Bi-directional Search in QoS Routing

F.A. Kuipers and P. Van Mieghem

Delft University of Technology
Electrical Engineering, Mathematics and Computer Science
P.O Box 5031, 2600 GA Delft, The Netherlands
{F.A.Kuipers, P.VanMieghem}@ewi.tudelft.nl

Abstract. The "bi-directional search method" used for unicast routing is briefly reviewed. The extension of this method unicast QoS routing is discussed and an exact hybrid QoS algorithm HAMCRA that is partly based on bi-directional search is proposed. HAMCRA uses the speed of a heuristic when the constraints are loose and efficiently maintains exactness where heuristics fail. The performance of HAMCRA is simulated.

1 Introduction

One of the classical shortest path algorithms was proposed by Dijkstra [6]. Ever since, many variations of shortest path algorithms have been proposed in the literature [8], [2], mainly based on different proposals for a priority queue [3]. Nearly all proposed shortest path algorithms are designed to find a shortest paths tree rooted at the source to all other nodes. In practice, these algorithms are often only used to find a path between a single source-destination pair. This particular use may be inefficient. The idea to improve the Dijkstra algorithm for source-destination routing was provided in 1960 by Dantzig [4] and concretized in 1966 by Nicholson [14]. Their *bi-directional search* idea consisted of building two shortest path trees alternating between the source and the destination. Bi-directional search can lead to significant savings in time, but unfortunately seems to have been overshadowed by classical shortest path routing. We briefly revisit the concept of bi-directional search and evaluate its application to Quality of Service (QoS) routing.

One of the key issues in QoS routing is how to determine paths that satisfy multiple QoS constraints such as constraints on bandwidth, delay, jitter, and reliability. We focus on this so-called *multi-constrained path* problem and assume that the network-state information is temporarily static, has been distributed throughout the network and is accurately maintained at each node. Before giving the formal definition of the multi-constrained path problem, we first 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 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. The vector $\boldsymbol{L}$ represents the set of m QoS constraints. The QoS measure of a path can either be additive (e.g., delay, jitter), 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) that do not satisfy the requested min(max) QoS constraints. Constraints on additive QoS measures cause more difficulties. Multiplicative QoS measures can be transformed into additive measures by taking their logarithm. Hence, without loss of generality, we assume all QoS measures to be additive. The multi-constrained path problem 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 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 A to a destination node B 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. In some cases it might be desirable to retrieve the path with smallest length $l(P)$ from the set of feasible paths. This more difficult problem is called the *multi-constrained optimal path* (MCOP) problem. In general, MCP, irrespective of path optimization, is known to be an NP-complete problem [9]. This explains why the lion's share of proposed QoS routing algorithms are heuristics [11].

The rest of this paper is structured as follows. In Section 2 we discuss multi-constrained bi-directional search. In Section 3 we propose an exact QoS routing algorithm HAMCRA that is partly based on bi-directional search and discuss its complexity. In Section 4 we present the simulation results of HAMCRA and we end in Section 5 with the conclusions.

2 Multi-Constrained Bi-directional Search

The basic idea behind bi-directional search originated after observing that the Dijkstra algorithm examines a number of "unnecessary" nodes. Especially when the shortest (sub)path grows towards the destination, it can make an increasingly number of unnecessary scans. To reduce the number of unnecessary scans, it is better to start scanning from the source node as well as from the destination node[1]. In that case a large part of the topology will not be scanned, resulting in a higher efficiency.

When the shortest path has an odd number of hops, the simple idea of alternating between two directions and meeting in the middle is not enough to

[1] In case of a directed graph, the scan-procedure from destination B towards A should proceed in the reversed direction of the links.

find the shortest path. We also need to keep track of the minimum path length found sofar. Since we execute the Dijkstra algorithm from two sides, we need two queues Q_A and Q_B. The bi-directional Dijkstra algorithm extracts a node u by alternating between Q_A and Q_B. If a node u has been extracted from Q_A and from Q_B and if the end-to-end path length is smaller or equal to the shortest discovered (but not extracted) shortest path sofar, then we have found the shortest path and can return it by concatenating the two sub-paths from A to u and u to B.

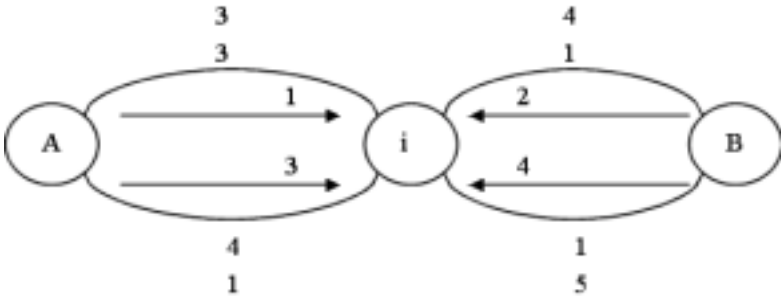

Fig. 1. Example of bi-directional search in two dimensions. The links represent paths, with their corresponding path weight vector.

Extending bi-directional search from $m = 1$ to $m > 1$ dimensions is not a trivial task. The complicating factor is the necessity of a non-linear length function for exact QoS routing [15]. A non-linear length causes that *subsections of shortest paths in $m > 1$ dimensions are not necessarily shortest paths themselves*. If two shortest paths (one originating in source node A and the other in destination node B) in $m > 1$ dimensions meet at an intermediate node, the resulting complete path is not necessarily the shortest path from A to B. We must keep track of the minimum length of the complete paths found sofar. Even if a new complete path exceeds the length of a previously found complete path, that new path cannot be discarded as illustrated in Fig. 1. We use the non-linear path length $l(P) = \max_{i=1,\ldots,m}(\frac{w_i(P)}{L_i})$. The arrows indicate the order of arrival of these sub-paths at node i. Once the first two sub-paths have arrived at node i, we have our first complete path with weight vector $\boldsymbol{w}(P) = (7,4)$. If the constraints are $\boldsymbol{L} = (10,10)$ then the length of this path equals 0.7. Once the third path arrives at node i, it makes a complete path with the second path, with total length 0.8. However, we cannot remove this sub-path, because combined with path 4, it forms the shortest path with link weight vector (5,6) and length 0.6. This example also indicates that we will have to connect at each intermediate node with $k \geq 1$ paths (even if they violate the constraints), where k can grow exponentially with the number of nodes N.

These problems in multiple dimensions complicate the determination of an efficient stop criterion for the MCOP problem. The bi-directional search in $m > 1$ dimensions has more potential for the MCP problem, where the routing algorithm can stop as soon as a complete path obeys the constraints. Unfortunately, when no feasible path is present in the graph, bi-directional search may require twice the time of uni-directional search. Again, an efficient stop criterion that foresees that there is no feasible path present seems difficult to find.

3 A Hybrid QoS algorithm

In the previous section we have argued that the use of bi-directional search has a higher potential for MCP than for MCOP. This need not be a disadvantage. When the constraints are loose and many feasible paths exist, then optimization is not very important and precious CPU time could be wasted in computing the optimal path. In a heavily loaded network, the need for optimization increases. Fortunately, under heavy load, the number of feasible paths is expected to be small, in which case the MCP problem approximates the MCOP problem.

Although applying bi-directional search to the MCP problem is possible, finding a clear stop criterion when no feasible paths are present is still problematic. This difficulty suggests to deviate from the alternating bi-directional search to a hybrid algorithm that uses concepts of bi-directional search. We have named our hybrid QoS algorithm HAMCRA, the Hybrid Auguring Multiple Constraints Routing Algorithm. HAMCRA is exact in solving the MCP problem, but is not always exact in solving the MCOP problem. The rest of this section will present HAMCRA, give its worst-case complexity and show that it is indeed exact in solving the MCP problem.

3.1 HAMCRA

HAMCRA is composed of the exact algorithm SAMCRA, the Self-Adaptive Multiple Constraints Routing Algorithm, [15] and its heuristic predecessor TAMCRA, the Tunable Accuracy Multiple Constraints Routing Algorithm, [5]. Both SAMCRA and TAMCRA are based on three fundamental concepts:

1. In order for any QoS algorithm to be exact, we must use a non-linear length function, such as
$$l(P) = \max_{i=1,\ldots,m} \left(\frac{w_i(P)}{L_i} \right) \tag{1}$$
where $w_i(P)$ is the i-th weight of path P and L_i is the i-th constraint. If $l(P) > 1$ then path P is not feasible.
2. If a non-linear length function like (1) is used, then the subsections of shortest paths in multiple dimensions are not necessarily shortest paths themselves. It may therefore be necessary to store in the computation more (sub-)paths then just the shortest. SAMCRA and TAMCRA use the k-shortest path

approach [7], where in TAMCRA k is predefined by the user and in SAMCRA k is adapted in the course of the computation to assure that the exact shortest path is found.

3. Both TAMCRA and SAMCRA only consider non-dominated paths, where a path P is called non-dominated if there[2] does not exist a path P^* for which $w_i(P^*) \leq w_i(P)$ for all link weight components i except for at least one j for which $w_j(P^*) < w_j(P)$.

Since HAMCRA is composed of SAMCRA and TAMCRA, it is also based on their three concepts. In HAMCRA, first the TAMCRA algorithm is executed with a queue-size $k = 1$ from the destination node to all other nodes in the graph. This is the similarity with bi-directional search, because we also scan from the destination node. However we do not alternate between the source and the destination. We can use TAMCRA with $k > 1$, which will lead to a better accuracy at the cost of increased complexity (of TAMCRA). The running time of TAMCRA with $k = 1$ is comparable to that of the Dijkstra algorithm. At each node, the path weight vector found by TAMCRA from that node to the destination is stored. These values will later be used to predict the end-to-end path length. If TAMCRA has found a path within the constraints between the source and the destination, HAMCRA can stop and return this path. If TAMCRA does not find a feasible path, HAMCRA continues by executing the SAMCRA algorithm from the source node. The difference between HAMCRA and SAMCRA is that HAMCRA uses the information obtained by TAMCRA and only stores predicted end-to-end lengths in the queue instead of the real lengths of the sub-paths. The idea for using predictions was originally presented in [10]. The predicted end-to-end length is found by summing the real weights of a path from source A to the intermediate node u with the weights of the TAMCRA path from u to the destination B. The algorithm continues searching in this way until a feasible path from A to B is found or until the queue is empty.

HAMCRA uses a fourth concept to reduce the search-space, namely that of lower bounds (LB) [13]. The LB concept uses the property that if $\sum_{i=1}^{m} \alpha_i w_i(P) > \sum_{i=1}^{m} \alpha_i L_i$, $\alpha_i \geq 0$, then path P is not feasible. By computing the shortest, according to the linear length function $l(P) = \sum_{i=1}^{m} \alpha_i w_i(P)$, paths $P^*_{B \to n}$ rooted at the destination B to each node n in the graph $G(N, E)$, we obtain the lower bounds $w_i\left(P^*_{B \to n}\right)$. These lower bounds can be used to check if a path $P_{A \to n}$ from source A to node n can possibly obey the constraints: if $\sum_{i=1}^{m} \alpha_i \left(w_i(P_{A \to n}) + w_i\left(P^*_{B \to n}\right)\right) > \sum_{i=1}^{m} \alpha_i L_i$, then path $P_{A \to n}$ need not be considered further. For the simulations we have chosen to compute via Dijkstra the shortest path tree rooted at the destination to each node n in the graph $G(N, E)$ for each of the m link weights separately. For measure j, the Dijkstra shortest path agrees with $\sum_{i=1}^{m} \alpha_i w_i(P)$, where $\alpha_i = 0$ for $i = 1, ..., m$ except

[2] If there are two or more different paths between the same pair of nodes that have an identical weight vector, only one of these paths suffices. We therefore assume one path out of the set of equal weight vector paths as being non-dominated and regard the others as dominated paths.

$\alpha_j = 1$. Hence, for each of the m link weight components, the lowest value from the destination to a node $n \in N$ is stored in the queue of that node n.

Note that the LB concept is also based on (lower bound) predictions for the end-to-end path length. We could therefore also use the LB predictions instead of the TAMCRA predictions. Because HAMCRA uses TAMCRA to predict the end-to-end path length, this prediction (if erroneous) could be larger than the real end-to-end path length. It may then happen that HAMCRA extracts a non-shortest path first. Therefore HAMCRA using TAMCRA cannot guarantee a solution to the MCOP problem. If $l_{predicted}(P) \leq l_{actual}(P)$ as is the case with LB predictions, a solution to MCOP can be guaranteed. Unfortunately, simulations (see Fig. 3) have shown that such predictions are usually not as good as the TAMCRA predictions, leading to an increased running time.

3.2 Worst-case complexity of HAMCRA

The total worst-case complexity of HAMCRA is constructed as follows. Executing heap-optimized Dijkstra m times leads to $mO(N \log N + E)$ and m times computing a length of a path leads to $mO(mN)$. Executing TAMCRA with $k = 1$ requires $O(N \log N + mE)$. The search from the destination adds $O(mN \log N + mE + m^2 N)$, which is polynomial in its input. The "SAMCRA" search from the source adds $O(kN \log(kN) + k^2 mE)$ [15]. Combining these contributions yields a total worst-case complexity of HAMCRA with $k = k_{\max}$ of $O(mN \log N + mE + m^2 N + kN \log(kN) + k^2 mE)$ or

$$C_{HAMCRA} = O(kN \log(kN) + k^2 mE) \tag{2}$$

where m is fixed and $m \leq k = k_{\max}$ and $k_{\max}$ is an upper bound on the number of paths in the search-space. For a single constraint ($m = 1$ and $k = 1$), this complexity reduces to the complexity of the Dijkstra algorithm $C_{Dijkstra} = O(N \log N + E)$. For $m > 1$, the complexity becomes NP-complete, since k can grow exponentially with N.

3.3 Proof that HAMCRA is exact

The proof that HAMCRA is exact in solving the MCP problem depends on the validity of the search-space reducing techniques. Obviously, if TAMCRA finds a feasible path at the beginning, then the MCP problem is exactly solved. For the case that this first step fails, we summarize the different steps in HAMCRA:

1. Paths with length $l(P) > 1$ need not be examined.
2. If in the k-shortest path algorithm the value of k is not restricted, HAMCRA returns all possible paths ordered in length between source and destination.
3. If, for all i, $w_i(P_1) \leq w_i(P_2)$, then $w_i(P_1) + u_i \leq w_i(P_2) + u_i$ for any u_i. For all definitions of length $l(.)$ satisfying the vector norm criteria (such as (1)) there holds that $l(\boldsymbol{w}(P_1) + \boldsymbol{u}) \leq l(\boldsymbol{w}(P_2) + \boldsymbol{u})$ for any vector $\boldsymbol{u}$. Hence, we certainly know that P_2 will never be shorter than P_1. Hence, dominated paths need not be stored in the queue.

4. If $\sum_{i=1}^{m} \alpha_i \left(w_i(P_{A\to n}) + w_i\left(P^*_{B\to n}\right)\right) > \sum_{i=1}^{m} \alpha_i L_i$, then sub-path $P_{A\to n}$ can never be complemented with a path $P^*_{B\to n}$ to satisfy the constraints $\boldsymbol{L}$. Hence, the sub-path $P_{A\to n}$ should not be considered further.
5. Finally, the insertion/extraction policy of the nodal queues uses a predicted end-to-end length instead of the real length of a (sub)-path. However, since the predicted end-to-end length and the real end-to-end length of a complete path between source and destination are the same, this path is returned if $l(P) \leq 1$ or removed if $l(P) > 1$. Thus, only feasible end-to-end paths will be examined and exactness is guaranteed.

4 Performance Evaluation of HAMCRA

In this section a performance evaluation of HAMCRA is presented based on simulation results. We have simulated on three classes of graphs, namely the class of random graphs $G_p(N)$ [1] with link-density $p = 0.2$, the class of Internet-like power law graphs with power $\tau = 2.4$ in the nodal degree distribution $\Pr[d = k] = k^{-\tau}$ and the extremely regular class of square two-dimensional lattices. We have performed two types of simulations.

The first type of simulations consisted of generating, for each simulation run, 10^4 graphs with all link weights independent uniformly distributed in the range (0,1]. In each graph, based on multiple constraints, we computed a path between two nodes (A and B) in the graph and stored the maximum queue-size k that was used. In all classes of graphs, the source A and destination B were randomly chosen. For the class of two-dimensional lattices, we also simulated with A chosen in the upper left corner and B in the lower right corner. We refer to this "worst-case" setting as Lattice 2 and to the case where the source and destination nodes are chosen randomly as Lattice 1. The constraints are chosen very strict, such that only one path can obey these constraints. In this case the MCP problem equals the more difficult MCOP problem.

The second type of simulations consisted of generating only one two-dimensional lattice with A chosen in the upper left corner and B in the lower right corner and then finding a path in this graph, subject to different constraints. To examine the influence of the constraints, we simulated with 10^4 different constraint vectors per topology.

4.1 Simulation type 1

Fig. 2 presents the results as a function of the number of nodes N. The expected queue-size $E[k]$ and the variance in queue-size $var[k]$ are very close to one for the class of random graphs. Similar results were also obtained with SAMCRA [15], suggesting that exact QoS routing in the class of random graphs with uniformly distributed link weights is easy. The class of power law graphs also has a moderate $E[k]$, although the variability $var[k]$ is larger than in the class of random graphs. This was expected because the power law graphs are less random than the random graphs. The class of two-dimensional lattices gives the

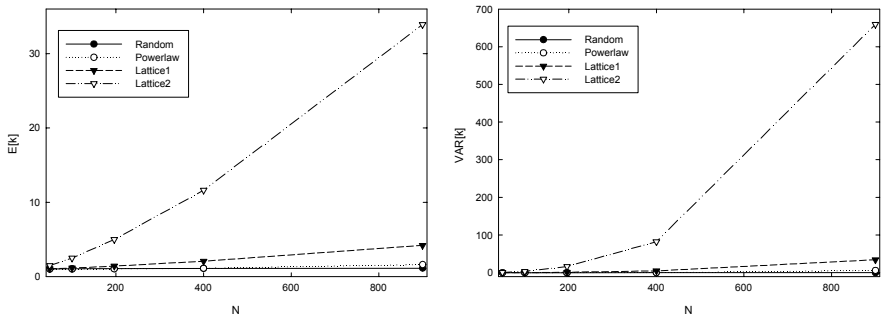

Fig. 2. The queue-size k for different topologies as a function of the number of nodes N.

worst performance, especially when the source and destination nodes are furthest apart. The two-dimensional lattices have a large expected hopcount and many paths between source and destination, making this class of graphs hard to solve. This was also observed and motivated in [12]. Finally, we have simulated the performance of HAMCRA as a function of the number of constraints m. The (undisplayed) results show that the queue-size k increases with m, but that this increase diminishes if m gets large enough. In fact, if $m \to \infty$, $E[k]$ will be exactly one as proved in [15].

4.2 Simulation type 2

In this subsection we present the results for the class of two-dimensional lattices with A chosen in the upper left corner and B in the lower right corner. We believe that this class of two-dimensional lattices represents worst-case topologies. We have simulated on a single lattice topology with 100 values for constraint L_1 and 100 values for constraint L_2, leading to a total of 10^4 computations per simulation. With our choice of the constraints it can occur that no feasible path exists. We have performed simulations for different levels of link correlation ρ. We only observed a high complexity for an extremely negative correlation. The results indicate that the values of the constraints and the correlation between the link weights can have a serious impact on the complexity. If the link weights are negatively correlated and the constraints are close to the weights of the m-dimensional shortest paths, the complexity is highest. The impact of correlation and constraints on the complexity of QoS routing has been discussed in [12]. Fig. 3 presents our results for HAMCRA with different predictive length functions. Our results show that HAMCRA with TAMCRA predictions has a good complexity if a feasible path is present and if the constraints are not too strict. The complexity is better than with Dijkstra lower bound predictions. Unfortunately, the complexity may be large if no feasible path is present or if only one path can

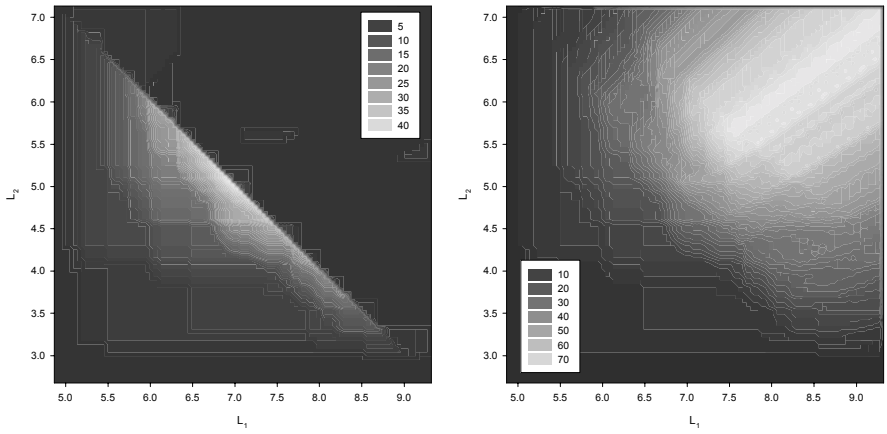

Fig. 3. The expected queue-size as a function of the constraints L_1 and L_2, whit $\rho = -1$, $N = 49$. (left) queue-size HAMCRA with TAMCRA predictions (right) queue-size HAMCRA with Dijkstra predictions.

obey the constraints. In the worst-case scenario with $\rho = -1$, it is possible to reduce the complexity by including lower bounds based on $\frac{1}{m}\sum_{i=1}^{m}\frac{w_i(P)}{L_i} \leq 1$. In this case we could have verified in polynomial time if a feasible path existed. If exactness for MCOP is required, we recommend to use the linear length function (see Section 3.1) $l(P) = \frac{1}{m}\sum_{i=1}^{m}\frac{w_i(P)}{L_i}$ for search-space reduction as well as end-to-end path length prediction (instead of TAMCRA).

5 Conclusions

In this paper we have revisited the use of bi-directional search in unicast routing. This method is powerful in (one-dimensional) unicast routing. To our knowledge an extension of one-dimensional bi-directional search towards multiple dimensions has never been examined. We have filled that gap in this paper and have shown that such an extension is not trivial. Some difficulties appear especially when the multi-dimensional shortest path is needed or when no feasible path is present. To avoid these difficulties, we have proposed and evaluated HAMCRA. This hybrid algorithm exactly solves the multi-constrained path (MCP) problem and is composed of the exact QoS algorithm SAMCRA and the heuristic QoS algorithm TAMCRA. HAMCRA uses TAMCRA to quickly follow a feasible path and uses SAMCRA to maintain its exactness for the MCP problem. Simulations with HAMCRA show that HAMCRA quickly finds a feasible path for nearly the entire range of feasible constraints. The complexity of HAMCRA can only be high when the constraints are closely situated around the weights of the multi-dimensional shortest paths, the link weights are negatively correlated and

we have a specific topology (like the two-dimensional lattice). We believe that in practice this situation is unlikely to occur and that HAMCRA is expected to have a low complexity. If our assumption holds, then it is pointless to consider heuristics for QoS routing that cannot even guarantee QoS requirements to be met.

References

1. B. Bollobas, *Random Graphs*, Cambridge University Press, second edition, 2001.
2. B.V. Cherkassky, A.V. Goldberg and T. Radzik, "Shortest paths algorithms: theory and experimental evaluation", Mathematical Programming, Series A, no. 73, pp. 129-174, 1996.
3. B.V. Cherkassky, A.V. Goldberg and C. Silverstein, "Buckets, heaps, lists and monotone priority queues", SIAM Journal on Computing, no. 28, pp. 1326-1346, 1999.
4. G. Dantzig, "On the shortest route through a network", Mgmt. Sci., vol. 6, pp. 187-190, 1960.
5. H. De Neve, and P. Van Mieghem, "TAMCRA: A Tunable Accuracy Multiple Constraints Routing Algorithm", Computer Communications, Vol. 23, pp. 667-679, 2000.
6. E.W. Dijkstra, "A note on two problems in connexion with graphs", Numerische Mathematik, no. 1, pp. 269-271, 1959.
7. D. Eppstein, "Finding the k Shortest Paths", SIAM J. Computing, 28(2):652-673, 1998.
8. G. Gallo and S. Pallottino, "Shortest Paths Algorithms", Annals of Operations Research, no. 13, pp. 3-79, 1988.
9. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.
10. P. Hart, N. Nilsson and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths", IEEE Transactions on Systems Science and Cybernetics, 2:100-107, 1968.
11. F.A. Kuipers, T. Korkmaz, M. Krunz and P. Van Mieghem, "An Overview of Constraint-Based Path Selection Algorithms for QoS Routing", IEEE Communications Magazine, vol. 40, no. 12, December 2002.
12. F.A. Kuipers and P. Van Mieghem, "The Impact of Correlated Link Weights on QoS Routing", Proc. of IEEE INFOCOM 2003, April 2003.
13. G. Liu and K.G. Ramakrishnan, "A*Prune: An Algorithm for Finding K Shortest Paths Subject to Multiple Constraints", Proc. of IEEE INFOCOM 2001, April 2001.
14. T. Nicholson, "Finding the shortest route between two points in a network", the computer journal, vol. 9, pp. 275-280, 1966.
15. P. Van Mieghem, H. De Neve, and F.A. Kuipers, "Hop-by-hop Quality of Service Routing", Computer Networks, vol. 37. No 3-4, pp. 407-423, 2001.