ISSN: 2278-0181
NCRTS-2015 Conference Proceedings

A Survey Paper on Efficient Utilization of Shortest Path Computation

Abhilasha S N[1]
2nd sem M-Tech,CSE
City Engineering College,
Bangalore

Nandhish A C[2]
P.G.Co-ordinator,Dept.of.CSE
City Engineering College,
Bangalore

Abstract--Computing constrained shortest paths is fundamental to some important network functions such as QoS routing, which is to find the cheapest path that satisfies certain constraints. In particular, finding the cheapest delay-constrained path is critical for real-time data flows such as voice calls. Because it is NP-complete, much research has been designing heuristic al-gorithms that solve the ε-approximation of the problem with an adjustable accuracy. A common approach is to discretize (i.e., scale and round) the link delay or link cost, which transforms the original problem to a simpler one solvable in polynomial time. The efficiency of the algorithms directly relates to the magnitude of the errors introduced during discretization. In this paper, we propose two techniques that reduce the discretization errors, which allows faster algorithms to be designed. Reducing the overhead of the costly computation for constrained shortest paths is practically important for the design of a high-throughput QoS router, which is limited by both processing power and memory space. Our simulations show that the new algorithms reduce the execution time by an order of magnitude on power-law topologies with 1000 nodes. The reduction in memory space is similar. When there are multiple constraints, the improvement is more dramatic.

I. INTRODUCTION

A major obstacle against implementing distributed multi- media applications (e.g., web broadcasting, video teleconfer- encing, and remote diagnosis) is the difficulty of ensuring QoS (Quality of Service) over the Internet. Besides the issues of packet scheduling, admission control, resource reservation, and traffic engineering, the QoS routing is a critical element for QoS provision. It is to find a constrained shortest path — a network path that satisfies a given set of constraints (e.g., minimum bandwidth requirement and bounded end-to- end delay) [1], [2]. For interactive real-time traffic, the delay- constrained least-cost path has particular importance. It is the cheapest path whose end-to-end delay is bounded by the delay requirement of a time-sensitive data flow such as a voice call. The additional bandwidth requirement, if there is one, can be easily handled by a pre-processing step that prunes the links without the required bandwidth from the graph.

A path that satisfies the delay requirement is called a *feasible path*. Finding the cheapest (least-cost) feasible path is NP-complete. There has been considerable work in designing heuristic solutions for this problem. Let m and n be the number of links and the number of nodes in the network, respectively. Juttner et al. applied the Lagrange relaxation method on the delay-constrained least-cost routing problem with a time complexity $O(m^2 \log^4 m)$ [4]. There is no theo- retical bound on how large the cost of the found path will be, comparing with the optimal path. Korkmaz and Krunz proposed a heuristic algorithm with the same time complexity as Dijkstra's algorithm [5]. However, it does not provide a theoretical bound on the property of the returned path, nor pro- vide conditional guarantee in finding a feasible path when one exists. In addition, because the construction of the algorithm ties to a particular destination, it is not suitable for computing constrained paths from one source to all destinations. For this task, it is slower than the algorithms proposed in this paper by two orders of magnitude based on our simulations.

Another thread of research in this area is to design poly- nomial time algorithms that solves the NP-complete problem with an accuracy that is theoretically bounded. Given a small constant ε, Hassin's algorithm [6] has a time complexity of $O(\frac{mn}{\varepsilon} \log \log \frac{UB}{LB})$, where UB and LB are the costs of the fastest path and the cheapest path from the source node to the destination node, respectively. The algorithm finds a feasible path if there exists one. The cost of the path is within the cost of the cheapest feasible path multiplied by $(1 + \varepsilon)$. Chen and Nahstedt solved a similar problem in time $O((m + n \log n)x)$, where $x = O(n/\varepsilon)$ in order to achieve the ε-approximation [7]. Goel et al.'s algorithm [8] has the best-known complexity of $O((m + n \log n) \frac{L}{\varepsilon})$, where L is the length (hops) of the longest path in the network. It computes a path whose cost is no more than the cost of the cheapest feasible path, while the delay of the path is within $(1 + \varepsilon)$ of the delay requirement.

One common technique of the above algorithms [6]–[8] is to discretize the link delay (or link cost). Due to the discretization, the possible number of different delay values (or cost values) for a path is reduced, which makes the problem solvable in polynomial time. The effectiveness of this technique depends on how much error is introduced during the discretization. The existing approaches either generate positive discretization errors for all links or generate negative errors.

for all links. Therefore, the discretization error on a path is statistically proportional to the path length as the errors on the links along the path add up. In order to bound the maximum error, the discretization has to be done at a fine level, which leads to high execution time of the algorithms.

Given the limited resources and ever-increasing tasks of the routers, it is practically important to improve the efficiency of network functions, particularly, the efficiency of expensive operations such as computing the constrained shortest paths. In this paper, we propose two techniques, *randomized discretiza- tion* and *path-delay discretization*, which reduce the discretiza- tion errors and allow faster algorithms to be designed. The ran- domized discretization cancels out the link errors along a path. The larger the topology, the greater the error reduction. The statistic mean of the discretization error on a path P is zero and the standard deviation is proportional to $\sqrt{l(P)}$, where $l(P)$ is the length of P. The path-delay discretization works on path delays instead of individual link delays, which eliminates the problem of error accumulation. Based on these techniques, we design fast algorithms for the constrained shortest-path problem. We prove the correctness of the algorithms, and demonstrate their efficiency by simulations.

II. PROBLEM DEFINITION AND EXISTING DISCRETIZATION APPROACHES

Consider a network $G\langle V, E\rangle$, where V is a set of n nodes and E is a set of m directed links connecting the nodes.

The delay and the cost of a link $(u, v) \in E$ are denoted as $d(u, v)$ and $c(u, v)$, respectively. The delay and the cost of a path P are denoted as $d(P)$ and $c(P)$, respectively. $d(P) = \sum_{(u,v)\in P} d(u,v)$, and $c(P) = \sum_{(u,v)\in P} c(u,v)$. Let $l(P)$ be the length (number of hops) of P, and L be the length of the longest path in the network.

Given a delay requirement r, P is called a *feasible path* if $d(P) \le r$. Given a source node s, let V_s be the set of nodes to which there exist feasible paths from s. For any $t \in V_s$, the *cheapest feasible path* $P_{s,t}$ from s to t is defined as

$$d(P_{s,t}) \le r$$

$$c(P_{s,t}) = \min\{c(P) \mid d(P) \le r, \forall \text{path } P \text{ from } s \text{ to } t\}$$

The *delay-constrained least-cost routing* problem (DCLC) is to find the cheapest feasible paths from s to all nodes in V_s, which is NP-complete [9]. However, if the link delays are all integers and the delay requirement is bounded by an integer λ, the problem can be solved in time $O((m + n \log n)\lambda)$ by Joksch's dynamic programming algorithm [10] or the extended Dijkstra's algorithm [7].

$\forall v \in V, i \in [0..\lambda]$, let $w[v, i]$ be a variable storing the cost of the cheapest path P from s to v with $d(P) \le i$, and $\pi[v, i]$ storing the last link of the path. Initially, $w[v, i] = \infty$, $\forall v = s$, and $w[s, i] = 0$. $\pi[v, i]$ = NIL. Assuming that all link delays are positive integers, Joksch's dynamic programming algorithm can be described as follows.

$$w[v,i] = \min\{w[v,i-1],\ w[u,i-d(u,v)] + c(u,v),\ \forall(u,v) \in E,\ d(u,v) \le i\}$$

Goel et al. enhenced the approach by allowing zero link delays [8]. Let G_Z be the subgraph consisting of all zero-delay links. For each $i \in [0..\lambda]$, immediately after Joksch's algorithm calculates $w[v, i]$, $\forall v \in V$, Dijkstra's algorithm is executed on G_Z to improve $w[v, i]$ on zero-delay paths.

The above integer-delay special case points out a heuristic solution for the general NP-complete problem, which is to discretize (scale and then round) arbitrary link delays to integers [6]–[8], [11]. There are two existing discretization approaches, *round to ceiling* [7] and *round to floor* [8]. Both approaches map the delay requirement r to a selected integer λ, while the link delays are discretized as follows.

Round to ceiling (RTC): For every link (u, v), the delay value is divided by $\frac{r}{\lambda}$. If the result is not an integer, it is rounded to the nearest larger integer.

$$d^c(u,v) = \left\lceil \frac{d(u,v)}{r}\lambda \right\rceil \qquad (1)$$

Round to floor (RTF): For every link (u, v), the delay value is divided by $\frac{r}{\lambda}$. If the result is not an integer, it is rounded to the nearest smaller integer.

$$d^f(u,v) = \left\lfloor \frac{d(u,v)}{r}\lambda \right\rfloor \qquad (2)$$

The discretization error of a link (u, v) is defined as

$$\Delta^f(u,v) = d(u,v) - d^f(u,v)\frac{r}{\lambda} \qquad (3)$$

$$\Delta^c(u,v) = d(u,v) - d^c(u,v)\frac{r}{\lambda} \qquad (4)$$

The discretization error of a path P is defined as

$$\Delta^f(P) = \sum_{(u,v) \text{ on } P} \Delta^f(u,v) \qquad (5)$$

$$\Delta^c(P) = \sum_{(u,v) \text{ on } P} \Delta^c(u,v) \qquad (6)$$

With either Joksch's algorithm or Geol's algorithm, both RTC and RTF can solve the ε-approximation of DCLC, which is to find a path P for every node $t \in V_s$, such that

$$d(P) \le (1+\varepsilon)r$$

$$c(P) \le c(P_{s,t})$$

where ε is a small percentage. The delay of the path is allowed to exceed the requirement by a percentage of no more than ε, while the cost should be no more than that of the cheapest feasible path $P_{s,t}$. Using RTF, the delay scaling algorithm (DSA) proposed by Goel et al. achieves the best time complexity $O((m + n \log n)L/\varepsilon)$ among all existing algorithms [8].

III. RANDOMIZED DISCRETIZATION

RTC creates positive rounding error on every link. The error accumulates along a path. The larger the topology, the longer a path, the larger the accumulated error. The same thing is true for RTF, which has negative rounding error on every link. The insight is that if we can reduce the error introduced by discretization, we can improve the performance of the algorithm. With a smaller error, the new problem after discretization is closer to the original problem. The solution to the new problem will also be closer to the solution of the original problem.

Our first approach is randomized discretization. It rounds to ceiling or to floor according to certain probabilities. The idea is for some links to have positive errors and some links to have negative errors. Positive errors and negative errors cancel out one another along a path in such a way that the accumulated error is minimized statistically.

Round randomly (RR): For every link (u, v), the delay value is divided by $\frac{r}{\lambda}$. If the result is not an integer, it is rounded to the nearest smaller integer or to the nearest larger integer randomly such that the mean error is zero.

$$d^r(u,v) = \begin{cases} \left\lceil \frac{d(u,v)}{r}\lambda \right\rceil & \text{with prob. } p_1 = \frac{d(u,v)}{r}\lambda - \left\lfloor \frac{d(u,v)}{r}\lambda \right\rfloor \\ \left\lfloor \frac{d(u,v)}{r}\lambda \right\rfloor & \text{with prob. } p_2 = 1 - p_1 \end{cases} \quad (7)$$

The discretization error of a link (u, v) is

$$\Delta^r(u,v) = d(u,v) - d^r(u,v)\frac{r}{\lambda} \quad (8)$$

and the discretization error of a path P is

$$\Delta^r(P) = \sum_{(u,v) \text{ on } P} \Delta^r(u,v) = d(P) - d^r(P)\frac{r}{\lambda} \quad (9)$$

Following the iterative approach of [8], the randomized discretization algorithm (RDA) is described below. Let λ_0 be a small constant. We use the extended Dijkstra's shortest path algorithm (EDSP), which is equivalent to Joksch's algorithm, except that $w[v, i]$ stores the cost of the cheapest path P from s to v with $d^r(P) = i$.

The algorithm assumes a preprocessing step that removes all nodes to which there are no feasible paths from s.

Initialize(V, s, λ)

for each vertex $v \in V$, each $i \in [0..\lambda]$ **do**
    $w[v, i] := \infty$, $\pi[v, i] :=$ NIL, $\delta[v, i] := \infty$
$w[s, 0] := 0$, $\delta[s, i] := 0$

Relax_RDA(u, v, i, λ)
$i' := i + d^r(u, v)$
$error := \delta[u, i] + \Delta^r(u, v)$
if $error < 0$ **then**
    $error := error + r/\lambda$
    $i' := i' - 1$
if $i' \le \lambda$ and $w[v, i'] > w[u, i] + c(u, v)$ **then**
    $w[v, i'] := w[u, i] + c(u, v)$
    $\pi[v, i'] := u$
    $\delta[v, i'] := \min\{\delta[v, i'], error\}$

EDSP_RDA(G, s, λ)
Initialize(V, s, λ)
for $i = 0$ to λ **do**
    $Q := V$
    while $Q \neq \emptyset$ **do**
        $u :=$ Extract Min(Q)
        $Q := Q - \{u\}$
        for every adjacent node v of u **do**
            Relax_RDA(u, v, i, λ)

RDA(G, s)
$\lambda := \lambda_0$
do
    $\lambda := 2\lambda$
    EDSP_RDA(G, s, λ)
while $\exists v \in V$, $d(P^v) > (1 + \epsilon)r$,

where P^v is the path with $\min\{w[v, i] \mid i \in [0..\lambda]\}$

Due to space limit, we omit all proofs.

Lemma 1: It always holds that $\delta[u, i] \ge 0$, $\forall u \in V, i \in [0..\lambda]$.

Lemma 2: Let P^u_i be the path stored by $\pi[u, i]$. It always holds that $d(P^u_i) \ge i\frac{r}{\lambda} + \delta[u, i]$, $\forall u \in V, i \in [0..\lambda]$.

Lemma 3: Let P^u_i be the path stored by $\pi[u, i]$. It always holds that $d(P^u_i) \le (i + l(P^u_i))\frac{r}{\lambda}$, $\forall u \in V, i \in [0..\lambda]$, where $l(P^u_i)$ is the length (hops) of P^u_i.

Theorem 1: RDA solves the ϵ-approximation of DCLC in time $O((m + n \log n)L/\epsilon)$.

It is easy to see why RDA has the same worst-case time complexity as DSA. It could happen that $d^r(u, v) = d^f(u, v)$, $\forall (u, v) \in E$, which makes RDA identical to DSA. However, with much larger probabilities, $d^r(u, v)$ follows a distribution with positive errors and negative errors cancelling out each other along a long path, which allows RDA to run much faster than DSA on an average case. For an arbitrary routing instance, it can be proved that if DSA can terminate with a λ value, then RDA must be able to terminate with the same λ value; the opposite statement is not true. RDA usually terminates with a smaller λ value than DSA.

IV. PATH DELAY DISCRETIZATION

RTF, RTC, and RR all perform discretization at the link level. Each link carries certain amount of error, which may accumulate along a path. Another way to control the total error is to perform discretization on the path level, using the interval partitioning method for combinatorial approximation [12]. Given a path P,

$$d^0(P) = \left\lfloor \frac{d(P)}{r}\lambda \right\rfloor \quad (10)$$

The error is independent of the path length. The path discretization algorithm (PDA) is shown below. EDSP_PDA is omitted because it is identical to EDSP_RDA except that it calls Relax_PDA.

Initialize(V, s, λ)

for each vertex $v \in V$, each $i \in [0..\lambda]$ **do**
    $w[v, i] := \infty$, $\pi[v, i] :=$ NIL, $z[v, i] := \infty$
$w[s, 0] := 0$, $z[s, i] := 0$

Relax_PDA(u, v, i, λ)

$i' := \left\lfloor \frac{z[u,i] + d(u,v)}{r}\lambda \right\rfloor$
if $i' \le \lambda$ and $w[v, i'] > w[u, i] + c(u, v)$ **the**

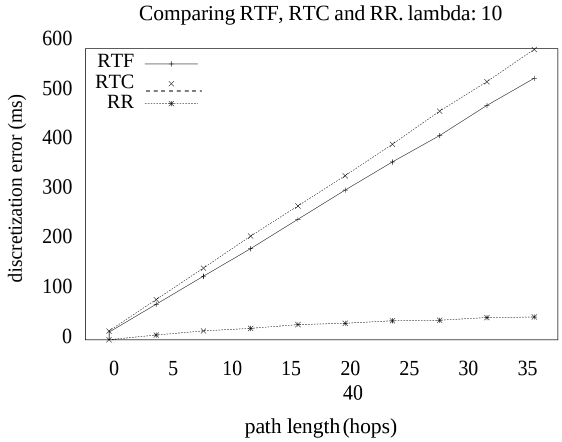


Fig. 1. Compare the average discretization errors of RTF, RTC and RR with respect to different path lengths. The vertical axis is the average of $|\Delta^f(P)|$, $|\Delta^c(P)|$, or $|\Delta^r(P)|$ over 10000 sample paths.

8. $z[v, i^0] := \min\{z[v, i^0], z[u, i] + d(u, v)\}$

PDA(G, s) 9. $\lambda := \lambda_0$

10. **do**

11. $\lambda := 2\lambda$

12. EDSP-PDA(G, s, λ)

13. **while** $\exists v \in V, d(P^v) > (1 + \varepsilon)r$,

where P^v is the path with $\min\{w[v, i] \mid i \in [0..\lambda]\}$

Lemma 4: Let P^u_i be the path stored by $\pi[u, i]$. It always holds that $z[u, i] \leq d(P^u_i)$, $\forall u \in V, i \in [0..\lambda]$.

Lemma 5: Let P^u_i be the path stored by $\pi[u, i]$. It always holds that $z[u, i] \geq i\frac{r}{\lambda}$, $\forall u \in V, i \in [0..\lambda]$.

Lemma 6: Let P^u_i be the path stored by $\pi[u, i]$. It always holds that $d(P^u_i) \leq (i + l(P^u_i))\frac{r}{\lambda}$, $\forall u \in V, i \in [0..\lambda]$, where $l(P^u_i)$ is the length (hops) of P^u_i.

Theorem 2: PDA solves the ε-approaximation of DCLC in time $O((m + n \log n)L/\varepsilon)$.

Both RDA and PDA can be easily extended to handle more than one constraints.

V. ANALYSIS

For RTF, the discretization error of every link is non-negative with a tight upper bound of $\frac{r}{\lambda}$. Hence, the discretiza- tion errors of links on a path P will add up to a non-negative value with a tight upper bound of $\frac{r}{\lambda}l(P)$, which is linear to the path length. Statistically, the longer the path, the larger the error. For instance, if $\Delta^f(u, v)$, $\forall(u, v) \in P$, is uniformly distributed in $[0, \frac{r}{\lambda})$, the mean of $\Delta^f(P)$ is $\frac{r}{2\lambda}l(P)$.

For RTC, the discretization error of every link is always non-positive with a tight lower bound of $-\frac{r}{\lambda}$. If $\Delta^c(u, v)$, $\forall(u, v) \in P$, is uniformly distributed in $(-\frac{r}{\lambda}, 0]$, the mean of $\Delta^c(P)$ is $-\frac{r}{2\lambda}l(P)$.

The error of the path delay discretization is always non- negative with a tight upper bound of $\frac{r}{\lambda}$, independent of the length of the path.

To study RR, we model $d(u, v)$, $\forall(u, v) \in E$, as a random variable, whose probability density function is $f_{u,v}(x)$, $x \in$ [0, +∞). For any path P, $\Delta^r(P)$ is also a random variable. Assume the delays of different links are independent.

Theorem 3: Given a path P, the mean of $\Delta^r(P)$ is zero and the standard deviation of $\Delta^r(P)$ is at most $\frac{r\sqrt{l(P)}}{2\lambda}$, regardless of the probability distributions of the link delays.

Fig. 1 shows how the discretization errors of RTF, RTC and RR grow with the path length. The link delay is randomly generated, following an exponential distribution with a mean at 100 ms. As shown in the figure, the discretization errors of RTF and RTC grow linearly with the path length,[1] while the error of RR grows sublinearly.

VI. SIMULATION

A. Simulation Setup

The simulation uses network topologies generated based on the Power-Law model [13]. The default simulation parameters are: The link delays (costs) are randomly generated, following the exponential distribution with a mean of 100. ε = 0.1. λ_0 = 3. Each data point is the average over 1000 randomly generated routing requests. More specifically, we randomly generate ten topologies. On each topology, 100 routing re- quests are generated with the source node randomly selected from the topology. We run DSA, RDA, and PDA to find a cheapest feasible path to every destination for which a feasible path exists. All simulations were done on a PC with PIV 2GHz CPU and 512 Megabytes memory.

The performance metrics used to evaluate the routing algo- rithms are defined as follows.

$$\text{avg execution time} = \frac{\text{total execution time for all requests}}{\text{total number of routing requests}}$$

$$\text{avg cost} = \frac{\text{total cost of returned paths}}{\text{number of returned paths}}$$

$$\text{success rate} = \frac{\text{number of returned paths that are feasible}}{\text{number of returned paths}}$$

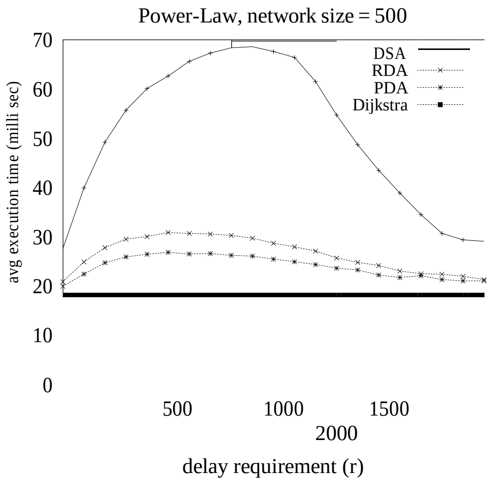
All algorithms under simulation guarantee that the delay of any returned path is bounded by $(1 + \varepsilon)r$.

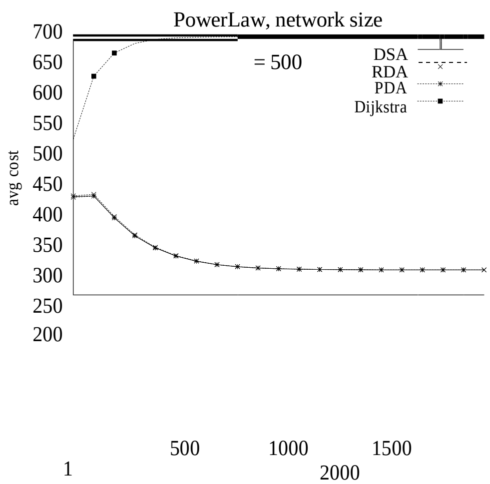
B. Comparing RDA and PDA with DSA

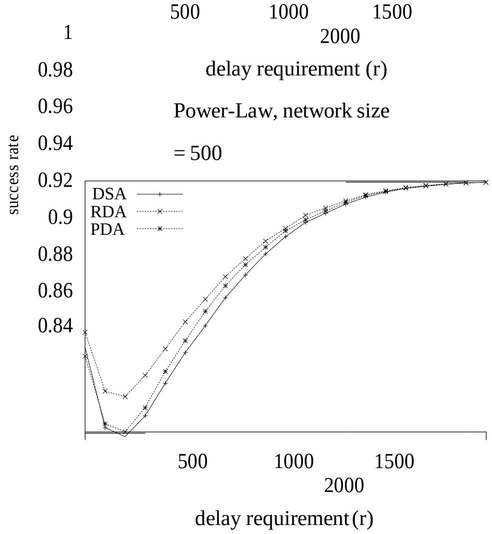
Fig. 2 compares DSA, RDA, and PDA on Power-Law topologies with 500 nodes. Both RDA and PDA are much faster than DSA, with PDA achieving the best execution time. The average costs of the three algorithms are comparable. The success ratio of RDA is slightly better than the other two. Because the three algorithms are close in terms of average cost and success rate in all simulations, we shall focus on execution time in the sequel.

Fig. 3 compares the scalability of the three algorithms with respect to the network size. The gains by RDA and PDA increase for larger topologies. The improvement exceeds an order of magnitude for 1000-node networks.

[1] When the link delay follows an exponential distribution, the average error caused by RTF is smaller than that caused by RTC. However, when the link delay follows a uniform distribution, the average error by RTF is the same as that by RTC.







In summary, the simulations confirmed our prediction that the execution time could be greatly improved by reducing the discretization error, which was achieved very effectively by RDA and PDA. Even with 1000 nodes and one constraint, RDA and PDA computes the constrained shortest paths within 38 milliseconds and 25 milliseconds, respectively, which makes them practical solutions for routers to compute the QoS routing paths periodically.

The last iteration of DSA, RDA, or PDA dominates in terms of both execution time and memory usage, which are $O((m + n \log n)\lambda)$ and $O(n\lambda)$, respectively. Therefore, the memory usage is proportional to the execution time. The previous comparison on execution time thus provides a relative comparison on memory usage as well.

VII. CONCLUSION

We proposed two techniques, randomized discretization and path delay discretization, to design fast algorithms for the delay-constrained least-cost routing problem. Our simulations showed that the new algorithms ran significantly faster than the best existing algorithm.

REFERENCES

[1] S. Chen and K. Nahrstedt, “An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions,” *IEEE Network, Special Issue on Transmission and Distribution of Digital Video*, December 1998.

[2] R. Guerin and A. Orda, “QoS-based Routing in Networks with Inaccurate Information: Theory and Algorithms,” *IEEE INFOCOM’97, Japan*, April 1997.

[3] H. F. Salama, D. S. Reeves, and Y. Viniotis, “A Distributed Algorithm for Delay-Constrained Unicast Routing,” *IEEE INFOCOM’1997*, pp. 84–91, April 1997.

[4] A. Juttner, B. Szviatovszki, I. Mecs, and Z. Rajko, “Lagrange Relaxation Based Method for the QoS Routing Problem,” *IEEE INFOCOM’2001*, April 2001.

[5] T. Korkmaz and M. Krunz, “Multi-Constrained Optimal Path Selection,” *IEEE INFOCOM’2001*, April 2001.

[6] R. Hassin, “Approximation Schemes for the Restricted Shortest Path Problem,” *Mathematics of Operations Research*, vol. 17, pp. 36–42, 1992.

Fig. 2. Compare DSA, RDA, and PDA on Power-Law topologies

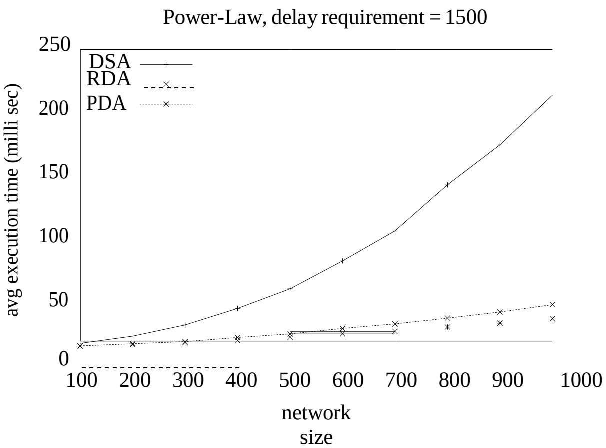


Fig. 3. Scalability comparison

[7] S. Chen and K. Nahrstedt, "On Finding Multi-Constrained Paths," *IEEE International Conference on Communications (ICC'98)*, June 1998.

[8] A. Goel, K. G. Ramakrishnan, D. Kataria, and D. Logothetis, "Efficient Computation of Delay-Sensitive Routes for One Source to All Destinations," *IEEE INFOCOM'2001*, April 2001.

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

[10] H. C. Joksch, "The Shortest Route Problem with Constraints," *Journal of Mathematical Analysis and Applications*, vol. 14, pp. 191–197, 1966.

[11] D. Lorenz and D. Raz, "A Simple Efficient Approximation Scheme for the Restricted Shortest Paths Problem," *Bell Labs Technical Memorandun*, 1999.

[12] S. Sahni, "General Techniques for Combinatorial Approximation," *Operations Research*, vol. 26, no. 5, 1977.

[13] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On Power-Law Relationships of the Internet Topology," *ACM Proceedings of SIGCOMM '99*, 1999.