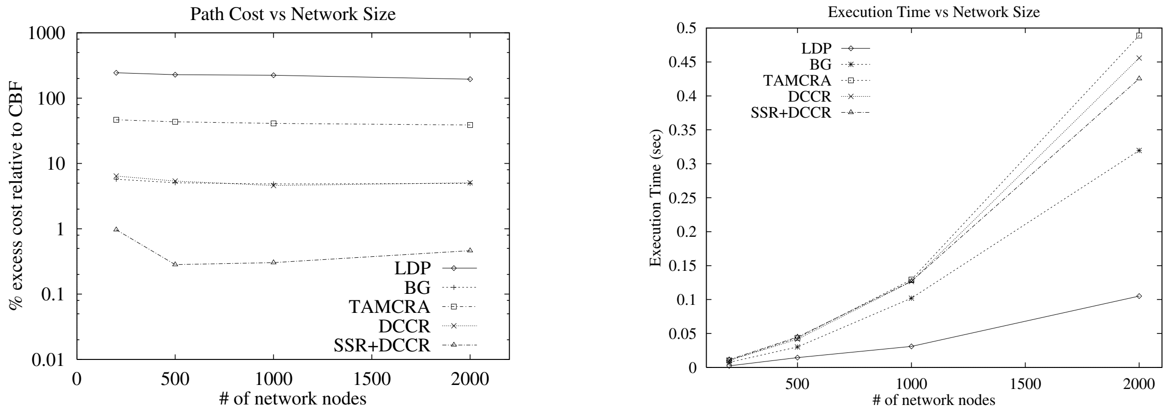
Figure 5 shows the effect of the delay bound on the performance. We can see that the relative excess cost of BG, TAMCRA, and DCCR, is increasing as the delay bound gets looser. This is because a looser bound will enlarge the solution space, thus the capability of these algorithms becomes limited by either the weakness of the linear weight function (cf. Figure 1) or by the fixed value of k (cf. Figure 4). On the contrary, the performance of SSR+DCCR appears to be less sensitive to the delay bound. As analyzed earlier, this is because the cost bound given by the BG heuristic is already tight enough to restrict the number of feasible paths.

All the above experiments assume that the link cost and link delay are negatively correlated. This assumption is valid for some networks, and increases the difficulty in finding the optimal path. We also note that in some cases, the link cost may not have any relationship with the link delay. Thus, in the next experiment, we assume that link cost is a random number, and is not correlated to link delay. Figure 6 shows the performance of all the investigated algorithms for different network sizes. The relative order of the heuristics remains the same, *i.e.* SSR+DCCR performs the best, followed by DCCR, BG, then TAMCRA. LDP performs the worst. The difference here is that now BG performs much closer to TAMCRA, and the improvement made to DCCR in SSR+DCCR is not so significant. However, we still confirm that the cost of the path returned by SSR+DCCR is very close to the optimal cost.

5 Conclusions

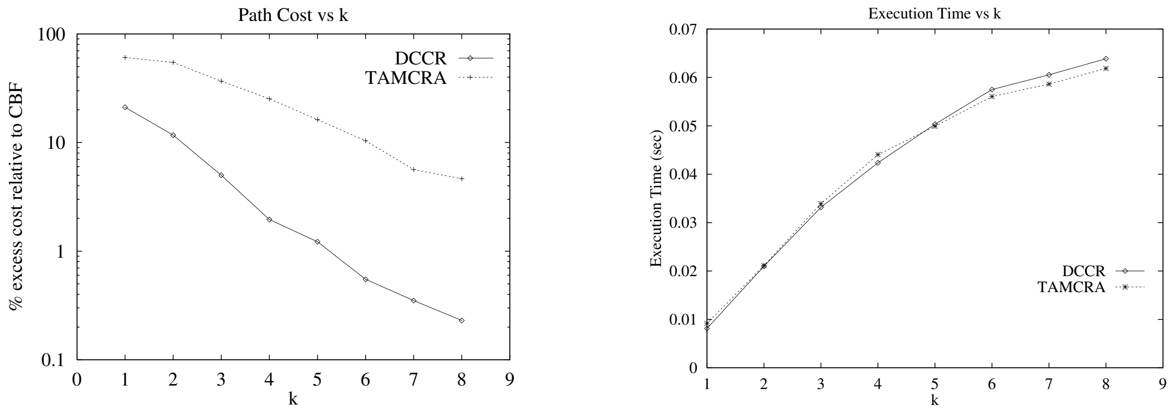
An efficient algorithm for obtaining a Delay-Constrained Least-Cost (DCLC) path is presented in this paper. This algorithm uses a non-linear path weight function and applies a k-shortest path heuristic to make the path search more accurate and faster. To further enhance the accuracy of the algorithm, we also propose to use another DCLC heuristic as a prelude to reduce the solution/search space, thus trading some extra execution time for a more accurate search. Results from extensive simulations show that even under the most difficult situation, *i.e.*, when link cost and link delay are negatively correlated, our improved SSR+DCCR algorithm always returns very quickly a feasible path whose cost is very close to that of the optimal one, which could only be found using a computationally prohibitive search method.

Our SSR+DCCR algorithm could be applied in multicast routing protocols to build a low-cost multicast tree. Since it is common that the membership of a multicast group is dynamic, and the network state (link delays and costs) is also dynamic, it is very hard, if not impossible, to maintain all the time an optimal cost multicast tree that also satisfies given performance (delay) constraints. One possible solution to this problem is to, whenever a new group member joins or an existing member becomes out-of-bound, add or replace the old path with a new delay-bounded path. Thus, reducing the cost of this delay-bounded path can further reduce the cost of the whole tree. We will investigate this approach in our future work.



(a) (b)

Figure 3. Path cost and execution time versus network size: average node degree=4.0, ρ = 0.5, T = 10.0, negative correlation between cost and delay.



(a) (b)

Figure 4. Path cost and execution time versus k: network size = 500, average node degree = 4.0, ρ = 0.5, T = 10.0, negative correlation between cost and delay.

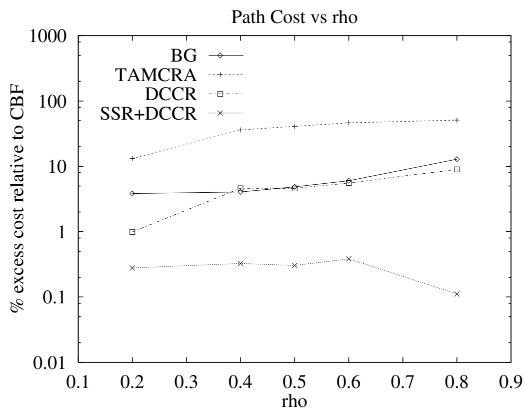
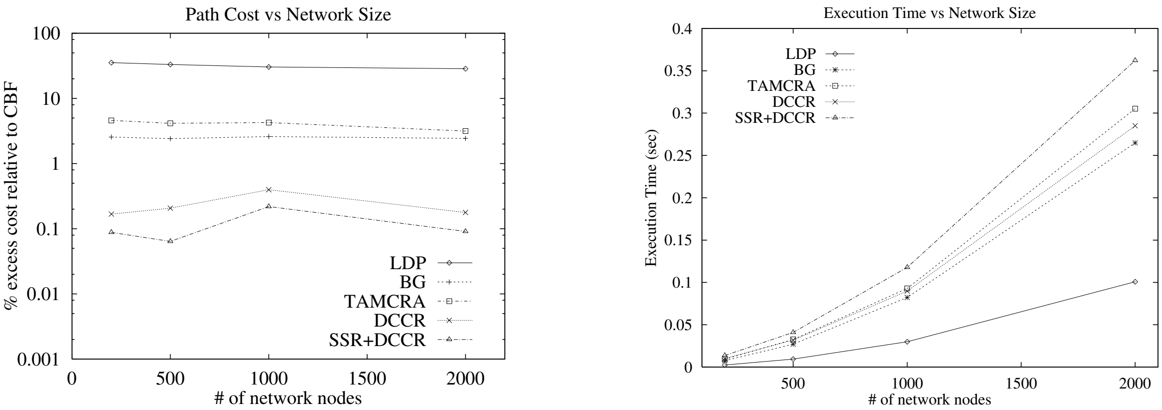


Figure 5. Path cost versus ρ: network size = 1000 nodes, average node degree = 4.0, T = 10.0, negative correlation between cost and delay.



(a) (b)

Figure 6. Path cost and execution time versus network size: average node degree=4.0, ρ = 0.5, T = 10.0, cost and delay are independent.

References

[1] D. Blokh and G. Gutin. An Approximation Algorithm for Combinatorial Optimization Problems with Two Parameters, 1995. manuscript.

[2] S. Chen and K. Nahrstedt. On Finding Multi-constrained Paths. In Proc. *ICC '98*, Atlanta, Georgia, 1998.

[3] E.I. Chong, S. Maddila, and S. Morley. On Finding Single-Source Single-Destination k Shortest Paths. In Proc. *International Conference on Computing and Information (ICCI) '95*, pages 40–47, Ontario, Canada, July 1995.

[4] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* The MIT Press and McGraw-Hill Book Company, 1990.

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

[6] L. Guo and I. Matta. Search Space Reduction in QoS Routing. Technical Report NU-CCS-98-09, College of Computer Science, Northeastern University, October 1998. Available from http://www.ccs.neu.edu/home/matta/publications.html.

[7] S.P. Hong, H. Lee, and B.H. Park. An Efficient Multicast Routing Algorithm for Delay-Sensitive Applications with Dynamic Membership. In Proc. *INFOCOM'98*, San Francisco, CA, March 1998.

[8] J.Jaffe. Algorithms for Finding Paths with Multiple Constraints. *Networks*, pages 95–116, Spring 1984.

[9] D.H. Lorenz and A. Orda. QoS Routing in Networks with Uncertain Parameters. In Proc. *IEEE INFOCOM '98*, San Francisco, CA, March 1998.

[10] Q. Ma and P. Steenkiste. Quality-of-Service Routing for Traffic with Performance Guarantees. In Proc. *IFIP Fifth International Workshop on Quality of Service*, pages 115–126, Columbia University, New York, May 1997.

[11] D. Neve and Miegeham. A Multiple Quality of Service Routing Algorithm for PNNI. In Proc. *IEEE ATM '98 Workshop*, pages 306–314, Fairfax, Virginia, May 1998.

[12] H.F. Salama, D.S. Reeves, and Y. Viniotis. A Distributed Algorithm for Delay-Constrained Unicast Routing. In Proc. *IEEE INFOCOM'97*, Japan, April 1997.

[13] H.F. Salama, D.S. Reeves, and Y. Viniotis. Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks. *IEEE J. Select. Areas Commun.*, 15:332–346, April 1997.

[14] Q. Sun and H. Langendorfer. A New Distributed Routing Algorithm with End-to-End Delay Guarantee. In Proc. *IFIP Fifth International Workshop on Quality of Service*, Columbia University, New York, May 1997.

[15] Z. Wang and J. Crowcroft. Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE J. Select. Areas Commun.*, 14(7):1188–1234, Sepetember 1996.

[16] B.M. Waxman. Routing of Multipoint Connections. *IEEE J. Select. Areas Commun.*, pages 1617–1622, December 1988.

[17] R. Widyono. The Design and Evaluation of Routing Algorithms for Real-Time Channels. Technical Report ICSI TR-94-024, International Computer Science Institute, U.C. Berkeley, June 1994.