A Review of Constraint-Based Routing Algorithms

F.A. Kuipers*, T. Korkmaz, M. Krunz and P. Van Mieghem

June 14, 2002

Abstract

Constraint-based routing is an invaluable part of a full-fledged Quality of Service (QoS) architecture. Unfortunately, routing with multiple additive constraints is known to be a NP-complete problem. Hence, accurate constraint-based routing algorithms with a fast running time are scarce, perhaps even non-existent. The expected impact of such a constrained-based routing algorithm has resulted in the proposal of numerous heuristics and a few exact QoS algorithms. Although at times an overview of QoS algorithms has been published, a comparative study to the performance of QoS algorithms is still missing. With this paper we attempt to fill this gap.

This paper aims to give a thorough, concise and fair evaluation of the most important constraint-based routing algorithms known today. We will provide a descriptive overview of *restricted shortest path* algorithms and *multi-constrained path* algorithms. A performance evaluation of these two classes of algorithms is presented based on complexity analysis and simulation results.

1 Introduction

The continuous demand for using multimedia applications over the Internet has triggered a spur of research on how to satisfy the Quality of Service (QoS) requirements of these applications, e.g. requirements regarding bandwidth, delay, jitter, and reliability. These efforts resulted in the proposals of several QoS-based frameworks, such as the Integrated Services (IntServ) [14], Differentiated Services (DiffServ) [12] and Multi-Protocol Label Switching (MPLS) [7], [71]. One of the key issues in providing QoS guarantees is *how to determine paths that satisfy QoS constraints*. Solving this problem is known as QoS routing or constraint-based routing. All the above mentioned QoS-based frameworks will benefit and therefore should deploy QoS routing, as described later.

The research community has extensively studied the QoS routing problem, resulting in many QoS routing algorithms. In 1998, Chen and Nahrstedt [18] provided an overview of the majority of QoS routing algorithms known at that time, however without comparing them. In the mean time, many other QoS routing algorithms have emerged, but still no comparative study has been performed. In this paper, we aim at filling this gap for unicast[1] QoS routing algorithms, which try to find a path between a *source* node and a *destination* node that satisfies a set of constraints.

Routing in general involves two identities [39], namely the *routing protocol* and the *routing algorithm*. The *routing protocol* has the task of capturing the state of the network and its available network

*Corresponding author, F.A.Kuipers@its.tudelft.nl

[1] Multicast QoS routing faces different conceptual problems as discussed in [53]. An overview of several multicast QoS algorithms has been given in [72] and more recently in [81].

resources and distributing this information throughout the network. The *routing algorithm* uses this information to compute shortest paths. Current best-effort routing performs these tasks based on a single measure like hopcount or delay. QoS routing however, must take into account multiple QoS requirements, link dynamics, as well as the implication of the selected routes on network utilization, turning QoS routing into a notoriously challenging problem. Despite its difficulty, we argue that QoS routing is invaluable in a network architecture that needs to satisfy traffic and service requirements ([23, 85, 61, 34]). For example, in the context of ATM (PNNI), QoS routing is performed by source nodes to determine suitable paths for connection requests. These connection requests specify QoS constraints that the path must obey. Since ATM is a connection-oriented technology, a path selected by PNNI will remain in use for a potentially long period of time. It is therefore important to choose a path with care. The IntServ/RSVP framework is also able to guarantee some specific QoS constraints. However, this framework relies on the underlying IP routing table to reserve its resources. As long as this routing table is not QoS-aware, paths may be assigned that cannot guarantee the constraints, which will result in blocking. In MPLS, which is a convergence of several efforts aimed at combining the best features of IP and ATM [19], a source node selects a path, possibly subject to QoS constraints, and uses a signaling protocol (e.g., RSVP or CR-LDP) to reserve resources along that path. In the case of DiffServ, QoS-based routes can be requested, for example, by network administrators for traffic engineering purposes. Such routes can be used to guarantee a certain service level agreement (SLA)[85]. These examples all indicate the importance of constraint-based routing algorithms, both in ATM and IP. Depending on the frequency at which constrained paths are requested, the computational complexity of finding a path subject to multiple constraints is often a complicating but decisive factor.

To enable QoS routing, it is necessary to implement state-dependent, QoS-aware networking protocols. Examples of such protocols are PNNI [6] of the ATM Forum and the QoS-enhanced OSPF protocol [5]. For the first task in routing (i.e., the representation and dissemination of network-state information), both OSPF and PNNI use link-state routing, in which every node tries to acquire a "map" of the underlying network topology and its available resources via flooding [11]. Despite its simplicity and reliability, flooding involves unnecessary communications and causes inefficient use of resources, particularly in the context of QoS routing that requires frequent distribution of multiple, dynamic parameters, e.g., using triggered updates [4]. Creating efficient QoS routing protocols is still an open issue that needs to be investigated further [55, 15]. Hereafter, we assume that the network-state information is temporarily static and has been distributed throughout the network and is accurately maintained at each node using QoS link-state routing protocols. Once a node acquires the network-state information, it performs the second task in QoS routing, namely computing paths based on multiple QoS constraints. In this paper, we mainly focus on this so-called *multi-constrained path* selection problem and consider numerous proposed algorithms. Before giving the formal definition of the *multi-constrained path* problem, let us describe the notation that is used throughout this paper.

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 number of QoS measures (e.g. delay, hopcount, ...) is denoted by m. Each link is characterized by a m-dimensional link weight vector, consisting of m non-negative QoS

weights ($w_i(u,v)$, $i = 1, ..., m$, $(u,v) \in E$) as components. The QoS measure of a path can either be additive (e.g., delay, jitter, the logarithm of packet loss), in which case the weight of that measure equals the sum of the QoS weights of the links defining that path. Or the weight of a QoS measure of a path can be the minimum(maximum) of the QoS weights along the path (e.g., available bandwidth and policy flags). Constraints on min(max) QoS measures can easily be treated by omitting all links (and possibly disconnected nodes) which do not satisfy the requested min(max) QoS constraints. We call this topology filtering. In contrast, constraints on additive QoS measures cause more difficulties. Hence, without loss of generality, we assume all QoS measures to be additive.

The basic problem considered in this paper can be defined as follows:

Definition 1 *Multi-Constrained Path (MCP) problem:* Consider a network $G(N,E)$. Each link $(u,v) \in E$ is specified by a link weight vector with as components m additive QoS weights $w_i(u,v) \geq 0$, $i = 1, ..., m$. Given m constraints L_i, $i = 1, ..., m$, the problem is to find a path P from a source node s to a destination node d such that:

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

A path that satisfies all m constraints is often referred to as a feasible path. There may be multiple different paths in the graph $G(N,E)$ that satisfy the constraints. According to **definition 1**, any of these paths is a solution to the MCP problem. However, it might be desirable to retrieve the path with smallest length $l(P)$ from the set of feasible paths. This problem is called the *multi-constrained optimal path* problem and is formally defined as follows:

Definition 2 *Multi-Constrained Optimal Path (MCOP) problem:* Consider a network $G(N,E)$. Each link $(u,v) \in E$ is specified by a link weight vector with as components m additive QoS weights $w_i(u,v) \geq 0$, $i = 1, ..., m$. Given m constraints L_i, $i = 1, ..., m$, the problem is to find a path P from a source node s to a destination node d such that:

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

(ii) $l(P) \leq l(P^*)$, $\forall P^*, P$ satisfying (i)

$l(P)$ can be any function of the weights $w_i(P)$, $i = 1, ..., m$, provided it obeys the criteria for "length" or "distance" in vector algebra (see [79], Appendix A). Minimizing a properly chosen length function, can result in an efficient use of the network resources and/or result in a reduction of monetary cost.

In general, MCP, irrespective of path optimization, is known to be a NP-complete problem [29]. Under the umbrella of the MCP problem, another NP-complete problem has received most attention, namely the *restricted shortest path* problem:

Definition 3 *Restricted Shortest Path (RSP) problem*: Consider a network $G(N,E)$. Each link $(u,v) \in E$ is specified by two nonnegative measures: 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 $d(P) \stackrel{\text{def}}{=} \sum_{(u,v) \in P} d(u,v)$.

Because the problems defined in **definitions 1-3** are NP-complete [1, 29], they are considered to be intractable for large networks. Accordingly, mostly heuristics have been proposed for these problems. In this paper, we briefly describe the lion's share of the published QoS algorithms and compare them based on extensive simulations. Complexity will be an important criterion for evaluating the algorithms. Complexity refers to the intrinsic minimum amount of resources needed to solve a problem or execute an algorithm. Complexity can be divided into *time* complexity and *space* complexity. In this paper we will only look at on the *computational time-complexity*. Throughout this paper, we consider both the worst-case complexity and the execution time. There can be a significant difference between these measures, as shown by Kuipers and Van Mieghem in [52]. They show that, under certain conditions and on average, the MCP problem can be solved in polynomial time despite its worst-case NP-complete complexity. Moreover, there exist specific classes of graphs, for which the MCP problem is not NP-complete at all, e.g. if each node has only two neighbors.

The remainder of this paper is divided into two parts, which can be read independently. In Section 2, we consider the RSP problem, give a survey of the algorithms that target this problem, evaluate their performance using simulations, and provide a conclusion. Section 3 adopts the same approach for the MCP problem. For completeness Section 4 touches upon QoS algorithms that were designed to operate in specific cases. Section 5 provides the summary and discussion.

2 RSP Algorithms

In this section, we consider the Restricted Shortest Path (RSP) problem. In the literature, the RSP problem is also studied under different names such as delay-constrained least-cost (DCLC) path, minimum-cost restricted-time (MCRT) path, or constrained shortest path (CSP). Garey and Johnson [29] and Ahuja *et al.* [1] have shown that the RSP problem is a NP-complete problem. To cope with this NP-completeness, researchers have mainly resorted to heuristics and approximation algorithms. We will first describe and classify the RSP algorithms according to their fundamental properties in Section 2.1. We then compare these algorithms in Section 2.2. Finally, we provide a brief conclusion in Section 2.3.

2.1 Overview of RSP Algorithms

In this subsection, we classify the RSP algorithms and briefly describe them.

2.1.1 Exact Algorithms

The exact solution to the RSP problem can be found by systematically examining every path in a brute-force manner. However, since the number of paths grows exponentially, this method requires an exponential number of operations in the worst case. Hence, it may not be useful in practice. To provide a practical exact solution to the RSP problem, Widyono [84] proposed an algorithm called Constrained Bellman-Ford (CBF). This algorithm, finds independent minimum cost paths between a source and a set of destination nodes[2] of which each has its own delay constraint. The basic idea behind this algorithm is to systematically discover the lowest-cost paths while monotonically increasing

[2] Here we only consider the single destination case.

delay. CBF maintains a list of paths from the source node to each other node with increasing cost and decreasing delay. The algorithm selects a node whose list contains a path that satisfies the delay constraint and that has the minimum cost. CBF then explores the neighbors of this node using a breadth-first search [22], and (if necessary) adds new paths to the list maintained at each neighbor. This process continues as long as the delay constraint is satisfied and there exists a path to explore further. Although CBF exactly solves the RSP problem, its execution time grows exponentially in the worst case. We have implemented this algorithm as a reference point for exact algorithms and have measured its execution time through simulations.

The RSP problem can also be solved exact via pseudo-polynomial-time algorithms. An example of an such algorithm is shown in Figure 1. It is assumed that $d(u, v)$ and Δ have a finite granularity and are represented by non-negative integers, i.e. they are expressed as an integer number of a basic unit. The idea here is to iteratively compute the minimum cost paths whose delays are equal to r,

Pseudo-polynomial-RSP$(G(N, E), s, t, \Delta)$
1 $\mathcal{C}_v[i] = \infty$ $\forall v$ and for $i = 0, 1, 2, \ldots, \Delta$
2 $\mathcal{C}_s[i] = 0$ for $i = 0, 1, 2, \ldots, \Delta$
3 $\mathcal{P}_v[i] = NIL$ $\forall v$ and for $i = 0, 1, 2, \ldots, \Delta$
4 **for** $r = 0$ to $r \leq \Delta$ do
5 **for** each $(u, v) \in E$ do
6 **if** $r + d(u, v) \leq \Delta$ and $\mathcal{C}_u[r] + c(u, v) < \mathcal{C}_v[r + d(u, v)]$ then
7 $\mathcal{C}_v[r + d(u, v)] = \mathcal{C}_u[r] + c(u, v)$
8 $\mathcal{P}_v[r + d(u, v)] = u$
9 **end if**
10 **end for**
11 **end for**
12 The cost of the optimal path is $\min\{\mathcal{C}_t[i] \mid i = 0, 1, 2, \ldots, \Delta\}$

Figure 1: Pseudo-polynomial-time algorithm for the RSP problem.

where $r = 0, 1, 2, \ldots, \Delta$. A cost vector $C_v[r]$ is associated with each node v and stores the minimum cost from s to v that has a total delay of r. The vector of $P_v[k]$ represents the previous node of v on the minimum cost path. The complexity of this algorithm is $O(\Delta E)$ since two nested loops in lines 4 and 5 are executed for $r = 0, 1, 2, \ldots, \Delta$ and for each link $(u, v) \in E$. In general, the complexity of pseudo-polynomial-time algorithms depends on the actual values of the input data (e.g., the given delay constraint) in addition to the size of the network. Pseudo-polynomial-time algorithms can require a large execution time if the value of the input data is large. This can happen if the granularity is very small.

2.1.2 ϵ-Optimal Approximation Algorithms

One general approach in dealing with NP-complete problems is to look for an approximation algorithm (i.e. heuristic) with a polynomial complexity that guarantees to find a quantifiable close solution to

the optimal one [45]. If an algorithm is ϵ-optimal, it returns a path with a length at most $(1+\epsilon)$ times the length of the optimal path, where $\epsilon > 0$. For the RSP problem, Hassin [37] provided two ϵ-optimal approximation algorithms with the complexities of $O((\frac{EN}{\epsilon}+1)\log\log B)$ and $O(\frac{EN^2}{\epsilon}\log(\frac{N}{\epsilon}))$, where B is an upper bound on the cost of a path. It is assumed that the link weights are positive integers.

The first ϵ-optimal approximation algorithm initially determines an upper bound (UB) and a lower bound (LB) on the optimal cost denoted by OPT. 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 these bounds are found, the approximation algorithm bounds the cost of each link by rounding and scaling it according to: $c'(u,v) = \left\lfloor \frac{c(u,v)(N-1)}{\epsilon LB} \right\rfloor \ \forall\ (u,v) \in E$. Finally, it applies a pseudo-polynomial-time algorithm on these modified weights that is similar to the one presented in Figure 1.

The second approximation algorithm uses a slightly different technique called interval partitioning, in which a set of positive numbers $Q = \{p_1, p_2, \ldots, p_m\}$ is partitioned into subsets $R_1, \ldots, R_{r+1}$ such that $p_i \in R_j$ if and only if $\frac{X(j-1)}{r} < p_i \leq \frac{Xj}{r}$ for $j = 1, \ldots, r$, and $p_i \in R_{r+1}$ if and only if $p_i > X$, where X is a given positive number. Phillips [68] provided another ϵ-optimal approximation using a Dijkstra-based algorithm with the complexity of $O(EN(1+\frac{1}{\epsilon}) + N^2(1+\frac{1}{\epsilon})(\log N + \log(1+\frac{1}{\epsilon})))$.

Approximation algorithms perform better in minimizing the cost of a returned feasible path as ϵ goes to zero. However, a smaller value for ϵ leads to an increased complexity. Orda [64] and Lorenz *et al.* [57] modified ϵ-optimal approximation algorithms to scale better in hierarchical networks. Ergun *et al.* [27] proposed an ϵ-optimal approximation algorithm for a RSP-related problem, in which one link weight is a function of the other. Goel *et al.* [30] considered a related problem, in which the least-cost path from a given source to all destinations is searched while satisfying the delay constraint Δ for each path. For this problem, Goel *et al.* provided an ϵ-approximation algorithm with the complexity of $O((E + N\log N)\frac{D}{\epsilon})$, where D can be at most $N-1$.

2.1.3 Backward-Forward Heuristic Algorithms

Reeves and Salama [70] proposed a distributed heuristic for the RSP problem, called the delay-constrained unicast routing (DCUR) algorithm. The complexity of this algorithm is $O(N^2)$. The algorithm explores the graph based on the concatenation of two segments: (1) the so-far explored path from the source s to an intermediate node u; and (2) the least-delay or the least-cost path from the node u to the destination d. Sun and Langendorfer [76] considered the same DCUR algorithm and improved the original DCUR by reducing the complexity to $O(N)$. Ishida *et al.* [41] considered a similar algorithm to DCUR and discussed its use in multipath routing. Sriram *et al.* [73] used similar ideas to provide a distributed algorithm based on probing and backtracking. This algorithm sends a probe packet over the preferred links one at a time. If it is accepted, the probe packet is forwarded to the next node. Otherwise, it is rejected and the algorithm tries the next preferred link. The least delay and least cost paths from every node u to destination d are used as in the previous algorithms.

While there are some differences in the distributed versions of the four above mentioned algorithms, the main property behind these algorithms leads to a centralized heuristic that involves searching the underlying graph in backward and forward directions. Since we have implemented other algorithms in a centralized manner, we have implemented the backward-forward heuristic (BFH) as follows: BFH first

determines the least-delay path (LDP) and the least-cost path (LCP) from every node u to destination d. This can be done by executing Reverse-Dijkstra [1] w.r.t. $d(u,v)$ and $c(u,v)$. BFH then starts from the source node s and explores the graph as in Dijkstra's algorithm with the following modification in the relaxation procedure: link (u,v) is relaxed if $d[u] + d(u,v) + d[\text{LDP from } v \text{ to } d] \leq \Delta$ and $c(u) + c(u,v) < c(v)$. BFH extracts nodes that have minimum cost. The computational complexity of BFH equals two times Reverse-Dijkstra plus one modified Dijkstra execution, summing up to tree times the complexity of Dijkstra's algorithm.

2.1.4 Lagrangian-based Linear Composition Algorithms

The Lagrangian-based linear composition algorithm linearly combines the delay and cost of each link and finds the shortest path w.r.t. this single measure. The weight of a link thus becomes $w(u,v) = \alpha d(u,v) + \beta c(u,v)$, where α, β are the multipliers. With this approach there is no guarantee that the returned path is within the delay constraint and minimizes the cost. A key issue here is how to determine the appropriate multipliers when combining the delay and cost. Aneja and Nair [3] proposed a systematic way of searching the appropriate multipliers to combine delay and cost as follows: the algorithm iteratively finds the shortest path w.r.t. a linear combination of delay and cost. At each iteration, it adjusts the multipliers of delay and cost in the linear combination to approach the optimal path. Aneja and Nair demonstrated the similarity of this method to the generalized Lagrangian technique (see also Kuhn-Tucker conditions [51]). They also showed that the search takes finite iterations of Dijkstra's algorithm assuming that the weights of the paths are uniformly distributed in the delay-cost space.

Several researchers [36], [13], [44] have considered the Lagrangian-based search and independently reached the above mentioned approach along with some extensions. For example, Handler and Zang [36] considered a k-shortest path algorithm [26] to close the gap between the optimal solution and the returned path based on the linear combination. Although the computational results indicate an order of magnitude savings, the amount of time to determine an optimal path may be excessive in some cases. Juttner *et al.* [44] showed that the worst-case complexity of our implemented algorithm is $O(E^2 \log^2(E))$.

2.1.5 Hybrid Algorithms

Guo and Matta [35] select the cost of the least-delay path as the cost constraint. Then they try to solve the RSP problem through the minimization of a nonlinear length function, analogue to TAMCRA (see Section 3.1.3), that gives more priority to lower cost paths. To minimize the nonlinear length function, they propose an algorithm called DCCR that uses a k-shortest path algorithm. The performance of the DCCR algorithm depends on k. If k is large, the algorithm gives good performance at the expense of an increased execution-time. In order to improve the performance with small values of k, Guo and Matta tried to reduce the search space and tighten the cost bound by using a Lagrangian-based algorithm before applying DCCR. This final hybrid algorithm is called SSR+DCCR. The complexity of this final algorithm depends on that of the Lagrangian-based algorithm and the k-shortest path algorithm. In our implementation, we used the above presented Lagrangian-based algorithm and the k-shortest paths algorithm in [20], leading to a total complexity of $O(xE \log N + kE \log(kN) + k^2 E)$.

2.2 Performance Comparison of RSP Algorithms

In this subsection, we compare the RSP algorithms through simulations, for which we use Waxman graphs [83], [80]. Waxman graphs are often chosen in simulations as topologies resembling communication networks. Moreover, these graphs are easy to generate, allowing us to evaluate 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 significant conclusions. In each simulation with 50, 100 and 200 nodes, we generated 10^4 Waxman graphs and selected node 1 and node N as the source and destination node, respectively. In each graph, the delay and the cost of every link (u, v) are independent uniformly distributed random variables.

We select the delay constraint Δ as follows: 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 $d(\text{LCP}) \leq \Delta$, then the LCP is the optimal path. Since these two cases are easy to deal with, we want to 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, we select per graph five different monotonically increasing values for Δ in the range $(d(\text{LDP}), d(\text{LCP}))$, as follows:

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

All considered RSP algorithms are capable of finding a feasible path (if any) that satisfies the given delay constraint Δ. With our choice of the delay constraint there is always a feasible path present. 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. The considered RSP algorithms may return different feasible paths with different costs. We compare the algorithms based on how *efficient* or *inefficient* they are in minimizing the cost of a returned feasible path, when compared to the exact algorithm that finds a feasible path with the minimum cost. The *inefficiency* of an algorithm A is defined as

$$inefficiency_A \stackrel{\text{def}}{=} \frac{cost(A) - cost(Exact\_Algorithm)}{cost(Exact\_Algorithm)}$$

where $cost(\cdot)$ is the average cost of the feasible paths that are returned by a given algorithm. This performance measure is also used in other papers, such as [35]. The other performance measure we use is the *complexity*. In addition to the worst-case complexity reported in the previous subsection, we measure the *execution time* of the compared algorithms. These execution times are normalized by the execution time of Dijkstra's algorithm.

Initial simulation results have indicated that the execution times of the ϵ-optimal approximation algorithms (even when $\epsilon = 1$) are much larger than that of the other compared algorithms. Therefore, we excluded the ϵ-approximations from our simulations. We compared the following algorithms: the exact CBF, the least delay path (LDP), Lagrangian-based Linear Composition (LLC), Backward-forward heuristic (BFH), DCCR, and SSR+DCCR. The behavior of these algorithms for different delay constraints is similar for $N = 50, 100, 200$. Therefore we have averaged the results for the five delay constraints. We have plotted the *inefficiency* and the *execution time* of the algorithms in Figure 2. In all cases, the basic LDP algorithm had the highest *inefficiency* and the lowest *complexity*.

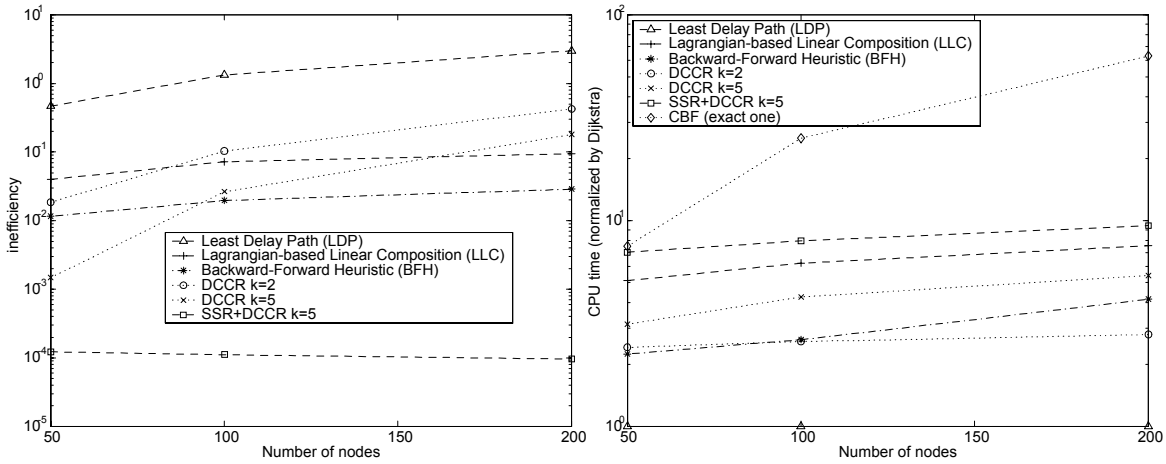

Figure 2: Scaling of the performance measures with N.

With a slight increase in execution time (on average two times that of Dijkstra's algorithm), BFH has a significantly lower *inefficiency* than the LDP algorithm. Actually, BFH also has a lower *inefficiency* (even in 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* than the other algorithms as k increases, at the expense of a longer execution time.

The complexity of the exact CBF algorithm is linearly increasing with the value of Δ while the complexity of other algorithms does not significantly change with Δ, suggesting that CBF can be used when Δ is small. The *inefficiency* of all algorithms except for SSR+DCCR increases as Δ increases. The reason is that as Δ increases, more paths with small cost become feasible and the search space becomes larger. However, since the other algorithms do not reduce their search space as SSR+DCCR does, their chance of finding an optimal path is often decreased as Δ increases. SSR+DCCR circumvents this situation by reducing its search space, and achieves a lower *inefficiency* than the other simulated algorithms.

2.3 RSP Conclusions

Our conclusions for the *restricted shortest path* problem are valid for the considered class of Waxman graphs, with independent uniformly distributed link weights. According to [80], the conclusions will also be valid for the class of random graphs, with the same link weight distribution.

In general, the simulations indicated that a higher *efficiency* is 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. Such an algorithm should start with BFH instead of LLC and (if needed) continue to use a k-shortest path algorithm with a nonlinear length function, as in DCCR. The main advantage of a hybrid algorithm would be to initially determine a good path with a small execution time and to improve the *efficiency* while controlling the *complexity* with the value of k.

Summarizing, the concepts that render the best RSP algorithm among the set of evaluated RSP algorithms, are: a nonlinear length function, search space reduction, tunable accuracy through a k-shortest path algorithm and a look-ahead (predictive) property. These concepts will also lead to a

better performance for the MCP problem in the following section.

3 MCP Algorithms

In this section we focus on the MCP algorithms, including some MCOP algorithms. The MCP and MCOP problems are defined in **definitions 1** and **2**, respectively.

In Section 3.1 we briefly describe the MCP algorithms. Subsequently we compare these algorithms via simulations in Section 3.2. Finally, we provide a brief conclusion in Section 3.3.

3.1 Overview of MCP Algorithms

3.1.1 Jaffe's Approximate Algorithm

Jaffe [43] presented two algorithms to deal with the MCP problem under two constraints. The first method is an exact pseudo-polynomial-time algorithm that has an unattractive worst-case complexity of $O(N^5 b \log Nb)$, where b is the largest weight in the graph. Because of this prohibitive complexity, we will only discuss Jaffe's second algorithm, which we will further refer to as Jaffe's algorithm. Jaffe proposed to use a shortest path algorithm on a linear combination of the two link weights:

$$w(u,v) = d_1 \cdot w_1(u,v) + d_2 \cdot w_2(u,v) \tag{1}$$

where d_1 and d_2 are positive multipliers. Figure 3 visualizes how the shortest path w.r.t. the linear combination of two link weights is determined.

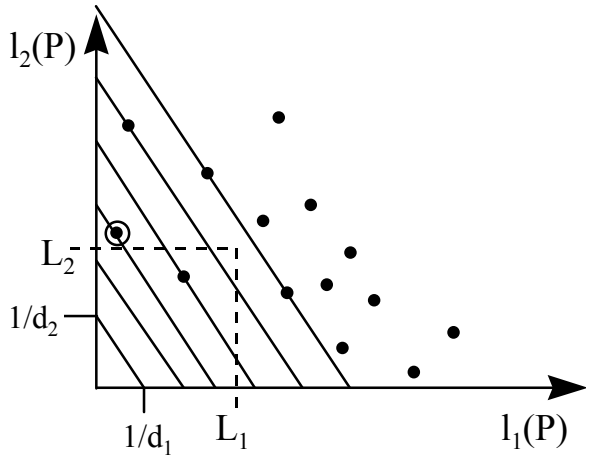

Figure 3: Distribution of the paths. Jaffe's scanning procedure first encounters the encircled node, which accordingly represents the path with minimal length.

Each line in Figure 3 shows equilength paths w.r.t. the linear length definition (1). Jaffe's search is visualized by a line starting in the origin and moving outward according to the values of the multipliers. As soon as this line hits a path (i.e., encircled black dot), the algorithm returns this path as the shortest one w.r.t. the linear length definition (1). Figure 3 also illustrates that the shortest path based on a linear combination of link weights does not necessarily reside within the constraints.

Jaffe had also noticed this and therefore provided the following nonlinear definition for the path length $f(P) = \max\{w_1(P), L_1\} + \max\{w_2(P), L_2\}$, whose minimization can guarantee to find a feasible path if such a path exists. However, because no simple shortest path algorithm can cope with this nonlinear length function, Jaffe tried to approximate it by using the above mentioned linear length function (1). Andrew and Kusuma [2] extended Jaffe's analysis to an arbitrary number of constraints m, extending the linear length function to

$$l(P) = \sum_{i=1}^{m} d_i w_i(P) \tag{2}$$

and the nonlinear function to

$$f(P) = \sum_{i=1}^{m} \max\left(w_i(P),\, L_i\right)$$

Andrew and Kusuma proved that for all positive values of the multipliers $\frac{l(P)}{f(P)} \leq 2$. This performance bound on $\frac{l(P)}{f(P)}$ can be reduced to $2 - \frac{1}{m}$ by choosing $d_i = \frac{1}{L_i^{1/m}}$. Obviously this works best for small m. However, under different circumstances, a different choice of the multipliers may render a better performance. For the simulations we have used $d_i = \frac{1}{L_i}$. Furthermore, we have used Dijkstra's algorithm along with Fibonacci heaps, leading to a complexity for Jaffe's algorithm of $O(N \log N + mE)$.

If the returned path is not feasible, then Jaffe's algorithm returns this path and stops, but the search could be continued by using different values for d_i, which might lead to finding a feasible path. Unfortunately, in some cases, even if all possible combinations of d_i are exhausted, a feasible path may not be found using linear search. For exactness, it is therefore necessary to use a nonlinear length function, even though such a function cannot be minimized through a simple shortest path algorithm.

3.1.2 Iwata's Algorithm

Iwata *et al.* [42] proposed a polynomial-time algorithm to solve the MCP problem. The algorithm 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. A similar approach has been proposed by Lee *et al.* [54]. In the simulations we will evaluate the algorithm of Iwata *et al.* and refer to it as Iwata's algorithm.

The problem with the proposed approach is that there is no guarantee that any of the shortest paths for each measure individually, is close to a path within the constraints. This is illustrated in Figure 4, which shows the twenty shortest paths of a two-constraint problem applied to a random graph with 100 nodes. Here we see that only the 2nd and 3rd shortest path for measure 1 and the 2nd and 4th shortest path for measure 2 lie within the constraints.

Iwata's (and likewise Jaffe's) algorithm will perform best if the link weights are positively correlated, because then if one weight is small, the other weights are most likely also relatively small.

In our simulations we will only consider one shortest path per QoS measure computed via Dijkstra's algorithm, leading to a complexity of $O(mN \log N + mE)$.

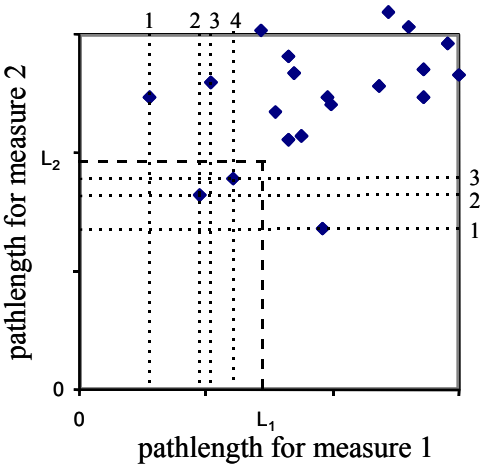

Figure 4: Twenty shortest paths for a two-constraint problem. Each path is represented as a dot and the coordinates of each dot are its path-length for each metric individually.

3.1.3 SAMCRA: A Self-Adaptive Multiple Constraints Routing Algorithm

SAMCRA [79] is the exact successor of TAMCRA, a Tunable Accuracy Multiple Constraints Routing Algorithm [25], [24]. TAMCRA and SAMCRA are based on three fundamental concepts: (1) a nonlinear measure for the path length, (2) a k-shortest path approach [20] and (3) the principle of non-dominated paths [38]:

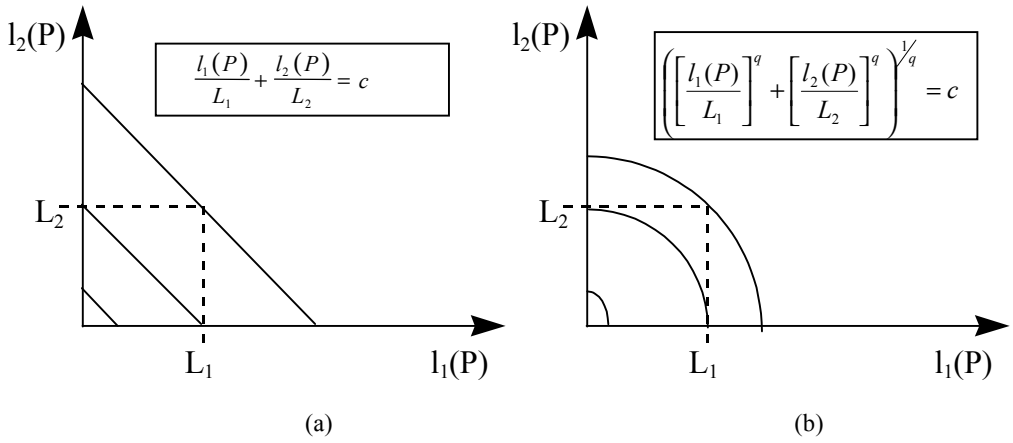

Figure 5: Scanning procedure with (a) straight equilength lines. (b) curved equilength lines.

1. Figure 5 illustrates that using curved equilength lines (a nonlinear length function) to scan the constraints area is more efficient than the straight equilength line approach as performed by Jaffe's algorithm. The formula in Figure 5b is derived from Holder's q-vector norm [32]. Ideally, the equilength lines should perfectly match the boundaries of the constraints, scanning the constraint area without ever selecting a solution outside the constraint area, which is only achieved when $q \rightarrow \infty$. Motivated by the geometry of the constraints surface in m-dimensional space, the

length of a path P is defined, equivalent to Holder's q-vector norm with $q \to \infty$, as follows [25]:

$$l(P) = \max_{1 \leq i \leq m} \left(\frac{w_i(P)}{L_i} \right) \tag{3}$$

where $w_i(P) = \sum_{(u,v) \in P} w_i(u,v)$.

A solution to the MCP problem is a path whose weights are all within the constraints, i.e. $l(P) \leq 1$. However, SAMCRA is a MCOP algorithm. Depending on the specifics of a constrained optimization problem, SAMCRA can be used with different length functions, provided they obey the criteria for length in vector algebra. Example length functions are given in [79] and we have implemented such a different length function in SAMCRAcost, which minimizes only one "monetary cost" measure of the paths within the constraints. By using length function (3) we treat all QoS measures as equally important. This could result in a more efficient use of the network resources. An important corollary of a nonlinear path length as (3) is that *the subsections of shortest paths in multiple dimensions are not necessarily shortest paths.* This suggests to consider in the computation more paths than only the shortest one, leading to the k-shortest path approach.

2. The k-shortest path algorithm as presented in [20] 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. In SAMCRA the k shortest path concept is applied to the intermediate nodes i on the path from source node s to destination node d, to keep track of multiple sub-paths from s to i. Not all sub-paths are stored, but a distinction is made based on non-dominance.

3. The principle of non-dominance is the third concept in SAMCRA. A path Q is dominated by a path P if $w_i(P) \leq w_i(Q)$ for $i = 1, .., m$, with an inequality for at least one i. SAMCRA only considers non-dominated (sub)-paths. This property allows to efficiently reduce the search-space without compromising the solution. "Dominance" can be regarded as a multidimensional relaxation. The latter is a key fundament of single parameter shortest path algorithms (such as Dijkstra and Bellman-Ford).

SAMCRA and TAMCRA have a worst-case complexity of

$$O(kN \log(kN) + k^2 mE)$$

For TAMCRA the k is fixed and hence the complexity is polynomial. However, for SAMCRA this k can grow exponentially in the worst case. Knowledge about k is crucial to the complexity of SAMCRA, because a large k could make the algorithm useless. As an upper-bound for k, we could use $k_{\max} = \lfloor e(N-2)! \rfloor$, which is an upper-bound on the total number of paths between a source and destination in $G(N, E)$ [80]. If the constraints/measures have a finite granularity, another upper-bound applies:

$$k_{\max} = \min \left(\frac{\prod_{i=1}^{m} L_i}{\max_j (L_j)}, \lfloor e(N-2)! \rfloor \right)$$

where the constraints L_i are expressed as an integer number of a basic unit.

SAMCRA guarantees to find a path within the constraints, provided such a path exists. In this process, SAMCRA only allocates queue-space when truly needed and self-adaptively adjusts the number of stored paths k in each node. This explains the S in SAMCRA. In TAMCRA the allocated queue-space is predefined via k. During the simulations with TAMCRA we chose $k = 2$, because this small value for k already renders good results. Of course a better performance is achieved when k is increased. Simulation results for different values for k can be found in [25].

3.1.4 Chen's Approximate Algorithm

Chen and Nahrstedt [16], [17] provided an approximate algorithm for the MCP problem. This algorithm first transforms the MCP problem into a simpler problem by scaling down $m-1$ (real) link weights to integer weights as follows:

$$w_i^*(u,v) = \left\lceil \frac{w_i(u,v) \cdot x_i}{L_i} \right\rceil \text{ for } i = 2, 3, \ldots, m,$$

where x_i are predefined positive integers. The simplified problem constitutes of finding a path P for which $w_1(P) \leq L_1$ and $w_i^*(P) \leq x_i$, $2 \leq i \leq m$. A solution to this simplified problem is also a solution to the original MCP problem, but not vice versa, because the conditions of the simplified problem are more strict. Since the simplified problem can be solved exactly, Chen and Nahrstedt have shown that *the MCP problem can be solved exact in polynomial time, when at least $m-1$ QoS measures have bounded integer weights.*

To solve the simplified MCP problem, Chen and Nahrstedt proposed two algorithms based on dynamic programming: the Extended Dijkstra's Shortest Path algorithm (EDSP) and the Extended Bellman-Ford algorithm (EBF). The algorithms return a path that minimizes the first (real) weight provided that the other $m-1$ (integer) weights are within the constraints. The EBF algorithm is expected to give the better performance in terms of execution time when the graph is sparse and the number of nodes relatively large. We have chosen to implement the EBF version for our simulations.

The complexities of EDSP and EBF are $O(x_2^2 \cdots x_m^2 N^2)$ and $O(x_2 \cdots x_m NE)$, respectively. To achieve a good performance, high x_i's are needed, which makes this approach rather computationally intensive for practical purposes. By adopting the concept of non-dominance, like in SAMCRA, this algorithm could reduce its search-space, resulting in a faster execution time. (We have simulated all algorithms in their original form, without any possible improvements)

3.1.5 Randomized Algorithm

Korkmaz and Krunz [48] have proposed a randomized heuristic for the MCP problem. The concept behind randomization is to make random decisions during the execution of an algorithm [58, 62] so that unforeseen traps can potentially be avoided when searching for a feasible path. The proposed

randomized algorithm is divided into two parts: the initialization phase and the randomized search. In the initialization phase, the algorithm computes the shortest paths from every node u to the destination node d w.r.t. each QoS measure and the linear combination of all m measures. Based on the information obtained in the initialization phase, the algorithm can decide whether there is a chance of finding a feasible path or not. If so, the algorithm starts from the source node s and explores the graph using a randomized breadth-first search (BFS). In contrast to conventional BFS, which systematically discovers every node that is reachable from a source node s, the randomized BFS discovers nodes from which there is a good chance to reach a destination node d. By using the information obtained in the initialization phase, the randomized BFS can check whether this chance exists before discovering a node. If there is no chance, the algorithm can foresee the trap and does not explore such nodes further. The randomized BFS continues searching by randomly selecting discovered nodes until the destination node is reached. If the randomized BFS fails in the first attempt, it is possible to execute only the randomized BFS again so that the probability of finding a feasible path can be increased.

Under the same network conditions, multiple executions of the randomized algorithm may return different paths between the same source and destination pair, providing some load-balancing. However, some applications might require the same path again. In such cases, path caching can be used [67].

The worst-case complexity of the randomized algorithm is $O(mN \log N + mE)$. For the simulations we only executed one iteration of the randomized BFS.

3.1.6 H_MCOP

Korkmaz and Krunz [49] also provided a heuristic called H_MCOP. This heuristic tries to find a path within the constraints by using the nonlinear path length function (3) of TAMCRA. In addition, H_MCOP tries to simultaneously minimize the weight of a single "cost" measure along the path. To achieve both objectives simultaneously, H_MCOP executes two modified versions of Dijkstra's algorithm in backward and forward directions. In the backward direction, H_MCOP uses the Reverse-Dijkstra algorithm for computing the shortest paths from every node to the destination node d w.r.t. $w(u,v) = \sum_{i=1}^{m} \frac{w_i(u,v)}{L_i}$. This is the same as using Jaffe's algorithm in reverse mode, where d_i in (2) is equal to $\frac{1}{L_i}$ for $i = 1, \ldots, m$. Later on, these (reverse) paths from every node u to the destination node d are used to estimate how suitable the remaining sub-paths are. In the forward direction, H_MCOP uses a modified version of Dijkstra's algorithm called Look_Ahead_Dijkstra. Look_Ahead_Dijkstra starts from the source node s and discovers each node u based on a path P, where P is a heuristically determined complete s-d path that is obtained by concatenating the already traveled sub-path from s to u and the estimated remaining sub-path from u to d. Since H_MCOP considers complete paths before reaching the destination, it can foresee several infeasible paths during the search. If paths seem feasible, then the algorithm can switch to explore these feasible paths based on the minimization of the single measure.

The complexity of the H_MCOP algorithm is $O(N \log N + mE)$. If one deals only with the MCP problem, then H_MCOP should be stopped whenever a feasible path is found during the search in the backward direction, reducing the computational complexity. In the simulations, we use H_MCOP with the objective of minimizing the weight of a single measure. The performance of H_MCOP in

finding feasible paths can be improved by using the k-shortest path algorithm and by eliminating dominated paths [50].

3.1.7 Limited Path Heuristic

Yuan and Liu [86, 87] presented two heuristics for the MCP problem. The first "limited granularity" heuristic has a complexity of $O(N^m E)$, whereas the second "limited path" heuristic (LPH) has a complexity of $O(k^2 NE)$, where k corresponds to the queue-size at each node. The authors claim that when $k = O(N^2 \log_2 N)$, the limited path heuristic has a very high probability of finding a feasible path, provided that such a path exists. However, applying this value results in an excessive execution time.

The performance of both algorithms is comparable when $m \leq 3$, but for $m > 3$ the limited path heuristic is better than the limited granularity heuristic. Hence, we will only evaluate the limited path heuristic. Another reason for omitting an evaluation of the limited granularity heuristic is that it closely resembles the algorithm from Chen and Nahrstedt (Section 3.1.4).

The limited path heuristic is an extended Bellman-Ford algorithm that uses two of the fundamental concepts of TAMCRA. Both use the concept of non-dominance and maintain at most k paths per node. However, TAMCRA uses a k-**shortest** path approach, while LPH stores the first (and not necessarily shortest) k paths. Furthermore LPH does not check whether a sub-path obeys the constraints, it only does this at the end for the destination node. An obvious difference is that LPH uses a Bellman-Ford approach, while TAMCRA uses a Dijkstra-like search. The simulations revealed that Bellman-Ford-like implementations require more *execution time* than Dijkstra-like implementations, especially when the graphs are dense.

Conform the queue-size allocated for TAMCRA, we also allocated $k = 2$ in the simulations for LPH.

3.1.8 A*Prune

Liu and Ramakrishnan [56] considered the problem of finding not only one but multiple (K) shortest paths satisfying the constraints. The length function used is the same as Jaffe's length function (2). Liu and Ramakrishnan proposed an exact algorithm called A*Prune. If there are no K feasible paths present, the algorithm will only return those that are within the constraints. For the simulations we took $K = 1$.

A*Prune first calculates for each QoS measure the shortest paths from the source s to all $i \in N \backslash \{s\}$ and from the destination d to all $i \in N \backslash \{d\}$. The weights of these paths will be used to evaluate whether a certain sub-path can indeed become a feasible path (similar look ahead features were also deployed by Korkmaz and Krunz [48], [49]). After this initialization phase the algorithm proceeds in a Dijkstra-like fashion. The node with the shortest predicted end-to-end length[3] is extracted from a heap and then all of its neighbors are examined. The neighbors that cause a loop or lead to a violation of the constraints are pruned. The A*Prune algorithm continues extracting/pruning nodes until K constrained shortest paths from s to d are found or until the heap is empty.

[3] The length function is a linear function of all measures (2). If there are multiple sub-paths with equal predicted end-to-end length, the one with the shortest length so-far is chosen.

If Q is the number of stored paths, then the worst-case complexity is $O(QN(m+h+\log Q))$, where h is the number of hops of the retrieved path. This complexity is exponential, because Q can grow exponentially with $G(N, E)$. Liu and Ramakrishnan [56] do mention that it is possible to implement a Bounded A*Prune algorithm, which runs polynomial in time at the risk of loosing exactness.

3.2 Performance Comparison of MCP Algorithms

In this section we will present and discuss the simulations results for the MCP problem. The simulations consist of creating a Waxman topology [83], [80], through which the evaluated algorithms compute a path based on a set of constraints. After storing the desired results, this procedure is repeated. All simulations consisted of generating 10^4 topologies. The weights of a link were assigned independent uniformly distributed random variables in the range [0, 1].

The choice of the constraints is important, because it determines how many (if any) feasible paths exist. We adopt two sets of constraints, referred to as $L1$ and $L2$:

- $L1 : L_i = w_i(P),\ i = 1, ..., m$, where P is the shortest path according to (3)
- $L2 : L_i = \max_{j=1,...,m} (w_i(SP_j)),\ i = 1, ..., m$, where SP_j is the shortest path based on the j-th measure.

The first set of constraints, denoted by $L1$, is very strict, such that there is only one feasible path present in the graph. The second set of constraints ($L2$) is based on the weights of the shortest paths for each QoS measure. We use Dijkstra to compute these shortest paths and for each of these m paths we store their path weight vectors. We then choose for each measure i the maximum i-th component of these m path weight vectors. (Iwata's algorithm can always find a feasible path with this set of constraints)

During all simulations we stored the *success rate (SR)* and the *normalized execution time (NET)*. They are defined as follows:

$$SR = \frac{\#\text{ returned feasible paths}}{\#\text{ examined graphs}}$$

$$NET = \frac{\text{CPU time algorithm}}{\text{CPU time Dijkstra}}$$

Our simulations revealed that the Bellman-Ford-like algorithms (Chen's algorithm and the Limited Path Heuristic) consume significantly more *execution time* than their Dijkstra-like counterparts. We therefore omitted them from the results presented in this paper.

Figure 6 gives the *success rate* for four different topology sizes ($N = 50$, 100, 200 and 400), with $m = 2$. The exact algorithms SAMCRA and A*Prune always give the highest *success rate* possible. The difference in the *success rate* of the heuristics is especially noticeable when the constraints are strict. In this case Jaffe's algorithm and Iwata's algorithm perform significantly worse than the others. The only heuristic that is not affected much by strict constraints is the randomized algorithm. However, its *execution time* is comparable to that of the exact algorithms.

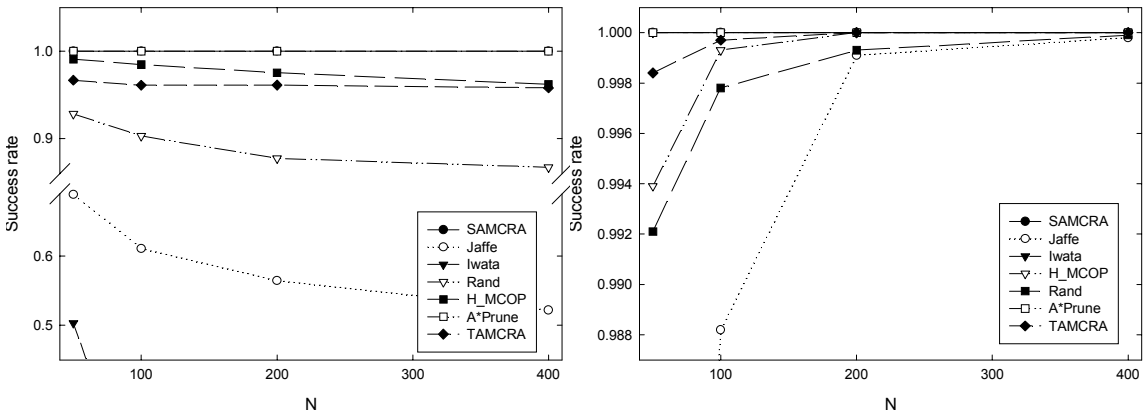

Figure 6: The success rate for $m = 2$. The results for the set of constraints $L1$ is depicted on the left and for $L2$ on the right.

Figure 7 displays the *normalized execution time*. It is interesting to observe that the *execution time* of the exact algorithm SAMCRA, does not deviate much from the polynomial time heuristics. This difference increases with the number of nodes, but an exponential growing difference is not noticeable! A first step towards understanding this phenomenon was provided by Kuipers and Van Mieghem in [52]. Furthermore, it is noticeable that when the constraints get looser, the *execution time* increases. The algorithms to which this applies, all try to minimize some length function (MCOP). When constraints get loose, this means that there will be more paths within the constraints, among which the shortest path has to be found. Searching through this larger set results in an increased *execution time*. If optimization is not strived for (MCP), then it is easier to find a feasible path if the constraints are loose, then when they are strict.

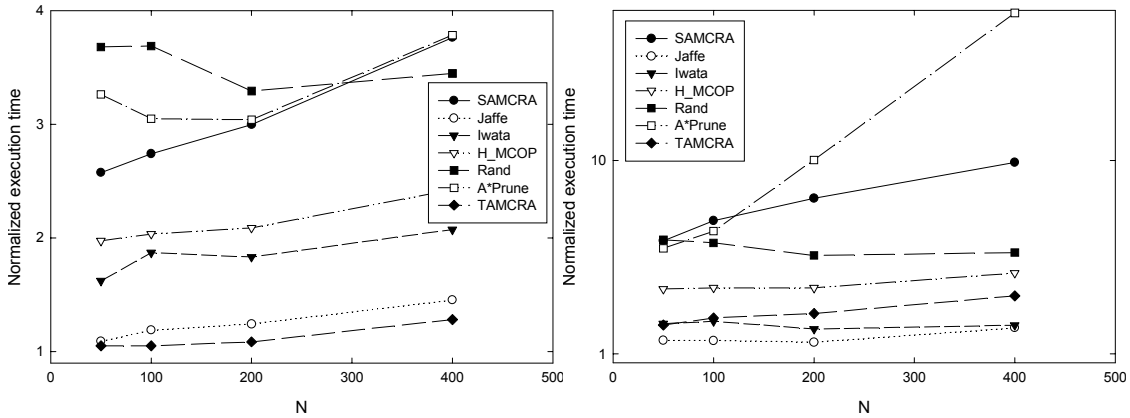

Figure 7: The normalized execution times for $m = 2$. The results for the set of constraints $L1$ are plotted on the left and for $L2$ on the right.

We have also simulated the performance of the algorithms as a function of m ($m = 2$, 4, 8 and 16). The results are plotted in Figures 8 and 9. We can see that the algorithms display a similar ranking in

success rate as in Figure 6. All link weights are independent uniformly distributed random variables. Under independent link weights, the larger m, the larger the set of non-dominated paths to evaluate. However, at a certain threshold point (m), the constraint values will become dominant, leading to an increasing number of paths that violate the constraints and hence less paths to evaluate. This property is explained in [79]. The impact of the constraint values can also be seen by comparing the execution times in Figures 8 and 9. If the constraints are loose, then a significant difference in *execution time* is noticeable between the exact algorithms SAMCRA and A*Prune. This can be attributed to the look ahead property of A*Prune, which can foresee whether sub-paths can lead to feasible end-to-end paths. Again, note that we do not see any NP-complete behavior in the *execution times*.

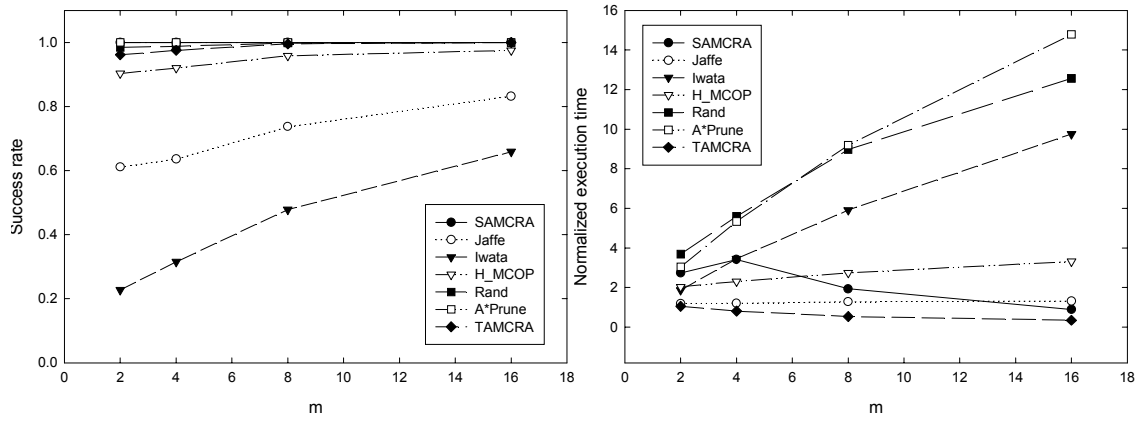

Figure 8: The success rate and normalized execution time in a 100-node network, as a function of m, with the set of constraints $L1$.

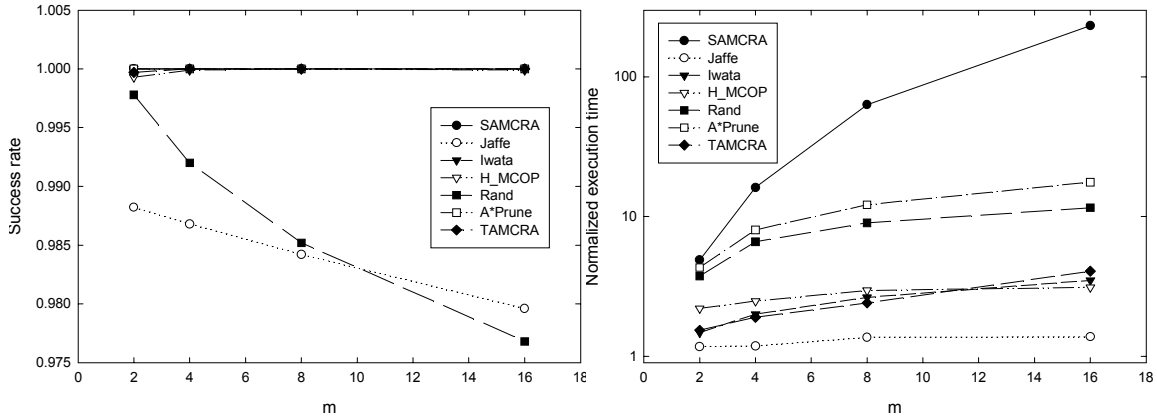

Figure 9: The success rate and normalized execution time in a 100-node network, as a function of m, with the set of constraints $L2$.

Based on these results we can rank the heuristics according their *success rate* and *execution time* as follows: TAMCRA, H_MCOP, Randomized algorithm, Jaffe's algorithm, Iwata's algorithm.

Finally we have evaluated the *optimal decision rate (ODR)*, which is defined as follows:

$$ODR = \frac{\#\text{ returned feasible paths with shortest length}}{\#\text{ feasible paths with shortest length}}$$

We have evaluated the *optimal decision rate* of the best heuristics TAMCRA (vs. SAMCRA) and H_MCOP (vs. SAMCRAcost). The exact algorithms SAMCRA and SAMCRAcost always have an *optimal decision rate* of 1.0. Figure 10 displays the *optimal decision rate* of TAMCRA with the length function (3) and the constraints set $L2$. Because the constraints set $L1$ allows only for one feasible path, the *success rate* (in Figure 6) for this set equals the *optimal decision rate*.

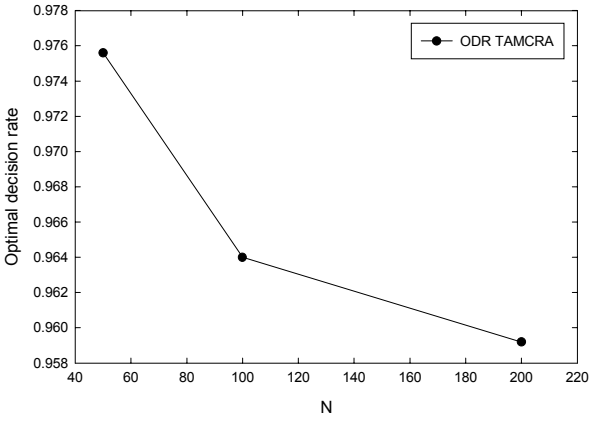

Figure 10: The optimal decision rate of TAMCRA with $k = 2$, for $m = 2$.

In roughly 95% of the cases TAMCRA finds the shortest path. The *optimal decision rate* decreases slightly with increasing N. The *optimal decision rate* for H_MCOP is plotted in Figure 11. The length function used here is fully determined by a single unconstrained "cost" measure, e.g., monetary cost.

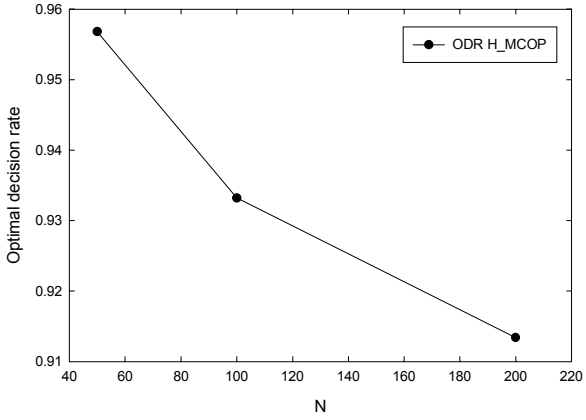

Figure 11: The optimal decision rate of H_MCOP for $m = 2$.

The ability of H_MCOP to find the shortest path decreases somewhat faster as compared to

TAMCRA in Figure 10. Two reasons can be given: (1) The *success rate* of H_MCOP is smaller than that of TAMCRA. (2) H_MCOP uses an extra "cost" measure.

3.3 MCP Conclusions

We will present conclusions for the considered class of graphs, namely the Waxman graphs (and according to [80] also random graphs) with independent uniformly distributed link weights.

For the MCP problem, we observe that TAMCRA-like algorithms have a higher *success rate* than linear approximations and Bellman-Ford based algorithms. This higher *success rate* is attributed to the following features of TAMCRA-like algorithms:

1. *Using a Dijkstra-like search along 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 give a high *success rate* in finding feasible paths, but under different circumstances the returned path may violate the constraints by 100%.
   A Bellman-Ford-like search runs better on sparse graphs, however our simulations indicated that even on sparse graphs (link-density ≈ 0.2), the Dijkstra-like heap-optimized search runs significantly faster.

2. *Reducing the search-space through the concept of non-dominance*
   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 is small.

3. *Tunable accuracy through a k-shortest path functionality*
   Routing with multiple constraints may require that multiple paths be stored at a node, leading to a k-shortest path approach.

4. *Predicting the feasibility of paths (look ahead property)*
   First calculating a path in polynomial time between the source and destination and using this information to find a feasible path between the same source and destination is especially useful when graphs become "hard to solve", i.e. N, E and m are large.

The exactness of the TAMCRA-like algorithms depends on the liberty of k. If k is not restricted, then both MCP and MCOP problems can be solved exact, 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. If a slightly larger *execution time* is permitted, then such exact algorithms are a good option. Furthermore, simulation results show that TAMCRA-like algorithms with small values of k render near-exact solutions with a Dijkstra-like (polynomial) complexity. For example, TAMCRA with $k = 2$ has almost the same *success rate* as the exact algorithms.

4 QoS Routing Algorithms for Special Cases

Several works in the literature have aimed at addressing special yet important sub-problems in QoS routing. For example, researchers addressed QoS routing in the context of bandwidth and delay. Routing with these two measures is not NP-complete. Wang and Crowcroft [82] presented a *bandwidth-delay based routing algorithm* which simply prunes all links that do not satisfy the bandwidth constraint and then finds the shortest path w.r.t. the delay in the pruned graph. Several path selection algorithms based on different combinations of bandwidth, delay, and hopcount were discussed in [59, 60, 64] (e.g., widest-shortest path and shortest-widest path). In addition, new algorithms were proposed to find more than one feasible path w.r.t. bandwidth and delay (e.g., Maximally Disjoint Shortest and Widest Paths) [77]. Kodialam and Lakshman [46] proposed bandwidth guaranteed dynamic routing algorithms. Orda and Sprintson [65] considered pre-computation of paths with minimum hopcount and bandwidth guarantees. They also provided some approximation algorithms that take into account certain constraints during the pre-computation. Guerin and Orda [33] focussed on the impact of reserving in advance on the path selection process. They describe possible extensions to path selection algorithms in order to make them advance-reservation aware, and evaluate the added complexity introduced by these extensions. Fortz and Thorup [28] investigated how to set link weights based on previous measurements so that the shortest paths can provide better load balancing and can meet the desired QoS constraints. When there exist certain specific dependencies between the QoS measures, due to specific scheduling schemes at network routers, the path selection problem is also simplified [59, 69]. Specifically, if Weighted Fair Queueing scheduling [31, 75] is being used and the constraints are on bandwidth, queueing delay, jitter, and loss, then the problem can be reduced to a standard shortest path problem by representing all the constraints in terms of bandwidth. However, care must be taken, because although queueing delay can be formulated as a function of bandwidth, this is not the case for the propagation delay, which cannot be ignored in high-speed networks.

5 Summary and Discussion

The state of the network in terms of Quality of Service (QoS) measures, e.g., available bandwidth and delay, is often volatile, making it hard to provide each node in the network with a consistent, accurate view of the network. For instance, consider ad hoc networks [66], where besides the dynamic behavior of the link weights also the nodes move around. It is therefore important to find a suitable *QoS protocol* that keeps the nodes' view of the state of the network up to date. There are some attempts towards such a protocol (e.g., in [5]), but still much work needs to be done. For instance how and when to distribute the link-state information without overloading the network, but at the same time to keep every node up to date, is still an unsolved problem. Furthermore, these QoS protocols operate within an AS (a single domain) and inter-AS protocols like BGP are currently not able to provide QoS information between ASs. Efficiently providing end-to-end QoS is therefore still an utopia and will require a coherent cooperation of a set of networking tools, like a good link-state update policy, traffic management (resource reservation, packet classification, queue management, scheduling, admission control, policy control, bandwidth broker, ...) and hierarchical structuring of the topology.

Once a suitable *QoS routing protocol* has been found and each node in the network has an up

to date view of the network a second challenging task in QoS routing appears, namely (based on this information) to find a path subject to multiple constraints. In other words, we need a suitable *QoS routing algorithm*. Several researchers investigated this constraint-based path selection problem and proposed various algorithms, mostly heuristics. This paper, divided into two parts, describes these algorithms as proposed for the *restricted shortest path* and *multi-constrained (optimal) path* problems, and evaluates their performance through simulations in the class of Waxman graphs with independent uniformly distributed random link weights. Table 1 displays the complexities of the algorithms discussed in this paper.

Algorithm	Worst-case complexity
Pseudo-polynomial-RSP	$O(\Delta E)$
ϵ-approximate from Hassin	$O((\frac{EN}{\epsilon}+1)\log(\log B)), O(\frac{EN^2}{\epsilon}\log(\frac{N}{\epsilon}))$
ϵ-approximate from Philips	$O(EN(1+1/\epsilon)+N^2(1+1/\epsilon)(\log N+\log(1+1/\epsilon)))$
ϵ-approximate from Goel et al.	$O(\frac{D}{\epsilon}(E+N\log N))$
DCUR	$O(N)$
BFH	$O(N\log N+E)$
Lagrangian algorithm	$O(E^2\log^2(E))$
SSR+DCCR	$O(xE\log N+kE\log(kN)+k^2E)$
Jaffe's algorithm	$O(N\log N+mE)$
Iwata's algorithm	$O(mN\log N+mE)$
SAMCRA, TAMCRA	$O(kN\log(kN)+k^2mE)$
EDSP, EBF	$O(x_2^2\cdots x_m^2N^2)$, $O(x_2\cdots x_mNE)$
Randomized algorithm	$O(mN\log N+mE)$
H_MCOP	$O(N\log N+mE)$
LPH	$O(k^2NE)$
A*Prune	$O(QN(m+N+\log h))$

Table 1: Worst-case complexities of QoS routing algorithms.

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. In general, the simulation results indicate that TAMCRA-like algorithms that use a k-shortest path algorithm along with a nonlinear length function while eliminating dominated paths and possibly applying other search-space reducing techniques, give the better performance for considered problems. The performance and complexity of these algorithms is easily adjusted by controlling the value of k. When k is not restricted, the TAMCRA-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 almost linearly with the number of nodes in $G(N, E)$, contrary to the expected exponential increase.

References

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Inc., 1993.

[2] L.H. Andrew and A.A.N. Kusuma, *Generalized Analysis of a QoS-aware routing algorithm*, IEEE GLOBECOM 1998, Piscataway, NJ, USA, vol. 1, pp. 1-6, 1998.

[3] Y.P. Aneja and K.P.K. Nair, *The constrained shortest path problem*, Naval Research Logistics Quarterly, 25:549–555, 1978.

[4] G. Apostolopoulos, R. Guerin, S. Kamat and S.K. Tripathi, *Quality of Service Based Routing: A performance perspective*, Proceedings of the ACM SIGCOMM '98 Conference, Vancouver, British Columbia, Canada, pp. 17-28, August/September, 1998.

[5] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda and T. Przygienda, *QoS Routing Mechanisms and OSPF Extensions*, RFC 2676, August 1999.

[6] The ATM Forum, *Private Network-to-Network Interface Specification Version 1.0 (PNNI 1.0)*, af-pnni-0055.000, March 1996.

[7] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell and J. McManus, *Requirements for Traffic Engineering over MPLS*, RFC 2702, September 1999.

[8] A. Banerjee, J. Drake, J.P. Lang, B. Turner, K. Kompella and Y. Rekhter, *Generalized Multiprotocol Label Switching: An Overview of Routing and Management Enhancements*, IEEE Communications Magazine, vol. 39, issue 1, pp. 144-150, January 2001.

[9] E. Basturk and P. Stirpe, *A Hybrid Spanning Tree Algorithm For Efficient Topology Distribution in PNNI*, Proceedings of the 1st IEEE International Conference on ATM (ICATM '98), pp. 385-394, 1998.

[10] B. Bellur and R.G. Ogier, *A reliable, efficient topology broadcast protocol for dynamic networks*, Proceedings of the INFOCOM'99 Conference, pp. 178-186, 1999.

[11] D. Bertsekas, *Data networks*, Prentice Hall, Inc., 1992.

[12] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, *An Architecture for Differentiated Services*, RFC 2475, December 1998.

[13] D. Blokh and G. Gutin, *An approximate algorithm for combinatorial optimization problems with two parameters*, Australasian Journal of Combinatorics, 14:157–164, 1996.

[14] R. Braden, D. Clark and S. Shenker, *Integrated Services in the Internet Architecture: an Overview*, RFC 1633, June 1994.

[15] B. Cain, *Fast link state flooding*, Global Telecommunications Conference, IEEE GLOBECOM'00, vol. 1, pp. 465-469, 2000.

[16] S. Chen and K. Nahrstedt, *On finding multi-constrained paths*, proceedings of ICC '98, IEEE, New York, pp. 874-879, 1998.

[17] S. Chen and K. Nahrstedt, *On Finding Multi-constrained Paths*, Technical Report UIUCDCS-R-97-2026, Dept. of Computer Science, University of Illinois, Urbana-Champaign, August, 1997.

[18] S. Chen and K. Nahrstedt, *An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions*, IEEE Network, November/December 1998.

[19] T.M. Chen and T.H. Oh, *Reliable Services in MPLS*, IEEE Communication Magazine, vol. 37, no. 12, pp. 58-62, Dec. 1999.

[20] E.I. Chong, S. Maddila, S. Morley, *On Finding Single-Source Single-Destination k Shortest Paths*, J. Computing and Information, 1995, special issue ICCI'95, pp. 40-47.

[21] R. Coltun, *The OSPF Opaque LSA Option*, RFC 2370, July 1998.

[22] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms*, The MIT Press, Cambridge, 2000.

[23] E. Crawley, R. Nair, B. Rajagopalan and H. Sandick, *A Framework for QoS-based Routing in the Internet*, RFC 2386, August 1998.

[24] H. De Neve and P. Van Mieghem, *A multiple quality of service routing algorithm for PNNI*, IEEE ATM workshop, Fairfax, May 26-29, 1998, pp. 324-328.

[25] H. De Neve and P. Van Mieghem, *TAMCRA: A Tunable Accuracy Multiple Constraints Routing Algorithm*, Computer Communications, 2000, vol. 23, pp. 667-679.

[26] D. Eppstein, *Finding the k Shortest Paths*, SIAM J. Computing, 28(2):652-673,1998.

[27] F. Ergun, R. Sinha and L. Zhang, *QoS routing with performance-dependent costs*, Proceedings of INFOCOM 2000, vol. 1, pp. 137-146, 2000.

[28] B. Fortz and M. Thorup, *Internet traffic engineering by optimizing OSPF weights*, Proceedings of INFOCOM 2000, vol. 2, pp. 519-528, 2000.

[29] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.

[30] A. Goel, K.G. Ramakrishnan, D. Kataria and D. Logothetis, *Efficient computation of delay-sensitive routes from one source to all destinations*, in Proceedings of the INFOCOM 2001 Conference, volume 2, pages 854–858. IEEE, April 2001.

[31] S.J. Golestani, *A self-clocked fair queueing scheme for broadband applications*, Proceedings of INFOCOM'94, vol. 2, pp. 636-646, 1994.

[32] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 1st ed., North Oxford Academic, Oxford, 1983.

[33] R. Guerin and A. Orda, *Networks with advance reservations: The routing perspective*, Proceedings of INFOCOM 2000, Israel, March 26-30, 2000.

[34] R. Guerin and V. Peris, *Quality-of-Service in Packet Networks: Basic Mechanisms and Directions*, Computer Networks, vol. 31, no. 3, pp. 169-179, Feb. 1999.

[35] L. Guo and I. Matta, *Search space reduction in QoS routing*, Proc. of the 19th III Int. Conference on Distributed Computing Systems, III, May 1999, pp. 142-149.

[36] G.Y. Handler and I. Zang, *A dual algorithm for the constrained shortest path problem*, Networks, 10:293–310, 1980.

[37] R. Hassin, *Approximation schemes for the restricted shortest path problem*, Mathematics of Operations Research, 17(1):36–42, 1992.

[38] M.I. Henig, *The shortest path problem with two objective functions*, European J. of Operational Research, 1985, vol. 25, pp. 281-291.

[39] C. Huitema, *Routing in the Internet*, Prentice Hall, Inc., 1995.

[40] P.A. Humblet and S.R. Soloway, *Topology Broadcast Algorithms*, Computer Networks and ISDN Systems, vol. 16, pp. 179-186, 1988/89

[41] K. Ishida, K. Amano and N. Kannari, *A delay-constrained least-cost path routing protocol and the synthesis method*, in Proceedings of the Fifth International Conference on Real-Time Computing Systems and Applications, pages 58 – 65. IEEE, Oct. 1998.

[42] A. Iwata, R. Izmailov, D.-S. Lee, B. Sengupta, G. Ramamurthy and H. Suzuki, *ATM Routing Algorithms with Multiple QoS Requirements for Multimedia Internetworking*, IEICE Transactions and Communications E79-B, no. 8, pp. 999-1006, 1996.

[43] J.M. Jaffe, *Algorithms for finding paths with multiple constraints*, Networks 14, pp. 95-116, 1984.

[44] A. Juttner, B. Szviatovszki, I. Mecs and Z. Rajko, *Lagrange relaxation based method for the QoS routing problem*, in Proceedings of the INFOCOM 2001 Conference, volume 2, pages 859–868. IEEE, April 2001.

[45] P.N. Klein and N.E. Young, *Approximation Algorithms for NP-Hard Optimization Problems*, in Algorithms and Theory of Computation Handbook, ed. M.J. Atallah, CRC Press, ch. 34, pp. 34.1-34.19, 1999.

[46] M. Kodialam and T.V. Lakshman, *Dynamic routing of bandwidth guaranteed tunnels with restoration*, Proceedings of INFOCOM 2000, pp. 902-911, 2000.

[47] T. Korkmaz, M. Krunz and S. Tragoudas, *An Efficient Algorithm for Finding a Path Subject to Two Additive Constraints*, Proceedings of the ACM SIGMETRICS '00 Conference, Santa Clara, CA, vol. 1, pp. 318-327, June, 2000.

[48] T. Korkmaz and M. Krunz, *A randomized algorithm for finding a path subject to multiple QoS requirements*, Computer Networks, vol. 36, pp. 251-268, 2001.

[49] T. Korkmaz and M. Krunz, *Multi-Constrained Optimal Path Selection*, IEEE INFOCOM 2001.

[50] T. Korkmaz, *QoS Routing in Packet Networks*, The University of Arizona, 2001.

[51] H.W. Kuhn and A.W. Tucker, *Nonlinear Programming*, Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probability, pp. 481-492, Berkeley, CA, 1961.

[52] F.A. Kuipers and P. Van Mieghem, *QoS routing: Average Complexity and Hopcount in m Dimensions*, Proc. of Second COST 263 International Workshop, QofIS 2001, Coimbra, Portugal, pp. 110-126, September 24-26, 2001.

[53] F.A. Kuipers and P. Van Mieghem, *MAMCRA: A Constrained-Based Multicast Routing Algorithm*, Computer Communications, vol. 25/8, pp. 801-810, May 2002.

[54] W.C. Lee, M.G. Hluchyi and P.A. Humblet, *Routing Subject to Quality of Service Constraints in Integrated Communication Networks*, IEEE Network, pp. 46-55, July/August, 1995.

[55] B. Lekovic and P. Van Mieghem, *Link State Update Policies for Quality of Service Routing*, IEEE Eighth Symposium on Communications and Vehicular Technology in the Benelux (SCVT2001), Delft, The Netherlands, pp. 123-128, October 18 2001.

[56] G. Liu and K.G. Ramakrishnan, *A*Prune: An Algorithm for Finding K Shortest Paths Subject to Multiple Constraints*, IEEE INFOCOM 2001.

[57] D.H. Lorenz, A. Orda, D. Raz and Y. Shavitt, *Efficient QoS Partition and Routing of Unicast and Multicast*, Proceedings of IWQoS 2000, pp. 75-83, June, 2000.

[58] L. Lovasz, *Randomized Algorithms in Combinatorial Optimization*, in Combinatorial Optimization (Series in Discrete Mathematics and Theoretical Computer Science), eds. W. Cook, L. Lovasz and P. Seymour, vol. 20, pp. 153-179, American Mathematical Society, 1995.

[59] Q. Ma and P. Steenkiste, *On Path Selection for Traffic with Bandwidth Guarantees*, Proceedings of the IEEE International Conference on Network Protocols (ICNP '97), Atlanta, Georgia, pp. 191-202, October, 1997.

[60] Q. Ma and P. Steenkiste, *Routing Traffic with Quality-of-ServiceGuarantees in Integrated Services Networks*, Proceedings of NOSSDAV'98, July 1998.

[61] D.E. McDysan, *QoS & traffic management in IP & ATM networks*, McGraw-Hill, 2000.

[62] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[63] J. Moy, *OSPF Version 2*, RFC 2328, April 1998.

[64] 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.

[65] A. Orda and A. Sprintson, *QoS routing: the precomputation perspective*, Proceedings of INFOCOM 2000, pp. 128-136, 2000.

[66] C.E. Perkins (ed.), *Ad Hoc Networking*, Addison-Wesley, 2001, ISBN: 0-201-30976-9.

[67] M. Peyravian and A.D. Kshemkalyani, *Network path caching: Issues, algorithms and a simulation study*, Performance Evaluation, vol. 20, no. 8, pp. 605-614, 1997.

[68] C.A. Phillips, *The network inhibition problem*, in Proceedings of the 25th Annual ACM Symposium on the Theory of Computing (STOC), pages 776–785, May 1993.

[69] C. Pornavalai, G. Chakraborty and N. Shiratori, *QoS Based Routing Algorithm in Integrated Services Packet Networks*, Proceedings of IEEE ICNP'97, pp. 167-174, 1997.

[70] D.S. Reeves and H. F. Salama, *A distributed algorithm for delay-constrained unicast routing*, IEEE/ACM Transactions on Networking, 8(2):239–250, April 2000.

[71] E. Rosen, A. Viswanathan and R. Callon, *Multiprotocol Label Switching Architecture*, RFC 3031, January 2001.

[72] H.F. Salama, D.S. Reeves and Y. Viniotis, *Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks*, IEEE JSAC, 15(3), pp. 332-345, April 1997.

[73] R. Sriram, G. Manimaran, and C. S. R. Murthy, *Preferred link based delay-constrained least-cost routing in wide area networks*, Computer Communications, 21:1655–1669, 1998.

[74] M. Steenstrup, *Routing in Communications Networks*, Prentice Hall, Inc., 1995.

[75] I. Stoica and H. Zhang, *Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks*, Proceedings of the ACM SIGCOMM '99 conference on Applications, technologies, architectures, and protocols for computer communication, pp. 81-94, 1999.

[76] Q. Sun and H. Langendorfer, *A new distributed routing algorithm for supporting delay-sensitive applications*, Computer Communications, 21:572–578, 1998.

[77] N. Taft-Plotkin, B. Bellur and R. Ogier, *Quality-of-Service routing using maximally disjoint paths*, The Seventh International Workshop on Quality of Service (IWQoS'99), London, England, pp. 119-128, May/June, 1999.

[78] M. van der Zee, *Quality of Service Routing - State of the Art Report*, Ericsson, 1/0362-FCP NB 102 88 Uen, http://searchpdf.adobe.com/proxies/0/9/25/62.html, 1999.

[79] P. Van Mieghem, H. De Neve and F.A. Kuipers, *Hop-by-Hop Quality of Service Routing*, Computer Networks, vol. 37/3-4, pp. 407-423, October 2001.

[80] P. Van Mieghem, *Paths in the simple Random Graph and the Waxman Graph*, Probability in the Engineering and Informational Sciences (PEIS), vol. 15, pp. 535-555, 2001.

[81] B. Wang and J.C. Hou, *Multicast routing and its QoS extension: problems, algorithms, and protocols*, IEEE Network, vol. 14, no. 1, pp. 22-36, Jan.-Feb 2000.

[82] Z. Wang and J. Crowcroft, *Quality-of-Service Routing for Supporting Multimedia Applications*, IEEE JSAC, vol. 14, no. 7, pp. 1228-1234, September, 1996.

[83] B.M. Waxman, *Routing of multipoint connections*, IEEE JSAC, 6(9):1617-1622, december 1998.

[84] 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.

[85] X. Xiao and L.M. Ni, *Internet QoS: A big picture*, IEEE Network, vol. 13, no. 2, pp. 8-18, March-April 1999.

[86] X. Yuan and X. Liu, *Heuristic Algorithms for Multi-Constrained Quality of Service Routing*, IEEE INFOCOM 2001.

[87] X. Yuan, *Heuristic Algorithms for Multiconstrained Quality-of-Service Routing*, IEEE/ACM Transactions on Networking, vol. 10, no. 2, April 2002.