On the Complexity of QoS Routing

P. Van Mieghem and F.A. Kuipers*

Delft University of Technology
Information Technology and Systems
P.O Box 5031, 2600 GA Delft, The Netherlands

Abstract

We present SAMCRA, an exact QoS routing algorithm that guarantees to find a feasible path if such a path exists. Because SAMCRA is an exact algorithm, its complexity also characterizes that of QoS routing in general. The complexity of SAMCRA is simulated in specific classes of graphs. Since the complexity of an algorithm involves a scaling of relevant parameters, the second part of this paper analyzes how routing with multiple independent link weights affects the hopcount distribution. Both the complexity and the hopcount analysis indicate that for a special class of networks, QoS routing exhibits features similar to single-parameter routing. These results suggest that there may exist classes of graphs in which QoS routing is not NP-complete.

1 QoS Routing in Perspective.

After almost five years since the appearance of our "Aspects of QoS routing" [21], a brief update seems desirable because more understanding has been gained since then although basically little on the concepts has been changed. Our starting point here is the complementarity between routing algorithm and routing protocol. We quote from [21]

> Network routing essentially consists of two identities, the routing algorithm and the routing protocol. The routing algorithm assumes a temporarily static or frozen view of the network topology. [...] The routing protocol, on the other hand, provides each node in the network topology with a consistent view of that topology at some moment...

If this duality is acceptable, it says that capturing the dynamics of the topology and the link weights is taken care of by the routing protocol. Once the graph where each link is specified by a QoS link weight vector with as components for example delay, cell/packet loss, available bandwidth, monetary cost, etc... is offered by the routing protocol to a node, a local QoS routing algorithm is applied to determine in that graph a path from A to B subject to a QoS constraint vector. If an effective QoS routing protocol exists together with a QoS routing algorithm, we argue intuitively that the load in the network will be distributed automatically by QoS routing (both protocol and algorithm) and will tend closely to an optimally load-balanced network. Thus, under these assumptions, many

*email: {P.VanMieghem,F.A.Kuipers}@its.tudelft.nl of the current goals in traffic engineering (ietf-tewg) would be achieved as a natural spin-off of QoS routing. We believe that this potential additionally underlines the importance of QoS routing.

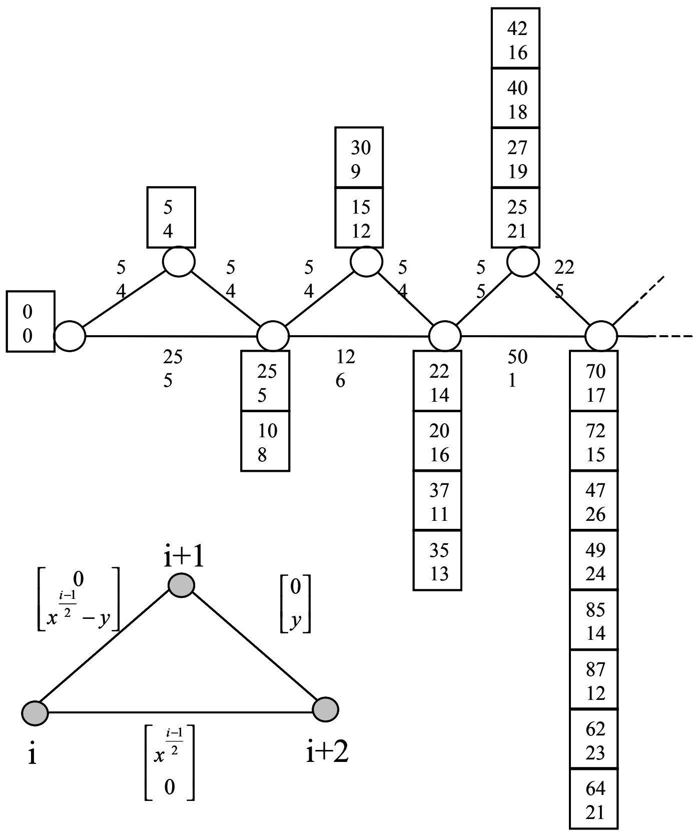
Relatively many papers have been devoted to QoS routing algorithms, but very few to the QoS routing protocol ([3], [2]), which seems to indicate that the latter poses considerably more difficulty than the former as solutions are (very) rare. Undoubtedly, the difficulty lies in the QoS routing protocol and less in the QoS routing algorithm. Even stronger as commented below, we claim that the 'QoS routing algorithm'-part of the dual problem is nearly entirely solved. Although a promising QoS routing protocol in the connection oriented world (PNNI in ATM) has been standardized, the complex parts (apart from the routing algorithm) have been left over as 'vendor specific parts', in particular, the topology and link weight vector update strategy. As a side remark and an argument why these difficult issues have been treated in a stepmotherly fashion, many operators are reluctant to outsource the control of their network to 'intelligent' QoS routing software.

A corresponding protocol in the connectionless world (Internet) is not available due to a number of reasons. First, the Internet suffers from a routing legacy. Whereas the intradomain link state protocols (e.g. OSPF) could be upgraded relatively easy if, as in PNNI, a consistent topology and link weight vector update strategy existed, the situation for the interdomain routing protocols (e.g. BGP-4 and updates) is more problematic. Mainly the chaotic and unstandardized way in which peering agreements are signed makes current end-to-end (QoS) routing opaque and inefficient. Recent measurements [7] of the end-to-end delay show that the length of the end-to-end path can be several times (even an order of magnitude) longer than the shortest possible distance. Further, the average number of hops in the Internet [20] is about 15 while telephony operates with only half that number. Hence, there is room for improvements on the interdomain level! However, even if the interdomain routing would have been solved, the connectionless or hop-by-hop paradigm in Internet implies that QoS routing needs an exact algorithm (and concepts of active networking) as proved in [22].

The need for an exact routing algorithm brings us from the protocol to the algorithm, which is the focal point of this paper. The difficulty does not lie in the existence of an exact QoS algorithm - we have proposed SAMCRA (see section 2) - but in its complexity. For some time already, it is known [12], [25] that QoS routing with additive link weights is an NP-complete problem[1] which is interpreted, in practice, as unfeasible. Hence, only approximations (heuristics) with polynomial time complexity of the QoS algorithm are considered feasible. This point of view resulted in the publication of a wealth of heuristics [17], each of them claiming some attractive features over the others. Thus, these heuristics introduced a blurring factor in the already complex field of QoS routing because, as we claim here, their need or existence is argued as superfluous. Indeed, after about five years of extensive simulations on a huge number of graphs (of a particular class), we have never observed tendencies towards NP-complete behavior (e.g. see [22, 11]). The goal of this paper is to provide some arguments why we believe that, for a large class of graphs (and most likely also realistic networks), the QoS

[1]A problem is NP-complete if it cannot be solved in polynomial time. This means that the number of elementary computations or complexity C grows faster than any polynomial function if the problem parameters increase. Mathematically, let be a the basic problem parameter of problem P. Problem P is said to be NP-complete if, for $\epsilon > 0$, $C_P = O(\exp(\epsilon a))$ as $a \to \infty$ for any algorithm that solves problem P. If at least one algorithm is known, for which $C_P = O(a^\varepsilon)$ for some ε, then P is not NP-complete or is a polynomial problem. More details can be found in Garey and Johnson [12].

routing problem is not NP-complete. Unfortunately, so far, for that class of graphs we are unable to prove that SAMCRA (or equivalent QoS routing since SAMCRA is exact[2]) is *not* NP-complete.



0 0
5 4
5 4
5 4
25 5
5 4
25 5
10 8
5 4
30 9
15 12
5 4
12 6
5 4
22 14
20 16
37 11
35 13
5 5
42 16
40 18
27 19
25 21
22 5
50 1
70 17
72 15
47 26
49 24
85 14
87 12
62 23
64 21
i+1
$\begin{bmatrix} 0 \\ x^{\frac{i-1}{2}} - y \end{bmatrix}$
$\begin{bmatrix} 0 \\ y \end{bmatrix}$
i
$\begin{bmatrix} x^{\frac{i-1}{2}} \\ 0 \end{bmatrix}$
i+2

Figure 1: An example of a 'chain' topology that, when continued for large N (dotted lines), illustrates the NP-completeness of QoS routing. The basic ingredients for NP-completeness in this example are both the special topology structure that regularly (in the $i+2$ nodes) forces paths to coincide and the link weight structure that is correlated over the entire chain.

Proving or disproving NP-completeness is known to be notoriously difficult. The fact that QoS routing is, in general, NP-complete follows from Figure 1: if the chain topology is continued according to the triangular structure (bottom left), at each node i, the path vector or sum of the link weight vectors from left on up to node i (that are not dominated, explained below) grows as a power of 2. This example learns two basic features. First, NP-completeness involves a scaling of the relevant parameters. For example, if the number of added nodes N in the chain topology increases without bound, the number of entries (non-dominated path vectors) to make a QoS routing decision grows as $O(2^{N/2})$. Second, the cause of NP-completeness lies hidden in the special structure of the topology and, perhaps more important, in the specifics of the link weight vector. The latter is seen to be correlated over the path from A (left node) to B (node at the right). The proof that QoS routing is NP-complete in [25] heavily relies on the correlation of the link weight vectors. Furthermore, it suffices

[2]In [22], it has been proved that SAMCRA is exact. However, it has not been shown that SAMCRA is the most efficient algorithm possible. Therefore, the complexity derived from SAMCRA is strictly speaking an upper bound for the complexity of QoS routing.

to show that there exists one case (a specific topology and link weight structure) in which QoS routing is NP-complete to catalogue QoS routing *as a whole* as an NP-complete problem. Both the chain topology and the correlated link weight vectors are unlikely to occur in realistic networks and this article suggests that QoS routing in realistic networks may *not* be NP-complete. In other words, the QoS routing problem *confined to a particular class of networks* (topology and link weight structure) may *not* be NP-complete. We know that there are classes of graphs that are polynomially solvable. For example, QoS routing for the class of graphs where all nodes have degree 2 can always be solved in polynomial time irrespective of the link weight structure. Another "polynomially solvable" case is the class of graphs where all link weight vectors are identical. In summary, the difficulty remains in determining the full (or at least a large) set of classes for which QoS routing is not NP-complete and this article is a first, tiny step towards this goal.

After a brief presentation of SAMCRA [22], an exact Self-Adaptive Multiple Constraints Routing Algorithm, in section 2, the complexity is studied through simulations. The results show that for certain classes of graphs, the complexity of QoS routing does not seem to point towards a NP-complete behavior. The scaling (limit of N to infinity) cannot be simulated. Therefore, the second part (section 4) of this paper analyses the hopcount of the shortest path in multiple dimensions. This section is an extension to previous work on the hopcount in the Internet [20]. The analytic results indicate that, for the class of random graphs with uniformly distributed link weight components, further denoted as the RGU class, the hopcount in multiple dimensions behaves as a single dimension routing problem with modified link weight distribution. Hence, the combination of both results hints (unfortunately does not prove) that, for a large class of graphs, the QoS routing problem is not NP-complete.

2 SAMCRA: a Self-Adaptive Multiple Constraints Routing Algorithm

A network topology is denoted by $G(N, E)$, where N is the set of nodes and E is the set of links. With a slight abuse of notation we will also denote by N and E respectively the number of nodes and the number of links. A network supporting QoS consists of link weight vectors with m non-negative QoS measures ($w_i(e)$, $i = 1, ..., m$, $e \in E$) as components. The QoS measure of a path can either be additive in which case it is the sum of the QoS measures along the path (such as delay, jitter, the logarithm of packet loss, cost of a link, etc.) or it can be the minimum(maximum) of the QoS measures along the path (typically, available bandwidth and policy flags). Min(max) QoS measures are treated by omitting all links (and possibly disconnected nodes) which do not satisfy the requested min(max) QoS constraints. We call this topology filtering. Additive QoS measures cause more difficulties: *the multiple constrained path (MCP) problem, defined as finding a path subject to more than one additive constraint (L_i), is known to be NP-complete* [12], [25] and hence considered as intractable for large networks.

2.1 The concepts behind SAMCRA.

SAMCRA is the successor of TAMCRA, a Tunable Accuracy Multiple Constraints Routing Algorithm [11], [10]. As opposed to TAMCRA, SAMCRA guarantees to find a path within the constraints, provided such a path exists. Furthermore, SAMCRA only allocates queue-space (memory) when truly needed, whereas in TAMCRA the allocated queue-space is predefined. The major performance criterion for SAMCRA, namely the running-time/complexity, will be discussed in paragraph 2.2. Similar to TAMCRA, SAMCRA is based on three fundamental concepts:

1. **A non-linear measure for the path length**. Motivated by the geometry of the constraints surface in m-dimensional space, we defined the length of a path P as follows [11]:

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

   where $w_i(P) = \sum\limits_{e \in P} w_i(e)$.
   The definition of the path length has to be non-linear in order to guarantee that a retrieved path lies within the constraints, i.e. $l(P) \le 1$. A solution to the MCP problem is a path whose link weights are all within the constraints, $w_i(P) \le L_i$, for all $i = 1, .., m$. SAMCRA can also be applied to solve multiple constrained optimization problems, e.g. delay-constrained least-cost routing. 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. Examples of length functions different from (1) are given in [22]. In [13] and [15], TAMCRA-based algorithms with specific length functions are proposed and evaluated. By using the length function (1), all QoS measures are considered as equally important. An important corollary of a non-linear path length as (1) 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 us naturally to the k-shortest path approach.

2. **The k-shortest path approach.** The k-shortest path algorithm [8] is essentially Dijkstra's algorithm that does not stop when the destination is reached, but continues until the destination has been reached k times. If k is not restricted, the k-shortest path algorithm returns all possible paths between A and B. The k-shortest path approach is applied to the intermediate nodes i on the path from source node s to destination node d, where we keep track of multiple sub-paths from s to i. Not all sub-paths are stored, but an efficient distinction based on non-dominance is made.

3. **The principle of non-dominated paths** [14]. A (sub)-path Q is dominated by a (sub)-path P if $w_i(P) \le w_i(Q)$ for $i = 1, .., m$, with an inequality for at least one link weight component i. SAMCRA only considers non-dominated (sub)-paths. The principle of non-dominance enables an efficient reduction in the search-space (all paths between the source and destination) without compromising the solution.

2.2 The Complexity of SAMCRA

2.2.1 Worst-case complexity.

In the graph $G(N, E)$, the queue in SAMCRA can never contain more than kN path lengths, where k denotes the number of non-dominated paths (within the constraints) that are stored in the queue of a node. Because SAMCRA self-adaptively allocates the queue-size at each node, k may differ per node. When using a Fibonacci (or relaxed) heap to structure the queues [9], selecting the minimum path length among kN different path lengths takes at most a calculation time of the order of $log(kN)$. As each node can at most be selected k times from the queue, the **extract_min** function (explained in [9]) in line 6 of SAMCRA's meta-code (see Appendix B) takes $O(kNlog(kN))$ at most. The for-loop starting on line 11 is invoked at most k times from each side of each link in the graph. Calculating the length takes $O(m)$ when there are m metrics in the graph while verifying path dominance takes $O(km)$ at most. Adding or replacing a path length in the queue takes $O(1)$. Adding the contributions yields a worst-case complexity C_{SAMCRA} with $k = k_{\max}$ of

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

where $k_{\max}$ is an upper-bound on the number of non-dominated paths in $G(N, E)$. A precise expression for k is difficult to find. However, two upper-bounds can be determined. The first upper-bound $k_{\max}$ for k,

$$k_{\max} = \lfloor e(N-2)! \rfloor$$

is an upper-bound on the total number of paths between a source and destination in $G(N, E)$ [23].

If the constraints (or link weight components) have a finite granularity, a second upper-bound applies,

$$k_{\max} = \frac{\prod_{i=1}^{m} L_i}{\max_j(L_j)} \tag{3}$$

where the constraints L_i are expressed as an integer number of a basic metric unit. For instance, if the finest granularity for time measurements is 1 msec, then the constraint value for e.g. end-to-end delay is expressed in an integer number times 1 msec. This bound (3) has been published earlier in [11] but first proved here in Appendix A.

Clearly, for a single constraint ($m = 1$ and $k = 1$), the complexity (2) reduces to that of Dijkstra's algorithm. As mentioned above, for multiple (additive) metrics the worst-case complexity of SAMCRA is NP-complete. The NP-complete behavior also follows from (2) and the two expressions for $k_{\max}$.

2.2.2 Average-case complexity.

Although there exist many problems that are NP-complete, the average-case complexity might not be intractable, suggesting that such an algorithm could have a good performance in practice. The theory of average-case complexity was first advocated by Levin [18]. We will now give a calculation that suggests that *the average and even amortized*[3] *complexity of SAMCRA is polynomial in time* for fixed m and all weights w_i independent random variables.

[3] Amortized analysis differs from average-case analysis in that probability is not involved; an amortized analysis guarantees the average performance of each operation in the worst-case [9].

Lemma 1: *The expected number of non-dominated vectors in a set of T i.i.d. vectors in m dimensions is upper bounded by* $(\ln T)^{m-1}$.

A proof of Lemma 1 can be found by adopting a similar approach as presented by Barndorff-Nielsen and Sobel [4] or by Bentley *et al.* [5]. To gain some insight into the number of non-dominated paths in a graph, we will assume that the path-vectors are i.i.d. vectors. When we apply lemma 1 to the complexity of SAMCRA, we see that, in the worst-case, SAMCRA examines $T = \lfloor e(N-2)! \rfloor$ paths, leading us to an expected number of non-dominated paths between the source and destination in the worst-case of

$$(\ln T)^{m-1} = (\ln(\lfloor e(N-2)! \rfloor))^{m-1} \leq (1 + (N-2)\ln(N-2))^{m-1}$$

Hence, the amortized complexity of SAMCRA is given by (2) with $k = (\ln T)^{m-1} = (\ln(\lfloor e(N-2)! \rfloor))^{m-1}$, which is polynomial in N for fixed m.

In the limit $m \to \infty$ and for w_j independent random variables, all paths in $G(N,E)$ are non-dominated, which leads to the following lemma (proved in [22]).

Lemma 2: *If the m components of the link weight vectors are independent random variables and the constraints L_j are such that $0 \leq \frac{w_j}{L_j} \leq 1$, then any path with K hops has precisely a length (as defined by (1)) equal to K in the limit $m \to \infty$.*

This means that for $m \to \infty$ it suffices to calculate the minimum hop path, irrespective of the link weight distribution of the m independent components. Since the minimum hop problem is an instance of a single metric shortest path problem, it has polynomial complexity. Of course, this limit case $m \to \infty$ mainly has theoretically value. In realistic QoS routing, only a few link weights are expected to occur. The number m of QoS link weights is a design choice for the QoS routing protocol.

In summary, if the link weights are independent random variables, there are two properties reducing the search-space (the number of paths to be examined) of SAMCRA. For m small the concept of non-dominance is very powerful, resulting in the presence of only a small number of non-dominated paths between two points in a graph. At the other extreme, for m large, the values L_j of the constraints cause the largest search-space reduction, because only a few paths between the source and destination lie within the constraints. Even if the constraints are chosen infinitely large, SAMCRA may lower them in the course of the computation (by means of `endvalue`, line 25/26 meta-code) without affecting the solution.

The two properties complement each other, resulting in an overall good average performance of SAMCRA. The simulation results of the next section indicate that the average complexity of SAMCRA is $O(N \log N + mE)$ which follows from (2) with $E[k] \approx 1$, for the class $G_p(N)$ of random graphs [6], with independent uniformly distributed link weights, further denoted by RGU.

3 Simulation results on complexity

3.1 The class RGU

The simulations were performed on random graphs of the type $G_p(N)$ [6]. The expected link density p also equals the probability that there is a link between two nodes.. The m components of the link weight vector, $w_i(e)$, are independent uniformly distributed random variables in the range [0, 1].

SAMCRA searches for the shortest path within the constraints instead of just any path obeying the constraints. Since all link weights are in the range [0, 1], the highest possible weight of a loop-free path $w_i(P)$ is $N-1$. By choosing all the constraints equal to N, all possible (loop-free) paths between the source and destination will be able to satisfy these constraints. When the goal is to find the shortest path within a set of constrained paths, choosing the constraints large and hence increasing the amount of paths that need to be evaluated, will increase the complexity of finding the shortest path. Hence, we are simulating a worst-case scenario by choosing the constraints equal to N.

For each simulation run, we used SAMCRA to compute the shortest path between a source node (labelled 1) and destination node (labelled N) for a large number (minimum of 10^5) of random graphs of the type $G_p(N)$. For each computation we stored the hopcount and minimum queue-size ($k_{\min}$) needed to find the shortest path within the constraints. If TAMCRA had used that particular $k_{\min}$ under the same conditions, it would have found the same shortest path as SAMCRA did, but if a smaller value had been used TAMCRA would not have found the shortest path.

For all simulations in this section, we have computed the 95% confidence intervals, which were all less than 0.52% of the mean. Furthermore, we have validated our simulations with the chain topology of Figure 1. These simulations confirmed that, once the constraints are large enough, $k_{\min}$ grows exponentially (as $O(2^{\frac{N}{2}})$) with N. Moreover, if for the same topology $w_i(e) = w_i(e^*)$, for $i = 1, ..., m$, $\forall e, e^* \in E$ or if $w_i(e) = w(e)$, for $i = 1, ..., m$, $e \in E$, then we found that $k_{\min} = 1$. This shows that the link weight structure is very important.

We will start by presenting the simulation results for $m = 2$. Figure 2 gives the minimum queue-size ($k_{\min}$) needed to find the shortest path within the constraints. Figure 2 also gives the statistics corresponding to the results for $k_{\min}$.

Figure 2 shows that an increase of N does not result in an exponential increase in $k_{\min}$. The expectation $E[k]$ remains close to 1 and hardly increases with N. If we extrapolate these results to $N \to \infty$, Figure 2 suggests that for the class of random graphs $G_p(N)$ with 2 independent uniformly distributed link weights, the average complexity of SAMCRA is approximately $O(N \log N + 2E)$.

Figures 3 and 4 show a similar behavior ($E[k] \approx 1$). The only difference is that the worst-case values ($\max[k]$) have slightly increased with m, as a result of the increased expected number of non-dominated paths. However, since $E[k]$ stays close to one, the simulation results suggest that the two search-space-reducing concepts, dominance and constraint values, are so strong that the average complexity of SAMCRA is not significantly influenced by the number of metrics m. The behavior of $k_{\min}$ as a function of the number of constraints m is illustrated in Figure 5.

In the previous paragraph we indicated that there are two concepts reducing the search-space of SAMCRA. For small m the dominant factor is the non-dominance property, whereas for $m \to \infty$ the constraint values are more dominant. Because these properties are most effective in a certain range of m's, we expect the worst-case behavior to occur with an m that is neither small nor large. Figure 5 shows the k-distribution for different values of m. The best performance is achieved with $m = 2$ and the worst performance is for m around 8. However, as Figure 4 illustrates, $E[k]$ for $m = 8$ is still approximately 1, leading us to believe that for the class of random graphs $G_p(N)$ with independent uniformly distributed weights, the average complexity of SAMCRA is approximately $O(N \log N + mE)$ for every m.

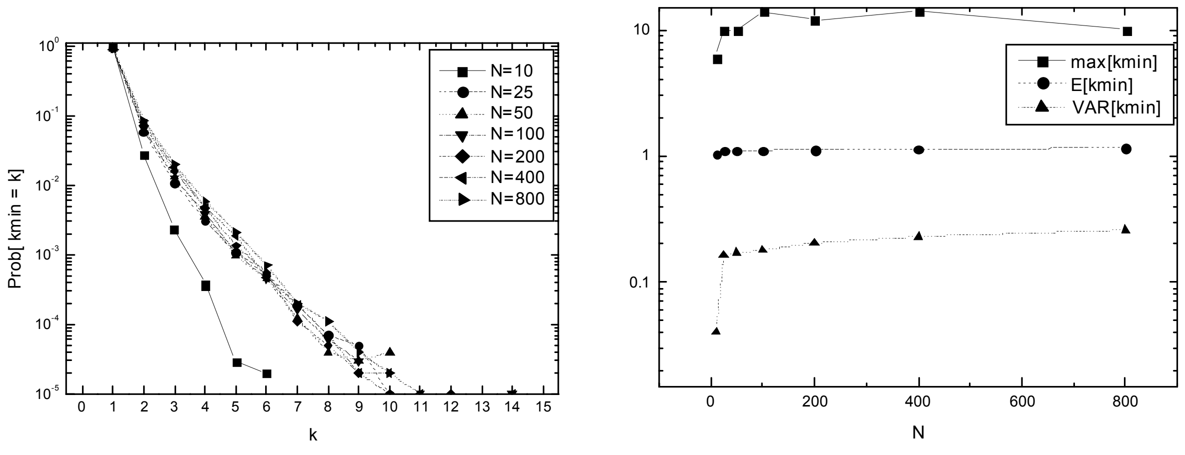


Figure 2: Results for $k_{\min}$ with $m = 2$.

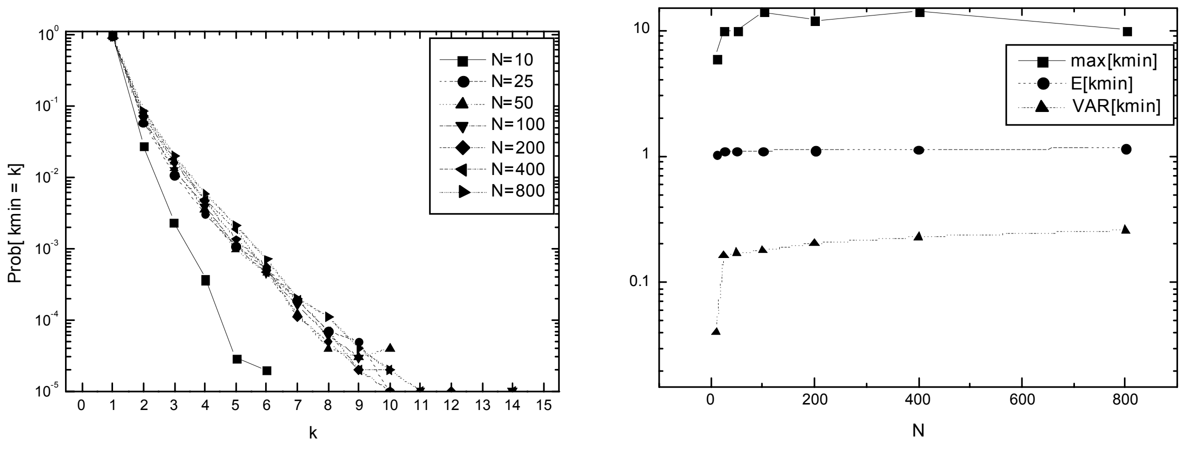


Figure 3: Results for $k_{\min}$ with $m = 4$

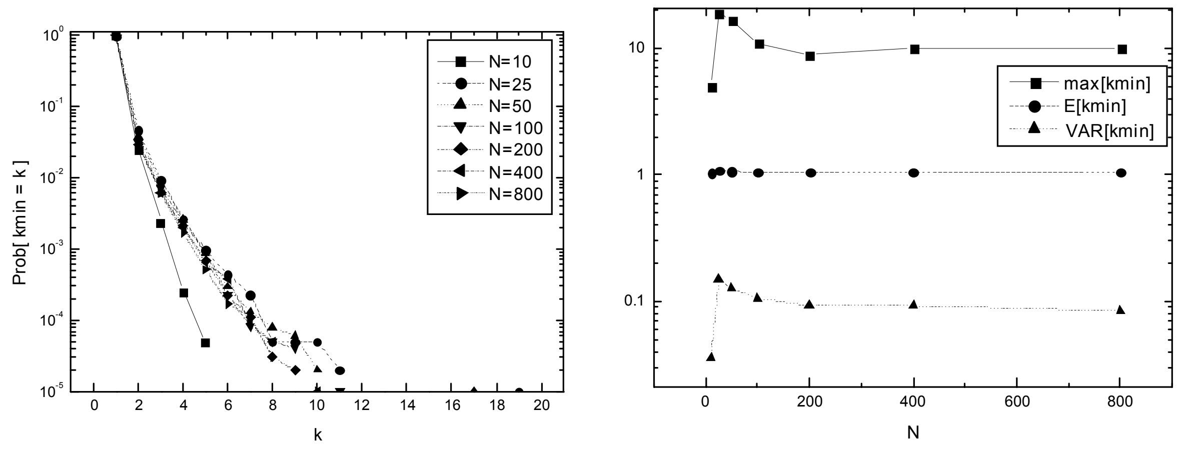


Figure 4: Results for $k_{\min}$ with $m = 8$

Our last simulation concerns the granularity of the constraints (or link weights). When the granularity is finite, an upper-bound, in terms of the constraints, on the number of non-dominated paths can be found (3). The finer the granularity, the larger this upper-bound. Figure 6 confirms this behavior (for $N = 20$ and $p = 0.2$). In practice the constraints (or link weights) have a finite granularity, which according to Figure 6 will positively influence the running-time of SAMCRA.

Because SAMCRA solves the MCP problem exactly, and since the simulations suggest that SAMCRA's average complexity is polynomial for $G_p(N)$ with independent uniformly distributed link weights, the MCP problem for that class RGU seems, on average, solvable in polynomial time. The observation that $E[k] \approx 1$ indicates that, for the sizes N considered, routing in multiple dimensions is analogous to routing in a single dimension ($m = 1$).

3.2 A two-dimensional lattice.

Contrary to the class of RGU graphs, this section is devoted to extremely regular graphs, namely two-dimensional lattices. The regular structure of Figure 1 has inspired us to investigate the complexity of SAMCRA in lattices. The m components of the link weight vector, $w_i(e)$, remain independent uniformly distributed random variables. We refer to the class of two-dimensional lattices with uniformly distributed link weight vectors as 2LGU. Figure 7 gives an example of a two-dimensional lattice consisting of 16 nodes: every internal node has degree 4 while nodes on the boundary have degree 3 and the 4 corner nodes have degree 2.

For our simulations we only generated square-shaped lattices. Since the sides of this square consist of $\sqrt{N}$ nodes, $\sqrt{N}$ must be an integer number. The source node is chosen in the upper left corner and the destination node in the right lower corner. This location of source and destination is expected to lead to worst-case behavior for SAMCRA because it yields the longest hop-count path. All simulations were performed on 10^5 graphs in the class 2LGU.

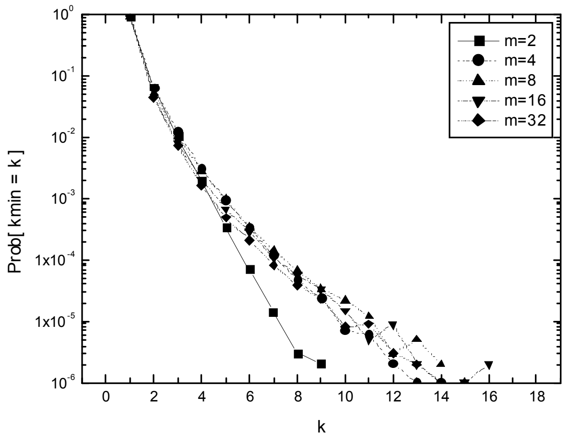


Figure 5: P.d.f. of k_{min}, $N = 20$

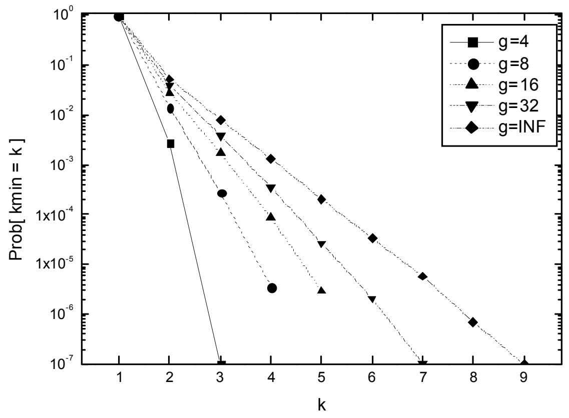


Figure 6: k_{min} for different granularity

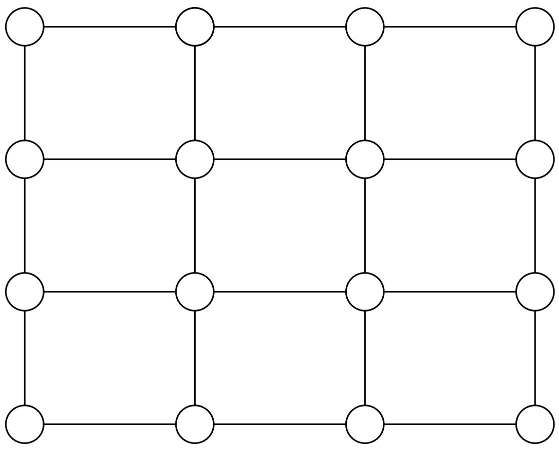


Figure 7: A two-dimensional lattice of 16 nodes

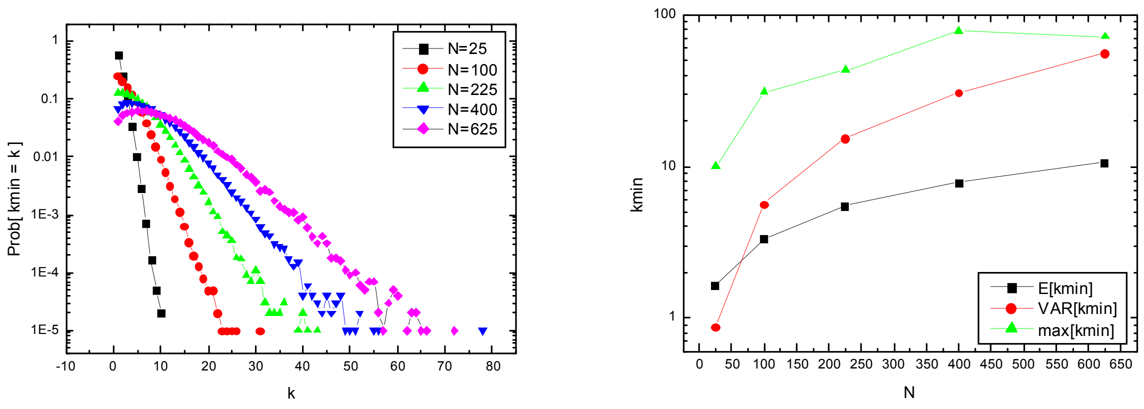


Figure 8: Results for $k_{\min}$ with $m = 2$ as a function of the number of nodes N.

Figure 8 displays the probability distribution of $k_{\min}$ with the most important statistical parameters. As expected, Figure 8 illustrates that $k_{\min}$ indeed increases faster than for the class RGU with same size N. Figure 8 also shows that $\max(k_{\min})$ increases only linearly with N (in the range up to 625 nodes). Hence, if this linear tendency can be extrapolated to large N, we would be able to conclude that also the class 2LGU does not cause SAMCRA's complexity to be NP-complete.

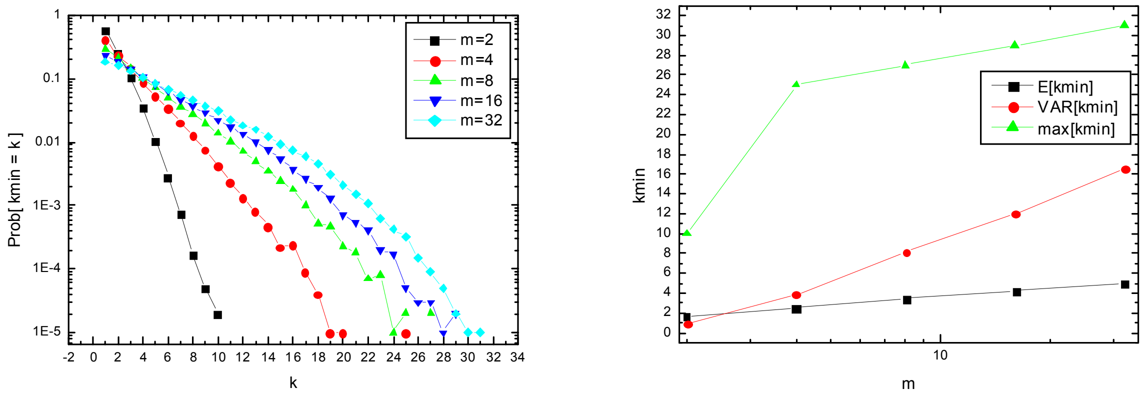


Figure 9: Results for $k_{\min}$ with $N = 25$ for different dimensions m.

Figure 9 shows the behavior of $k_{\min}$ as function of the dimension m (at a fixed size $N = 25$). Up to $m = 32$, the statistics for $k_{\min}$ increase with m. For the class RGU displayed in Figure 5, a similar increase of $k_{\min}$ up to $m = 8$ is observed, but $k_{\min}$ decreases afterwards. This decrease is due to the constraint values, that are chosen "infinitely" large but are reduced in the course of the path computation to the path weight vector of an already discovered path that obeys the constraints (see Appendix B line 25 and 26 in SAMCRA's metacode). The minimum number of hops between the source and destination is on average larger in the class 2LGU than in the class RGU. As a result the constraint values are reduced at a later point in the path computation and therefore we do not see a decrease in Figure 9 as soon as we did in Figure 5. In reality the constraint values will not be chosen infinitely large, leading to a smaller search-space and consequently smaller complexity.

4 The expected hopcount $E[h_N]$ for the random graph $G_p(N)$

Only the class RGU is considered here. Each link is specified by a weight vector with m independent components possessing the same distribution function

$$F_w(x) = \Pr[w \leq x] = x^{\alpha} 1_{[0,1]}(x) + 1_{(1,\infty)}(x), \quad \alpha > 0 \tag{4}$$

For this network topology, the expected hopcount $E[h_N]$ or the average number of traversed routers along a path between two arbitrarily chosen nodes in the network will be computed. The behavior of the expected hopcount $E[h_N]$ in multiple dimension QoS routing will be related to the single metric case ($m = 1$). That case $m = 1$ has been treated previously in [20], where it has been shown, under quite general assumptions, that

$$E[h_N] \sim \frac{\ln N}{\alpha}$$
$$var[h_N] \sim \frac{\ln N}{\alpha^2}$$

Lemma 2 shows that for $m \to \infty$ in the class RGU, the shortest path is the one with minimal hopcount. Thus the derivation for a single weight metric in [20] for $G_p(N)$ with all link weights 1 is also valid for $m \to \infty$. The first order (asymptotic) calculus as presented in [20] will be extended to $m \geq 2$ for large N. In that paper, the estimate

$$\Pr[h_N = k, w_N \leq z] \simeq N^{k-1} p^k F_w^{k*}(z),$$

has been proved, where the distribution function $F_w^{k*}(z)$ is the probability that a sum of k independent random variables each with d.f. F_w is at most z and is given by the k-fold convolution

$$F_w^{k*}(z) = \int_0^z F_w^{(k-1)*}(z-y) f_w(y)\, dy, \quad k \geq 2,$$

and where $F_w^{1*} = F_w$. By induction it follows from (4), that for $z \downarrow 0$,

$$F_w^{k*}(z) \sim \frac{z^{\alpha k} (\alpha \Gamma(\alpha))^k}{\Gamma(\alpha k + 1)}.$$

In multiple (m) dimensions, SAMCRA's definition of the path length (1) requires the maximum link weight of the individual components $w_N(\gamma) = \max_{i=1,\ldots,m} [w_i(\gamma)]$ along some path γ. Since we have assumed that the individual links weight components are i.i.d random variables, and hence $\Pr[w_N \leq z] = (\Pr[w_i \leq z])^m$, this implies for m-dimensions that

$$F_w^{k*}(z) \sim \left[\frac{z^{\alpha k} (\alpha \Gamma(\alpha))^k}{\Gamma(\alpha k + 1)} \right]^m$$

such that

$$\Pr[h_N = k, w_N \leq z] \simeq N^{k-1} p^k \left[\frac{z^{\alpha k} (\alpha \Gamma(\alpha))^k}{\Gamma(\alpha k + 1)} \right]^m$$

We will further confine to the case $\alpha = 1$, i.e. each link weight component is uniformly distributed over $[0, 1]$ as in the class RGU,

$$\Pr[h_N = k, w_N \leq z] \simeq \frac{1}{N} \frac{(Npz^m)^k}{(k!)^m} \tag{5}$$

For a typical value of z, the probabilities in (5) should sum to 1,

$$1 = \frac{1}{N} \sum_{k=1}^{N-1} \frac{(Npz^m)^k}{(k!)^m}$$

At last, for a typical value of z, $\Pr[w_N \leq z]$ is close to unity resulting in

$$\Pr[h_N = k, w_N \leq z] \simeq \Pr[h_N = k]$$

Let us denote with $y = Npz^m$,

$$S_m(y) = \sum_{k=0}^{N-1} \frac{y^k}{(k!)^m} \tag{6}$$

subject to

$$N + 1 = S_m(y) \tag{7}$$

Hence, the typical value y of the end-to-end link weight that obeys (7) is independent on the link density p for large N. Also the average hopcount and the variance can be written in function of $S_m(y)$ as

$$\begin{aligned} E\left[h_N\right] &= \frac{y}{N} S_m^{'}(y) & (8) \\ var\left[h_N\right] &= \frac{1}{N}\left[y^2 S_m^{"}(y) + y S_m^{'}(y) - \frac{y^2}{N}\left(S_m^{'}(y)\right)^2\right] & (9) \end{aligned}$$

We will first compute good order approximations for $E[h_N]$ in the general case and only $var\left[h_N\right]$ and the ratio $\alpha = \frac{E[h_N]}{var[h_N]}$ in case $m = 2$. Let us further concentrate on

$$V_m(y) = \sum_{k=0}^{\infty} \frac{y^k}{(k!)^m} \tag{10}$$

Clearly, $V_m(y) = \lim_{N \to \infty} S_m(y)$. It is shown in [16, Appendix] that

$$V_m(y) \sim A_m(y) \exp\left[m y^{1/m}\right] \tag{11}$$

with

$$A_m(y) = \frac{(2\pi)^{\frac{1-m}{2}}}{\sqrt{m}} y^{-\frac{1}{2}\left(1-\frac{1}{m}\right)} \tag{12}$$

After taking the logarithmic derivative of relation (11), we obtain

$$V_m'(y) \sim V_m(y)\left[\frac{A_m'(y)}{A_m(y)} + y^{\frac{1}{m}-1}\right]$$

In view of (7), y is a solution of $V_m(y) \sim N$, such that the average (8) becomes

$$E\left[h_N\right] \sim \frac{y}{N} V_m^{'}(y) \sim \frac{V_m(y)}{N}\left[y\frac{A_m'(y)}{A_m(y)} + y^{\frac{1}{m}}\right]$$

or

$$E\left[h_N\right] \sim y^{\frac{1}{m}} + y\frac{A_m'(y)}{A_m(y)} \tag{13}$$

Using (12) in (13), we arrive at

$$E\left[h_N\right] \sim y^{\frac{1}{m}} - \frac{1}{2}\left(1 - \frac{1}{m}\right) \tag{14}$$

where y is a solution of $V_m(y) \sim N$. As shown in [16], the solution y is

$$y \sim \left(\frac{\ln N}{m}\right)^m + \left(\frac{\ln N}{m}\right)^{m-1}\left[\frac{1}{2}(m-1)\ln(\ln N) + Q\right] + O\left(\ln(\ln N)\ln^{m-2} N\right).$$

Introduced into (14), the average hopcount follows as

$$\begin{aligned} E\left[h_N\right] \sim\ & \frac{\ln N}{m} + \frac{1}{2}\left(1 - \frac{1}{m}\right)\ln\left(\ln N\right) + \frac{\ln m}{2m} - \frac{1}{2}\left(1 - \frac{1}{m}\right)\left(\ln\left(\frac{m}{2\pi}\right) + 1\right) \\ & + O\left(\frac{\ln\left(\ln N\right)}{\ln N}\right) \end{aligned} \tag{15}$$

This formula indicates that, to a first order, $m = \alpha$. The simulations (Figures 10 and 11) show that, for higher values of m, the expectation of the hopcount tends slower to the asymptotic $E\left[h_N\right]$-regime given by (15).

For the computation of the variance, we confine ourselves to the case $m = 2$, for which $V_2(y) = \sum_{k=0}^{\infty} \frac{y^k}{(k!)^2} = I_0(2\sqrt{y})$ where $I_0(z)$ denotes the modified Bessel function of order zero [1, sec. 9.6]. The variance of the hopcount from (9) with $S_m^{''}(y) = \frac{d^2 I_0(2\sqrt{y})}{dy^2} = \frac{I_0(2\sqrt{y})}{y} - \frac{I_1(2\sqrt{y})}{y\sqrt{y}}$

$$\begin{aligned} var\left[h_{N,2}\right] &\sim \frac{y}{N} I_0(2\sqrt{y}) - \frac{\sqrt{y}}{N} I_1(2\sqrt{y}) + E\left[h_{N,2}\right] - \left(E\left[h_{N,2}\right]\right)^2 \\ &\sim y - \left(E\left[h_{N,2}\right]\right)^2 \end{aligned}$$

At this point, we must take the difference between $I_0(x)$ and $I_1(x)$ into account otherwise we end up with $var\left[h_N\right] \sim 0$. For large x,

$$I_0(x) \sim \frac{e^x}{\sqrt{2\pi x}}\left(1 + \frac{1}{8x} + O\left(x^{-2}\right)\right)$$

and

$$I_1(x) \sim \frac{e^x}{\sqrt{2\pi x}}\left(1 - \frac{3}{8x} + O\left(x^{-2}\right)\right)$$

such that

$$I_1(x) \sim I_0(x)\left(1 - \frac{1}{2x} + O\left(x^{-2}\right)\right)$$

$$\begin{aligned} E\left[h_{N,2}\right] &\sim \frac{y}{N} I_1(2\sqrt{y}) \frac{1}{\sqrt{y}} \sim \frac{I_0(2\sqrt{y})}{N}\left(\sqrt{y} - \frac{1}{4} + O\left(y^{-1}\right)\right) \\ &\sim \frac{\ln(N)}{2} + \frac{\ln\left(\ln(N)\right)}{4} - \frac{1}{4} + O\left(\frac{1}{\ln(N)}\right) \end{aligned} \tag{16}$$

Thus,

$$var\left[h_{N,2}\right] \sim y - \left(\sqrt{y} - \frac{1}{4}\right)^2 = \frac{\sqrt{y}}{2} - \frac{1}{16} + O\left(\frac{1}{\sqrt{y}}\right) \tag{17}$$

and

$$\alpha = \frac{E\left[h_{N,2}\right]}{var\left[h_{N,2}\right]} \sim 2 - \frac{1}{4\sqrt{y}} + O\left(y^{-1}\right) \sim 2 - \frac{\sqrt{2}}{4\sqrt{\ln N}} + O\left(\frac{1}{\ln(N)}\right) \tag{18}$$

This corresponds well with the simulations shown in Figure 9. In addition, the average and variance of the hopcount for $m = 2$ dimensions scales with N in a similar fashion as the same quantities in $G_p(N)$ with a single link weight, but polynomially distributed with $F_w\left[w \leq x\right] = x^2$.

In summary, the asymptotic analysis reveals that, for the class RGU, the hopcount in m dimensions behaves similarly as in the random graph $G_p(N)$ in $m = 1$ dimension with polynomially distributed

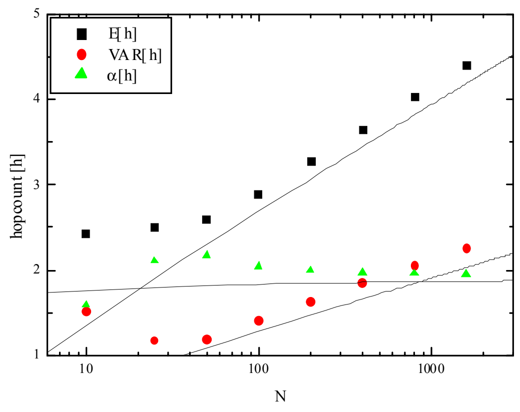
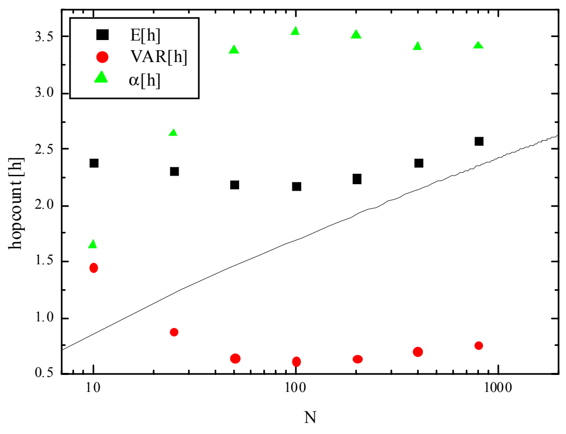


Figure 10: The average $E[h_{N,2}]$, the variance $var[h_{N,2}]$ and the ratio $\alpha = \frac{E[h_{N,2}]}{var[h_{N,2}]}$ of the shortest path found by SAMCRA, as a function of the size of the random graph N with two link metrics ($m = 2$). The full lines are the theoretical asymptotics



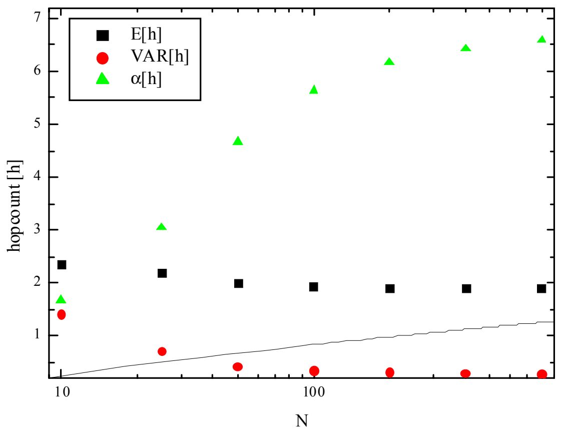


Figure 11: Hopcount statistics for $m = 4$ (left) and $m = 8$ (right).

link weights specified via (4) where the polynomial degree α is precisely equal to the dimension m. This result, independent of the simulations of the complexity of SAMCRA, suggests a transformation of shortest path properties in multiple dimensions to the single parameter routing case, especially when the link weight components are independent. As argued in [20], the dependence of the hopcount on a particular topology is less sensitive than on the link weight structure, which this analysis supports.

5 Conclusions

Since constrained-based routing is an essential building block for a future QoS-aware network architecture, we have proposed a multiple constraints, exact routing algorithm called SAMCRA. Although the worst-case complexity is NP-complete (which is inherent to the fact that the multiple constraints problem is NP-complete), when the link weight vectors are i.i.d. vectors the average complexity in the worst-case is polynomial. A large amount of simulations on random graphs with independent link weight components seem to suggest that the worst-case complexity is polynomial for this class of graphs and that the average-case complexity is similar to the complexity in the single parameter case. Simulations on a different class of graphs, namely the regular two-dimensional lattice graphs with uniformly distributed link weights display, as expected, a higher computational complexity. However, also these simulations suggest a polynomial complexity in the worst case. For the considered classes, the MCP problem thus seems tractable.

The second part of this paper was devoted the study of the hopcount in multiple dimensions as in QoS–aware networks. For random graphs of the class RGU, a general formula for the expected hopcount in m dimensions has been derived and only extended to the variance $var\left[h_N\right]$ as well in $m = 2$ dimensions, in order to compute the variance and the ratio of the expected hopcount and its variance. To first order, with the network size $N >> m$ large enough, the expected hopcount behaves asymptotically similar as the expected hopcount in $m = 1$ dimension with a polynomial distribution function $(x^\alpha 1_{[0,1]}(x) + 1_{(1,\infty)}(x))$ and polynomial degree $\alpha = m$.

Both the complexity analysis and the hopcount computation suggests that for special classes of networks, among which random graphs of the class RGU in m dimensions, the QoS routing problem exhibits features similar to the one dimensional (single parameter) case. The complexity analysis suggested this correspondence for small N, whereas the asymptotic analysis for the hopcount revealed the connection for $N \to \infty$. We showed that there are indeed classes of graphs for which the MCP problem is not NP-complete. The problem is to determine the full set of classes of graphs which posses a polynomial, rather than non-polynomial *worst* case complexity. Further, what is the influence of the correlation structure between the link weight components because lemma 1 and the simulations suggest that independence of these link weight components seems to destroy NP-completeness. Moreover, we notice that the proof presented in [25] strongly relies on the choice of the link weights. At last, if our claims about the NP-completeness would be correct, how large is then the class of networks that really lead to an NP-complete behavior of the MCP problem? In view of the large amount of simulations performed over several years by now, it seems that this last class fortunately must be small, which suggests that, in practice, the QoS-routing problems may turn out to be feasible.

References

[1] Abramowitz, M. and I.A. Stegun (eds.), *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*, Dover Pubns, June 1974, ISBN: 0486612724.

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

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

[4] Barndorff-Nielsen, O. and M. Sobel, "On the distribution of the number of admissible points in a vector random sample", Theory of Probability and its Applications, vol. 11, no. 2, 1966.

[5] Bentley, J.L., H.T. Kung, M. Schkolnick and C.D. Thompson., "On the Average Number of Maxima in a Set of Vectors and Applications", Journal of ACM, vol. 25, no. 4, pp. 536-543, 1978.

[6] Bollobas, B., *Random Graphs*, Academic Press, London, 1985.

[7] Bovy, C. J., H. T. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal and P. Van Mieghem, "Analysis of End-to-end Delay Measurements in Internet", accepted for PAM2002.

[8] Chong, E.I., 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.

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

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

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

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

[13] Guo, L. 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.

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

[15] Korkmaz, K. and M. Krunz, "Multi-Constrained Optimal Path Selection", IEEE INFOCOM 2001.

[16] Kuipers, F. A. and P. Van Mieghem, "QoS Routing: Average complexity and hopcount in m dimensions", Proceedings of 2nd International Workshop on Quality of Future Internet Services, QofIS2001 [edited by M. Smirnov et al. in Springer Verlag LNCS 2156], Coimbra, Portugal, September 24-26, pp. 110-126, 2001.

[17] Kuipers, F. A., T. Korkmaz, M. Krunz and P. Van Mieghem, "A Review of Constraint-Based Routing Algorithms", submitted to Proceedings of the IEEE.

[18] Levin, L.A., "Average Case Complete Problems", SIAM J. Comput., 1986, 15(1):285-286.

[19] Titchmarsh, E. C., *The Theory of Functions*, Oxford University Press, 1964.

[20] Van Mieghem, P., G. Hooghiemstra and R. van der Hofstad, "A Scaling Law for the Hopcount in Internet", Delft University of Technology, report 2000125, 2000, http://wwwtvs.et.tudelft.nl/people/piet/telconference.html.

[21] Van Mieghem, P. and H. De Neve, "Aspects of Quality of Service Routing", SPIE'98, Nov. 1-6, Boston (USA), 3529A-05, 1998

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

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

[24] Van Mieghem, P., "The Asymptotic Behaviour of Queueing Systems: Large Deviations Theory and Dominant Pole Approximation", Queueing Systems 23, 27-55, 1996.

[25] Wang, Z. and J. Crowcroft, 1996, "QoS Routing for supporting Multimedia Applications", IEEE JSAC, 14(7): 1188-1234, September 1996.

A Proof of (3) for $k_{\max}$.

Recall that all weight components have a finite granularity, which implies that they are expressed in an *integer* number times a basic unit.

Definition 1 *Non-dominance*: A path P is called non-dominated if there does not exist a path P^* for which $w_i(P^*) \leq w_i(P), \forall i$ and $\exists j : w_j(P^*) < w_j(P)$.

Ad definition 1: 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. In the sequel we will therefore extend definition 1 to denote one path out of the set of equal weight vector paths as being non-dominated and regard the others as dominated paths.

Theorem 1 *The number of non-dominated paths within the constraints cannot exceed (3).*

Proof: Without loss of generality we assume that $L_1 \leq L_2 \leq ... \leq L_m$ such that (3) reduces to $k_{\max} = \prod_{i=1}^{m-1} L_i$.

First, if $m = 1$, there is only one shortest and non-dominated (according to **Ad definition 1)** path possible within the constraint L_1. This case reduces to single parameter shortest path routing with $k_{\max} = 1$.

For $m \geq 2$, the maximum number of distinct paths[4] is $\prod_{i=1}^{m} L_i$. Two paths P_1 and P_2 do not dominate each other if, for at least two different link weight components $1 \leq a \neq b \leq m$ holds that $w_a(P_1) < w_a(P_2)$ and $w_b(P_1) > w_b(P_2)$. This definition implies that, for any couple of non-dominated paths P_1 and P_2, at least two components of the m-dimensional vector $\vec{w}(P_1)$ must be different from

[4]Two paths are called distinct if their path weight vectors are not identical.

$\vec{w}(P_2)$. Equivalently, if we consider a $(m-1)$-dimensional subvector $\vec{v}$ by discarding the j-th component in $\vec{w}$, at least one component of $\vec{v}(P_1)$ must differ from $\vec{v}(P_2)$. The maximum number of different subvectors $\vec{v}$ equals $\prod_{i=1;i\neq j}^{m} L_i$. If $j \neq m$ such that $L_j < L_m$, within the $\prod_{i=1;i\neq j}^{m} L_i$ possibilities, there are paths for which only the j-th and/or m-th component differ while all the other components are equal. In order for these paths not to dominate each other, the j-th and m-th component must satisfy the condition that if $w_m(P_1) > w_m(P_2)$, then $w_j(P_1) < w_j(P_2)$ or vice versa. For the m-th component, there are L_m different paths for which $w_m(P_1) = L_m > w_m(P_2) = L_m - 1 > ... > w_m(P_{L_m}) = 1$. Since $L_j < L_m$, there are only L_j paths for which $w_j(P_1) = 1 < w_j(P_2) = 2 < ... < w_j(P_{L_j}) = L_j$. Therefore, there are paths P_1 and P_2 for which $w_m(P_1) > w_m(P_2)$, but $w_j(P_1) = w_j(P_2)$. Hence, only L_j instead of L_m non-dominated paths are possible, leading to a total of $\frac{L_j}{L_m}\prod_{i=1;i\neq j}^{m} L_i = \prod_{i=1}^{m-1} L_i$ non-dominated paths. This proofs the upperbound $k_{\max} = \prod_{i=1}^{m-1} L_i$. □

Corollary 2 *(3) is strict.*

Proof: Without loss of generality assume that $L_1 \leq L_2 \leq ... \leq L_m$. We will show that there exist sequences $\{L_1, L_2, ..., L_m\}$ for which the bound (3) is achieved.

If for each pair of paths P_i, P_j the m-th link weight component obey

$$w_m(P_i) \geq w_m(P_j) + \sum_{k=1}^{m-1} (w_k(P_j) - w_k(P_i)) \tag{19}$$

then (3) is a strict, attainable bound.

Formula (19) is found by recursively applying the following prerequisite recalling that the smallest difference between two weight components is one unit. If for two paths P_1, P_2 applies that $w_j(P_1) - w_j(P_2) = 1$ (in units of the j-th weight component) for only one j of the first $m-1$ metrics and $w_i(P_1) - w_i(P_2) = 0$ for the other $1 \leq i \neq j \leq m-1$, then for non-dominance to apply, the m-th weight components must satisfy $w_m(P_1) - w_m(P_2) \leq -1$. If (19) is not obeyed, then $w_m(P_1) > w_m(P_2) - 1$, i.e. $w_i(P_1) \geq w_i(P_2)$ for $i = 1, ..., m$ and according to the definition of non-dominance P_1 is then dominated by P_2.

The largest possible difference between two path vectors provides us with a lower-bound on L_m,

$$L_m \geq 1 + \sum_{i=1}^{m-1} (L_i - 1)$$

When this bound is not satisfied, then the number of non-dominated paths within the constraints is smaller than (3). □

For example, in $m = 5$ dimensions, with $L_1 = 1,\ L_2 = 2,\ L_3 = 3,\ L_4 = 3,\ L_5 = 6$ $(\geq 1+1+2+2)$, all 18 non-dominated vectors are

$$\begin{vmatrix}1\\1\\1\\1\\6\end{vmatrix}\begin{vmatrix}1\\1\\1\\2\\5\end{vmatrix}\begin{vmatrix}1\\1\\1\\3\\4\end{vmatrix}\begin{vmatrix}1\\1\\2\\1\\5\end{vmatrix}\begin{vmatrix}1\\1\\2\\2\\4\end{vmatrix}\begin{vmatrix}1\\1\\2\\3\\3\end{vmatrix}\begin{vmatrix}1\\1\\3\\1\\4\end{vmatrix}\begin{vmatrix}1\\1\\3\\2\\3\end{vmatrix}\begin{vmatrix}1\\1\\3\\3\\2\end{vmatrix}\begin{vmatrix}1\\2\\1\\1\\5\end{vmatrix}\begin{vmatrix}1\\2\\1\\2\\4\end{vmatrix}\begin{vmatrix}1\\2\\1\\3\\3\end{vmatrix}\begin{vmatrix}1\\2\\2\\1\\4\end{vmatrix}\begin{vmatrix}1\\2\\2\\2\\3\end{vmatrix}\begin{vmatrix}1\\2\\2\\3\\2\end{vmatrix}\begin{vmatrix}1\\2\\3\\1\\3\end{vmatrix}\begin{vmatrix}1\\2\\3\\2\\2\end{vmatrix}\begin{vmatrix}1\\2\\3\\3\\1\end{vmatrix}$$

B SAMCRA's meta-code

SAMCRA(G, s, d, L)

G: graph, s: source node, d: destination node, L: constraints

```
1.  counter = 0 for all nodes
2.  endvalue = 1.0
3.  path(s[1]) = NULL and length(s[1]) = 0
4.  put s[1] in queue
5.  while(queue ≠ empty)
6.      extract_min(queue) -> u[i]
7.      u[i] = marked grey
8.      if(u = d)
9.          stop
10.      else
11.          for each v ∈ adjacency_list(u)
12.              if(v ≠ previous node of u[i])
13.                  PATH = path(u[i]) + (u,v)
14.                  LENGTH = length(PATH)
15.                  check all non-black paths at v and PATH
            for dominancy & endvalue → mark obsolete paths black
16.                  if(LENGTH ≤ endvalue and PATH non-dominated)
17.                      if(no black paths)
18.                          counter(v) = counter(v)+1
19.                          j = counter(v)
20.                          path(v[j]) = PATH
21.                          length(v[j]) = LENGTH
22.                          put v[j] in queue
23.                      else
24.                          replace a black path with PATH
25.                      if(v = d and LENGTH < endvalue)
26.                          endvalue = LENGTH
```

For a detailed explanation of this meta-code, we refer to [22].