A Comparison of Exact and ε-Approximation Algorithms for Constrained Routing

Fernando Kuipers[1], Ariel Orda[2], Danny Raz[2], and Piet Van Mieghem[1]

[1] Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands
{F.A.Kuipers, P.VanMieghem}@ewi.tudelft.nl
[2] Technion, Israel Institute of Technology, Haifa, Israel 32000
{ariel@ee, danny@cs}.technion.ac.il

Abstract. The Constrained Routing Problem is a multi-criteria optimization problem that captures the most important aspects of Quality of Service routing, and appears in many other practical problems. The problem is NP-hard, which causes exact solutions to require an intractable running time in the worst case. ε-approximation algorithms provide a guaranteed approximate solution for all inputs while incurring a tractable (i.e., polynomial) computation time. This paper presents a performance evaluation of these two types of algorithms. The main performance criteria are accuracy and speed.

Keywords: QoS routing, performance evaluation, RSP algorithms.

1 Introduction

One of the key issues in providing guaranteed Quality of Service (QoS) is *how to determine paths that satisfy QoS constraints*. Solving this problem is known as *Constrained routing* or *QoS routing*. The research community has extensively studied this problem, resulting in many QoS routing algorithms (see [5] for an overview and performance evaluation). Research has mainly focused on a two-parameter optimization problem called the Restricted Shortest Path (*RSP*) problem. Before presenting the formal definition of the RSP problem, we introduce some terminology and notation.

Let $G(N, L)$ denote a network topology, where $\{N\}$ is the set of N nodes and $\{L\}$ is the set of L links. The number of QoS measures (e.g., delay, hop count) is denoted by m. Each link is characterized by an m-dimensional link weight vector, consisting of m non-negative QoS weights ($w_i(u, v)$, $i = 1, ..., m$, $(u, v) \in \{L\}$) as components. The QoS measure of a path can be either *additive* (e.g., delay, jitter, the logarithm of packet loss), in which case the weight of a path equals the sum of the weights of its links, or *bottleneck* (e.g., available bandwidth), in which case the weight of a path is the minimum (or maximum) of the weights of its links. Without loss of generality [9], we assume all QoS measures to be additive.

The RSP problem is formally defined as follows.

Definition 1. *Restricted Shortest Path (RSP) problem*: Consider a network $G(N, L)$. Each link $(u, v) \in \{L\}$ is specified by $m = 2$ nonnegative measures:

F. Boavida et al. (Eds.): NETWORKING 2006, LNCS 3976, pp. 197–208, 2006.

a *cost* $c(u, v)$ and a *delay* $d(u, v)$. Given a *delay constraint* Δ, the RSP problem consists of finding a path P^* from a source node s to a destination node d such that $d(P^*) \leq \Delta$ and $c(P^*) \leq c(P)\ \forall P : d(P) \leq \Delta$, where $c(P) \stackrel{\text{def}}{=} \sum_{(u,v)\in P} c(u, v)$ and $d(P) \stackrel{\text{def}}{=} \sum_{(u,v)\in P} d(u, v)$.

The RSP problem is known to be NP-hard [1]. To cope with this worst-case intractability, heuristics and ε-approximations have been proposed, as well as a few exact algorithms.

As described in [5], many studies focused on heuristic solutions, which may perform well in certain scenarios. However, in the most general case they cannot provide any performance guarantee, which makes them unpredictable. We focus on the two classes of exact and ε-approximation algorithms, which can (rigorously) provide a predefined level of QoS guarantees. For the ε-approximation algorithms mainly theoretical results exist and no empirical results are published. Exact algorithms provide the optimal solution, however their running time may be very high in the worst case. In this paper we evaluate two representative algorithms, distinguish their worst cases, provide empirical results and discuss and compare the relative strengths of the two approaches.

The outline of the paper is as follows. In Section 2 we describe the two algorithms: we choose SAMCRA [9] as a representative of the class of exact RSP algorithms and SEA [6] as a representative of the class of RSP ε-approximation algorithms. In Section 3 we delineate the worst-case scenarios of each of the two algorithms. In Section 4 we conduct an empirical comparison between the two algorithms. Finally, we discuss some open problems in Section 5 and provide a brief conclusion in Section 6.

2 RSP Algorithms

2.1 SAMCRA

SAMCRA [9] stands for Self-Adaptive Multiple Constraints Routing Algorithm and is a general exact QoS algorithm, which incorporates four fundamental concepts: (1) a nonlinear measure for the path length. When minimizing a linear function of the weights, solutions outside the constraints area may be returned. An important corollary of a nonlinear path length is that *the subsections of shortest paths in multiple dimensions are not necessarily shortest paths themselves.* This necessitates to consider in the computation more paths than only the shortest one, leading to (2) a k-shortest path approach. The k-shortest path algorithm is essentially Dijkstra's algorithm that does not stop when the destination is reached, but continues until the destination has been reached by k different paths, which succeed each other in length. To reduce the search space we use (3) the principle of non-dominated paths[1], and (4) the look-ahead concept. The latter precomputes (via Dijkstra's algorithm) one or multiple shortest

[1] Often also referred to as Pareto optimality. A path P is dominated by a path Q if $w_i(Q) \leq w_i(P)$, for $i = 1, ...m$, with inequality for at least one i.

path trees rooted at the destination and then uses this information to compute end-to-end lower bounds to reduce the search space. SAMCRA can be used with different length functions, and can therefore be easily adapted to solve the RSP problem. The nonlinear length that we have used is:

$$l(P) = \begin{cases} c(P), & \text{if } d(P) \leq \Delta \\ \infty, & \text{else} \end{cases} \quad (1)$$

By employing this length function, SAMCRA can guarantee to find the minimum-cost path within the delay constraint.

2.2 SEA

SEA [6] stands for Simple Efficient Approximation and is an ε-approximation algorithm that (like most ε-approximation algorithms) specifically targets the RSP problem. ε-approximation algorithms are characterized by a polynomial complexity and ε-optimal performance. An algorithm is said to be ε-optimal if it returns a path whose cost is at most $(1+\varepsilon)$ times the optimal value, where $\varepsilon > 0$ and the delay constraint is strictly obeyed. ε-approximation algorithms perform better in minimizing the cost of a returned feasible path as ε goes to zero. However, the computational complexity is proportional to $1/\varepsilon$, making these algorithms impractical for very small values of ε. SEA is based on Hassin's algorithm [3], which has a complexity of $O((\frac{LN}{\varepsilon} + 1) \log \log B)$, where B is an upper bound on the cost of a path. It is assumed that the link weights are positive integers. This ε-approximation algorithm initially determines an upper bound (UB) and a lower bound (LB) on the optimal cost. For this, the algorithm initially starts with $LB = 1$ and $UB =$ sum of (N-1) largest link-costs, and then systematically adjusts them using a *testing* procedure. Once suitable bounds are found, the approximation algorithm bounds the cost of each link by rounding and scaling it according to: $c'(u,v) = \left\lfloor \frac{c(u,v)(N+1)}{\varepsilon LB} \right\rfloor + 1 \; \forall \; (u,v) \in \{L\}$. Finally, it applies a pseudo-polynomial-time algorithm on these modified weights. SEA improves upon Hassin's algorithm by finding better upper and lower bounds and by improving the testing procedure. In this way SEA obtains the polynomial complexity of $O(LN(\log \log N + \frac{1}{\varepsilon}))$.

It is also worth mentioning that there is another class of approximation algorithms, e.g. [2], that approximate the delay constraint rather than the cost. Indeed, this is a heavier compromise, but the reward is in terms of a smaller running time. Yet another approach is to specialize on the network topology (e.g., assume a hierarchical structure) and thus provide an exact and computationally tractable solution [7].

3 Worst-Case Scenarios

NP-hard problems may be solvable in some (or even many) instances, while displaying intractability in the worst case. It is therefore important to gain some understanding at what constitutes a worst-case scenario for a particular problem or algorithm.

3.1 Exact Algorithms

Worst-case scenarios for exact QoS algorithms were identified in [4], and according to [5] they also resulted to be worst-case scenarios for several heuristics. Summarizing [4], the intractability of the constrained routing problem hinges on four factors, namely: (1) The underlying topology, because the number of paths in some classes of topologies can be bounded by a polynomial function of N; based on empirical results [4], other classes of topologies, like the class of random graphs that have a small expected hop count, also appear to be computationally solvable. (2) Link weights that can grow arbitrarily large or have an infinite granularity; when link weights are bounded and have a finite granularity, which is often the case in practice, it can be proved that the constrained routing problem is solvable in polynomial time; in fact, this is the property that ε-approximation algorithms rely on to guarantee a polynomial complexity. (3) A very negative correlation among the link weights; empirical results [4] indicate that there is hardly any "intractability" for the entire range of correlation coefficients $\rho \in [-1, 1]$, except for extreme negative values. (4) The values of the constraints: if they are very large, then it is easy to find a path within the constraints, while if they are very small, then it is easy to verify that there is no path that meets the constraints. If, indeed, the four above-mentioned conditions are all necessary to "induce intractability," they could allow network and service providers to properly dimension their infrastructures so as to avoid intractable scenarios.

3.2 ε-Approximation Algorithms

The class of ε-approximation algorithms are based on entirely different concepts and may not be affected by the worst-case scenarios of exact algorithms. In this section we delineate the worst-case scenarios for ε-approximation algorithms, and in particular for SEA.

The rounding and scaling performed by SEA prevents that a solution that is exactly a factor $(1+\varepsilon)$ larger than optimal can be returned. The scaled weights are computed via $c'(u,v) = \left\lfloor \frac{c(u,v)(N+1)}{\varepsilon LB} \right\rfloor + 1 \ \forall \ (u,v) \in \{L\}$ and hence we have that $c(u,v) \leq \frac{c'(u,v)\varepsilon LB}{N+1} \leq c(u,v) + \frac{\varepsilon LB}{N+1}$. The maximum error that can be made along any path therefore equals $\frac{(N-1)\varepsilon LB}{N+1} \leq \frac{(N-1)\varepsilon c(P^*)}{N+1} < \varepsilon c(P^*)$. The maximum path error of $\varepsilon c(P^*)$ can only be approximated from below for large N.

The factor that affects performance is not so much the topology as the distribution of weights over the links. Let us consider two nodes, s and d, interconnected by two links as displayed in Figure 1. Let the delay of each link be 1, and let the costs be $c(l_1) = 1$, $c(l_2) = 1 + \frac{\varepsilon LB}{N+2}$. We assume that $\frac{N+1}{\varepsilon LB}$ is an integer number, then scaling the link costs results in $c'(l_1) = c'(l_2) = \frac{N+1}{\varepsilon LB} + 1$. Hence, due to the scaling performed by the algorithm, the weights of the two links would appear identical, and the algorithm may pick link l_2, which is a factor $(1 + \frac{\varepsilon}{4})$ more costly than link l_1, when $LB = c(P^*) = 1$. SEA cannot return

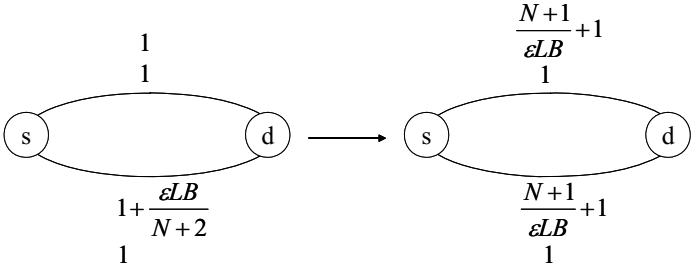

Fig. 1. Example topology consisting of two nodes and two links, where each link is characterized by a cost and a delay. The left topology represents the original weights, while the right topology gives the scaled weights (according to SEA).

a path that is a factor $(1 + \frac{\varepsilon}{3})$ more costly than optimal in this topology. Note that, depending on the implementation details of the algorithm, either of the two paths could be chosen. This source of "randomness" reduces the expected error over multiple graphs.

Another measure that determines the worst-case error of SEA is the value of the lower bound LB. SEA first determines upper and lower bounds, such that $\frac{UB}{LB} \leq N$. Hence, for the lower bound holds that $\frac{c(P^*)}{N} \leq LB \leq c(P^*)$. In case $LB = \frac{c(P^*)}{N}$, the worst-case error that SEA could make is upper bounded by $\frac{\varepsilon c(P^*)}{N}$.

Finally, in a general topology, the weights are unlikely to constitute worst-case errors. To obtain a worst-case error, the link weights should be chosen from two classes, namely link weights that, when scaled and rounded, do not lead to an error and link weights that, when scaled and rounded, give the maximum attainable error. The optimal path would then consist of the "error-free" link weights, while the approximation algorithm could return in the worst case a path that includes only the "erroneous" link weights. If the weights are randomly assigned to the links, then there is a smoothing effect over the various links. So, for pushing the algorithm to its limit, one could (either or both):

- Consider very simple topologies, with a small number of edges and low connectivity.
- Assume some correlation among the weights of consecutive links, in an attempt to cancel the "smoothing effect." In addition, the weights of the links should be chosen out of a small set, in which the differences are such that the scaling operation would incur the maximal possible error.
- We should focus on large values of the weights and the delay constraint, since for small values a pseudo-polynomial algorithm would provide a solution that is both optimal and computationally solvable.

4 Performance Evaluation

We have performed a comprehensive set of simulations to compare between SAMCRA and SEA. We have used Waxman graphs [8], complete graphs, random

graphs of the type $G_p(N)$, where p is the link density, power-law graphs, and lattices. In each class of graphs, the delay and cost of every link $(u, v) \in \{L\}$ were taken as independent uniformly distributed random integers in the range $[1, M]$. However, for the class of lattices, the delay and the cost of every link (u, v) were also negatively correlated: the delay was chosen uniformly from the range $[1, M]$ and the corresponding cost was set to $M+1$ minus the delay. Simulations for different values of M did not display any significant differences, so we have chosen $M = 10^5$. In each simulation experiment, we generated 10^4 graphs and selected nodes 1 and N as the source and destination, respectively. For lattices, this corresponds to a source in the upper left corner and a destination in the lower right corner, leading to the largest minimum hop count. For power-law graphs, this corresponds to a source that has the highest nodal degree and a destination that has the lowest nodal degree in the graph. For the other classes of graphs, this is equivalent to choosing two random nodes.

The delay constraint Δ was selected as follows. First, we computed the least-delay path (LDP) and the least-cost path (LCP) between the source and the destination using Dijkstra's algorithm. If the delay constraint $\Delta < d(\text{LDP})$, then there is no feasible path. If $d(\text{LCP}) \leq \Delta$, then the LCP is the optimal path. Since these two cases are easy to deal with, we compared between the algorithms considering the values $d(\text{LDP}) < \Delta < d(\text{LCP})$, as follows:

$$\Delta = d(\text{LDP}) + \frac{x}{4}(d(\text{LCP}) - d(\text{LDP})) \tag{2}$$

In all simulations we chose $x = 2$, except when evaluating the influence of the constraints, in which case we considered $x = 0, 1, 2, 3, 4$.

4.1 Simulation Results

SAMCRA always finds the optimal path within the delay constraint. We therefore evaluated SEA based on how successful it is in minimizing the cost of a returned feasible path, when compared to SAMCRA. The effective approximation α of SEA is defined as

$$\alpha = \frac{c(P_{SEA})}{c(P_{SAMCRA})} - 1$$

where $c(P_x)$ is the cost of the feasible paths that are returned by algorithm x. We plot $E[\alpha]$, $var[\alpha]$, and $\max[\alpha]$ based on the 10^4 iterations. We also report the *execution time* of the compared algorithms. Figure 2 displays the effective approximation α and execution time as a function of ε for lattice graphs with $N = 100$, and independent uniformly distributed random link weights.

We can clearly see that $\alpha << \varepsilon$, which means that SEA hardly or never reaches a worst-case performance. Even the performance for $\varepsilon = 1$ is surprisingly good. The reason that $\alpha << \varepsilon$ is partly due to the assignment of the link weights according to a uniform distribution. Given that the link costs are uniformly distributed in the range $[1, M]$, then the scaled and rounded costs are approximately uniformly distributed in the range $[\left\lfloor \frac{(N+1)}{\varepsilon LB} \right\rfloor + 1, \left\lfloor \frac{M(N+1)}{\varepsilon LB} \right\rfloor + 1]$.

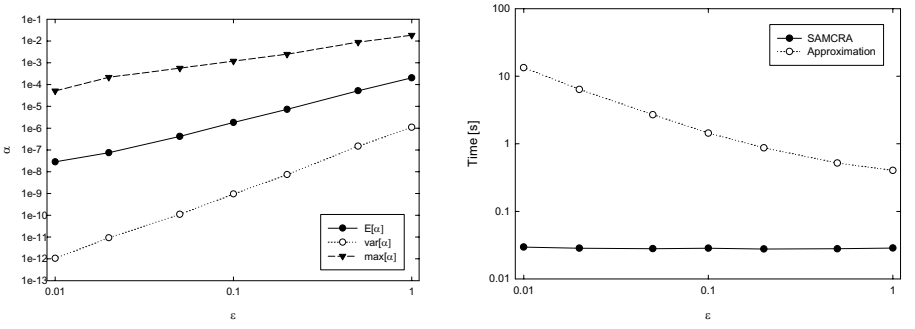

Fig. 2. Effective approximation α and execution time as a function of ε. The results are for Lattice graphs with $N = 100$, and the link weights are independent and uniformly distributed random variables.

As any real number x can be written as $x = \lfloor x \rfloor + \langle x \rangle$, where $\lfloor x \rfloor$ denotes the largest integer smaller or equal to x and where $\langle x \rangle \in [0, 1)$ denotes the fractional part of x, the round-off error of link (u, v) equals $1 - \left\langle \frac{c(u,v)(N+1)}{\varepsilon LB} \right\rangle$, for which holds $0 \leq 1 - \left\langle \frac{c(u,v)(N+1)}{\varepsilon LB} \right\rangle \leq 1$. Assuming that $\frac{(N+1)}{\varepsilon LB}$ is a fixed fractional number that is known to SEA before it executes its main procedure, the size of the round-off error is determined by the costs $c(u, v)$. Since these costs are uniformly distributed, we believe that the round-off errors are well approximated by a uniform distribution. If this holds, then our expected round-off error on a link is only half its worst-case value.

The expected α displays an approximately linear increase on the log-log scale, with a slope that is almost equal to 2. Therefore, in our simulated range, changing the value of ε has a quadratic impact on the effective approximation α. We can also see a clear correspondence between ε and the execution time: the larger ε, the smaller the execution time. The results approximately follow a linear line with a slope of -1 on a log-log scale, which indicates that the time is inversely proportional to ε, as was expected from the worst-case time complexity $O(LN(\log \log N + \frac{1}{\varepsilon}))$. However, even for $\varepsilon = 1$ the execution time of SEA is still by an order of magnitude larger than the execution time of SAMCRA. Figure 3 plots the effective approximation α as a function of the constraint values. A larger constraint means that more paths obey it. This larger search space results in a higher probability of making an erroneous decision (within the ε margin). The execution times of SAMCRA and SEA seem hardly influenced by the different constraints. Actually, by choosing x in Equation (2) as $x = 0$ or $x = 4$, the RSP problem is polynomially solvable, with solutions LDP and LCP respectively. For $x = 1, 2, 3$ SAMCRA is able to solve the RSP problem in a similar time span, suggesting that these simulated instances were also polynomially solvable.

Figure 4 displays the effective approximation α and execution time as a function of N.

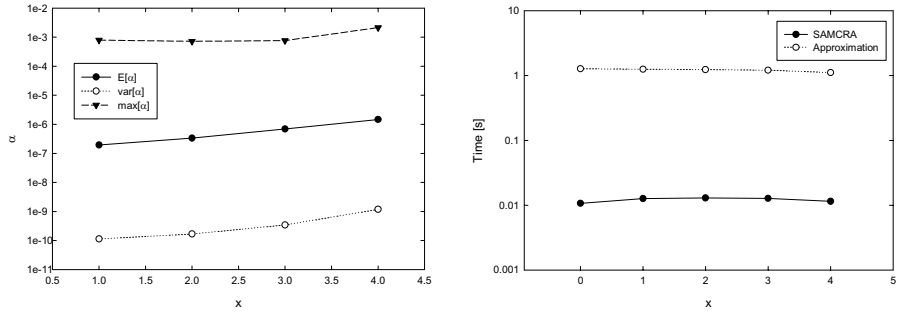

Fig. 3. Effective approximation α and execution time as a function of x in equation (2). The results are for Lattice graphs with $\varepsilon = 0.1$ and $N = 100$, and the link weights are independent and uniformly distributed random variables.

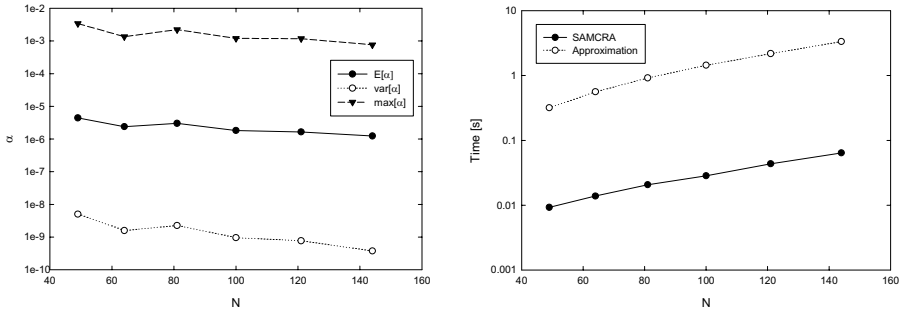

Fig. 4. Effective approximation α and execution time as a function of N with $\varepsilon = 0.1$. The results are for lattice graphs, and the link weights are independent and uniformly distributed random variables.

We can see that α slightly decreases with N. If N grows, there may be many paths that have a length close to the shortest feasible path. Finding one of these paths is less difficult than finding the true RSP path. The relative difference in time between SAMCRA and SEA remains fairly constant: SAMCRA is more than 10 times faster than SEA.

Figure 5 displays the results for negatively correlated random link weights. According to [4], this simulation setting corresponds to a worst-case scenario for exact algorithms.

Contrary to the decrease of α in Figure 4, we observe an increase of $E[\alpha]$ with N. Also, the values of α are much higher (considering the smaller values of N). Therefore, this worst-case scenario for exact algorithms also seems to affect ε-approximation algorithms, although not to the extent of constituting a worst-case scenario for SEA. The difference in execution time is clear: SAMCRA incurs an exponential computation time, whereas SEA is (always) a polynomial-time algorithm. Therefore, there is a cross-over point (at $N = 40$), where SAMCRA starts to run slower than SEA.

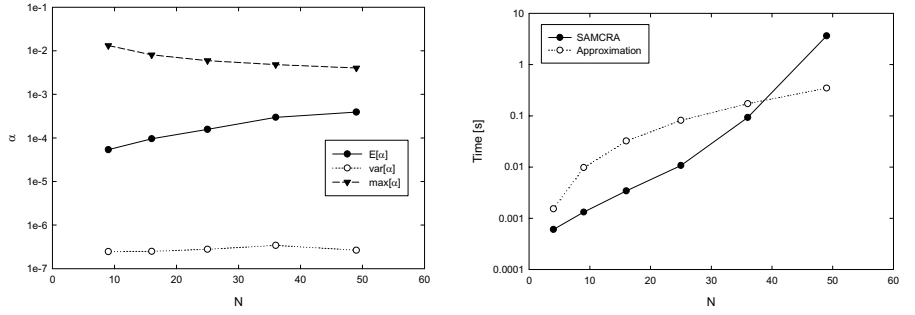

Fig. 5. Effective approximation α and execution time as a function of N with $\varepsilon = 0.1$. The results are for Lattice graphs, and the link weights are negatively correlated, uniformly distributed random variables.

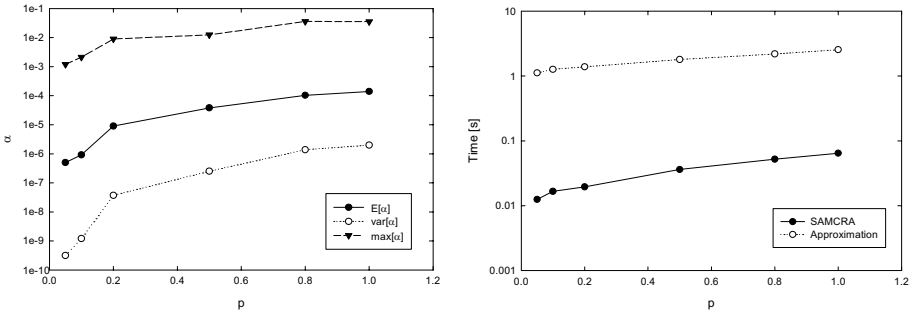

Fig. 6. Effective approximation α and execution time as a function of the link density p. The results are for random graphs with $\varepsilon = 0.1$, $N = 100$, and the link weights are independent and uniformly distributed random variables.

We have simulated in the class of random graphs with different link densities p ($p = 1$ corresponds to the class of complete graphs).

The values of α in Figure 6 increase with p, which suggests that SEA has more difficulty with dense graphs. Dense graphs have more links than sparse graphs and hence the probability of making round-off errors increases. Also, the denser a graph becomes, the shorter the expected hop count will be. With a short expected hop count, situations like in Figure 1 are more likely to occur than when the expected hop count is large, like in the class of lattices. A small effective approximation was also observed for the sparse Waxman graphs. The effective approximation α and execution time, as function of ε and N, in the class of Waxman graphs displayed a similar trend as in Figure 2 for the class of lattices, and hence are not plotted here.

We have also simulated in the class of power-law graphs, which are considered to contain the Internet graph. In power-law graphs the nodal degree distribution

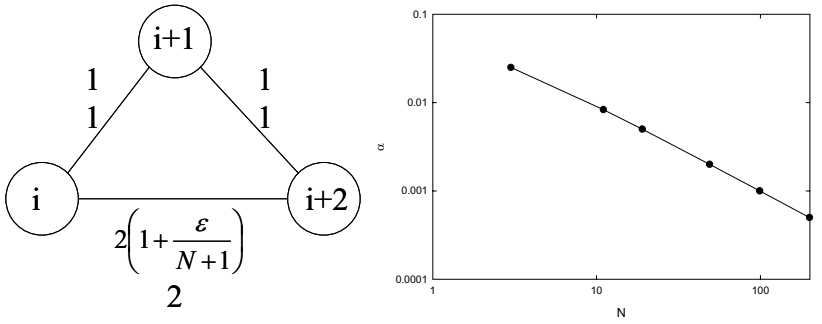

Fig. 7. Effective approximation α as a function of N. The results are for the chain topology (on the left, $i = 1, ..., N-2$), with $\varepsilon = 0.1$.

is $\Pr[d = i] = ci^{-\tau}$, where c is a constant such that $\sum_{i=1}^{N-1} ci^{-\tau} = 1$. Measurements in the Internet suggest that $\tau \approx 2.4$ and therefore we have chosen this value for the generation of our power-law graphs. Since the source referred to the node with the highest degree and the destination to the node with the lowest degree, the probability that there is only one path between source and destination is much higher in this class of power-law graphs than in the other considered classes of graphs. Our simulations for different ε showed that for $N = 100$ and $\varepsilon < 0.1$, α was zero. Furthermore, we deduced that the effective approximation α in the class of power-law graphs with $\tau \approx 2.4$ was the lowest among the considered classes of graphs.

Finally, we have simulated with a chain topology. By choosing the weights as in Figure 7, the error when rounding and scaling link $(i, i+2)$ equals $\frac{2\varepsilon}{N+1}$ and the total error that can be accumulated in the worst case along the lower path with $\frac{N-1}{2}$ hops is $\frac{N-1}{N+1}\varepsilon$. Since the optimal cost equals $N-1$, the effective approximation α can be found to obey $\alpha = \frac{(N-1)\varepsilon}{(N-1)(N+1)} = \frac{\varepsilon}{N+1}$, which perfectly matches our result in Figure 7, as seen by the straight line on a log-log scale.

4.2 Simulation Conclusions

In this subsection we summarize the conclusions that can be drawn from our simulation results.

1. Besides the better performance, the running time of the exact algorithm SAMCRA was at least ten times faster than the running time of SEA, in all simulated scenarios except for the constructed worst-case scenario of Figure 5.
2. The actual performance of SEA, as measured by the effective approximation α, was much better than the theoretical $(1 + \varepsilon)$ upper bound.
3. The combination of many paths with a small hop count between source and destination leads to larger α on average than in the case of a large hop count or very few paths.

4. Changing the value of ε seems to have a quadratic impact on the effective approximation α.
5. The correspondence between the time t that SEA needs to solve an instance of the RSP problem and ε, nicely follows $t \sim \frac{1}{\varepsilon}$.

5 Discussion

SEA has a much better performance than the theoretical $(1 + \varepsilon)$ bound. The question therefore rises if we can make this bound sharper without increasing the time complexity. For instance, instead of only rounding up, one could consider rounding to the nearest number (e.g., if the granularity is 0.1 then $0.57 \rightarrow 0.6$ and $0.52 \rightarrow 0.5$). The overall worst-case error ε will then be halved and the expected error might tend to 0 (under a uniform distribution of the link weights).

The extension of RSP approximation algorithms like SEA to the more general QoS algorithms that handle $m > 2$ constraints would still be polynomial. However, the complexity would increase with $O(N^m)$, which may be prohibitive.

It is possible to devise approximation schemes that can also work with real weights. This can be done via an extra phase of rounding and scaling. The solution will still be polynomial, but a second source of inaccuracy (that can also be bounded) is introduced.

How to take advantage of the strengths of SAMCRA and SEA? One approach is to invoke SAMCRA with a running time "budget" T, within which it attempts to retrieve the optimal solution. In case SAMCRA encounters a hard instance, T may not suffice to accomplish this task. In this case, SAMCRA is halted and the SEA algorithm is invoked. The combined approach has the following properties: it guarantees an ε-optimal solution, it has a polynomial worst-case running time, and empirical evidence shows that, usually, an optimal solution would be found quickly.

6 Conclusions

The Restricted Shortest Path (RSP) problem seeks to minimize the cost of a path while obeying a delay constraint. The importance of this problem is undisputed, since it appears in many different research fields and plays a key role in Quality of Service (QoS) routing. Unfortunately, the RSP problem is NP-hard. Many algorithms have been proposed, which can be subdivided into the classes of exact solutions, ε-approximations, and heuristics. Only the first two classes can provide some (rigorous) level of guarantee on the optimality of the solution. We have therefore focused on these two classes, represented by the exact SAMCRA algorithm and the ε-approximation algorithm SEA. ε-approximation algorithms mainly have been studied theoretically, providing worst-case bounds, but not empirically. We have therefore compared SEA to SAMCRA. In worst-case scenarios, the complexity of SAMCRA is prohibitively high, but in most instances it ran significantly faster than SEA. SEA, on the other hand has a very good accuracy and polynomial running time.

References

1. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.
2. A. Goel, K.G. Ramakrishnan, D. Kataria, D. Logothetis, "Efficient Computation of Delay-sensitive Routes from One Source to All Destinations," Proc. of IEEE INFOCOM, pp. 854-858, 2001.
3. R. Hassin, "Approximation schemes for the restricted shortest path problem," Mathematics of Operations Research, vol. 17, no. 1, pp. 36-42, February 1992.
4. F.A. Kuipers and P. Van Mieghem, "The impact of correlated link weights on QoS routing," Proc. of the IEEE INFOCOM Conference, vol. 2, pp. 1425-1434, April 2003.
5. F.A. Kuipers, T. Korkmaz, M. Krunz and P. Van Mieghem, "Performance Evaluation of Constraint-Based Path Selection Algorithms," IEEE Network, vol. 18, no. 5, pp. 16-23, September/October 2004.
6. D.H. Lorenz and D. Raz, "A simple efficient approximation scheme for the restricted shortest path problem," Operations Research Letter, vol. 28, no. 5, pp. 213-219, June 2001.
7. A. Orda, "Routing with end-to-end QoS guarantees in broadband networks," IEEE/ACM Transactions on Networking, vol. 7, no. 3, pp. 365-374, 1999.
8. P. Van Mieghem, "Paths in the simple random graph and the waxman graph," Probability in the Engineering and Informational Sciences (PEIS), no. 15, pp. 535-555, 2001.
9. P. Van Mieghem and F.A. Kuipers, "Concepts of exact quality of service algorithms," IEEE/ACM Transactions on Networking, vol. 12, no. 5, pp. 851-864, October 2004.