TAMCRA:

A Tunable Accuracy Multiple Constraints Routing Algorithm

Hans De Neve and Piet Van Mieghem

Alcatel Corporate Research

Francis Wellesplein 1, B-2018 Antwerp (BELGIUM)

Abstract

Routing with more than one additive constraints is known to be an NP-complete problem, and hence considered as intractable. We present a new QoS routing algorithm, called Tunable Accuracy Multiple Constraints Routing algorithm (TAMCRA), which solves the problem in a fast and accurate way.

1. INTRODUCTION

Although the current internet is a best-effort network, serious efforts are spent to offer QoS and some notion of guarantee. The IETF working groups that focus on the broad implications of providing QoS are QoSR (QoS based routing), RSVP (signalling for resource reservations), Integrated Services (traffic aspects) and RTP (time information). In the ATM Forum on the other hand, PNNI is promoted as the future-safe, powerful protocol that inherits the full QoS capabilities of ATM. Thus, in short, QoS aspects in dynamic, distributed networks deserve today due attention and we refer for a broad discussion on aspects of QoS to the recent book by Ferguson and Huston (1998).

A user may demand for several QoS requirements (such as delay, cost, bandwidth, cell loss, etc.) which are further called QoS measures. In principle, all possible combinations of these QoS measures should be provided by the network. QoS measures can be divided into two classes. The QoS measure of a path can either be additive (the sum of the QoS measures along the path) or it can be the minimum(maximum) of the QoS measures along the path. Topology filtering applies to (min)max QoS measures: all nodes and links not satisfying the requested min(max) QoS measure, are omitted from the graph before starting to calculate a path. This explains why a multiple QoS constraints routing problem only features additive QoS measures (as delay, hopcount, the cell loss ratio (after a logarithmic transform), link cost,...).

The general QoS routing problem consists of choosing and determining which QoS measures are relevant, of a QoS routing protocol that dynamically supplies each node in the network with a consistent view of the topology and finally of the routing algorithm. Several papers (Orda, 1998) treat a combination of the first and second topic. In this paper, we merely confine ourselves to the routing algorithm and assume that a completely defined topology with the relevant QoS measures per link are given. QoS routing with partial information (Lorentz and Orda, 1998) due to hierarchical condensation are discussed in Van Mieghem (1998a,b).

Routing with more than one additive constraints is a NP-complete problem (Garey and Johnson, 1979) and hence it becomes intractable for large networks, demanding good approximate solutions or heuristics. In the past a number of approximate algorithms for the multiple constraints routing problem have been proposed, most of which rely on a linear representation of the path length (Jaffe, 1984; Henig, 1985; Iwata, 1996; Lee, 1993; Wang, 1996). We propose a new algorithm which is based on a combination of a non-linear representation of the path length (Vogel, 1996), a *k*-shortest path algorithm (Chong, 1995)

and the concept of non-dominated paths (Henig, 1985). The accuracy of this algorithm is expressed as the probability that a path satisfying all constraints is found when indeed such a path exists. By tuning the single, integer parameter k of the k-shortest path algorithm, the accuracy of the algorithm can be adapted. Therefore, we named the algorithm TAMCRA: Tunable Accuracy Multiple Constraints Routing Algorithm. As opposed to several QoS routing proposals that present approximate algorithms (e.g. based on ε-optimality (Guerin and Orda, 1997;Hassin, 1992), TAMCRA yields *exact* results for the integer constraints problem.

The remainder of the paper is organized as follows. In section 2 to section 4 the principles behind TAMCRA are explained. In section 5 we discuss the criterion to evaluate TAMCRA and the type of random graphs used.. The results of this performance evaluation are found in section 6.

2. NON-LINEAR REPRESENTATION OF THE PATH LENGTH

Although the definition of the multiple constraints routing problem can be found in various publications (Jaffe, 1984; Wang, 1996) we rephrase it to avoid any misconception.

Consider a graph $G(V,E)$ with V nodes and E links where each link connecting two nodes u and v is specified by a link vector $l(u,v)$ having m additive link metrics $l_i(u,v) > 0$ with $i=1,...,m$. On the additivity of the link metrics, a path $P(s,k,...,l,d)$ connecting a source node s and a destination node d is specified by a link vector $l(P)$ where for each component the following holds $l_i(P) = l_i(s,k) + l_i(k,r) +...+ l_i(u,l) + l_i(l,d)$. In order words, the link vector $l(P)$ of the path P consists of the vector sum of the link vectors corresponding to links characterizing the path P. Given m positive constraints L_i, the multiple constraints routing

problem asks for a path $P(s,k,...,l,d)$ obeying the constraints $l_i(P) \leq L_i$ for each component $i=1,...,m$.

A first key problem is the definition of the concept of path length. In order to illustrate our choice, the definition (7) below, and its impact, we start examining the straightforward choice of a linear path length as presented by Jaffe (1984). Concentrating on m=2 constraints (L_1, L_2), Jaffe proposes to use the definition

$$l(u,v) = d_1.l_1(u,v) + d_2.l_2(u,v) \qquad (1)$$

where d_1 and d_2 are positive real numbers. By replacing each link vector via (1) by a single metric in the graph G, Jaffe reduces the m = 2 parameter problem to a single parameter one allowing the use of Dijkstra's well-known shortest path algorithm. Running the Dijkstra algorithm on the reduced graph will return a path $P(s,k,...,l,d)$ for which

$$\begin{aligned} l(P) &= [d_1.l_1(s,k) + d_2.l_2(s,k)] + ... + [d_1.l_1(l,d) + d_2.l_2(l,d)] \\ &= d_1.[l_1(s,k) + ... + l_1(l,d)] + d_2.[l_2(s,k) + ... + l_2(l,d)] \\ &= d_1.l_1(P) + d_2.l_2(P) \end{aligned} \qquad (2)$$

is minimal. Each path (2) between the source node s and the destination node d can be represented in a ($l_1(P)$,$l_2(P)$) -plane.

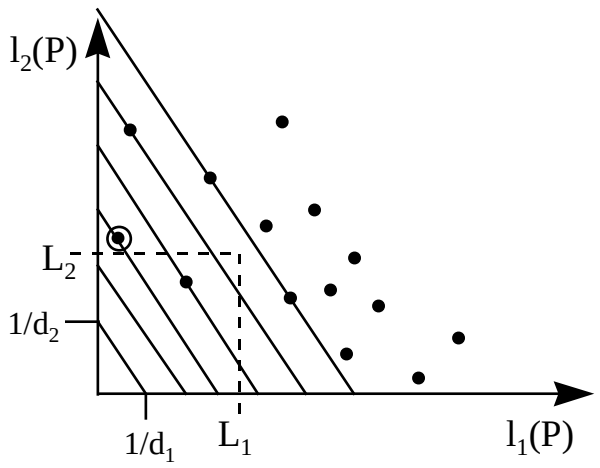

Figure 1: distribution of paths in the (l1(P),l2(P)) plane. The solution retrieved by the Dijkstra algorithm is encircled.

Each of the parallel lines

$$d_1.l_1(P) + d_2.l_2(P) = c \qquad (3)$$

intersects solutions with equal length *c*. All solutions lying above a certain line have a length (according to (1)) larger than the ones below or on the line. The shortest path returned by Dijkstra after condensation of the two link metrics (l_1,l_2) to a single metric $l=d_1.l_1 + d_2.l_2$ will be the first solution intersected by a set of parallel lines with slope (d_1/d_2) as shown in Figure 1.

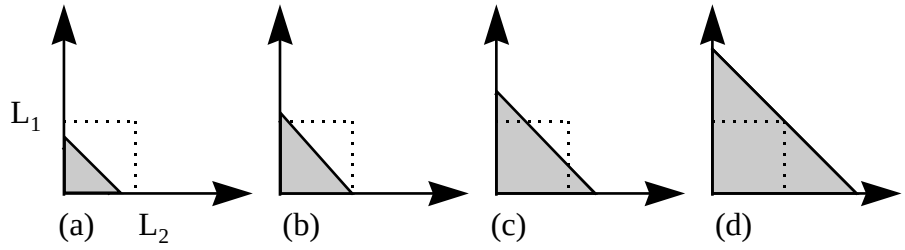

Figure 2: Scanning the constraint area with a straight equilength line

When scanning the solution space with a straight equilength line as in Figure 2, the area which is scanned outside the constraint region is minimized if the slope of the straight equilength lines satisfies

$$d_1 / d_2 = L_2 / L_1 \qquad (4)$$

With (4), always half of the constraint area is scanned before running the risk to select a solution outside the constraint area. This is shown in Figure 3-a. The area which is scanned outside the constraint area can be further reduced if the straight equilength lines are replaced by curved equilength lines that more closely approach the boundary of the constraints area as shown in Figure 3-b.

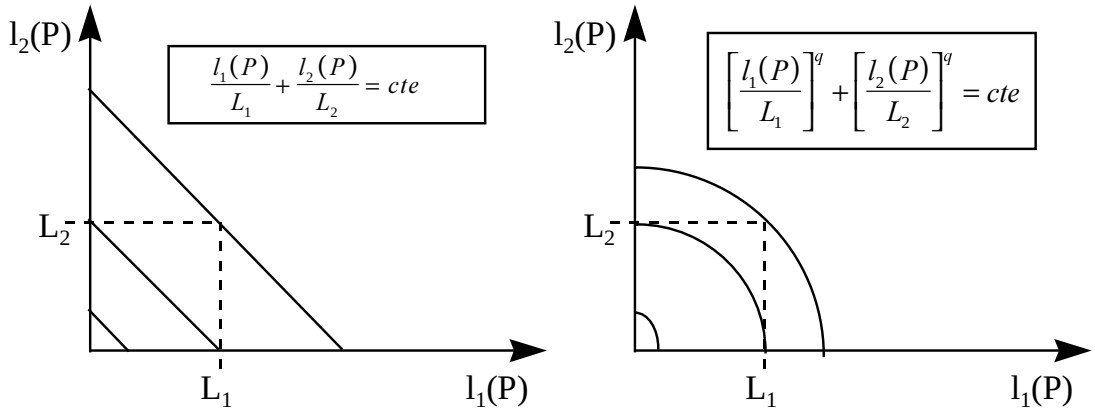

Figure 3: (a) straight equilength lines ; (b) curved equilength lines

Sorting paths according to such curved equilength lines necessitates another definition of the path length as

$$l(P) = \left(\sum_{i=1,...,m} \left(\frac{l_i(P)}{L_i} \right)^q \right)^{1/q} \qquad (5)$$

with $q > 1$ and

$$l_i(P) = \sum_{e:s \to d} l_i(e) \qquad (6)$$

The definition (5) is known as Holder's q-vector norm (Golub and Van Loan, 1983) and fundamental in the theory of classical Banach spaces (see Royden, 1988, chapt. 6). Ideally the equilength lines should perfectly match the boundaries of the rectangular constraints, scanning the constraint area without ever selecting a solution outside the constraint area, which is achieved when $q \to \infty$. In that case the value of $l(P)$ in (5) will be completely dominated by the largest term in the sum (5) and expression (5) reduces to

$$l(P) = \max\left(\frac{l_1(P)}{L_1}, \frac{l_2(P)}{L_2}, ..., \frac{l_m(P)}{L_m} \right) \qquad (7)$$

When there are only two metrics, (7) can be visualised as in Figure 4.

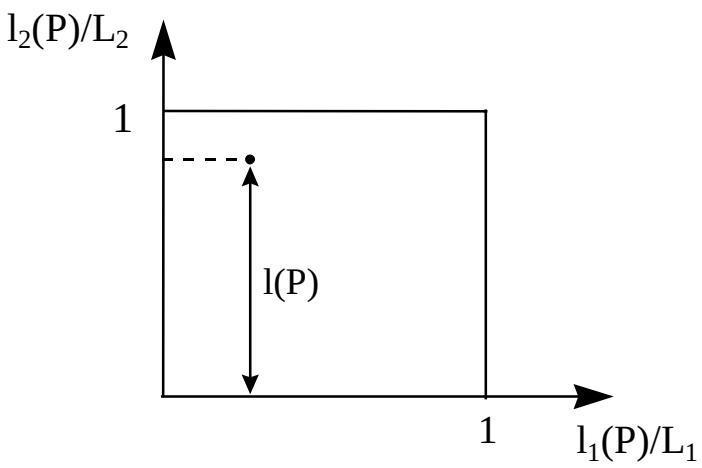

Figure 4: representation of a path (dot) and its pathlength *l*(*P*) according to (7) in case the path is subject to two constraints L_1 and L_2.

This figure shows that the pathlength is determined by that metric (component) closest to the constraints boundary. A similar approach can be found in Vogel (1996) where the distance between the position of the path in the plane and the boundary for a specific dimension is called the *availability* of the path in that dimension. In this terminology, the pathlength is determined by the dimension with the minimum availability.

With definition (7), the path length outside the constraint area is always larger than the path length inside the constraint area. Thus, if the shortest path lies outside the constraint area then there is no path satisfying all constraints. Hence, the problem of finding a path obeying *m* constraints is "reduced" to the problem of finding the shortest path according to definition of the path length in (7).

3. THE TAMCRA ALGORITHM

As subsections of shortest paths in multiple dimensions are not necessarily shortest paths (as demonstrated in Appendix A), a simple shortest path algorithm such as Dijkstra's algorithm will not always work satisfactorily. When the Dijkstra algorithm calculates the shortest path between a source and a destination node, it stores only the shortest path to

each intermediate node. When dealing with multiple constraints and the non-linear path length corollary, the shortest path to an intermediate node does not always lead to the shortest path to the final destination node. Therefore more than one path needs be stored for each intermediate node. The need of storing several paths leads us to consider a k-shortest path algorithm. The literature abounds with shortest path algorithms which calculate the *k* shortest path but one of those algorithms, by Chong et al. (1995), turns out to be very useful when combined with the definition of the pathlength as in (7). The k-shortest path algorithm from Chong et al. operates like the Dijkstra algorithm except that the queue which stores intermediate path lengths can contain more than one - in general *k* - entries per node. Evidently, the larger the value of *k*, the larger the probability that the subpath between the source and the intermediate node which eventually leads to the shortest path between source and destination, belongs to the set of the *k* shortest paths contained in the queue of the algorithm. The integer parameter *k* then naturally arises as 'accuracy tuning index', which has suggested to call our algorithm TAMCRA, a *Tunable* Accuracy Multiple Constraints Routing Algorithm.

Yet, TAMCRA is more than a combination of the proper *k*-shortest path algorithm with a specially chosen non-linear definition of the pathlength as in (7). The third key idea is that efficiency can be dramatically increased using *dominated paths*, a concept first introduced by Henig (1985). Consider two paths P_1 and P_2 between the source and some intermediate node for an m=2 constraints problem with pathlength vector (x_1, y_1) and (x_2, y_2) respectively. Both paths can be represented in a plane and there are two possible scenarios, shown in Figure 5 a-b.

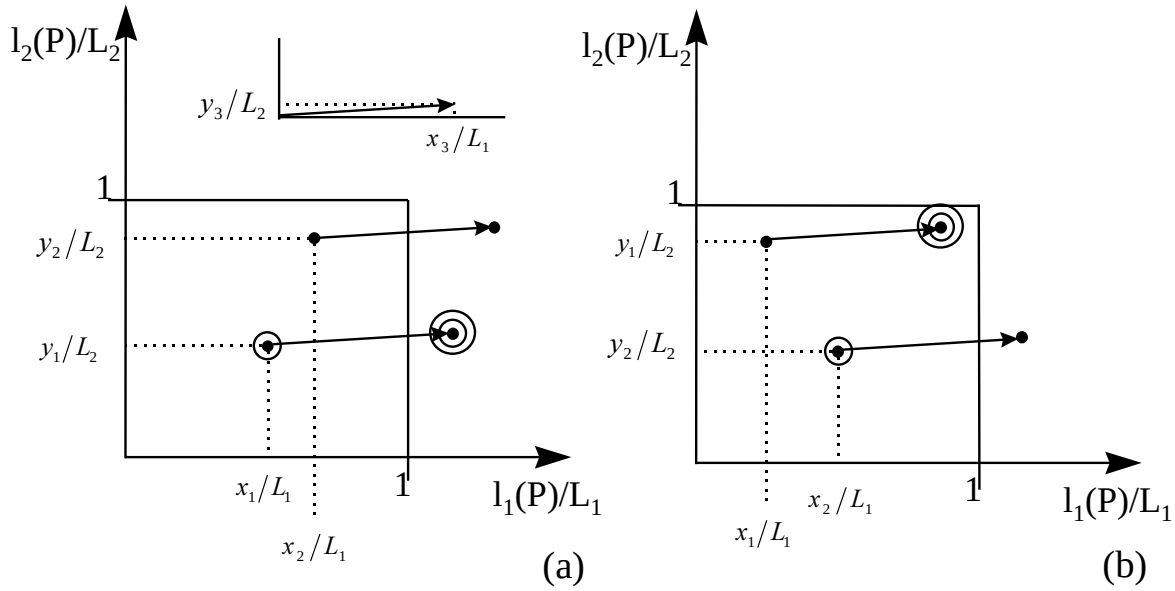

Figure 5: Representation of the two paths between the source node and some intermediate node. The length vector (x_3,y_3) represents the remaining part of the shortest path between source and destination. The shortest path is encircled twice.

In scenario (*a*), P_1 is shorter than P_2. P_1 is even shorter both for metric 1 (x) and for metric 2 (y). In that case, *any path between the source and final destination node which makes use of P_1 will be shorter than any other path between the source and destination which makes use of P_2*. This can be demonstrated as follows.

If

$$x_1/L_1 < x_2/L_1 \quad \text{and} \quad y_1/L_2 < y_2/L_2 \tag{8}$$

then also

$$\left(x_1 + x_3\right)/L_1 < \left(x_2 + x_3\right)/L_1 \quad \text{and} \quad \left(y_1 + y_3\right)/L_2 < \left(y_2 + y_3\right)/L_2 \tag{9}$$

for any value of x_3, y_3 and thus

$$\max\left(\frac{x_1 + x_3}{L_1}, \frac{y_1 + y_3}{L_2}\right) < \max\left(\frac{x_2 + x_3}{L_1}, \frac{y_2 + y_3}{L_2}\right) \tag{10}$$

Thus, we certainly know that P_2 will never be a subpath of a shortest path and therefore P_2 should not be stored in the queue. Using the terminology of Henig (1985), P_2 is said to be dominated by P_1.

In scenario (*b*) both paths have crossing abscissa and ordinate points: P_2 has a smaller value for metric 2 while P_1 has a smaller value for metric 1. Only in this situation, the shortest path between the source and some intermediate node is not necessarily part of the shortest path between source and destination. This is shown in Figure 5-*b* by adding the path length vector (x_3,y_3) which completes the path until the destination.

In short, if the abscissa-ordinate values of two partial paths cross, both values should be stored in the queue, while if these values do not cross, only the shortest should be stored in the queue. The same applies if there are *m* constraints. If path P_1 is already in the queue and a new path P_2 to the same intermediate node is found for which

$$\frac{l_i(P_1)}{L_i} < \frac{l_i(P_2)}{L_i}, \quad \text{for each } i=1,...,m \tag{11}$$

then path P_2 should not be stored in the queue. Of course, this implies that when a new pathlength is stored in the queue, all the individual pathlengths for this path also have to be stored somewhere, to verify whether future path lengths to the same node are not dominated by the paths whose path length is already stored in the queue.

Another consequence of the path dominance is that the worst-case amount of simultaneously stored paths is determined by the granularity of the constraints. In reality most protocols will only allocate a fixed number of bits for carrying the metric. In that case the constraints L_i can be expressed as an integer number of a basic metric unit and the worst-case number of partial paths which have to be maintained in parallel for each node are shown in Figure 6.

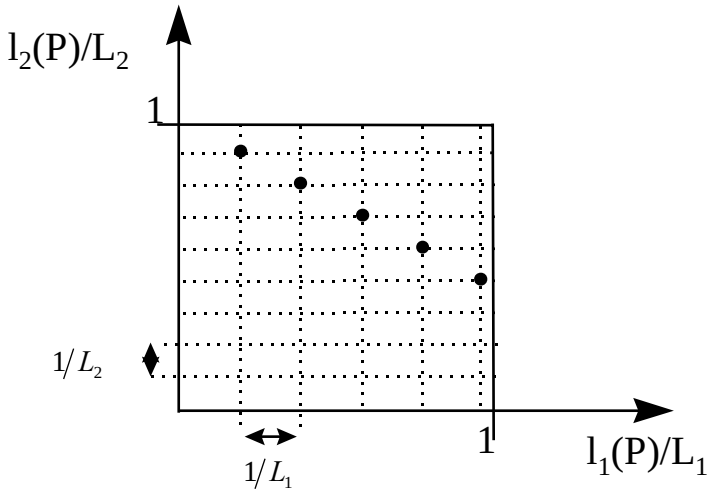

Figure 6: A worst-case situation in which L_1 partial paths are maintained in parallel

If $m=2$ and $L_1 < L_2$ as in Figure 6, there can never be more than L_1 partial paths with crossing abscissa-ordinate values as in Figure 5-a. Any additional partial path will relate to any of these L_1 partial paths as in Figure 7a-b.

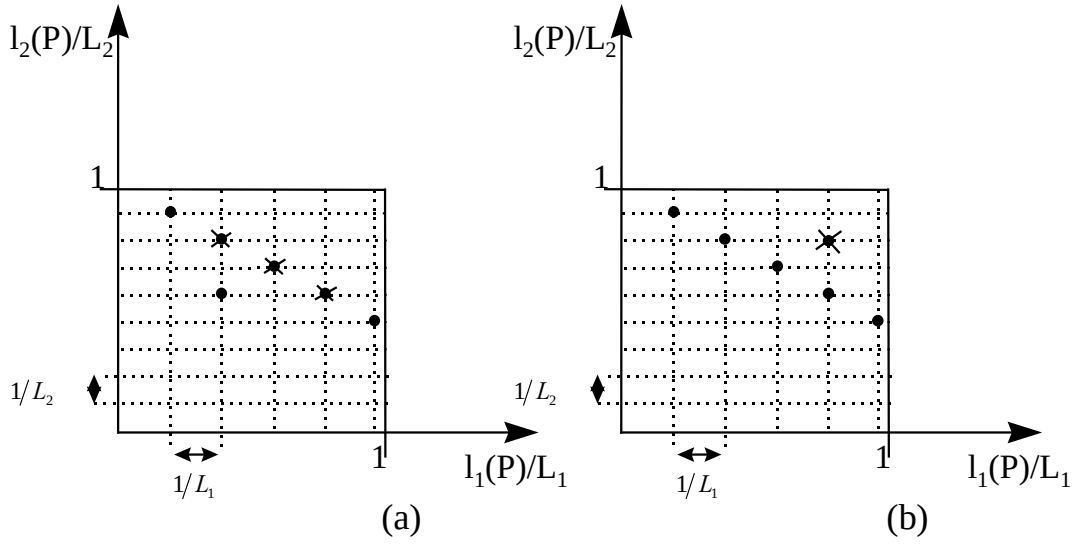

Figure 7: (a) only three partial paths should be maintained in parallel; (b) any longer path should not be maintained.

In both cases the number of paths which are stored in the queue is not increased.

Thus, for m=2, the upper limit for k in the TAMCRA algorithm is

$$k_{\max} = \min(L_1, L_2) \qquad (12)$$

If there are more than two constraints the maximum number of paths to be calculated is given by

$$k_{\max} = \prod_{i=1,\ldots,m} L_i \qquad , L_i \neq \max_j \left(L_j \right) \tag{13}$$

With $k = k_{max}$, the multiple constraints routing problem can be solved *exactly.* Nevertheless, in most circumstances k_{max} will be too large to be of any practical use. In this paper we want to demonstrate that small values of k are sufficient to solve the NP-complete problem in an approximate way.

4. THE COMPLEXITY OF THE TAMCRA ALGORITHM

The meta-code of the TAMCRA algorithm is listed below:

TAMCRA(*G*,s,d,L,k)

G: graph, s: source node, d: destination node, L: constraints, k: tunable k-parameter

```
1   counter=0, for all nodes
2   length(s[1]) = 0                    syntaxis: s[1] is the first path to node s
3   ADD s[1] to queue
4   while (queue ≠ empty)
5         u[i] = EXTRACT_MIN from queue
6         if( u = destination node) → STOP
7         else
8               for each v ∈ adjacent_list(u)
9               if(v ≠ previous node of u[i])
10                    PATH = u[i] + (u,v)
11                    LENGTH = length of PATH
12                    check if PATH is non-dominated
13                    if(LENGTH ≤ 1 and non-dominated)
14                          if (counter(v) < k)
15                                counter(v)=counter(v)+1
```

```
16                                  j=counter(v)
17                                  v[j] = PATH
18                                  length(v[j])=LENGTH
19                                  ADD v[j] to queue
20                          else
21                                  old v[j] = path in queue with maximum length to v
22                                  if (LENGTH < length(old v[j]))
23                                          new v[j] = PATH
24                                          REPLACE in queue old v[j] with new v[j]
```

The EXTRACT_MIN function (see Cormen *et al.,* 1991) in line 5 selects the minimum path length in the queue and returns the associated node. Line 8 and 9 describe the scanning procedure, in line 11 the non-linear path length is calculated according to the definition in (7). Line 12 checks whether the new path for the node is not dominated by the existing paths for that node. The existing paths for that node are those whose path length has been stored in the queue before. Line 13 removes any path lengths beyond 1. Line 15 to 19 describes how path lengths for a node can be added to the queue as long as the maximum number specified by k is not attained while the procedure from line 21 to line 24 is invoked when there are already k crossing paths to a certain node in the queue. In that case, only the k-shortest paths are maintained.

Complexity

If V is the number of nodes, the queue can never contain more than kV path lengths. When using a Fibonacci heap to structure the queue, selecting the minimum path length among kV different path lengths takes at most a calculation time of the order of $\log(kV)$ (Cormen *et al*., 1991). As each node can only be selected k times from the queue, the EXTRACT_MIN

function in line 5 takes O(kV.log(kV)) at most. The for-loop starting on line 8 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 the crossing condition takes O(km) at most. Adding a path length in the queue takes O(1) while replacing a path length takes O(k) at most because the largest of the k entries for that node must be found. Adding all those contributions yields a worst-case time-complexity of

$$O\left(kV\log(kV)+k^3mE\right) \qquad (14)$$

With a single constraint (m=1) and k=1, this reduces to the time-complexity of the Dijkstra algorithm given by

$$O\left(V\log V+E\right) \qquad (15)$$

This means that, for a fixed number of constraints m and a fixed value of k, the TAMCRA algorithm scales like the Dijkstra algorithm for a varying number of nodes V and links E.

The complexity (14) is non-polynomial (NP-complete) despite the fact that expression (14) is a polynomial function and the worst-case value of k (13) which is needed to solve the problem exactly is also a polynomial function. The reason why expression (14) is nevertheless called non-polynomial is because the complexity of an algorithm is only polynomial if it is of polynomial order in "the size of the graph", i.e., the number of nodes V, the number of links E and the number of constraints m.(Garey, 1979) The *value* of the *constraints* is not regarded as part of "the size of the graph" and therefore the complexity (14) with k given by (13) is non-polynomial or NP-complete. In such a case, the term pseudo-polynomiality applies.

5. EVALUATING THE TAMCRA ALGORITHM

The performance of the TAMCRA algorithm is tested on a number of random graphs (Bollobas, 1985). An important class of random graphs are the Waxman graphs for which the probability that two nodes are connected by a link is expressed by [Calvert 1997]

$$P = \alpha e^{-d/(\beta M)} \qquad (16)$$

d is the Euclidean distance between the two nodes and M is the length along one side of the square region which bounds the graph. The probability P in (16) that two nodes are connected decreases exponentially with the distance d between the nodes and can be tuned with the parameters α and β. The first parameter determines the local interconnection density while β determines the scale at which links occur. The larger β, the larger the distances which are bridged by some links.

Waxman graphs reflect the property of most networks that neighbouring nodes are more likely interconnected than remote nodes. An example of such a graph is shown in Figure 8 for 100 nodes**.** The parameters are chosen such that there are approximately 200 links which typically only span a fraction of the largest possible distance in the graph.

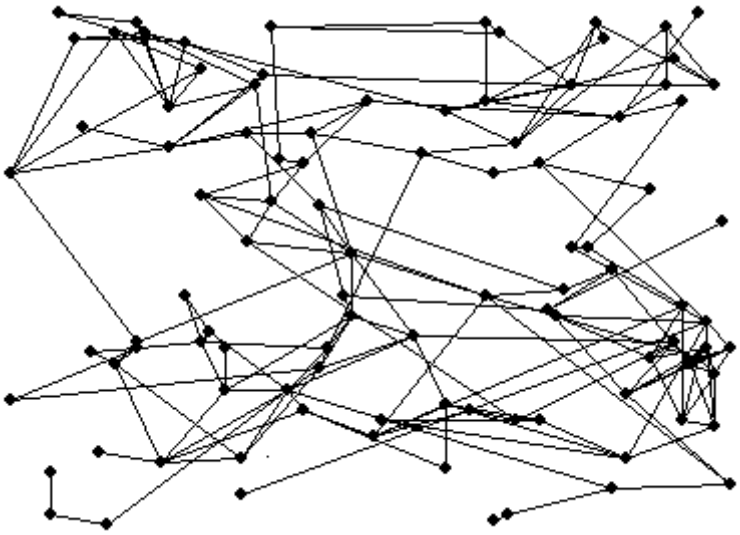

Figure 8: a Waxman graph with 100 nodes and approximately 200 links. The link density is determined by (16) with α=1, β=0.06 and L=100

TAMCRA is evaluated on this type of random graphs. Throughout the document they are simply referred to as Waxman graphs. Each link is bi-directional with the same characteristics in both directions. For each metric, the link weights are uniformly distributed random numbers between 0 and 1. Simulation results have indicated that this is a worst-case situation in the sense that any other scheme to attribute weights to the links yielded better results in terms of the probability to find the shortest path.

The performance of the TAMCRA algorithm is measured by the number of times it erroneously decides that the path with the requested constraints does not exist while, in fact there is such a path. The erroneous decision rate (EDR) is the percentage of wrong decisions on the total number of routing requests and can also be interpreted as the probability of finding a path outside the constraints while there are paths within the constraints. The EDR strongly depends on the number of paths which lie within the constraints. If there is more than one path within the constraints, TAMCRA can afford to miss the shortest path provided it returns another path which still lies within the constraints. Thus, if the constraints are large, many paths will satisfy the constraints and the EDR will be very small.

In this paper we will focus on the worst case EDR: every time that TAMCRA misses the shortest path this is considered to be an error, even if the non-shortest path which is found would satisfy all constraints. This is the EDR which occurs when the constraints are so tight that only one path satisfies all constraints: in that case only the shortest path satisfies all constraints while all non-shortest paths do not satisfy the constraints. All results should be interpreted as such. Throughout the rest of the document the term “EDR” should thus always be interpreted as the worst-case EDR unless it is specified otherwise.

6. RESULTS

6.1 TWO EQUAL CONSTRAINTS

As explained before the TAMCRA algorithm is tuned with a single parameter *k*. If *k* is too small, the path which is returned by the TAMCRA algorithm might not be the shortest path.

For each source-destination pair in a graph, the minimum value of *k* which is needed to find the shortest path can be calculated. This is shown in Figure 9 for 200 different locally connected random graphs of 100 nodes each yielding 200 x 100 x 99 different source-destination pairs. This large number of source-destination pairs provides a comfortably low statistical error as shown in appendix B.

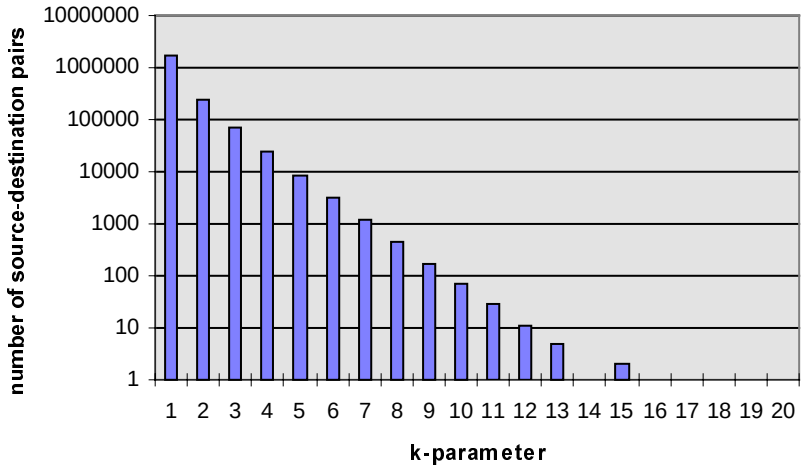

Figure 9: number of source-destination pairs requiring a minimum value of *k* in order to find the shortest path with the TAMCRA algorithm for Waxman graphs with 100 nodes, approximately 200 links and 2 link values per link.

From these data the worst-case EDR can be derived. The latter represents the probability that the shortest path is not found with a certain value of *k*. This can be found by summing

the number of source-destination pairs beyond a certain value of *k* in Figure 9 and dividing this sum by the total number of source-destination pairs (200 x 100 x 99) This is shown in Figure 10.

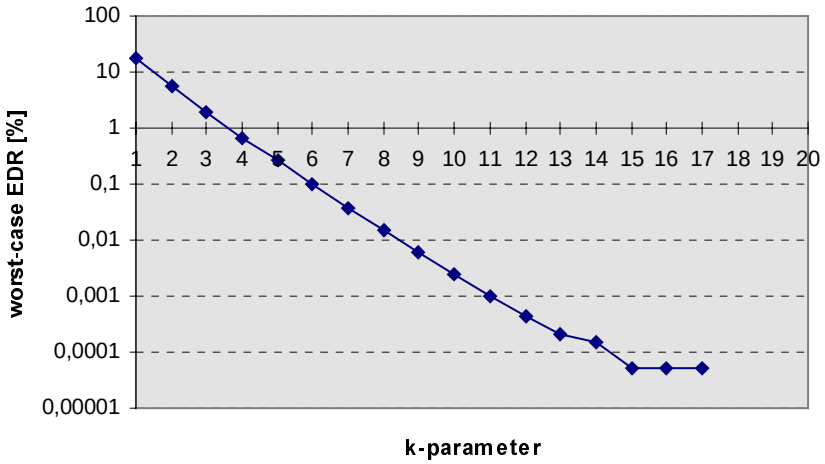

Figure 10: the worst-case EDR as a function of the value of *k* which is used in the TAMCRA algorithm. The Waxman graphs have 100 nodes connected by approximately 200 links and each link is characterised by 2 link values.

If TAMCRA is applied with *k*=1, the worst-case EDR for this graph is about 17% which means that for 83% of all source-destination pairs the two-constraint problem is solved with a Dijkstra-like complexity. The EDR rapidly decreases as *k* is increased: with *k*=4 the EDR has dropped below 1%, which means that the TAMCRA algorithm with *k*=4 perfectly solves the two-constraints problem in more than 99% of all cases for a graph with 100 nodes.

We have also monitored the time needed for one node to calculate the shortest path to all other nodes in the graph as a function of *k.* The calculation is done on a Sun SPARC Ultra Workstation and Figure 11 displays the result in absolute time units. As a reference we have added the running time of the Dijkstra algorithm (*m*=1, *k*=1).

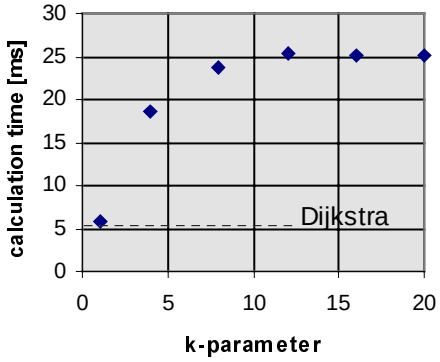

Figure 11: time needed by the TAMCRA algorithm to calculate the shortest path with 2 constraints from one source to all destinations in a Waxman graph with 100 nodes and approximately 200 links.

To solve a two-constraint problem with an accuracy of 99% in a graph with 100 nodes, 200 links and with two constraints, TAMCRA needs k=4. According to Figure 11 the running time of TAMCRA with k=4 is only 3.7 times the running time of the Dijkstra algorithm.

For the Waxman graphs with 100 nodes and two constraints, on the average only three to four paths per node have crossing abscissa-ordinate values. Hence, TAMCRA does not store more than these three to four paths even if it is allowed to store more. This explains the saturation of the observed calculation time as the value of k increases.

If the time to perform the routing calculation lies above the saturation level, k can be as large as needed to solve the multiple constraints problem exactly. In Figure 11 it can be seen that the calculation time for a graph with 100 nodes and two metrics saturates at a level which is about 5 times the time-complexity of the Dijkstra algorithm. Thus, using TAMCRA, multiple constraints problems in a graph with 100 nodes and 2 metrics can be solved exactly with a time-complexity which is only 5 times the time-complexity of the Dijkstra algorithm.

6.2 MORE THAN TWO EQUAL CONSTRAINTS

When the number of constraints increases, also the number of crossing or non-dominated paths is likely to increase. Thus, each node has to store much more paths in parallel and this is reflected in the calculation time shown in Figure 12.

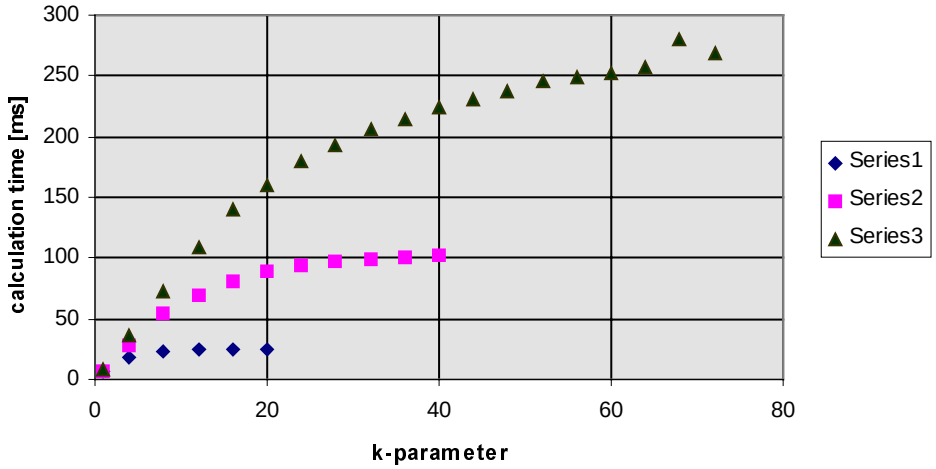

Figure 12: time needed by the TAMCRA algorithm to calculate the shortest path in a Waxman graph with 100 nodes and approximately 200 links for 2, 4 and 6 constraints.

The calculation time for m=4 and m=6 saturates at larger values of k than the calculation time for m=2 because each node has much more crossing or non-dominated paths. This implies that solving exactly an m=4 problem on a graph with 100 nodes would take 20 times the calculation time of the Dijkstra algorithm instead of only 5 times with m=2. Solving exactly an m=6 problem already requires 40 times the calculation time of the Dijkstra algorithm.

This confirms the concern of Henig (1985) who argued that calculating all the non-dominated paths is not possible when the number of constraints grows large. Fortunately, unless the multiple constraints problem needs to be solved exactly, TAMCRA does not require that all non-dominated paths are calculated. Moreover, beyond a certain number of

constraints m the probability of missing the shortest path seems even independent of the number of constraints as illustrated in Figure 13.

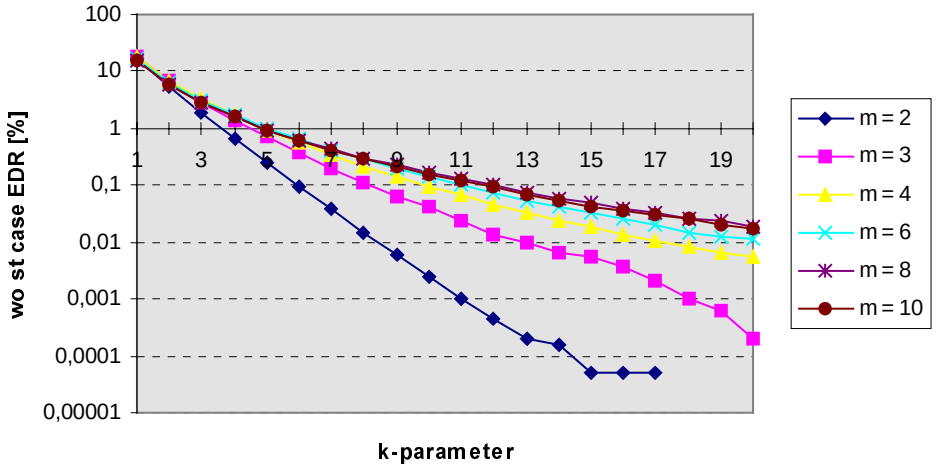

Figure 13: worst-case EDR in a Waxman graph with 100 nodes for 2, 3, 4, 6, 8 and 10 equal constraints.

As the number of constraints increases, the worst-case EDR also increases but it saturates at a certain level, despite the fact that the number of non-dominated paths increases. Figure 13 shows that solving a multiple constraints problem with an accuracy of 1% in a graph with 100 nodes requires k=5 only, no matter how many constraints there are. Of course, as expressed by (14) the calculation time of the algorithm does depend on the number of constraints as shown for a graph of 100 nodes and k=5 in

Figure 14.

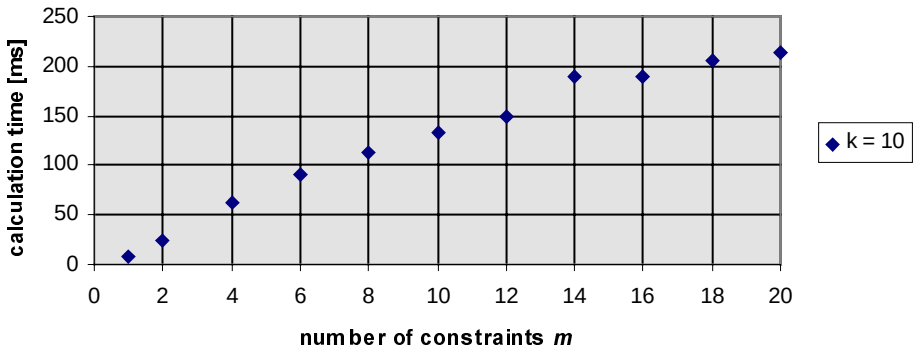

Figure 14: calculation time of TAMCRA as a function of the number of constraints *m* for a fixed value of the parameter *k*.

The calculation time is seen to increase *linearly* with the number of constraints. Hence, solving the *m*-constraints problem with a non-zero but fixed probability of missing the shortest path only requires a linear increase in the calculation time, despite the much larger increase of the number of non-dominated paths.

The number of constraints *m* will depend on the specific protocol in use. Currently PNNI is the only routing and signalling ATM specification which explicitly lists a number of QoS measures, three of which are additive QoS measures. If we include the additional constraint that the same physical path needs to be used in both directions, the total number of additive constraints doubles to six, for the time being.

6.3 UNEQUAL CONSTRAINTS

So far we have only concentrated on a situation where all constraints are equal. This is equivalent to a situation where all constraints are equally stringent. To make sure that this special set of constraints is representative for all sets of constraints, Figure 15 shows the EDR for the random set of 8 unequal constraints (20, 40, 100, 20, 300, 50, 30, 100) compared to the EDR of 8 equal constraints.

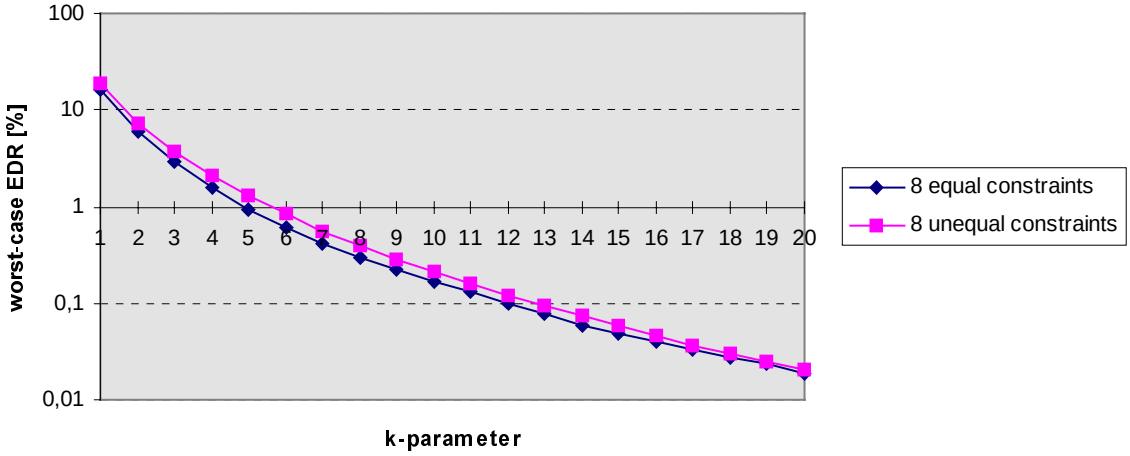

Figure 15: worst-case EDR for 8 equal and unequal constraints

In both cases the worst-case EDR is nearly identical in contrast to a two-constraints problem.. Figure 16 shows the worst-case EDR when one of the two constraints is 10 time larger than the other one. The EDR rapidly decreases which can be readily understood: if one of the constraints is relaxed (increases), its influence on the path which is returned by TAMCRA, decreases. In the limit, when one of the two constraints becomes very large, the problem is reduced to a single constraint problem for which the EDR = 0 for every *k*.

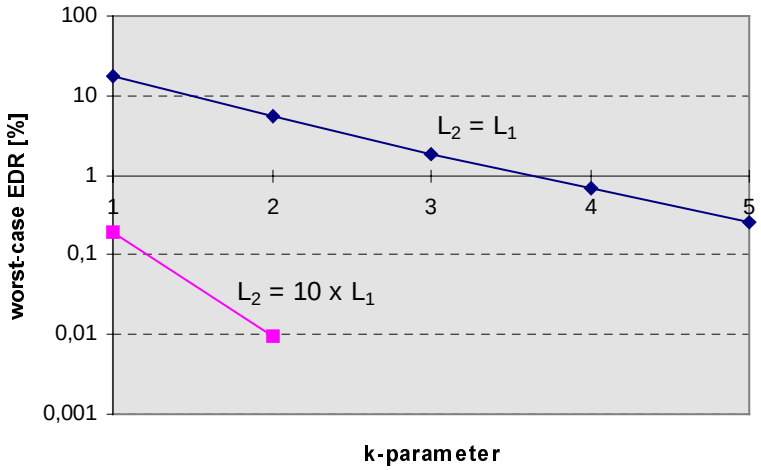

Figure 16: worst-case EDR for 2 equal and unequal constraints

6.4 SCALING BEHAVIOUR

The value of *k* which is necessary to achieve a specified value for the worst-case EDR also depends on the size of the graph. Table 1 shows the parameters of the Waxman graphs used for this scaling study. The parameters refer to the probability (16) that two nodes are connected. The graphs are scaled such that the local density of nodes and links does not change. Therefore the size of the graph scales as

$$M = \sqrt{2}.V \qquad (17)$$

In this way, the average distance between the nodes does not change. As the size of the graph increases the value of β is diminished such that the product $\beta.L$ remains constant. This guarantees that the average distance which is spanned by the links also remains constant. The third parameter α is then adjusted to keep the average number of links per node (E/V) constant.

V	M	α	β	E/V
50	71	1	0.11	2.4
100	100	1	0.08	2.8
200	141	0.9	0.056	2.8
400	200	0.78	0.04	2.8
800	283	0.78	0.028	2.8
1600	400	0.75	0.02	2.8

Table 1: parameters of the Waxman graphs in expression (16) used for the scaling study

The scaling of graphs is illustrated in Figure 17: the graph with 200 nodes is like four graphs with 50 nodes each, a local observer should not see the difference.

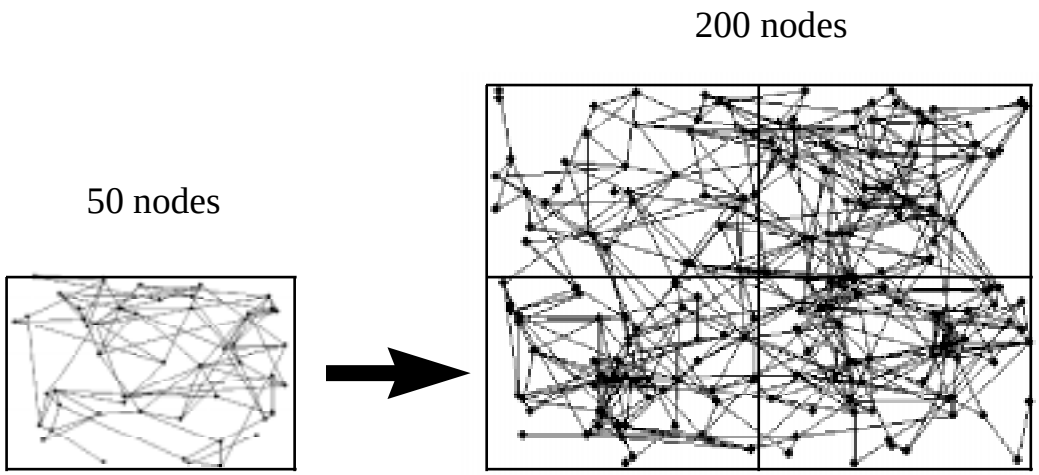

Figure 17: a Waxman graph with 50 nodes and a Waxman graph with 200 nodes according to the parameters given in table 1.

The worst-case EDR for the six different graphs of Table 1 are shown in Figure 18. The value of k which is needed to obtain a certain worst-case EDR steadily increases as the number of nodes V in the graphs increases.

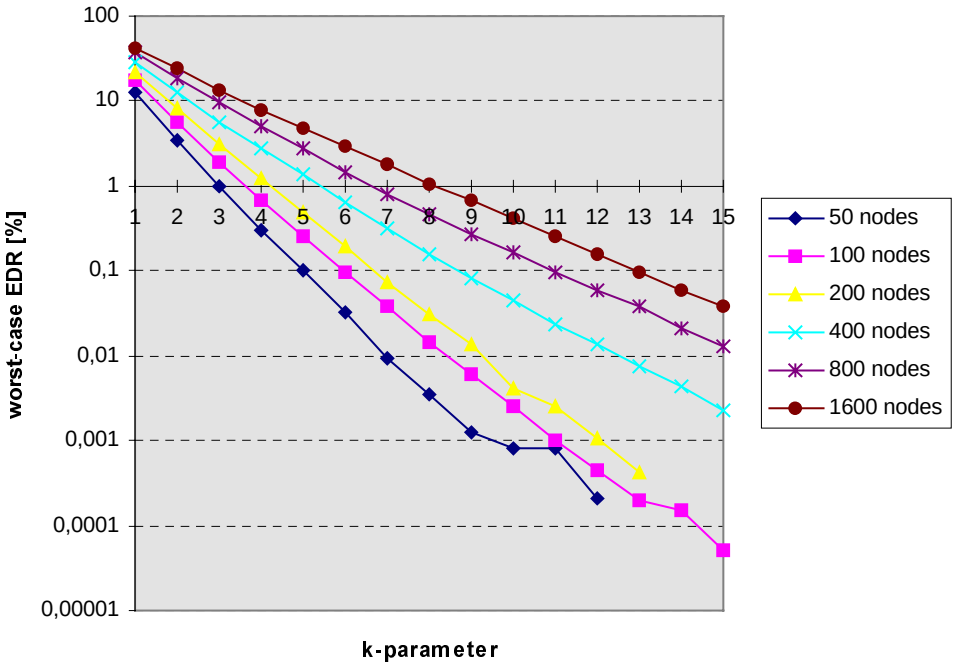

Figure 18: the worst-case EDR for 2 equal constraints as a function of the size of the Waxman graphs.

For each EDR, every doubling of the size induces an equidistant shift of the curve. This is confirmed when we plot the value of k which is needed to obtain worst-case EDR of 1% as a function of the number of nodes.

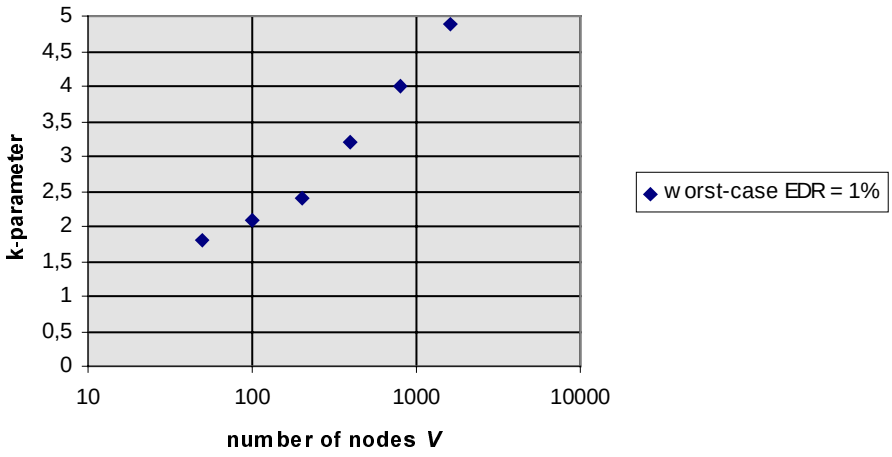

Figure 19: value of k which is needed to achieve a worst-case EDR of 1% as a function of the number of nodes V in the graph

For sizes varying between 200 and 1600 nodes, the value of k which is needed to maintain the worst-case EDR at a fixed level is proportional to $\log(V)$.

$$k(V) \sim \log(V) \qquad ,200 < V < 1600 \qquad (18)$$

As for the Dijkstra algorithm, the calculation time of the TAMCRA algorithm depends on the number of nodes V and the number of links E. The calculation time of the Dijkstra algorithm for a graph with 400 nodes is 4.36 time the calculation time for a graph with 100 nodes. If we divide the calculation time of the TAMCRA algorithm for the graph with 400 nodes by this number, both curves coincide as seen in Figure 20. This demonstrates that the TAMCRA algorithm scales exactly like the Dijkstra algorithm for any change of the number of nodes V and the number of links E as we already concluded from the expression (14) for the worst-case time-complexity.

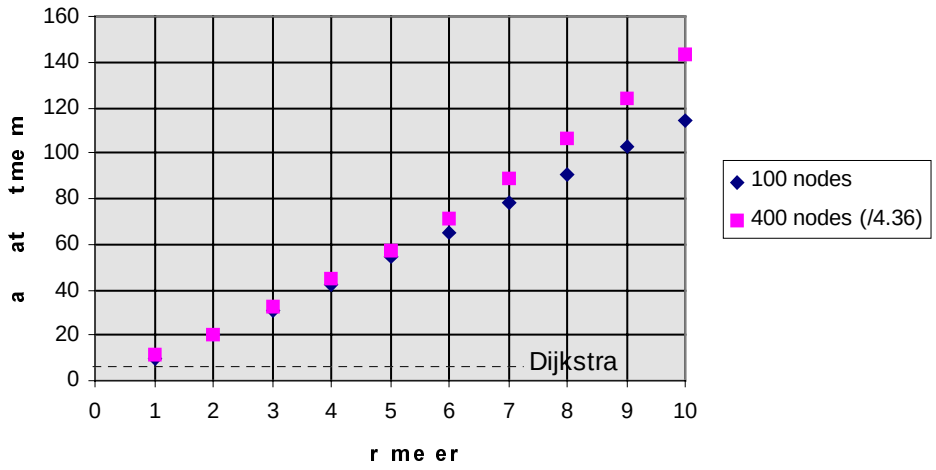

Figure 20: calculation time of TAMCRA for 8 equal constraints, applied to Waxman graphs with 100 nodes and 400 nodes.

7. CONCLUSIONS

The problem of routing with more than two additive constraints has been widely studied in literature. We reviewed a number of existing algorithms which are designed to solve these routing problems.

The main set of algorithms represent the length of a path by a linear combination of the *m* individual path lengths and calculate the shortest path by means of the Dijkstra algorithm. However, in that case the shortest path is not necessarily the path within the constraints and we argued that these algorithms might run in trouble trying to solve particular instances of the routing problem.

Based on this analysis we have proposed a different definition of the path length such that the shortest path can never lie outside the constraints when there is at least one path within the constraints. To find this shortest path we designed a new algorithm: the Tunable Accuracy Multiple Constraints Routing Algorithm or TAMCRA.

TAMCRA was evaluated on a number of random Waxman graphs. The probability of missing the shortest path, defined as the "worst-case Erroneous Decision Rate (EDR)" has been investigated.

The results learn that the main advantages of the TAMCRA algorithm are the following:

- Tunable trade-off between probability of finding the shortest path and the polynomial time-complexity with a single integer parameter *k*
- Exact solution of the multiple constraints problem with a good pseudo-polynomial time-complexity
- Exponential decrease of the probability of missing the shortest path as the value of *k* increases while the calculation time only increases linearly with *k*.
- Saturation of the calculation time above a certain value of *k* which allows an exact solution of the multiple constraints problem
- Calculation time of TAMCRA increases linearly with the number of constraints.
- Probability of missing the shortest path is independent of the number of constraints, above a certain number of constraints.
- Calculation time of TAMCRA scales as the Dijkstra algorithm for an increasing number of nodes *V* and links *E*.
- The value of *k* which is needed to keep the probability of missing the shortest path fixed increases as log(*V*).

These results demonstrate that TAMCRA is scaleable and that multiple constraints routing problems can be solved to a very high accuracy within a short time-frame as long as we accept a non-zero probability of missing the shortest path.

REFERENCES

Bollobas B., 1985, *Random Graphs*, Wiley, New York

Calvert K., M.Doar, E.Zegura, 1997, "Modelling Internet Topology", IEEE Comm. Magazine, pp.160-163

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

Cormen T.H, C.E.Leiserson, R.L.Rivest, 1991, *Algorithms*, MIT Press,

De Neve H., P.Van Mieghem, 1998a, "A multiple quality of service routing algorithm for PNNI", proceedings of IEEE ATM 98 workshop, Fairfax USA, pp.324-328

Ferguson, P. and G. Huston, 1998, *Quality of Service, Delivering QoS on the Internet and in Corporate Networks*, Wiley, New York.

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

Golub, G. H. and C. F. Van Loan, 1983, *Matrix Computations*, North Oxford Academic, first edition, Oxford.

Guerin, R. and A.Orda, 1997, "QoS-based Routing in Networks with Inaccurate Information: Theory and Algorithms", IEEE INFOCOM '97, pp.75-83.

Hassin, R., 1992, "Approximation Schemes for Restricted Shortest Path Problem", Mathematics of Operations Research, vol. 17, No. 1, pp. 36-42.

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

Iwata A, R.Izmailov, D-S.Lee, B.Sengupta, G.Ramamurthy, and H.Suzuki, 1996, "ATM Routing Algorithms with Multiple QOS Requirements for Multimedia Internetworking", IEICE Trans. Commun., vol.E79-B,no.8, pp.999-1007

Jaffe J.M, 1984, "Algorithms for Finding Paths with Multiple Constraints", Networks, vol.14, pp.95-116

Lee W, M.Hluchyi, and P.Humblet, 1995, "Routing Subject to Quality of Service Constraints in Integrated Communication Networks", IEEE Network July/August 1995, pp.46-55

Lee W, M.Hluchyj, and P.Humblet, 1993, "Rule-Based Call-by-Call Source Routing for Integrated Communication Networks", IEEE INFOCOM '93, pp.987-993

Lorenz, D. H. and A. Orda, 1998, " QoS Routing in Networks with Uncertain Parameters", IEEE INFOCOM'98.

Orda, A., 1998, "Routing with End to End QoS Guarantees in Broadband Networks", IEEE INFOCOM'98.

Royden, H. L., 1988, *Real Analysis*, Macmillan Publishing Company, 3th edition, New York.

Sanjay, M., 1998, “Bounds on End-to-End Performance via Greedy, Multi-path Routing in Integrated Services Networks”, IEEE INFOCOM’98.

Van Mieghem, P., 1998a, “Node and Link Aggregation in a Hierarchy”, submitted to Computer Networks and ISDN Systems

Van Mieghem, P., 1998b, “Routing in a hierarchical structure”, Proceedings of First IEEE International Conference on ATM (ICATM’98) pp.378-384, June 22-24, Colmar, France.

Vogel R., R.Herrtwich, W.Kalfa, H.Wittig, and L.Wolf, 1996, “QoS-Based Routing of Multimedia Streams in Computer Networks”, IEEE Journal on selected areas in communications, vol.14, no.7, pp.1235-1244

Wang Z and J.Crowcroft, 1996, “Quality-of-Service Routing for Supporting Multimedia Applications”, IEEE J. Selected Areas in Communications, vol.14, pp.1228-1234

APPENDIX A

An important corollary of non-linear path length definitions as (5) and (7) is that *subpaths of shortest paths are not necessarily shortest paths.* The demonstration relies on the triangle inequality, one of the three properties of vector norms (or path lengths) (Golub and Van Loan, 1983), stating that

$$l(p+q) \le l(p) + l(q) \qquad \text{(A.1)}$$

where p and q are two link vectors. Consider two paths P_1 and P_2 for which $l(P_1) < l(P_2)$ and assume that, by adding a same link a to both paths, we can construct the paths P_3 and P_4. Let us first focus on the case where the equality sign in (A.1) holds, typically if $q = 1$ in (5). By construction, we have $l(P_3) = l(P_1 + a)$ and by (A.1) with the equality sign, $l(P_1 + a) = l(P_1) + l(a)$ and analogously, $l(P_4) = l(P_2 + a) = l(P_2) + l(a)$. Since $l(P_1) < l(P_2)$, there holds that $l(P_3) < l(P_4)$ or, the subpaths of shortest path with linear definition of path length are again shortest paths, leading to the well-known and intuitive result. When the inequality sign holds in (A.1), typically if q > 1 as readily verified from (5), we arrive in a similar fashion at the set of inequalities $l(P_4) = l(P_2 + a) < l(P_2) + l(a)$; $l(P_3) = l(P_1 + a) < l(P_1) +$

$l(a)$ and $l(P_3) < l(P_4)$. However, from this set, it cannot be concluded whether $l(P_3) \leq l(P_4)$ or $l(P_3) > l(P_4)$. It suffices to show that the latter situation may exist in order to prove the corollary. This can be illustrated on the example graph shown below with three additive measures for each link. The constraints were chosen to be 14, 11 and 12 respectively.

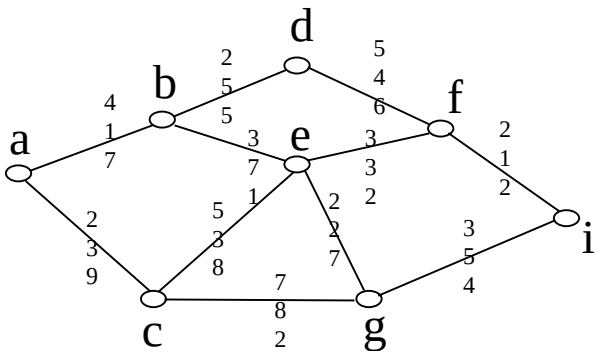

Figure A.1: example graph with three metrics per link

For these constraints, the shortest path between node *a* and node *i* runs over node *c*, *e* and *f* as shown below. According to the definition (7), the pathlength of this path equals 0.95 which means it satisfies all constraints.

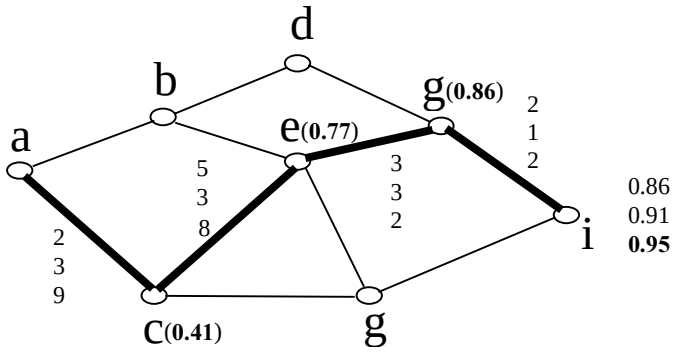

Figure A.2: shortest path between node *a* and node *i*

However, the shortest path between node *a* and node *e* does not run over node *c* but over node *b* as easily verified below. QED.

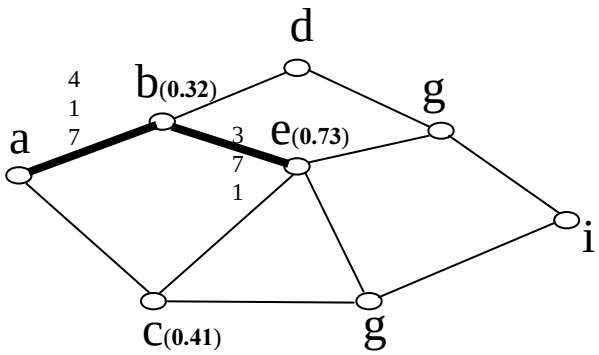

Figure A.3: shortest path between node *a* and node *e*

APPENDIX B

Figure B.1 compares the number of source-destination pairs as a function of the *k* which is needed to find the shortest path in between them, when averaged over a different number of random graphs. The calculation is done on Waxman graphs with 100 nodes and approximately 200 links.

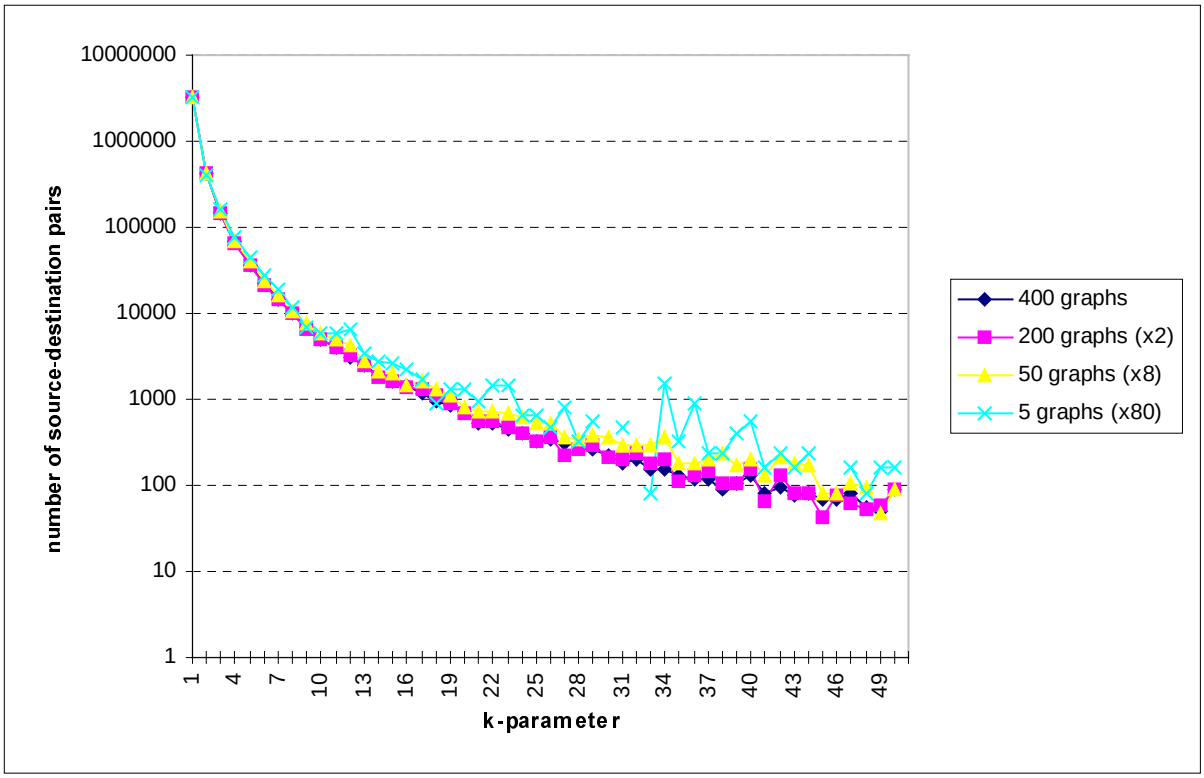

Figure B.1: number of source-destination pairs requiring a minimum value of *k* in order to find the shortest path with the TAMCRA algorithm for Waxman graphs with 100 nodes and approximately 200 links

Figure B.1 demonstrates that apart from the statistical noise, the result for 5 graphs is the same as the result for 400 graphs. Thus, the amount of correlation which is present in a single graph is very weak. This could be expected from the fact that TAMCRA is not even symmetrical: if the shortest path between source A and destination B is found which some value of *k*, the reverse path between source B and destination A might require a different value of *k* and there seems to be little or no correlation between the two.

In the case of graphs with 100 nodes, 200 graphs are required to generate a sufficient amount of source-destination pairs for all values of *k* and thus reduce the statistical noise.