Performance Evaluation of Constraint-Based Path Selection Algorithms

Fernando Kuipers, Delft University of Technology
Turgay Korkmaz, University of Texas at San Antonio
Marwan Krunz, University of Arizona
Piet Van Mieghem, Delft University of Technology

Abstract

Constraint-based path selection is an invaluable part of a full-fledged quality of service (QoS) architecture. Internet service providers want to be able to select paths for QoS flows that optimize network utilization and satisfy user requirements and as such increase revenues. Unfortunately, finding a path subject to multiple constraints is known to be an NP-complete problem. Hence, accurate constraint-based path selection algorithms with a fast running time are scarce. Numerous heuristics and a few exact algorithms have been proposed. In this article we compare most of these algorithms. We focus on restricted shortest path algorithms and multi-constrained path algorithms. The performance evaluation of these two classes of algorithms is presented based on complexity analysis and simulation results and may shed some light on the difficult task of selecting the proper algorithm for a QoS-capable network.

T here is a continuous demand for a mission-critical Internet that can provide various levels of quality-of-service (QoS) guarantees and/or service differentiation to voice, video, and data applications in a unified manner. Realizing the potential benefits of being able to use the Internet as a unified transport technology, the research community and industry have been paying significant attention to enabling QoS-based networking in the Internet. Users require tailormade services with high QoS and reliability, which a best-effort paradigm cannot provide. On the other hand, Internet service providers seek a more commercial Internet, which enables them to provide differentiated services, optimize network throughput, and possibly increase profit. To accommodate the need for QoS, the research community has proposed a variety of QoS-capable frameworks (e.g., IntServ, DiffServ, MPLS) [1]. Since these frameworks largely rely on the underlying routing table, one of the key issues in QoS-proficient architectures is how to determine efficient paths that can satisfy the QoS requirements of multimedia applications. This problem is commonly known as constraint-based path selection and has been shown to be NP-complete. Accordingly, the research community has proposed many heuristics and only a few exact algorithms. In this article we provide a descriptive overview of these algorithms and focus on their performance evaluation using extensive simulations. To the best of our knowledge, no comparative study has been conducted before, except for the limited simulation studies in the original papers describing these algorithms.

Before giving the formal definition of the problem, we describe the notation that is used throughout this article. Let $G(N,E)$ denote a network topology, where N is the set of nodes and E is the set of links. With a slight abuse of notation, we also use N and E to denote the number of nodes and the number of links, respectively. The source and destination nodes are denoted by s and d. P denotes a path between s

and d, and h reflects the hopcount of a path. The number of QoS measures (e.g., delay, hopcount) is denoted by m. Each link $(u, v) \in E$ is characterized by an m-dimensional link weight vector, consisting of m nonnegative QoS weights $w_i(u, v)$, $i = 1, \ldots, m$. QoS measures can be roughly classified into *additive*[1] (e.g., delay) and *non-additive* (e.g., available bandwidth). In case of an additive measure, the weight of a path is equal to the sum of the corresponding weights of the links along that path. For a non-additive measure, the weight of a path is the minimum (or maximum) link weight along that path. The QoS constraints are denoted by L_i, $i = 1, \ldots, m$. In general, constraints on non-additive measures can be dealt with by pruning from the graph all links (and possibly disconnected nodes) that do not satisfy the requested QoS constraint. For our performance evaluation we only consider the more difficult, additive measures. Furthermore, the compared algorithms assume that the network-state information (i.e., link weights) is accurately maintained at every node via QoS-aware networking protocols. All algorithms that are evaluated in this article were designed to solve (an instance of) the multi-constrained path selection problem.

Definition 1 — Multi-constrained path (MCP) problem: Consider a network $G(N,E)$. Each link $(u, v) \in E$ is associated with m additive weights $w_i(u, v) \geq 0$, $i = 1, \ldots, m$. Given m constraints L_i, $i = 1, \ldots, m$, the problem is to find a path P from s to d such that:

$$w_i(P) \overset{def}{=} \sum_{(u,v) \in P} w_i(u,v) \leq L_i \text{ for } i = 1, \ldots, m.$$

---

[1] *Multiplicative measures can be transformed into additive measures by taking their logarithm.*

A path obeying the above condition is said to be *feasible*. Note that there may be multiple feasible paths between *s* and *d*. A modified (and more difficult) instance of the MCP problem is to retrieve the shortest path among the set of feasible paths. This problem is known as the *multi-constrained optimal path* (MCOP) problem and is attained by adding a second condition on the path *P* in Definition 1: $l(P) \leq l(Q)$ for any *feasible* path *Q* between *s* and *d*, where *l*(.) is a path length function. A solution to the MCOP problem is also a solution to the MCP problem, but not necessarily vice versa. Considerable work in the literature has focused on a special case of the MCOP problem known as the *restricted shortest path* (RSP) problem. In this problem the goal is to find the least-cost path among those that satisfy only one constraint denoted by Δ, which bounds the permissible delay of a path. The MCP problem and its variants are known to be NP-complete [2].

In this article, we compare most QoS algorithms based on extensive simulations. For these algorithms, we investigate:
• How often the algorithms return feasible paths.
• Their complexity.
Complexity refers to the intrinsic minimum amount of resources needed to solve a problem or execute an algorithm. Complexity can be divided into *computational-time* complexity and *space complexity*. Here we focus on the computational-time complexity. We consider both the worst-case complexity and the empirical execution times. Table 1 summarizes the worst-case time and space complexities of all considered algorithms. All algorithms have been implemented with the same data structure, namely Fibonacci heaps. The simulations presented in this article consisted of creating a Waxman topology or a square lattice, through which the algorithms computed an RSP or MCP path. Waxman graphs belong to the class of random graphs where the probability of existence of a link between two nodes decays exponentially with the geographic distance between those two nodes. Such graphs are often chosen because of their resemblence to the actual network topologies, and also because they are easy to generate, allowing for the evaluation of a large number of topologies. This last property is crucial in an extensive algorithmic study, where it is necessary to evaluate many scenarios in order to be able to draw confident conclusions. We have chosen the Waxman parameters such that the longest minimum hopcount between two nodes in a 100-node graph is around 7. The class of lattices was chosen to reflect a hard topology class, as motivated in [3]. All simulations are based on 10^4 generated topologies, leading to an accuracy of roughly two digits.

The remainder of this article is organized as follows. In the next section, we consider the RSP problem, present the algorithms that target this problem, evaluate their performance using simulations, and provide conclusions. The third section adopts the same approach for the MCP problem. The fourth section concludes with a summary and discussion.

RSP Algorithms

In this section, we first briefly describe the most important RSP algorithms. For a more in depth discussion we refer the reader to [4] and the references therein. After our description of the RSP algorithms, we present the simulation results followed by conclusions.

Description of RSP Algorithms
Exact Algorithms — Widyono [5] presented an exact solution to the RSP problem called Constrained Bellman-Ford (CBF). CBF maintains a list of paths from *s* to every other node, ordered in increasing cost and decreasing delay. It selects a node whose list contains a path that satisfies the delay con-

straint Δ and that has the minimum cost. CBF then explores the neighbors of this node using a breadth-first search, and (if necessary) adds new paths to the list maintained at each neighbor. This process continues as long as Δ is satisfied and there exists a path to be explored.

∈*-Optimal Approximation* — One general approach for dealing with NP-complete problems is to look for an approximation algorithm with a polynomial time complexity. An algorithm is said to be ∈-optimal if it returns a path whose cost is at most $(1 + \in)$ times the cost of the optimal path, where $\in > 0$. Approximation algorithms perform better in minimizing the cost of a returned feasible path as ∈ goes to zero. However, the computational complexity is proportional to 1/∈, making these algorithms impractical for small values of ∈. In our simulations, we consider Hassin's algorithm [6] as a representative ∈-optimal approximation algorithm.

Backward-Forward Heuristic — The backward-forward heuristic (BFH) first determines the least-delay path (LDP) and the least-cost path (LCP) from every node *u* to *d* [7]. It then starts from *s* and explores the graph by concatenating two segments:
• The so-far explored path from *s* to an intermediate node *u*.
• The LCP or the LDP from node *u* to *d*.
BFH simply uses Dijkstra's algorithm with the following modification in the relaxation procedure: a link (u, v) is relaxed if it reduces the total cost from *s* to *v*, while its approximated end-to-end delay obeys the delay constraint.

Lagrangian-based Linear Composition — The Lagrangian-based linear composition algorithm combines the delay and cost of each link into a link weight $w(u, v) = \alpha d(u, v) + \beta c(u, v)$ and finds the shortest path with respect to $w(u, v)$. The parameters α and β are called the *multipliers*, and a key issue is how to determine appropriate values for them. This can be done by iteratively finding the shortest path with respect to the linear combination and adjusting the multipliers' values in the direction of the optimal solution [8]. Several refinements have been proposed to the basic Lagrangian-based composition approach. For example, one can use the *k*-shortest path algorithm[2] to close the gap between the optimal solution and the returned path. For our simulations, we use the approach of Juttner *et al.* [8].

Hybrid Algorithms — Hybrid algorithms use combinations of the aforementioned approaches. One such heuristic called DCCR was provided in [9]. DCCR tries to solve the RSP problem through the minimization of a nonlinear length function and using a *k*-shortest path algorithm. In order to improve the performance with small values of *k*, the search space is reduced using a Lagrangian-based algorithm (with *y* iterations) before applying DCCR. This final hybrid algorithm is called SSR+DCCR.

Performance Evaluation of RSP Algorithms
In this subsection, we compare the RSP algorithms by simulations using Waxman graphs and lattices. In each Waxman graph the delay and cost of every link (u, v) are taken as independent uniformly distributed random variables in the range [1, 100]. In the class of lattices the delay and the cost of every link (u, v) are negatively correlated: the delay is chosen uniformly from the range [1, 100] and the corresponding

---

[2] *A* k-*shortest path algorithm does not stop when the destination has been reached for the first time, but continues until it has been reached through* k *different paths succeeding each other in length.*
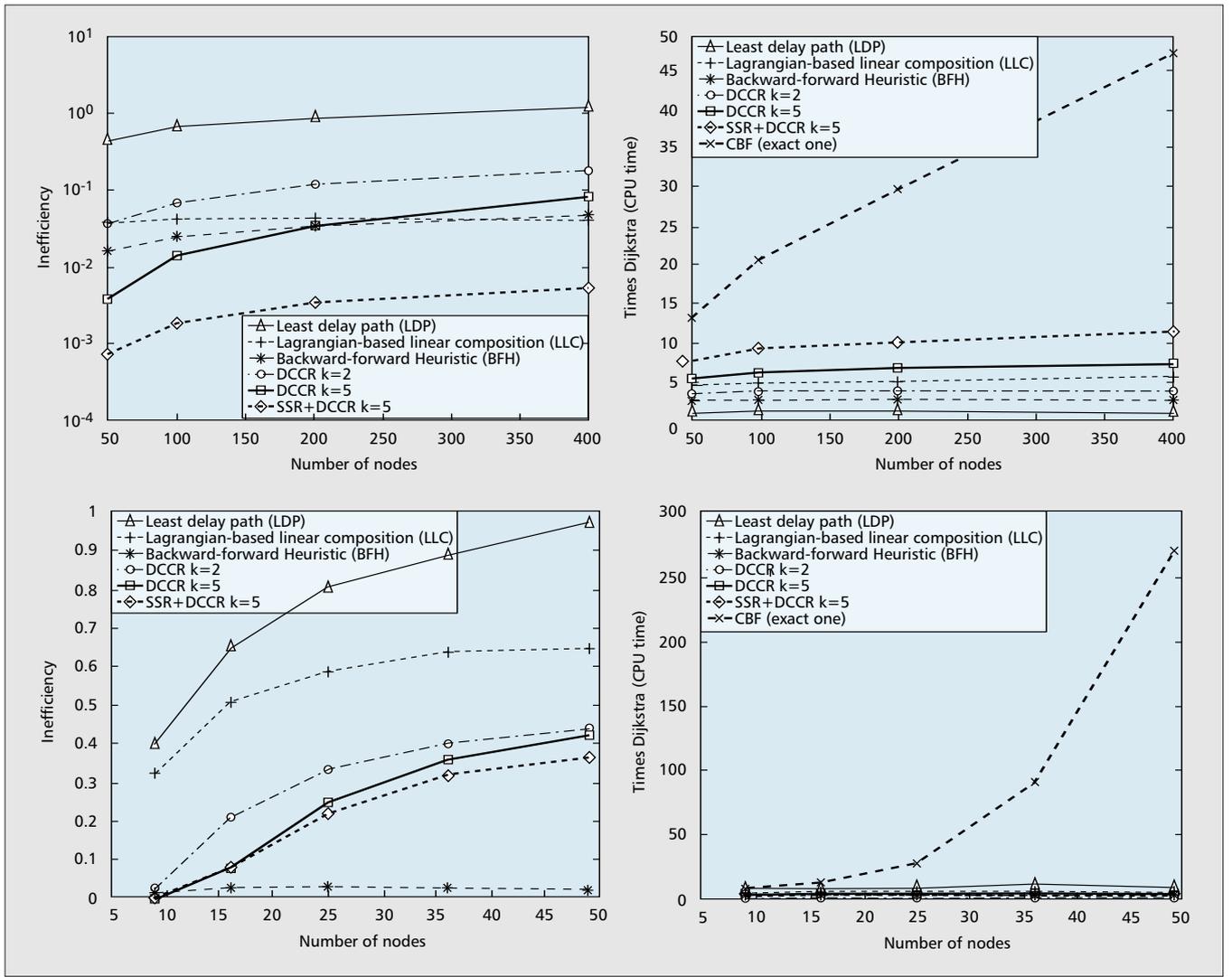
■ Figure 1. *Scaling of the performance measures with* N. *The inefficiency and normalized CPU time are shown for the class of Waxman graphs (above) and for the class of square lattices (below).*

cost is set to 101 minus the delay. In each simulation experiment we generate 10^4 graphs and select node 1 and node N as the source and destination node. For the lattices this corresponds to a source in the upper left corner and a destination in the lower right corner, leading to the largest minimum hopcount.

We select the delay constraint Δ as follows. First, we compute 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 $d(\text{LCP}) \le \Delta$, then the LCP is the optimal path. Since these two cases are easy to deal with, we compare the algorithms under the more difficult cases where $d(\text{LDP}) < \Delta < d(\text{LCP})$. To investigate how the different values of the delay constraint affect the performance of the compared algorithms, for each graph we select five different values for Δ in the range $(d(\text{LDP}), d(\text{LCP}))$, as follows:

$$\Delta(x) = d(\text{LDP}) + \frac{x}{5}(d(\text{LCP}) - d(\text{LDP})), \quad x = 1,2,3,4,5. \quad (1)$$

All considered RSP algorithms were found capable of finding a feasible path that satisfies the delay constraint Δ. Therefore, the challenging part of the RSP problem is not to find a feasible path, but the ability of the algorithm to minimize the cost of a selected feasible path. We compare the algorithms

based on how *inefficient* they are in minimizing the cost of a returned feasible path, when compared to the exact algorithm (CBF) that finds a feasible path with the minimum cost. The *inefficiency* of an algorithm A is defined as

$$inefficiency_A \overset{def}{=} \frac{c(A) - c(CBF)}{c(CBF)}$$

where $c(\cdot)$ is the average cost of the feasible paths that are returned by an algorithm. We also report the execution time of the compared algorithms. To make the results machine-independent, the execution times are normalized by the *execution time of Dijkstra's algorithm* (LDP).

The simulation results indicate that for the considered graphs, the execution times of the $\in$-optimal approximation algorithms (even when $\in$ = 1) are much larger than those of the other compared algorithms. Therefore, we exclude the $\in$-approximations from our plots. We compare the following algorithms: the exact CBF, the least delay path (LDP), Lagrangian-based Linear Composition (LLC), Backward-forward heuristic (BFH), DCCR with $k = 2$ and $k = 5$ (where k refers to the maximum number of paths that can be stored at a node), and SSR+DCCR with $k = 5$. Since the relative differences between the algorithms (in terms of *inefficiency* and *execution time*) do not vary significantly with x in $\Delta(x)$, we only

report the average of the inefficiency and the execution time of the algorithms.[3]

The results are shown in Fig. 1 as a function of the number of nodes N. In all cases the basic LDP algorithm has the highest inefficiency and the lowest execution time. With a slight increase in execution time, BFH gives a significantly lower inefficiency than the LDP algorithm. Actually, BFH also has a lower inefficiency (even with less computational time) than LLC and DCCR with $k = 2$. Since the inefficiency of DCCR and SSR+DCCR is controlled by the value of k, they can give a lower inefficiency if k increases. However, this will result in a longer execution time. Moreover, in some cases (e.g., the lattices with negatively correlated link delay and cost) no significant improvement can be obtained with the small values of k (e.g., $k \leq 5$). In this case, many subpaths with small cost and high delay will initially be stored. These subpaths are likely to lead to infeasible paths. Only if k is high enough, the paths with higher cost and lower delay will be stored, which may lead to the optimal solution. The BFH concept is more valuable in this scenario because it foresees whether a path may be able to improve the cost or obey the constraint.

RSP Conclusions

Our conclusions for the RSP problem are confined to the considered classes of graphs with the specified link weight distribution.

The simulation results indicated that a lower inefficiency is generally only obtained at the expense of increased execution time. Therefore, a hybrid algorithm similar to SSR+DCCR seems to be a good solution for the RSP problem. The main advantage of a hybrid algorithm would be to initially determine a good path with a small execution time (e.g., by using BFH) and to improve the inefficiency with a k-shortest path approach while controlling the *complexity* with the value of k. From this, we conclude that rather than using a single approach, a combination of key concepts (e.g., a nonlinear length function, search space reduction, tunable accuracy through a k-shortest path algorithm, and backward-forward search) leads to efficient algorithms for the RSP problem.

MCP Algorithms

In this section we start with a brief description of the MCP algorithms considered in our performance study. A more in depth discussion of these algorithms can be found in [4, 10], and the references therein. Subsequently, we present the simulation results and conclusions.

Description of MCP Algorithms

Jaffe's Approximation — Jaffe [11] proposed a Lagrange-based approximation for the MCP problem based on minimizing a linear combination of the link weights:

$$w(u,v) = \sum_{i=1}^{m} d_i w_i(u,v),$$

where d_i are positive multipliers. By choosing $d_i = (1/L_i)$, the largest volume inside the constraints volume is scanned, before a possibly infeasible path can be selected.

Iwata's Fallback Algorithm — Iwata *et al.* [12] proposed a heuristic that first computes one (or more) shortest path(s) based on one QoS measure and then checks if all the constraints are met. If this is not the case, the procedure is repeated with another measure until a feasible path is found or all QoS measures are examined. In our simulations we only consider one shortest path per QoS measure.

SAMCRA and TAMCRA — Both TAMCRA [13] and its successor SAMCRA [14] incorporate three fundamental concepts:
1 A nonlinear measure for the path length

$$l(P) = \max_{j=1,...m} \left(\frac{w_j(P)}{L_j} \right).$$

2 A k-shortest path approach.
3 The principle of non-dominated paths[4] to reduce the search-space.
SAMCRA also includes a fourth "look-ahead" concept. Similar to BFH, the look-ahead concept precomputes one or multiple shortest path trees rooted at the destination and then uses this information to reduce the search-space. In TAMCRA, k is fixed (giving it a polynomial complexity), but with SAMCRA this k can grow exponentially in the worst case. For the simulations with TAMCRA we set $k = 2$. A better performance can be achieved when k is increased (if k is unrestricted, all paths between s and d are returned, ordered according to their lengths).

Chen's Approximate Algorithms — Chen and Nahrstedt [15] provided two heuristics for the MCP problem: EDSP based on Dijkstra and EBF based on Bellman-Ford. In these algorithms the MCP problem is simplified by scaling down $m - 1$ (real) link weights. The user has to provide $m - 1$ values for x_i, $i = 1, ..., m - 1$, that are used as scaling factors. The algorithms then adopt a dynamic programming approach to return a path that minimizes the first (real) weight provided that the other $m - 1$ (scaled down integer) weights are within the constraints. We have chosen to implement the EBF version for our simulations. Unfortunately, to achieve a good performance, high x_i's are needed, which makes this approach computationally intensive for practical purposes.

Randomized Algorithm — Korkmaz and Krunz [16] proposed a randomized heuristic for the MCP problem that uses the concept of look-ahead. Based on look-ahead information, the algorithm randomly selects nodes that are likely to lead to a feasible path. Under the same network conditions, multiple executions of the randomized algorithm may return different paths between the same source and destination pair. For our simulations, we execute only one iteration of the randomized heuristic.

H_MCOP — Korkmaz and Krunz [17] also provided a heuristic called H_MCOP. This heuristic tries to find a path within the constraints by using the nonlinear path length function of TAMCRA and the concept of look-ahead. In addition, H_MCOP tries to simultaneously minimize the weight of a single "cost" measure along the path. H_MCOP uses two modified Dijkstra executions.

Limited Path Heuristic — Yuan [18] presented two heuristics for the MCP problem. The first "limited granularity" heuristic

---

[3] *Although not plotted here, the inefficiency of all algorithms except for SSR+DCCR increases as Δ increases. As Δ increases, more paths with small cost become feasible and the search space becomes larger. Since most algorithms do not reduce their search space, the chance of finding an optimal path is often decreased as Δ increases. SSR+DCCR can sometimes circumvent this situation by reducing the search space at the cost of increasing the execution time.*

[4] *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*.*
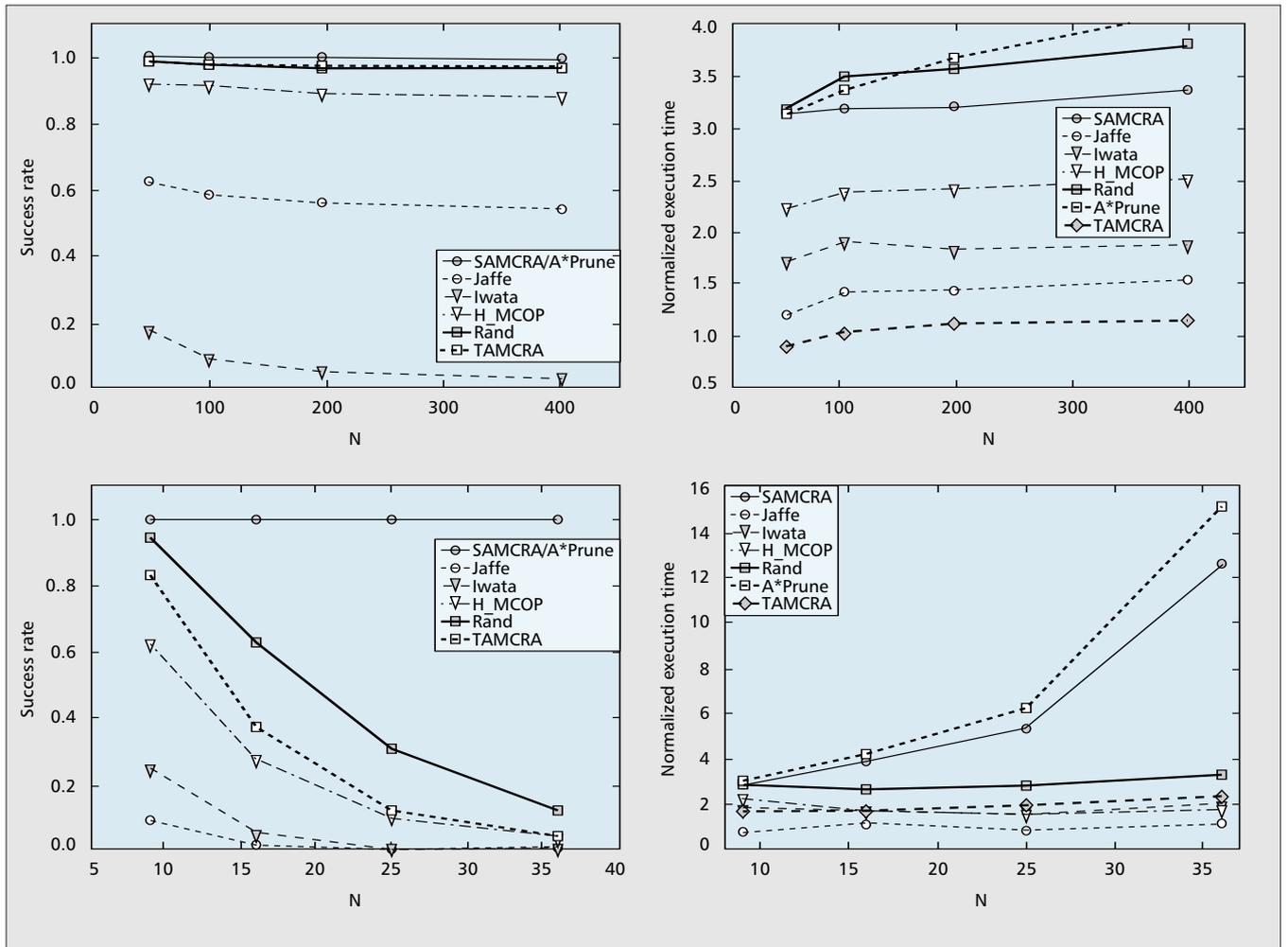
Figure 2. *Success rate (left) and normalized execution time (right) for the class of Waxman graphs (above) and lattices (below) as a function of the number of nodes (m = 2 and strict constraints).*

resembles the algorithm of Chen and Nahrstedt [15]. Therefore, we only consider the second heuristic, known as "limited path" heuristic (LPH). LPH is an extended Bellman-Ford algorithm that uses concepts (2) and (3) of TAMCRA. To conform with the queue-size allocated for TAMCRA, we let $k = 2$ for LPH.

A*Prune* — Liu and Ramakrishnan proposed A*Prune [19] and considered the problem of finding not only one but K shortest paths satisfying the constraints. For our simulations, we take $K = 1$. A*Prune uses the concept of look-ahead and then starts extracting/pruning nodes in a Dijkstra-like fashion until K feasible paths are found. A*Prune uses Jaffe's length function

$$\left(l(P) = \sum_{i=1}^{m} \frac{w_i(P)}{L_i} \right)$$

on the predicted (look-ahead) end-to-end path weights.

Performance Evaluation of MCP Algorithms

In this subsection we present and discuss the simulation results for the MCP problem. For the Waxman graphs and lattices, the weights of a link are sampled from independent and uniformly distributed random variables in the range (0, 1]. For the lattices, we consider two negatively correlated QoS measures, for which the link weights are assigned as follows: w_1 is uniformly distributed in the range (0, 1] and $w_2 = 1 - w_1$.

The choice of the constraints is important, since it determines how many (if any) feasible paths exist. We experimented with strict as well as loose constraints. In the case of loose constraints, our simulation results (not shown here) indicate that all MCP algorithms provide near-optimal success rates with small *execution times*. Accordingly, we focus our attention on the case of strict constraints. Note that for MCOP algorithms, loose constraints increase the number of feasible paths and hence the search space. This makes it difficult to find the optimal path. Fortunately, MCOP algorithms can be easily adapted to solve only the MCP part of the problem, by stopping as soon as a feasible path is reached.

The set of strict constraints is chosen as follows:

$$L_i = w_i(P), i = 1, \ldots, m$$

where P is the path for which $\max_{j=1, \ldots, m} (w_j(P))$ is minimum. In this case, only one feasible path is present in the graph (hence, MCP ≡ MCOP). This also allows us to fairly compare MCP and MCOP algorithms.

We report the *success rate* and the *normalized execution time*. The success rate of an algorithm is defined as the number of times that an algorithm returned a feasible path divided by the total number of iterations. The normalized execution time of an algorithm is defined as the execution time of the algorithm (over all iterations) divided by the execution time of Dijkstra's algorithm.

Our simulations revealed that Bellman-Ford-based algorithms (Chen's algorithm and the Limited Path Heuristic)
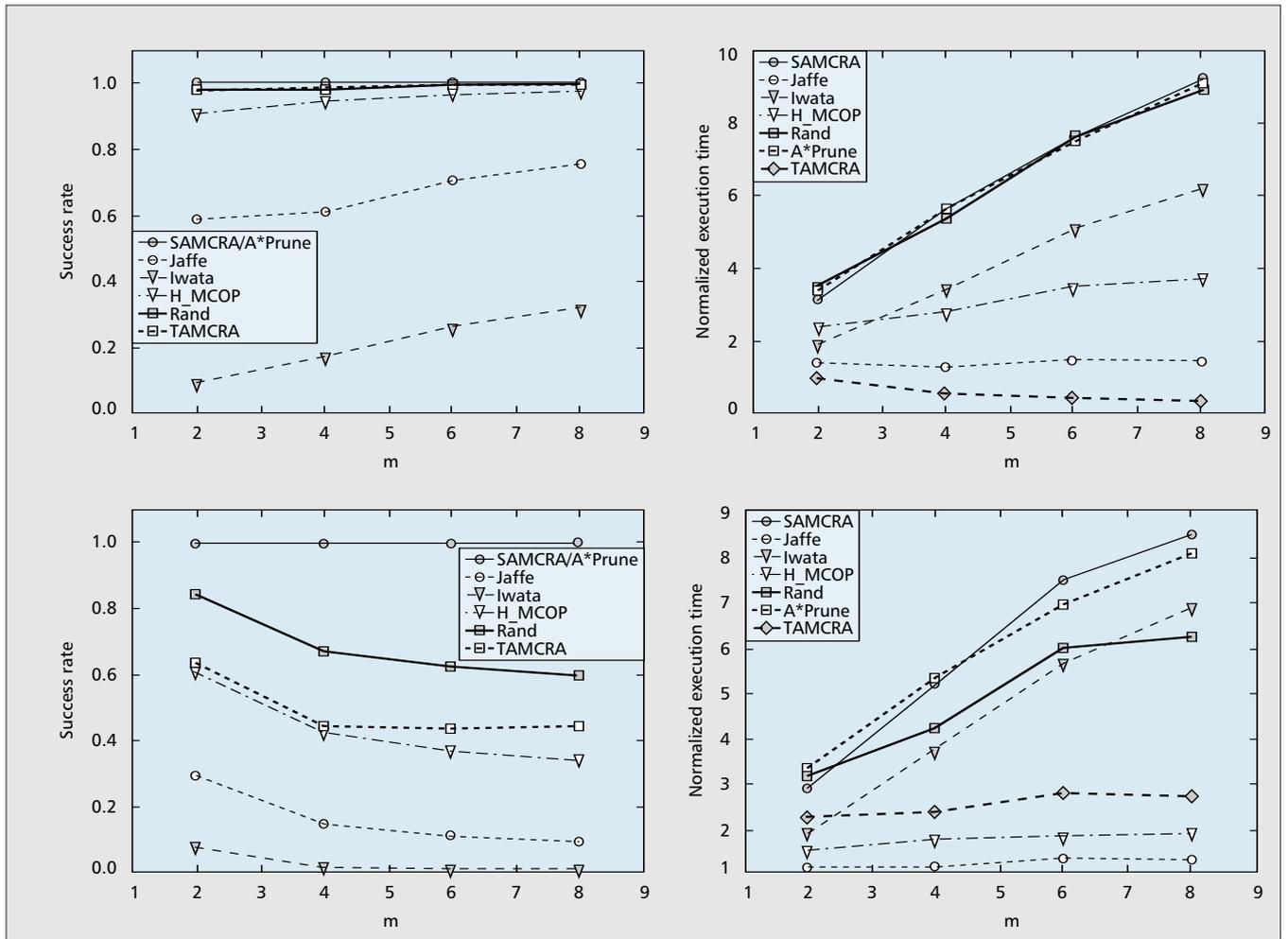
■ Figure 3. *Success rate and normalized execution time as a function of* m *under strict constraints. The results above are for Waxman graphs with* N = 100 *, and below for the lattices with* N = 49*. In both classes of graphs, the link weights are independent and uniformly distributed random variables.*

require significantly more execution time than their Dijkstra-based counterparts. Therefore, we do not include the former algorithms in the comparisons below.

Figure 2 gives the success rate and normalized execution time for the class of Waxman graphs and lattices (with negatively correlated link weights), with $m = 2$ under strict constraints. The exact algorithms SAMCRA and A*Prune always give a success rate of one. The difference in the success rate of the heuristics under strict constraints is significant. Jaffe's algorithm and Iwata's algorithm perform significantly worse than the others. In the class of two-dimensional lattices, this difference disappears as the success rates of *all* heuristics tend to zero as N increases, even for fairly moderate values of N.

The same figure also displays the normalized execution time that the algorithms used to obtain the corresponding success rate. For the class of Waxman graphs, the execution time of SAMCRA does not deviate much from the polynomial time heuristics. In fact, all algorithms display a polynomial execution time. For the class of lattices, the execution times of the exact algorithms grow exponentially, which is the price paid for exactness in hard topologies. SAMCRA and A*Prune use different length functions. The choice of a proper length function is very important, which opens the question of what is the best length function?

We have also simulated the performance of the algorithms as a function of the number of constraints m ($m = 2, 4, 6,$ and 8) under independent uniformly distributed link weights. The results for the class of Waxman graphs ($N = 100$) and Lattices

($N = 49$) are plotted in Fig. 3. We can see that the algorithms display a similar ranking in the success rate as in Fig 2. Some algorithms display a linear increase in the execution time. All these algorithms have an initialization phase in which they execute the Dijkstra algorithm m times. Finally, we can observe that if m grows, A*Prune slightly outperforms SAMCRA. This can be attributed to the non-dominance principle, which becomes of less significance as m increases. However, the time needed to check for non-dominance is only manifested in a small difference between the execution times of SAMCRA and A*Prune.

MCP Conclusions

We now present our conclusions for the considered classes of graphs, namely the Waxman graphs and the square lattices. The simulation results indicated that SAMCRA-like algorithms performed best at an acceptable computational cost, which can be attributed to the following features.

Dijkstra-Based Search with a Nonlinear Length Function: A nonlinear length function is a prerequisite for exactness. When the link weights are positively correlated, a linear approach may yield a high success rate in finding feasible paths, but under different circumstances the returned path may significantly violate the constraints. However, the choice of the best length function is not trivial. Our simulations indicated that, even on sparse graphs, a Dijkstra-like search runs significantly faster than a Bellman-Ford-like search.

Search-Space Reduction: Reducing the search-space is

always desirable because this reduces the execution time of an algorithm. The non-dominance principle is a very strong search-space reducing technique, especially when the number of constraints m is small. When m grows, the look-ahead concept together with the constraint values provide a better search-space reduction.

Tunable Accuracy through a k-shortest Path Functionality: Routing with multiple constraints and a nonlinear length function may require that multiple paths be stored at a node, necessitating a k-shortest path approach. By tuning the value of k, a good balance between success rate and computational complexity may be reached.

Look-Ahead Functionality: The look-ahead concept is based on information from path trees rooted at the destination, which are computed in polynomial time. These path trees are used to reduce the search-space and to facilitate the search for a feasible path. In the latter functionality, a predicted end-to-end path length may lead the search sooner in the correct direction, thereby saving in execution time.

The exactness of the TAMCRA-like algorithms depends on the value of k. If k is not restricted, then both MCP and MCOP problems can be solved exactly, as done by SAMCRA. Although k is not restricted in SAMCRA, simulations on Waxman graphs with independent uniformly distributed random link weights show that the execution time of this exact algorithm increases linearly with the number of nodes, providing a scalable solution to the MC(O)P problem. Simulation results also show that TAMCRA-like heuristics with small values of k render near-exact solutions. The results for the class of two-dimensional lattices with negatively correlated link weights are completely different. In such hard topologies, the heuristics are useless, whereas the exact algorithms display an exponential execution time. We believe that the best approach for such (unrealistic) graphs is via a hybrid algorithm that uses a good heuristic to make intelligent choices about which path to follow, combined with an exact SAMCRA-like algorithm that incorporates all of the four above mentioned concepts. If a solution to MCP suffices, then this algorithm should be stopped as soon as a feasible path is encountered.

Summary and Discussion

Several researchers investigated the constraint-based path selection problem and proposed various algorithms, mostly heuristics. This article has evaluated several of the algorithms that were proposed for the *restricted shortest path* and *multi-constrained (optimal) path* problems, via simulations in the class of Waxman graphs and the much harder class of two-dimensional lattices. Table 1 displays the worst-case complexities of the algorithms evaluated in this article.

The simulation results show that the worst-case complexities of Table 1 should be interpreted with care. For instance, the real execution time of H_MCOP will always be longer than that of Jaffe's algorithm under the same conditions, since H_MCOP executes Dijkstra's algorithm twice. In general, the simulation results indicate that SAMCRA-like algorithms that use a k-shortest path algorithm with a nonlinear length function while eliminating paths via the non-dominance and look-ahead concepts, provide the better performance for the considered problems (RSP, MCP, and MCOP). The performance and complexity of these algorithms is easily adjusted by controlling the value of k. When k is not restricted, the

Algorithm	Time	Space
CBF	$O(e^{\alpha N})$	$O(e^{\alpha N})$
LDP	$O(N \log N + E)$	$O(N)$
BFH	$O(N \log N + E)$	$O(N)$
LLC	$O(E^2 \log^2(E))$	$O(N)$
SSR + DCCR	$O(yE \log N + kE \log (kN) + k^2 E)$	$O(kN)$
Jaffe's algorithm	$O(N \log N + mE)$	$O(N)$
Iwata's algorithm	$O(mN \log N + mE)$	$O(N)$
SAMCRA, TAMCRA	$O(kN \log (kN) + k^2 mE)$	$O(kmN)$
EBF	$O(x_2 \ldots x_m NE)$	$O(x_2 \cdots x_m N)$
Randomized algorithm	$O(mN \log N + mE)$	$O(mN)$
H_MCOP	$O(N \log N + mE)$	$O(mN)$
A*Prune	$O(N!(m + N + N \log N))$	$O(mN!)$

■ Table 1. *Worst-case time and space complexities of the considered QoS path selection algorithms.*

SAMCRA-like algorithms lead to exact solutions. In the class of Waxman or random graphs with uniformly distributed link weights, simulation results suggest that the execution times of such exact algorithms increase linearly with the number of nodes. The exponential increase in execution time is only observed in the class of two-dimensional lattices. Heuristics perform poorly in such topologies, whereas exactness comes at a high price in complexity. In our simulations the polynomial-time ϵ-approximation schemes displayed an extensive execution time and were therefore omitted from the plots. More research is necessary to indicate whether these algorithms might provide a good alternative for exact algorithms in large and hard topologies.

References

[1] Z. Wang, *Internet QoS: Architectures and Mechanisms for Quality of Service*, Morgan Kaufman Publishers, 2001.
[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.
[3] F. A. Kuipers and P. Van Mieghem, "The Impact of Correlated Link Weights on QoS Routing," *Proc. IEEE INFOCOM Conf.*, vol. 2, Apr. 2003, pp. 1425–34.
[4] F. A. Kuipers *et al.*, "An Overview of Constraint-Based Path Selection Algorithms for QoS Routing," *IEEE Commun. Mag.*, vol. 40, no. 12, Dec. 2002, pp. 50–5.
[5] R. Widyono, "The Design and Evaluation of Routing Algorithms for Real-Time Channels," Technical Report TR-94-024, University of California at Berkeley & International Computer Science Institute, June 1994.
[6] R. Hassin, "Approximation Schemes for the Restricted Shortest Path Problem," *Mathematics of Operations Research*, vol. 17, no. 1, Feb. 1992, pp. 36–42.
[7] D. S. Reeves and H. F. Salama, "A Distributed Algorithm for Delay-Constrained Unicast Routing," *IEEE/ACM Trans. Net.*, vol. 8, no. 2, Apr. 2000, pp. 239–50.
[8] A. Juttner *et al.*, "Lagrange Relaxation-Based Method for the QoS Routing Problem," *Proc. IEEE INFOCOM Conf.*, vol. 2, April 2001, pp. 859–68.
[9] L. Guo and I. Matta, "Search Space Reduction in QoS Routing," *Proc. 19th IEEE Int'l. Conf. Distributed Computing Systems*, May 1999, pp. 142–49.
[10] P. Van Mieghem *et al.*, Eds., "Quality of Service Routing," Ch. 3 in *Quality of Future Internet Services*, EU-COST 263 Final Report, Ed. by Smirnov *et al.* in Springer LNCS 2856, 2003, pp. 80–117.

[11] J. M. Jaffe, "Algorithms for Finding Paths with Multiple Constraints," *Networks*, vol. 14, 1984, pp. 95–116.
[12] A. Iwata *et al.*, "ATM Routing Algorithms with Multiple QoS Requirements for Multimedia Internetworking," *IEICE Trans. and Commun.*, vol. E79-B, no. 8, 1996, pp. 999–1006.
[13] H. De Neve and P. Van Mieghem, "TAMCRA: A Tunable Accuracy Multiple Constraints Routing Algorithm," *Computer Communications*, vol. 23, 2000, pp. 667–79.
[14] P. Van Mieghem and F. A. Kuipers, "Concepts of Exact Quality of Service Algorithms," to appear in *IEEE/ACM Trans. Net.*, 2004.
[15] S. Chen and K. Nahrstedt, "On Finding Multi-Constrained Paths," *Proc. ICC'98 Conf.*, New York, 1998, pp. 874–9.
[16] T. Korkmaz and M. Krunz, "A Randomized Algorithm for Finding a Path Subject to Multiple QoS Requirements," *Computer Networks*, vol. 36, 2001, pp. 251–68.
[17] T. Korkmaz and M. Krunz, "Routing Multimedia Traffic with QoS Guarantees," *IEEE Trans. Multimedia*, vol. 5, no. 3, Sept. 2003, pp. 429–43.
[18] X. Yuan, "Heuristic Algorithms for Multiconstrained Quality-of-Service Routing," *IEEE/ACM Trans. Net.*, vol. 10, no. 2, April 2002, pp. 244–56.
[19] G. Liu and K. G. Ramakrishnan, "A*Prune: An Algorithm for Finding *k*-shortest Paths Subject to Multiple Constraints," *Proc. IEEE INFOCOM Conf.*, vol. 2, April 2001, pp. 743–9.

Biographies

FERNANDO A. KUIPERS (F.A.Kuipers@ewi.tudelft.nl) received the M.Sc. degree in electrical engineering at the Delft University of Technology in 2000. Currently he is working toward his Ph.D. in the network architectures and services group at the faculty of electrical engineering, mathematics and computer science (EEMCS), Delft University of Technology. He was a member of the DIOC (Delft Interdisciplinary Research Center) for the design and management of infrastructures, where he took part in the telecommunications project. His Ph.D. work focuses on the algorithmic aspects and complexity of Quality of Service routing.

TURGAY KORKMAZ received the B.Sc. degree with the first ranking from the department of computer science and engineering at Hacettepe University, Ankara, Turkey, in 1994, and two M.Sc. degrees in computer engineering from Bilkent University, Ankara, and computer and information science at Syracuse University, Syracuse, NY, in 1996 and 1997, respectively. He received his PhD in electrical and computer engineering from the University of Arizona, Tucson, AZ, in December 2001. In January 2002, he joined the University of Texas at San Antonio, where he is currently an assistant professor in the computer science department. His research interests include QoS-based routing, multi-constrained path selection, efficient dissemination of network-state information, topology aggregation in hierarchical networks, and performance evaluation of QoS-based routing protocols.

MARWAN M. KRUNZ is an associate professor of electrical and computer engineering at the University of Arizona. His current research interests include power control, routing, and MAC design for ad hoc networks; media streaming over wireless channels; Quality of Service and scheduling for packet-based cellular networks; constrained-based routing; and prefetching/caching for the web. He previously worked on traffic management and control issues for packet video. He has published more than 70 articles and peer-reviewed conference papers in the above topics and has filed for several related patents. He is a recipient of the National Science Foundation CAREER Award (1998). He currently serves on the editorial board for the *IEEE/ACM Transactions on Networking* and *Computer Communications Journal*. He was the guest-editor for a number of special issues in IEEE magazines. He is the program co-chair for the IEEE INFOCOM 2004 Conference in Hong Kong, and was the technical program co-chair for the 9th Hot Interconnects Symposium in 2001. He has served and continues to serve on the executive and technical program committees of numerous international conferences. He continuously serves on NSF panels, and is a consultant for several U.S. corporations.

PIET VAN MIEGHEM is a professor at the Delft University of Technology and chairman of the basic unit network architectures and services (NAS). His research focuses on new Internet-like architectures for the future, and broadband and QoS-aware networks. Professor Van Mieghem received Master's and Ph.D. degrees in electrical engineering from the K.U. Leuven (Belgium) in 1987 and 1991, respectively. Before joining Delft he worked at the Interuniversity Micro Electronic Center (IMEC) from 1987 to 1991. From 1992 to 1993 he was a visiting scientist at MIT in the department of electrical engineering. From 1993 to 1998 he was a member of the Alcatel Corporate Research Center in Antwerp, where he was engaged in performance analysis of ATM systems and in network architectural concepts of both ATM networks (PNNI) and the Internet.