Overview of Constraint-Based Path Selection Algorithms for QoS Routing

F.A. Kuipers, Delft University of Technology
T. Korkmaz, University of Texas at San Antonio
M. Krunz, University of Arizona
P. Van Mieghem, Delft University of Technology

September 2, 2002

Abstract

Constraint-based path selection is an invaluable component of a full-fledged quality of service (QoS) architecture. It aims at identifying a path that satisfies a set of constraints (e.g., bounded delay, packet loss rate, etc.). Unfortunately, multi-constrained path selection problems are NP-complete, and therefore often considered computationally intractable. This has led to the proposal of many heuristic algorithms for these problems. The aim of this paper is to give an overview of constraint-based routing algorithms, focusing on *restricted shortest path* and *multi-constrained path* algorithms.

1 Introduction

In recent years, there has been an increasing demand for Internet-based multimedia applications. In response to this demand, the research community has been extensively investigating several quality-of-service (QoS) based frameworks, including the Integrated Services (IntServ), the Differentiated Services (DiffServ), and Multi-Protocol Label Switching (MPLS). One of the key issues in all of these frameworks is how to identify efficient paths that can satisfy the given QoS constraints, or what is commonly known as the QoS-based routing problem.

In general, routing (be it QoS-based or not) involves two entities: *routing protocols* and *routing algorithms*. Routing protocols capture the network state information (e.g., available resources) and disseminate it throughout the network, while routing algorithms use this information to compute appropriate paths. The current best-effort routing simply performs these tasks based on a single, relatively static measure. In contrast, QoS-based routing needs to take into account both the applications requirements and the availability of network resources. As a result, QoS routing has to deal with some challenging issues that are not present in best-effort routing, including scalable dissemination of dynamic (state-dependent) information, state aggregation, computation of constrained paths, etc. In this paper, we focus on QoS-based *unicast* routing[1]. Our main goal is to shed some light on the myriad of existing multi-constrained path selection algorithms. In all of these algorithms, we assume

[1] QoS-based multicast routing often faces different conceptual problems.

that the network-state information is temporarily static and has been disseminated throughout the network using the underlying QoS-based routing protocol (e.g., QoS-enhanced OSPF).

Before formally defining the *multi-constrained path* problem, we first describe the notation 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 an 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. QoS measures can be roughly classified into *additive* (e.g., delay) and *non-additive* (e.g., available bandwidth, policy flags). In case of an additive measure, the QoS value of a path is equal to the sum of the corresponding weights of the links along that path. For a non-additive measure, the QoS value of a path is the minimum (or maximum) link weight along that path. Constraints are denoted by L_i, $i = 1, ..., m$. In general, non-additive measures can be easily dealt with by pruning from the graph all links (and possibly disconnected nodes) that do not satisfy the requested QoS constraint. Additive measures cause more difficulties. Hence, without loss of generality, we only consider additive measures. The basic problem considered in this paper can be stated as follows:

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 $w_i(u, v) \geq 0$, $i = 1, ..., m$. 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 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) version of the MCP problem is to retrieve the shortest "length" 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 (also known as the delay-constrained least-cost path, the minimum-cost restricted-time path, and the constrained shortest path). In the RSP problem, there is only one constraint Δ, which bounds the permissible delay of a path. The length measure is often referred to as the cost. For a path P, $l(P) \stackrel{\text{def}}{=} \sum_{(u,v)\in P} l(u, v)$, where $l(u, v)$ is a cost (= length) for the link (u, v). A solution to the RSP problem is one where the least-cost path among the set of delay-constrained paths is retrieved.

The MCP problem and its variants are known to be NP-complete [4]. Therefore, they are considered to be intractable for large networks. Accordingly, many heuristics have been proposed for these problems. In the rest of this paper, we briefly describe the lion's share of the published algorithms. To simplify the discussion, we divide the discussed algorithms into two categories: restricted shortest path algorithms in Section 2, and multi-constrained path algorithms in Section 3. Finally, we summarize and conclude this paper in Section 4.

2 RSP Algorithms

Before presenting some of the efficient solutions for the RSP problem, we start by discussing its exact (but computationally more strenuous) solutions.

2.1 Exact Algorithms

An exact solution to the RSP problem can be found by systematically examining every path between s and d in a brute-force manner (e.g., using depth-first search with backtracking). However, since the number of paths grows exponentially with the size of the network, this method requires an exponential number of operations in the worst case. Hence, it may not be useful in practice. An alternative exact solution is known as the Constrained Bellman-Ford (CBF) algorithm. This algorithm finds independent minimum-cost paths between a source and a set of destination nodes, each of which has its own delay constraint. The basic idea behind this algorithm is to systematically discover the lowest-cost paths while monotonically increasing their total delays. The algorithm maintains a list of paths from the source node to every other node with increasing cost and decreasing delay. It selects a node whose list contains a path that satisfies the delay constraint and that has the minimum cost. The algorithm 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 the delay constraint is satisfied and there exists a path to be explored. Although this algorithm exactly solves the RSP problem, its execution time grows exponentially in the worst case.

The RSP problem can also be solved exactly via pseudo-polynomial-time algorithms. In general, the complexity of pseudo-polynomial-time algorithms depends on the actual values of the input data (e.g., link delays and the given delay constraint) as well as the size of the network [4]. Pseudo-polynomial-time algorithms incur long execution times if the value of the input data is large. This can happen if the granularity is very small.

2.2 ϵ-Optimal Approximation Algorithms

One general approach to deal with NP-complete problems is to look for a polynomial-time algorithm that guarantees finding an approximate, quantifiable solution to the optimal one. An algorithm is said to be ϵ-optimal if it returns a path whose cost is at most $(1+\epsilon)$ times the cost of the optimal path, where $\epsilon > 0$. Two examples of ϵ-optimal approximation algorithms for the RSP problem were provided in [6]. Their complexities are $O((\frac{EN}{\epsilon} + 1) \log\log B)$ 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 equals to the sum of the $(N-1)$ largest link-costs, and then it systematically adjusts these bounds using a *testing* procedure. After computing LB and UB, the approximation algorithm bounds the cost of each link by rounding and scaling it as follows: $l'(u,v) = \left\lfloor \frac{l(u,v)(N-1)}{\epsilon LB} \right\rfloor \ \forall\ (u,v) \in E$. It then applies a pseudo-polynomial-time algorithm on these new weights. The second approximation algorithm is basically an extension of the first one, and is based on a technique called *interval partitioning*.

Approximation algorithms perform better in minimizing the cost of a returned feasible path as ϵ

goes to zero. However, a smaller value of ϵ leads to an increased complexity.

2.3 Backward-Forward Heuristic Algorithms

In *backward-forward* algorithms, the graph is explored 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 node u to the destination d. These algorithms can be used to find one or more paths, and can be implemented in centralized and distributed fashions. In one distributed implementation [12], probing and backtracking are used, as follows. The algorithm sends a probe packet over the preferred links one at a time. If the receiving node accepts the probe packet, it forwards it to the next node. Otherwise, if the packet is rejected the algorithm tries the next preferred link. In a centralized implementation, a backward-forward heuristic (BFH) can be implemented as follows: BFH first determines the least-delay path (LDP) and the least-cost path (LCP) from every node u to destination d. 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 [2]: link (u, v) is relaxed if $d[u] + d(u, v) + d[\text{LDP from } v \text{ to } d] \leq \Delta$ and $l[u] + l(u, v) < l[v]$ (we use the notation $d[u]$ and $l[u]$ to indicate, respectively, the total delay and total cost from source node s to node u). BFH extracts the next node that has the minimum cost from the heap. The computational complexity of the centralized BFH is three times that of Dijkstra's algorithm (i.e., $O(N \log N + E)$).

2.4 Lagrangian-Based Linear Composition Algorithms

In the Lagrangian-based composition approach, the algorithm linearly combines the delay and cost of each link and finds the shortest path with respect to (w.r.t.) the composite measure. Thus, the weight of a link becomes $w(u, v) = \alpha d(u, v) + \beta l(u, v)$, where α and β are called the multipliers. With this approach, there is actually no guarantee that the returned path satisfies the delay constraint. A key issue in the Lagrangian-based composition approach is how to determine appropriate values for the multipliers. This can be done systematically by iteratively finding the shortest path w.r.t. the linear combination and adjusting the multipliers' values in the direction of the optimal solution. The technique is similar to the well-known Lagrangian relaxation technique used in other constrained optimization problems. It can be shown that if the weights of the paths are uniformly distributed in the delay-cost space, then the search terminates after a finite number of iterations of Dijkstra's algorithm.

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 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 still excessive in some cases.

[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.

2.5 Hybrid Algorithms

It is also possible to devise efficient heuristics for the RSP problem using combinations of the aforementioned approaches. One such heuristic has been provided by Guo and Matta [5]. According to this heuristic, the cost of the least-delay path is selected as the cost constraint. The problem is then solved by minimizing a nonlinear length function, analogous to the one used in TAMCRA (see Section 3.3), that gives more priority to lower-cost paths. To minimize the nonlinear length function, a k-shortest-path-based algorithm called DCCR is used. 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 [5] tried to reduce the search space and tighten the cost bound by using a Lagrangian-based algorithm before applying DCCR. The complexity of this final hybrid algorithm, which is known as SSR+DCCR, depends on that of the Lagrangian-based algorithm and the k-shortest path algorithm. A worst-case complexity of $O(xE \log N + kE \log(kN) + k^2 E)$ was reported in [5], where x is an upperbound on the number of iterations involved in the search-space reduction.

3 MCP Algorithms

In this section, we present a representative sample of the algorithmic solutions for the MCP problem, and in some cases, for the more difficult MCOP problem.

3.1 Jaffe's Approximation Algorithm

In [7] Jaffe presented two algorithms for the MCP problem under two constraints ($m = 2$). The first is a 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 only discuss Jaffe's second algorithm, which we refer to simply as Jaffe's algorithm. For each link $(u, v) \in E$, the algorithm assigns a composite weight $w(u, v)$ that is obtained by linearly combining the original weights w_1 and w_2: $w(u, v) = d_1 \cdot w_1(u, v) + d_2 \cdot w_2(u, v)$, where d_1 and d_2 are positive multipliers. The algorithm then returns the path that minimizes the w weight. The minimization process is illustrated pictorially in Figure 1. In this figure, all possible paths between the source and destination nodes are indicated by black circles. Equal-length paths w.r.t. the composite weight w are indicated by a line. The search for the minimum-length path is equivalent to sliding this line outward from the origin until a path (black circle) is hit. This path is the one returned by the algorithm. Figure 1 also illustrates that the returned path does not necessarily reside within the feasibility region defined by the two constraints. Jaffe also noticed that and suggested using a nonlinear function whose minimization guarantees finding a feasible path, if such a path exists. However, because no simple shortest path algorithm is available to minimize such a nonlinear function, Jaffe did not pursue this option and approximated it by using the above mentioned linear length function.

Jaffe's work can be extended to an arbitrary number of constraints using the linear composition function $w(P) = \sum_{i=1}^{m} d_i w_i(P)$. In this case, it can be proven that for all positive values of the multipliers, if there exists a feasible path in the graph, then the returned path P can at most exceed each constraint by 100%.

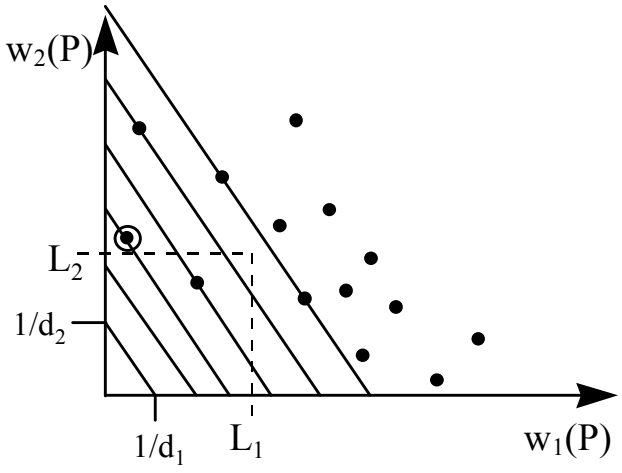

Figure 1: Pictorial representation of the search process in Jaffe's algorithm.

In the basic version of Jaffe's algorithm, if the returned path is not feasible, the algorithm terminates. However, the search could be continued by using different values for d_i, which might lead to finding a feasible path. Such an approach is similar to that used in the Lagrangian-based algorithms (see Section 2.4). Unfortunately, in some cases, even if all possible combinations of d_i are exhausted, a feasible path may not be found using a linear search. In such cases, approximate minimization of a *nonlinear* length function should be tried, as explained in Section 3.3.

3.2 Fallback Algorithm

In the fallback approach, the algorithm first computes one or more shortest paths based on one QoS measure, and then checks if all the constraints are met. If this is not the case, the procedure is repeated using another measure until a feasible path is found or until all QoS measures are examined. The complexity of the algorithm is polynomial. For example, if only one path is returned and Dijkstra's shortest path algorithm is used, the complexity is $O(mN \log N + mE)$. One problem with the fallback approach is that there is no guarantee that optimizing path selection w.r.t. any single measure would lead to a feasible path or even one that is close to being feasible. The fallback approach (just as Jaffe's algorithm) performs best when the link weights are positively correlated, because then if one weight is small, the other weights are also likely to be relatively small, resulting in a path farthest from the constraints.

3.3 TAMCRA and SAMCRA

TAMCRA [3] and its exact companion SAMCRA [13] are based on three fundamental concepts: (1) a nonlinear measure for the path length, (2) the k-shortest path approach, and (3) the principle of non-dominated paths. The first concept can be explained pictorially using Figure 2 (with $m = 2$). Part (a) of the figure depicts the search process using a *linear* composition function, similar to the one used in Jaffe's algorithm. If the two path weights are highly correlated, then the linear approach tends to perform well. However, if that is not the case, then a nonlinear function is more appropriate.

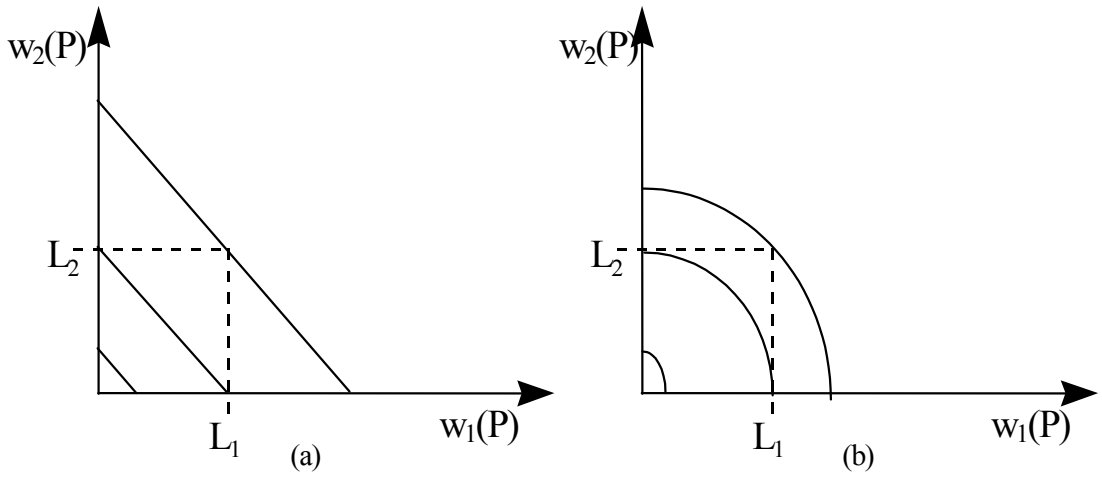

Figure 2: Searching for a feasible path by minimizing: (a) a linear composite function, (b) nonlinear composite function.

Part (b) of the figure depicts the search process using a nonlinear function. Ideally, the equal-length lines should perfectly match the boundaries of the constraints, scanning the constraint area without ever selecting a solution outside the constraint area. This can be achieved by taking:

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

where $w_i(P) \stackrel{\text{def}}{=} \sum_{(u,v) \in P} w_i(u,v)$. Any path P that satisfies $l(P) \le 1$ is a feasible path, and hence is an acceptable solution to the MCP problem. The obtained path, however, may not be optimal in terms of its length. SAMCRA addresses this issue by allowing different length functions[3]. Note that the function (1) treats all QoS measures equally.

An important characteristic of nonlinear path-length functions such as the one in (1) is that *sub-paths of shortest paths are not necessarily shortest paths.* In the path computation, this suggests considering more paths than only the shortest one, leading us to the k-shortest path approach. In SAMCRA the k-shortest path concept is applied to intermediate nodes i on the path from node s to node d. While traversing the graph, the algorithm keeps track of multiple sub-paths from s to i. Not all sub-paths are stored, but an efficient distinction is made based on the non-dominance of a path. We say that a path Q is dominated by a path P if $w_i(P) \le w_i(Q)$ for all $i = 1, .., m$, with an inequality for at least one i. SAMCRA only considers non-dominated (sub)-paths. This property allows it to efficiently reduce the search space without compromising the solution. Dominance can be regarded as a multidimensional relaxation; the latter being a key aspect of single-parameter shortest path algorithms (such as Dijkstra and Bellman-Ford).

SAMCRA and TAMCRA have a worst-case complexity of $O(kN \log(kN) + k^2 mE)$. Clearly, for a single constraint ($m = 1$ and $k = 1$), the complexity reduces to that of Dijkstra's algorithm. SAMCRA guarantees finding a feasible path, if one exists. Furthermore, it allocates buffer space only when truly

[3]SAMCRA provides a solution to the more difficult MCOP problem.

needed, and it self-adaptively adjusts the number of stored paths k at each node. Unfortunately, in the worst case, this could lead to an exponentially growing k. In TAMCRA the allocated buffer space k is predefined and fixed, and therefore its worst case complexity is polynomial.

3.4 Chen's Approximate Algorithm

Chen and Nahrstedt [1] provided an approximate algorithm for the MCP problem. This algorithm first transforms the MCP problem into a simpler one by scaling down $m-1$ (real-valued) link weights into integer weights, as follows: $w_i^*(u,v) = \left\lceil \frac{w_i(u,v)\cdot x_i}{L_i} \right\rceil$ for $i = 2, 3, \ldots, m$, where the x_i's are predefined positive integers. The simplified problem reduces to 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 (since the conditions of the simplified problem are stricter). Since the simplified problem can be solved exactly, Chen and Nahrstedt have shown that the MCP problem can be exactly solved 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. When the graph is sparse and the number of nodes is relatively large, EBF is expected to give better performance than EDSP in terms of execution time. 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 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, as in SAMCRA, this algorithm could reduce its search-space, resulting in a faster execution time.

3.5 Randomized Algorithm

Korkmaz and Krunz [9] proposed a randomized heuristic for the MCP problem. The concept behind randomization is to make random decisions during the execution of an algorithm so that unforeseen traps can potentially be avoided when searching for a feasible path. The proposed randomized algorithm is divided into two parts: an initialization phase and a 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 link weight and w.r.t. a linear combination of all weights. The algorithm then starts from the source node s and explores the graph using a randomized breadth-first search (BFS). In contrast to the 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 foresees the trap and does not consider such nodes any further. Otherwise, it continues searching by randomly selecting discovered nodes until the destination node is reached. If the first attempt of randomized BFS fails, the search can be repeated again. Because of the nature of randomization, subsequent attempts may succeed in returning a feasible path. The worst-case complexity of the randomized algorithm is $O(mN \log N + mE)$.

3.6 H_MCOP

Korkmaz and Krunz [8] also provided a heuristic algorithm for the MCOP problem called H_MCOP. This heuristic attempts to find a feasible path for any number of constraints while simultaneously minimizing a path length function. The search for a feasible path is done by approximating the nonlinear function (1), which is also used in TAMCRA. To achieve its objectives, H_MCOP executes two modified versions of Dijkstra's algorithm in backward and forward directions. In the backward direction, H_MCOP computes the shortest paths from every node to the destination node d w.r.t. $w(u,v) = \sum_{i=1}^{m} \frac{w_i(u,v)}{L_i}$. Later on, these (reverse) paths 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 travelled sub-path from s to u and the estimated remaining sub-path from u to d. Since the algorithm considers complete paths before reaching the destination, it can foresee some feasible 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 that defines the path length.

The complexity of H_MCOP algorithm is $O(N \log N + mE)$. If the algorithm is used for the MCP problem only, then the execution can be stopped once a feasible path is found during the backward search. This may reduce the execution time significantly. The performance of H_MCOP in finding feasible paths can be improved by using the k-shortest path algorithm and by eliminating dominated paths.

3.7 Limited Path Heuristic

Yuan [14] 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 author claims that when $k \sim O(N^2 \log_2 N)$, 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 $m \leq 3$. For $m > 3$, the limited path heuristic is better than the limited granularity heuristic.

The limited path heuristic is based on the Bellman-Ford algorithm and uses two of the fundamental concepts of TAMCRA. Both algorithms use the concept of non-dominance and both maintain at most k paths per node. However, while TAMCRA uses a k-shortest path approach, 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.

3.8 A*Prune

Liu and Ramakrishnan [10] considered the problem of finding not only one but multiple (K) shortest paths that are within the constraints. The linear length function used is the same as that of Jaffe's algorithm. The authors 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 each QoS measure, A*Prune calculates the shortest paths from s to all nodes $i \in N\backslash\{s\}$ and from 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 used in the H_MCOP algorithm). After this initialization phase, the algorithm proceeds in a Dijkstra-like fashion. The node with the shortest predicted end-to-end length[4] 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.

The worst-case complexity is $O(QN(m+h+\log Q))$, where h is the number of hops of the retrieved path and Q is the number of stored paths. This complexity grows exponentially with the size of the network. It is possible to implement a Bounded A*Prune algorithm, which runs polynomial in time at the risk of losing exactness.

4 Summary and Conclusions

QoS routing is a fundamental block of any QoS architecture. Its main objective is to identify a path that satisfies a set of QoS constraints. Several works in the literature 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. When there exist certain specific dependencies between the QoS measures (for example, as a result of using specific scheduling schemes at network routers), the path selection problem is also simplified [11]. Unfortunately, the general constraint-based path selection problems are NP-complete and may therefore become intractable in the worst case. We provided a high-level overview of the main solutions available in the literature for constraint-based path selection, focusing on the *restricted shortest path* problem and the *multi-constrained path* problem. Naturally, these solutions provide different trade-offs between computational complexity and accuracy. An important property of multidimensional routing is that a nonlinear length function is required to obtain exact results. QoS routing algorithms that use a linear definition for the path length will only prove useful when the link weights are positively correlated. In all other cases a nonlinear function is necessary, which significantly complicates the problem, since no simple shortest path algorithm is available to minimize such a nonlinear function. As a consequence, multiple paths must be evaluated, through the use of a k-shortest path algorithm. This increase of the search-space calls for efficient search-space reducing techniques. The most important encountered techniques are non-dominance, look-ahead, and the constraint values themselves. If exactness is not required, rounding and scaling the weights is also an option or simply bounding the number of paths to be evaluated. Depending on the processing resources available, these techniques allow for devising efficient tailor-made QoS algorithms.

[4] If there are multiple sub-paths with equal predicted end-to-end length, the one with the so-far shortest length is chosen.

Acknowledgments

The work of M. Krunz was supported by the National Science Foundation under grants ANI 9733143, CCR 9979310, and ANI 0095626.

References

[1] S. Chen and K. Nahrstedt. On finding multi-constrained paths. In *Proceedings of the ICC '98 Conference*, volume 2, pages 874–879. IEEE, 1998.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT press and McGraw-Hill book company, sixteenth edition, 1996.

[3] H. De Neve and P. Van Mieghem. TAMCRA: A tunable accuracy multiple constraints routing algorithm. *Computer Communications*, 23:667–679, 2000.

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

[5] L. Guo and I. Matta. Search space reduction in QoS routing. In *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems*, pages 142 – 149. IEEE, May 1999.

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

[7] J. M. Jaffe. Algorithms for finding paths with multiple constraints. *Networks*, 14:95–116, 1984.

[8] T. Korkmaz and M. Krunz. Multi-constrained optimal path selection. In *Proceedings of the INFOCOM 2001 Conference*, volume 2, pages 834–843, Anchorage, Alaska, April 2001. IEEE.

[9] T. Korkmaz and M. Krunz. A randomized algorithm for finding a path subject to multiple QoS constraints. *Computer Networks*, 36(2-3):251–268, 2001.

[10] G. Liu and K.G. Ramakrishnan. A*Prune: an algorithm for finding k shortest paths subject to multiple constraints. In *Proceedings of the INFOCOM 2001 Conference*, pages 743–749, Anchorage, Alaska, April 2001. IEEE.

[11] Q. Ma and P. Steenkiste. Quality-of-service routing for traffic with performance guarantees. In *Proceedings of the Fifth International IFIP Workshop on Quality of Service*, pages 115–126, May 1997.

[12] 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.

[13] P. Van Mieghem, H. De Neve and F.A. Kuipers. Hop-by-hop quality of service routing. *Computer Networks*, 37(3-4):407–423, October 2001.

[14] X. Yuan. Heuristic algorithms for multiconstrained quality-of-service routing. *IEEE/ACM Transactions on Networking*, 10(2):244–256, 2002.

Biographies

Fernando Kuipers (F.A.Kuipers@its.tudelft.nl): received the M.Sc. degree in Electrical Engineering at the Delft University of Technology in 2000. Currently he is working towards his Ph.D. degree in the Network Architectures and Services group at the same faculty. He is also a member of the DIOC (Delft interdisciplinary research center) for the Design and Management of Infrastructures, where he is taking part in the Telecommunications project. His Ph.D. work focuses on the algorithmic aspects and complexity of unicast Quality of Service routing, but also includes multicast QoS routing, multipath (link disjoint) QoS routing and efficient network-state update strategies for dynamic QoS-based networks.

Turgay Korkmaz (korkmaz@cs.utsa.edu): received the B.Sc. degree with the first ranking from Computer Science and Engineering at Hacettepe University, Ankara, Turkey, in 1994, and two M.Sc. degrees from Computer Engineering at Bilkent University, Ankara, and Computer and Information Science at Syracuse University, Syracuse, NY, in 1996 and 1997, respectively. He received his Ph.D. degree from Elec. and Computer Eng. at 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 of Computer Science department. His research interests include QoS-based routing, multiple constrained path selection, efficient dissemination of network-state information, topology aggregation in hierarchical networks, and performance evaluation of QoS-based routing protocols.

Marwan Krunz [M] (krunz@ece.arizona.edu): received his Ph.D. degree in Electrical Engineering from Michigan State University, Michigan, in 1995. From 1995 to 1997, he was a Postdoctoral Research Associate with the Department of Computer Science and the Institute for Advanced Computer Studies (UMIACS), University of Maryland, College Park. In January 1997, he joined the University of Arizona, where he is currently an Associate Professor of Electrical and Computer Engineering. Dr. Krunz's research interests lie in the field of computer networks, especially in its performance and traffic control aspects. His recent work has focused on the provisioning of quality of service (QoS) over wireless links, QoS routing (path selection, state aggregation), traffic modeling (video, WWW), bandwidth allocation, video-on-demand systems, and power-aware protocols for ad hoc networks. He has published more than 50 journal articles and refereed conference papers. Dr. Krunz is a recipient of the National Science Foundation CAREER Award (1998-2002). He currently serves on the editorial board for the IEEE/ACM Transactions on Networking, the Computer Communications Journal, and the IEEE Communications Interactive Magazine. He is a guest co-editor of a special issue on Hot Interconnects (IEEE Micro, Jan. 2002). Dr. Krunz was the Technical Program Co-chair for the 9th Hot Interconnects Symposium (Stanford University, August 2001). He has served and continues to serve on the executive and technical program committees of many international conferences. He served as a reviewer and a panelist for NSF proposals, and is a consultant for several corporations in the telecommunications industry.

Piet Van Mieghem (P.VanMieghem@its.tudelft.nl): is professor at the Delft University of Technology with a chair in telecommunication networks and chairman of the basic unit Network Architectures and Services (NAS). His main theme of the research are new Internet-like architectures for

future, broadband and QoS-aware networks. Professor Van Mieghem received a Master's and Ph.D. 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. During 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.