# Performance Evaluation of Efficient Solutions for the QoS Unicast Routing

Alia Bellabas, Samer Lahoud, Miklós Molnár

# Performance Evaluation of Efficient Solutions for the QoS Unicast Routing

A. Bellabas, S. Lahoud
Institut de Recherche en Informatique et Systèmes Aléatoires IRISA, Rennes, FRANCE
M. Molnár
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier LIRMM, Montpellier, FRANCE
Email: {alia.bellabas, samer.lahoud}@irisa.fr, miklos.molnar@lirmm.fr

***Abstract*— Quality of Service (QoS) routing known as multi-constrained routing is of crucial importance for the emerging network applications and has been attracting many research works. This NP-hard problem aims to compute paths that satisfy the QoS requirements based on multiple constraints such as the delay, the bandwidth or the jitter. In this paper, we propose two fast heuristics that quickly compute feasible paths if they exist. These heuristics are compared to the exact QoS routing algorithm: Self Adaptive Multiple Constraints Routing Algorithm (SAMCRA). For that, two main axes are explored. In the first axis, we limited the execution time of our heuristics. The simulation results show that the length of the computed paths is very close to the optimal ones that are computed by SAMCRA. Moreover, these heuristics satisfy more than $80\%$ of the feasible requests. In the second axis, to enforce our hypothesis about the relevancy of the proposed heuristics, we force our algorithms to compute paths until a feasible path is found if such a path exists. The success rate becomes then $100\%$. Moreover, the qualities of found solutions as well as the combinatorial complexity of our heuristics are still attractive.**

***Index Terms*— Routing, multi-constrained, quality of Service, heuristic**

## I. Introduction

Quality of Service (QoS) routing known as multi-constrained routing, consists in computing paths that meet a set of requirements such as delay, bandwidth, and cost. Most of the emerging multimedia applications become more stringent with QoS and require more guarantees. Wang and Crowcroft have proved that the multi-constrained routing problem is NP-hard [1]. For solving exactly or approximatively this problem, many solutions are proposed in the literature. Among the exact solutions, we mention the Depth First Search (DFS) approach which returns a feasible solution to the problem if such a solution exists [2]. Since the worst-case time complexity approach is exponential, Shane et al. proposed in 2001 a heuristic based on the DFS approach [3]. Another approach uses the Constrained Bellman-Ford algorithm for the delay-cost-constrained routing problem as done in [4]. The main idea of this algorithm is at first find minimal cost paths between the source and the other nodes. Then, the algorithm maintains a list of paths that increase the cost and decrease the delay. When the path with the smallest cost and acceptable delay is found the algorithm returns this path. However, in [5], it has been shown that a non-linear length is necessary to solve the QoS routing problem, and the authors replace the cost function by a non-linear length, and proposed the Self Adaptive Multiple Constraints Routing Algorithm SAMCRA. SAMCRA is an exact algorithm based on two main concepts: the use of a non-linear length function and the non dominance of paths. SAMCRA explores like Dijkstra's algorithm all nodes beginning by the source node and maintains a set of non-dominated feasible paths at each node. The algorithm stops when it computes the non dominated feasible path with the smallest non-linear length between the source and the destination nodes.

Since the optimal QoS routing with multiple constraints is NP-hard, heuristics are required for real network applications. A heuristic version of SAMCRA is given by TAMCRA [6], which was proposed earlier as a heuristic to solve the unicast QoS routing. Unlike SAMCRA, TAMCRA bounds the number of non dominated paths that can be stored at each node by a predefined integer $k$. Therefore, the found solution may not be the optimal one. In 2001, Yuan and Liu proposed an extended version of the Bellman-Ford algorithm that finds all optimal paths then chooses a feasible one if such a path exists [7]. In [8], Jaffe defined an approximation algorithm, which computes shortest paths based on a linear combination of the weight values of each link in one new weight. This algorithm was illustrated in the case of two metrics, a generalization for multiple metrics was proposed in [9]. H_MCOP is one of the well-known multi-constrained unicast algorithms that was introduced in [10]. This algorithm is based on the execution of two modified versions of Dijkstra's algorithm in forward and backward directions to compute the shortest paths between two nodes. Other works aim to solve the multi-constrained routing problem using Lagrange relaxation to mix the metrics. In this category, we can cite the algorithms proposed by Feng et al. in 2001 [11], 2002 [12], by Juttner et al. in 2001 [13] and by Guo and Matta in 1999 [14].

Recently, the research community is exploring the use of the metaheuristics to solve the multi-constrained routing problem, such as genetic algorithms [15], tabu search [7], and ant colonies [16].

Instead its effectiveness, the major drawback of SAMCRA resides in its complexity [17]. In this paper, we investigate the possibility to replace SAMCRA by two

simple heuristics that efficiently reduce the execution time and return satisfying solutions. These heuristics are based on the computation of the $k$ shortest paths. For this, we adapt the Yen's algorithm [18], which has the smallest combinatorial complexity [19]. The difference between the two heuristics lies in the metric used for shortest paths computation. Hop Count Approach (HCA) considers the hop count metric of a path, while the second heuristic Metric Linearization Approach (MLA) combines the QoS metrics into one weighted metric.

In the following, we first give a formal definition of the multi-constrained routing problem. In Section III, we outline SAMCRA algorithm. In Section IV, we give an overview of the proposed algorithms for computing the $k$ shortest paths. Our heuristics are presented in Section V. In Section VI, the performance of our heuristics is investigated through a large number of simulations.

## II. Problem Formulation

A communication network is modeled as an undirected weighted graph $G(N, E)$, where $N$ is the set of nodes and $E$ the set of links. Each link $e \in E$ of the network is associated with $m$ QoS parameters denoted by a weight vector: $\overrightarrow{w}(e) = [w_1(e), w_2(e), ..., w_m(e)]^T$. The QoS metrics can be classified into additive metrics such as delay, multiplicative metrics such as loss rate or bottleneck metrics such as available bandwidth. The end-to-end constraints of a given QoS request, from a source node $s$ to a destination node $d$, are given by an $m$-dimensional vector: $\overrightarrow{L} = [L_1, ..., L_m]^T$. In general, bottleneck metrics can be easily dealt with by pruning from the graph all the links that do not satisfy the QoS constraints, while the multiplicative metrics can be transformed into additive metrics by using a logarithm function. The additive metrics cause more difficulties. Therefore, and without loss of generality, we only consider additive metrics. The length of a path $p(s,d)$ corresponding to the metric $i$ is given by: $l_i(p(s,d)) = \sum_{e \in p(s,d)} w_i(e)$. Thus, we define a feasible path $p(s,d)$ as follows:

$$l_i(p(s,d_j)) \leq L_i, \ \forall i = 1, ..., m \tag{1}$$

Using the Pareto dominance, a path $p(s,d)$ dominates another path $p'(s,d)$ if:

$$\begin{cases} l_i(p(s,d)) \leq l_i(p'(s,d)), \ \forall i = 1, ..., m \\ l_j(p(s,d)) < l_j(p'(s,d)), \text{for at least one } j \end{cases} \tag{2}$$

In [5], the authors formulated the multi-constrained routing problem in two different ways.

**MCP problem** The Multi-Constraint Path (MCP) problem consists in finding a path $p(s,d)$ that satisfies a given constraint vector $\overrightarrow{L}$:

$$l_i(p(s,d)) \leq L_i, \ i \in \{1, ..., m\} \tag{3}$$

**MCOP problem** Considering a length function $l$ (e.g. $l(p(s,d)) = \frac{1}{m}\sum_1^m l_i(p(s,d))$), the Multi-Constraint Optimal Path (MCOP) problem consists in finding among the feasible paths, the path $p^*(s,d)$ with the smallest length $l(p^*(s,d))$.

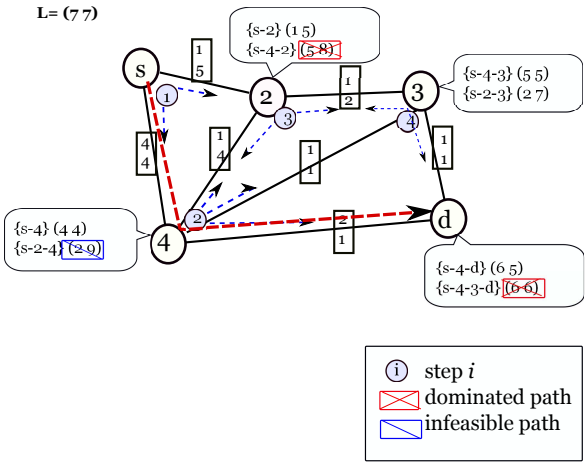


Figure 1. An example of SAMCRA

To evaluate the quality of a path $p(s,d)$, an interesting non-linear length function was defined in [5]:

$$l(p(s,d)) = max_{i=1,...,m}\left(\frac{\sum w_i(e)_{e \in p(s,d)}}{L_i}\right) \tag{4}$$

This length function considers the value of the most critical constraint of a path regarding the end-to-end requirements.

## III. SAMCRA

SAMCRA [5] is an exact multi-constrained routing algorithm that solves the MCOP problem using the non-linear length function and the dominance of paths.

For a node pair $(s,d)$, SAMCRA returns the shortest path that satisfies the constraint vector if such a path exists. Thus, SAMCRA begins by the source node $s$. At each iteration, the algorithm explores the neighbors of the current node and chooses the closest node using the non-linear length function defined in Equation 4. The dominated paths are regularly dropped, while all non dominated ones are memorized. SAMCRA ends when the destinations $d$ is reached, and there is no possibility to find a better path for this destination.

For instance, let us consider the example in Figure 1, where SAMCRA computes the path with the smallest non-linear length between $s$ and $d$. SAMCRA begins by exploring the neighbors of $s$ $\{2, 4\}$ , and chooses the node $4$ with the smallest non-linear length. Then, it explores its neighbors $\{2, 3, d\}$. The algorithm stops at the $4^{th}$ iteration, when the node $d$ is selected to having the smallest non-linear length. In the same figure, we notice that dominated paths like $(s-4-3-d)$ are dropped as well as the infeasible one $(s-2-4)$.

## IV. The $k$ shortest paths algorithms

Routing is usually associated with the computation of shortest paths (in number of hops for example). However,

when the shortest path does not satisfy the constraints of QoS, it becomes necessary to compute a set of $k$ shortest paths between a source node and destination node to find a feasible path. The $k$ shortest paths problem is a natural and long-studied generalization of the shortest path problem in which not one but several paths in an increasing order of length are required. The $k$ shortest paths problem in which paths can contain loops turns out to be significantly easier. An algorithm with a complexity in $O(|E| + k.|N|.log|N|)$ has been known since 1975 [20]; a recent improvement by Eppstein achieves the optimal complexity in: $O(|E| + |N|.log|N| + k)$ [21]. However, the problem of determining the $k$ shortest paths without loops has proved to be more challenging. The problem was first examined by Hoffman and Pavley [22]. For undirected graphs, the most efficient algorithm was proposed by Katoh et al. [23], which has a complexity in $O(k.(|E|+|N|.log|N|))$. Since undirected graphs can be transformed on directed graphs, by replacing the undirected link with two directed links with the same weights, on the most general case, the best known algorithm is that proposed by Yen in [18]. The Yen's algorithm was generalized by Lawler in [24] and has a complexity in $O(k.|N|(|E| + |N|.log|N|))$.

## V. Proposed Heuristics

### A. Motivation

It has been proved that the multi-constrained routing problem is NP-hard [1]. SAMCRA is an efficient algorithm that exactly solves this problem. Although its effectiveness, SAMCRA can be expensive with a combinatorial complexity in: $O(k|N|\log(k|N|)+k^2m|E|)$ [17], with $k$ the number of non dominated paths that can be stored at each node queue. Therefore, it may be more interesting to use an approximate algorithm with less combinatorial complexity expecting a feasible path between the source and the destination nodes. Reducing the execution time while giving satisfying solutions is our main motivation to propose our fast heuristics. These heuristics are based on the computation of the $k$ shortest paths. The idea of applying such an algorithm is that the shortest paths may be feasible. For that, we propose the modification of the well-known algorithm of Yen [18] to compute the paths between two nodes in increasing order of their length until (i) a feasible path that satisfies all constraints is found, (ii) or a given limitation of computed paths is reached.

To solve the multi-constrained unicast routing, we propose two heuristics based on the computation of shortest paths. Indeed, we argue that one of these computed paths will be feasible. Moreover, these heuristics have a combinatorial complexity that is bounded by the number of allowed computed shortest paths. The proposed heuristics are using one additive metric. The first heuristic computes the paths with the smallest number of links. The second heuristic uses a combination of the QoS metrics in a single one, using an efficient technique that will be explained in detail in Section V-C.

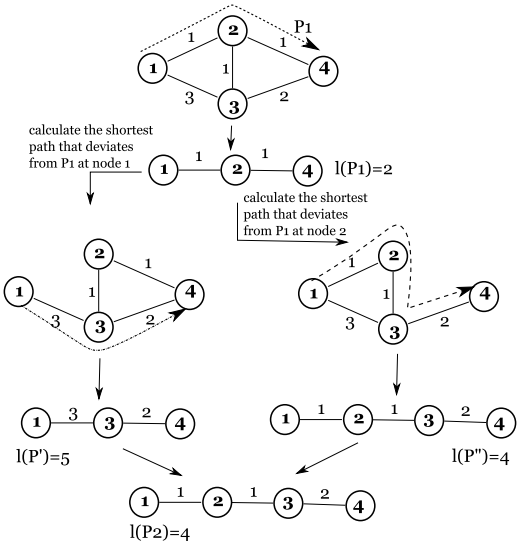


Figure 2. An example of Yen's algorithm processing

Both the proposed heuristics are based on Yen's algorithm, which we outline in the following.

### B. Yen's Algorithm

As shown in many studies [19], Yen's algorithm is the most pertinent and fast $k$ shortest paths algorithm that was introduced in [18].

For a given node pair $(s, d)$ and a given integer $k$, this iterative algorithm computes, using one additive metric, the $k$ shortest paths between these nodes. For that, it begins by computing the first shortest path using the Dijkstra algorithm. At the $i^{th}$ iteration, the algorithm computes the $i^{th}$ shortest path by considering all possible paths that deviate from the $(i-1)^{th}$ shortest path that are not already computed.

For instance, let us consider the example in Figure 2, where the first two shortest paths between the nodes 1 and 4 are required. The algorithm begins by computing the shortest path $P_1$. Then, it computes all shortest paths that deviate from $P_1$ at nodes 1 and 2. Two paths are computed $P^{'}$, $P^{''}$. The shortest one, here $P''$ with length 4, is the second shortest path $P_2$.

### C. Algorithmic Description of the Proposed Approaches

In this paper, we propose two fast heuristics that compute shortest paths in an increasing order, given an additive length function. These heuristics stop when (i) a path satisfying all the QoS constraints is found or (ii) an upper bound of the number of computed paths $k_{max}$ is reached. $k_{max}$ is an important parameter, since the combinatorial complexity and so the execution time of our heuristics depend on its value. Indeed, when $k_{max}$ is small, these heuristics are fast, and the number of satisfied request can be small. In parallel, when $k_{max}$ increases, the number of satisfied requests increases too. Furthermore, the proposed heuristics use a single additive metric obtained by two ways:

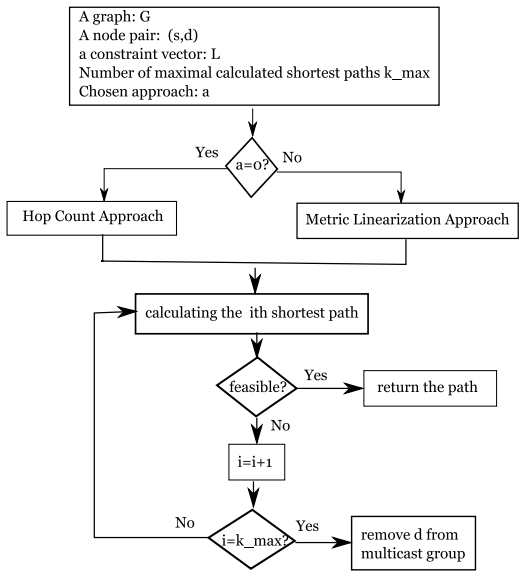


Figure 3. Proposed heuristics diagram

**Hop Count Approach (HCA):** in this approach, the algorithm searches for the shortest paths considering the number of hops as the only metric to optimize.

**Metric Linearization Approach (MLA)**: at first, it substitutes the weight vector $\vec{w}\ (e) = [w_1(e), .., w_m(e)]^T$, by a scalar weight $w^{'}(e) = \sum_{i=1,...,m} \alpha_i w_i(e)$. To calculate the parameters $\alpha_i$, the MLA approach computes $p_i^*(s, d_j),\ i = 1, ..., m$, the shortest path that optimizes the metric $i$. Then, the parameter $\alpha_i$ is calculated as follows:

$$\alpha_i = \frac{l_i(p_i^*(s, d_j))}{L_i} \tag{5}$$

$\alpha_i$ can be named *the criticality degree* of the constraint $L_i$. When $\alpha_i$ is close to 1, that means $p_i^*(s, d_j)$ is very close to $L_i$. Consequently, it is necessary at first to satisfy the constraint $L_i$.

Figure 3 summarizes the process of our heuristics HCA and MLA. For a given graph $G$, a given node pair $(s, d)$, with a constraint vector $\vec{L}$ and a chosen approach, the algorithm returns the fist feasible path between $s$ and $d$, if such a path is one of the $k_{max}$ shortest paths that are allowed to be computed. For that, the algorithm chooses the approach to use and compute the combined link weights in case of MLA. Then, it computes shortest paths in an increasing order. The process stops when a feasible path is found, or $k_{max}$ iterations are already done. A more detailed meta-code is presented in Algorithm 1 for MLA approach.

### *D. Limitations of the Yen's algorithm*

The proposed heuristics MLA and HCA are based on the computation of the $k$ shortest paths using the Yen's algorithm. However, Yen's algorithm computes the $k$ shortest paths without considering all the existing paths as we will demonstrate it in Figure 4.

**Algorithm 1** MLA meta-code

```
for (i = 1, ..., m) do
    Compute p_i^*(s, d)
    α_i = l_i(p_i^*(s,d)) / L_i
end for
for all e ∈ G do
    w'_i(e) = Σ_{i=1,...,m} α_i w_i(e)
end for
j ← 1, Find←false, k ← 1,
p_1(s, d) = Dijkstra(s, d), P, P'//paths
D = p_1(s, d) // candidate set
X = φ //shortest paths set
while ((Find≠true) and (k ≤ k_max)) do
    P ←shortest path in D

    if (p_j(s, d) is feasible) then
        Find←true
    else
        j ← j + 1
        v ← deviation[1](p_j(s, d), X)
        while (v ≠ d) do
            P ← Dijkstra(v, d)
            P' ← prefix[2](p_j(s, d), v) + P
            D ← D + P'
            v ←successor(v, p_j(s, d))
        end while
    end if
end while
```

On the left side (a), at first HCA computes the three shortest paths $P_1$,$P_2$ and $P_3$, that have the same hop count. When computing the fourth shortest path using the deviation concept at node 2, the algorithm will choose one of the two paths $P'$ and $P$". As the algorithm chooses one of the two latter paths, it can not choose the other one at the next iteration. This can lead to some cases where not all paths are explored and the feasible path can be skipped. For instance, if the constraints are given by the vector $\vec{L}\ (3, 3)$, only the path $P'$ will be feasible, and this path can be skipped by the original Yen's algorithm.

On the right side (b), we suppose that $\vec{L}= (4, 4)$. MLA starts by computing the shortest paths considering successively the two metrics. Here the path $(s - 2 - d)$ minimizes the first metric with the value 2, and the path $(s - 1 - d)$ minimizes the second metric with the value 2. Then, MLA computes the parameters $\alpha_1 = \frac{2}{4} = 0.5$ and $\alpha_2 = \frac{2}{4} = 0.5$. After adding the new weights to the links, MLA computes the first shortest path $P_1$. At the second iteration, when computing the shortest path at the deviation node $s$, MLA can chooses $P^*$ or $P^{**}$ since they have the same length 4. However, only $P^*$ is feasible.

This limitation prevents to compute feasible paths even if this path is one of the shortest paths. To cure this, we propose a modification of the Yen's algorithm. We replace the computation of the shortest path using Di-

[2]deviation is a function that returns the furthest node at which a path P deviates from a set of paths

[2]prefix retuns the sub-path of a path P, between the source node and a defined node v

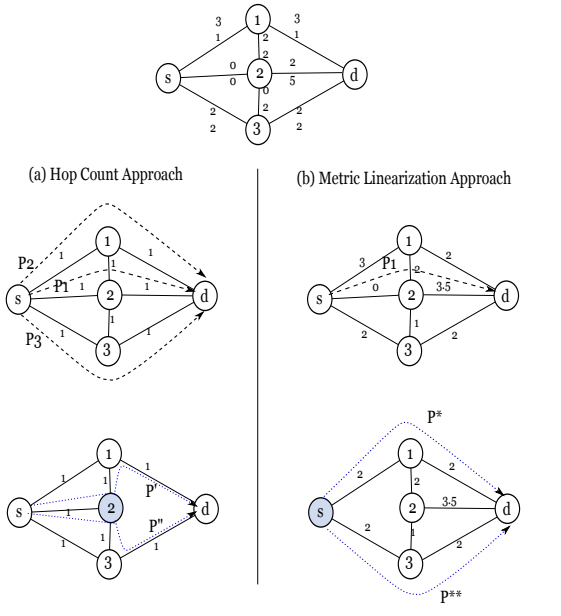


Figure 4. The relevance of enumerating all paths having the same length

jkstra's algorithm by a modified version denoted Mod-Dijkstra that computes all shortest paths that have the same length at the same time, if more than one exist. For this, instead of saving only one of the predecessors with the smallest length at each node, the algorithm saves all the predecessors that have the same smallest length.

### *E. Exact HCA and Exact MLA algorithms*

Exact HCA (E-HCA) and exact MLA (E-MLA) are two modified version of HCA and MLA respectively. These algorithms are based ob three main modifications.

- at each deviation node, the algorithm of computation of shortest paths computes all paths with the same additive length (hop count for E-HCA and metric linearization for E-MLA),
- the upper bound $k_{max}$ of computed paths become infinite ($k_{max} \rightarrow \infty$),
- the algorithm stops only when a solution is found for feasible requests.

Considering the example of Figure 4, two iterations will be sufficient for HCA to compute the feasible path $P$" without skipping this path. In the first iteration, Mod-Dijkstra returns the set $P_1, P_2, P_3$. Then, at the second iteration it returns both $P'$ and $P$". For MLA, the problem is less recurrent because of the new weights computation. Indeed, the paths have generally different lengths, and MLA cannot skip paths that have the same length.

## VI. Performance Evaluation and Simulations

The performance of the two proposed heuristics and SAMCRA are investigated through extensive simulations. For that, we use a realistic network with $50$ nodes and $82$ links denoted by Real-Topology [25]. Each link is associated with two additive weights. These weights are randomly generated using a uniform distribution in the interval $[1, 1024]$.

Different classes of constraints are also considered. For each pair of nodes $(s, d)$, the constraint vectors are generated in a way that they browse a defined space generation by areas from the strictest constraints to the loosest ones. In Figure 5, where only two metrics are considered, $P_1$ and $P_2$ denote the shortest paths between $s$ and $d$ that minimizes the first and second metric respectively. The shaded rectangle (B) delimited by $l_1(P_1),\ l_1(P_2)$ and $l_2(P_2),\ l_2(P_1)$ circumscribes the region where the constraints are selected. This region is divided in $10$ areas: area 1, area 2,..., area 10 (also denoted in the simulation figures by 1,2,...,10). The constraints are randomly selected within these areas. Outside the specified region, the QoS constraints are less interesting to be examined. Indeed, all constraints that are generated within space (A) are infeasible, while all constraints generated in space (C) are trivial and any polynomial algorithm will be sufficient to find solutions. We note that strict constraints are close to $l_1(P_1)$ and $l_2(P_2)$ (area 1), while loose constraints are close to $l_1(P_2)$ and $l_2(P_1)$ (area 10).

In the followings, two series of simulation are performed.

### *A. HCA and MLA performance evaluations*

In this part, several series of simulations have been performed. We randomly generate $100$ instances of link weights. For each instance of link weights, $100$ pair of nodes are randomly selected. Thereafter, $10$ routing requests are generated within each area, from the strictest constraints (area 1) to loosest ones (area 10). After that, the three algorithms: SAMCRA, HCA and MLA are executed independently to find a solution. Four performance measures are computed.

- *Success rate:* it is the number of satisfied routing requests from $100$ generated requests,
- *Quality of computed paths:*it corresponds to two lengths. The non-linear length and the average length of computed paths. The non-linear length is used by SAMCRA (Equation 4) and corresponds to the satisfactory degree of the most critical constraint. The average length ($l_{\text{avg}}(p(s,d)) = \frac{1}{m}\sum_1^m l_i(p(s,d))$) equivalently considers all the metrics, and computes the average quality of the computed paths.
- *Relative complexity:* it is the number of operations[3] that are performed to find a feasible solution,
- *Absolute complexity:* it is the number of operations that are performed before answering a given routing request, instead the request is infeasible,

We note that all these performance measures have been computed with $95\%$ confidence intervals according to the ten constraint generation areas. The upper bound $k_{max}$ of the computed shortest paths in our heuristics is fixed to three.

- **Success rate**

[3]an elementary operation corresponds to the visit of one node

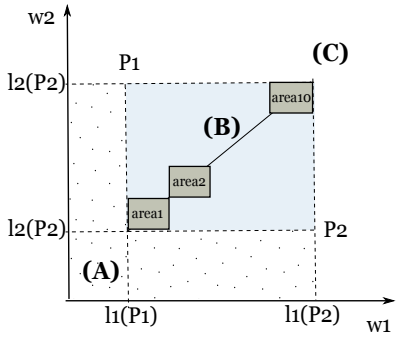


Figure 5. The constraints generation

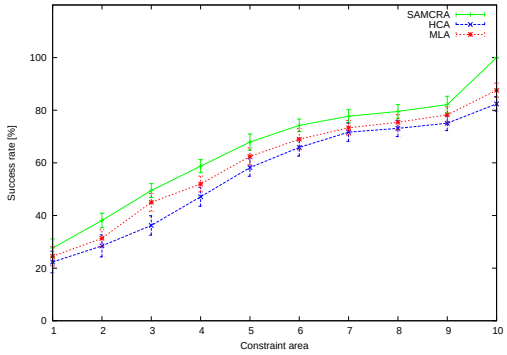


Figure 6. The success rate of the three algorithms SAMCRA,HCA and MLA

Figure 6 shows the success rate of the three algorithms: SAMCRA, HCA and MLA. Foremost, we notice that the success rate of the three algorithms is increasing. In fact, for strict constraints there is few feasible requests, and this number is increasing when constraints become loose. Since SAMCRA is an exact algorithm, it gives the highest success rate, while the upper bound of computed paths in our heuristics is fixed to three. The success rate of SAMCRA varies from $29\%$ for strict constraints to $100\%$ for the loose ones. For strict constraints, the gap between SAMCRA and our heuristics does not exceed $4\%$ with MLA and $7\%$ with HCA. For loose constraints (area $10$), the difference becomes $12\%$ with MLA and $16\%$ with HCA. Indeed, the area $10$ presents the trivial constraints for which SAMCRA always finds a solution, and the number of feasible requests is $100\%$.

The execution time is an important parameter when evaluating any routing algorithm. To evaluate more deeply the three algorithms, two kinds of complexity metrics are proposed. The relative complexity for feasible requests and the absolute complexity for the total generated requests.

- **Relative complexity**

Relative complexity is calculated only if a solution is found by the three algorithms. In Figure 7, we notice that the relative complexity of SAMCRA is significantly larger than both of HCA and MLA. Indeed, when the constraints are not strict, SAMCRA has more paths to explore before returning the optimal solution according to the non-linear length, while both approaches HCA and MLA stop at the first feasible path they find. This reduces their execution time.

- **Absolute complexity**

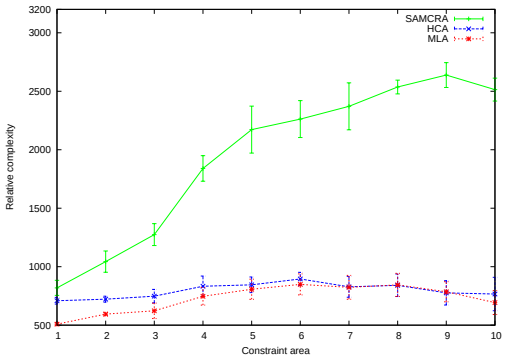


Figure 7. Relative complexity of the algorithms SAMCRA, HCA and MLA

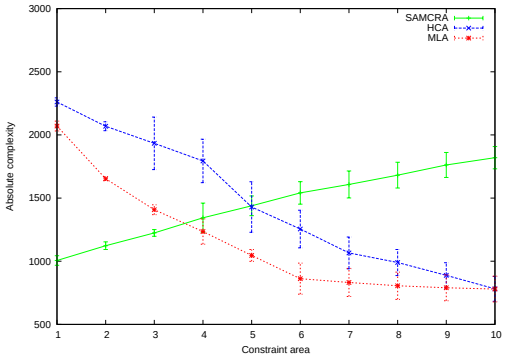


Figure 8. The absolute complexity of the algorithms SAMCRA, HCA and MLA

In Figure 8, we can state that the absolute complexity of both proposed heuristics HCA and MLA is greater than SAMCRA complexity for strict constraints. Indeed, when the constraints are strict, SAMCRA rapidly rejects non feasible requests, while both approaches explore the maximum number of paths (here fixed to three) without finding solutions. When constraints become less strict; the two proposed approaches find feasible solutions before computing the three paths, which significantly reduces their execution time.

- **Quality of computed paths**

In Figure 9, the solutions found by our approaches HCA and MLA are a little bit worse than those found by SAMCRA with $6.67\%$ and $2.39\%$ respectively. Obviously, SAMCRA finds the path with the smallest non-linear length, while both heuristics HCA and MLA stops at the first feasible path, which can be worse than the optimal one. Moreover, for strict constraints, the solutions computed by the three algorithms are significantly equal. In fact, for strict constraints, the number of feasible paths is very small, and if HCA or MLA computes a feasible path, this path has big chances to be the optimal one.

In Figure 10, the average length of MLA is better than that of SAMCRA with $1.59\%$, and the average length of HCA is very close to that of SAMCRA. Indeed, the linearization of the metrics involves the computation of paths based on both metrics, unlike SAMCRA which does not consider the variation between the metrics and

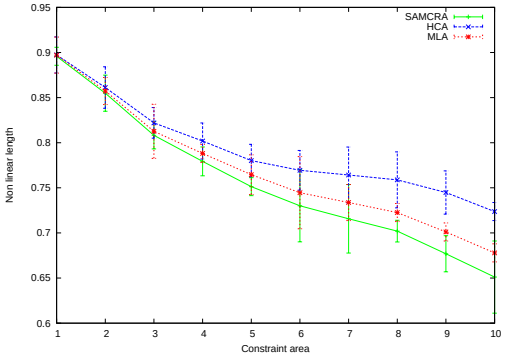


Figure 9. The non-linear length of the paths computed by SAMCRA, HCA and MLA

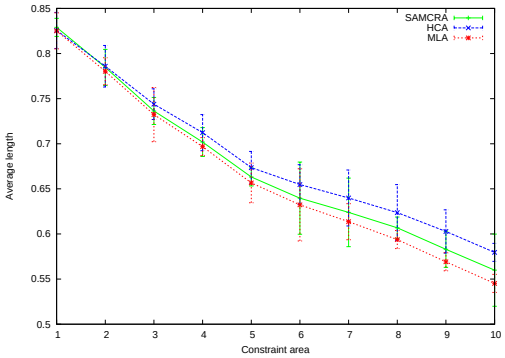


Figure 10. The average length of the paths computed by SAMCRA, HCA and MLA

focuses on the most critical one. The following example highlights the difference between non-linear length and average length evaluation. For that, let us consider two paths: the path computed by SAMCRA $p_1(s,d)$ : with the lengths $l_1(p_1(s,d)) = 0.6$, $l_2(p_1(s,d)) = 0.7$ corresponding to the metrics 1 and 2 respectively, and the path computed by HCA $p_2(s,d)$ : with the lengths $l_1(p_2(s,d)) = 0.1$, $l_2(p_2(s,d)) = 0.9$. It is clear that the non-linear length $l(p_1(s,d)) = 0.7 < l(p_2(s,d)) = 0.9$, while the average length $l_{avg}(p_1(s,d)) = 0.65 > l_{avg}(s,d) = 0.5$.

### *B. Exact-HCA and Exact-MLA Performance Evaluation*

In the second series of our study the exact version of the algorithms E-HCA and E-MLA has been analyzed. In fact, after the modification of the Yen's algorithm in order not to skip paths with the same length, E-HCA and E-MLA will explore all existing paths ($k_{max} \rightarrow \infty$) if necessary, between the source and the destination nodes. For this, they compute the paths in an increasing order according to the previously explained metrics. E-HCA uses the hop count metric, while E-MLA uses the linearization metric. In this series of simulations, the requests as well as the constraints are generated similarly to the first series of simulations. Two measure parameters are evaluated:

- *Number of computed paths:* is the average number of computed paths to find a feasible solution for feasible requests,
- *Exact complexity :* is the number of operations that are performed to find a solution for all feasible requests.

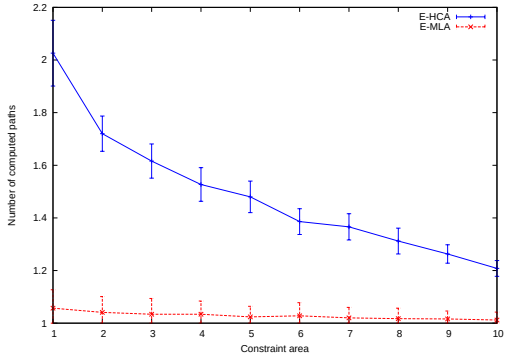


Figure 11. Number of computed paths before finding a solution by E-HCA and E-MLA

- **Number of computed paths**

Figure 11 shows the number of computed paths by E-HCA and E-MLA before finding a feasible path. For strict constraints, E-HCA needs 2.1 paths to find a solution while E-MLA does not need more than 1.13 paths. The number of computed paths is decreasing when the constraints are less strict. Instead numbers of computed paths by E-HCA and E-MLA are lower than the upper bound $k_{max}$ of HCA and MLA, E-MLA as well as E-MLA computes paths until a feasible one is found. In the 10% of cases where HCA and MLA do not find feasible paths they need to compute more than 3 paths. In the other cases, the computed paths is almost the first or the second one.

A simple demonstration can justify these small values. Let us suppose that HCA needs 1.5 paths to find solution for 23% of the generated requests as shown in Figure 6. In addition, E-HCA needs to compute 5 paths for the 5% to reach the success rate of 28% as SAMCRA. Since 23% presents 82% of 28%, the average number of computed paths $N_p$ will be computed as follows: $N_p = 0.82 * 1.5 + 0.18 * 5 = 2.13$. A similar demonstration can be done for MLA.

- **Exact complexity**

In Figure 12, the exact complexity of E-HCA is bigger than SAMCRA for strict constraints. This is due to the modified version of Yen's algorithm and the computation of additional paths. This high complexity can also be explained by the increasing number of explored paths and the small number of feasible ones. Let us suppose that the only feasible path is the second shortest one and there are 10 first shortest paths, and 10 second shortest paths. In this case, at least 10 paths and at most 19 paths will be explored before finding the feasible one. Since the number of computed paths in E-MLA is less than 1.13 paths, its exact complexity still the smallest one for both strict and loose constraints. Indeed, E-MLA computes paths using the linearization metric that simultaneously takes into account the two considered metrics.

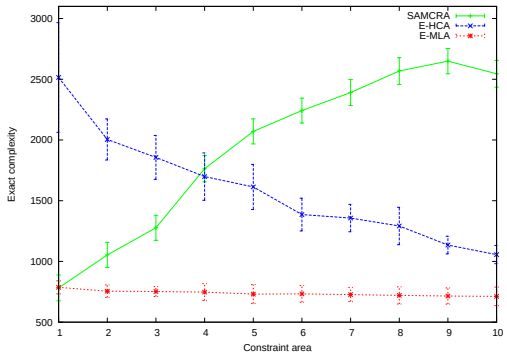


Figure 12. The exact complexity of SAMCRA, E-HCA and E-MLA

The exact complexity of E-MLA is very interesting since it is still smaller than SAMCRA, while having $100\%$ of success rate. However, the compromise done is regarding the quality of found solutions by E-MLA which may not be optimal but still feasible and this can be sufficient for most requests.

## VII. Conclusion

To solve the QoS routing problem, we proposed two fast heuristics HCA and MLA. These heuristics are simple to implement and give attractive results regarding the success rate, the quality of found solutions and they considerably reduce the execution time. The execution time is one of the most challenging parameters in the treated NP-hard problem. For this, extensive simulations are performed to compare our heuristics to the well-known algorithm SAMCRA. Two main ideas are developed then confirmed in this paper. The first idea is that HCA as well as MLA have a bounded combinatorial complexity that can be readjusted to reach a minimum success rate. The second idea focuses on the combinatorial complexity of HCA and MLA if they are transformed into exact algorithms, and the obtained results are very satisfying. Finally, since the current network applications become more exigent, it is interesting to explore the heuristic solutions to solve the NP-hard QoS routing problem.

## References

[1] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 1228–1234, 1996.

[2] R. E. Tarjan, "Depth-First Search and Linear Graph Algorithms," *SIAM Journal of Computers*, vol. 1, no. 2, pp. 146–160, 1972.

[3] D. W. Shin, E. K. P. Chong, and H. J. Siegel, "A multi-constraint QoS routing scheme using the depth-first search method with limited crankbacks," in *IEEE workshop on high performance switching and routing*, 2001, pp. 385–383.

[4] S. Chen and K. Nahrstedt, "On finding multi-constrained paths," in *ICC*, 1998.

[5] P. Van Mieghem, H. De Neve, and F. A. Kuipers, "Hop-by-hop quality of service routing," *Computer Networks*, vol. 37, no. 3/4, pp. 407–423, 2001.

[6] H. D. Neve and P. V. Mieghem, "Tamcra: a tunable accuracy multiple constraints routing algorithm," *Computer Communications*, vol. 23, no. 7, pp. 667–679, 2000.

[7] Y. Xin, "Heuristic algorithms for multiconstrained quality-of-service routing," *IEEE/ACM Transaction in Networks*, vol. 10, no. 2, pp. 244–256, 2002.

[8] J. M. Jaffe, "Algorithms for finding paths with multiple constraints," *Networks*, vol. 14, pp. 95–116, 1984.

[9] H. A. L. Lachlan and A. Kusumat, "Generalised analysis of a qos-aware routing algorithm," in *in IEEE GLOBECOM'98*, 1998, pp. 118–123.

[10] T. Korkmaz and M. Krunz, "Multi-constrained optimal path selection," vol. 2, pp. 834–843, 2001.

[11] G. Feng, K. Makki, N. Pissinou, and C. Douligeris, "An efficient approximation algotihm for delay-cost-constrained QoS routing," in *Tenth International conference on computer communications and networks*, 2001, pp. 395–400.

[12] G. Feng, C. Douligeris, K. Makki, and N. Pissinou, "Performance evaluation of delay-constrained least-cost QoS routing algorithms based on linear Lagrange relaxation," in *IEEE international conference on communications*, vol. 4, 2002, pp. 2273–2281.

[13] A. Jttner, B. Szviatovszki, I. Mcs, and Z. Rajk, "Lagrange relaxation based method for the qos routing problem," 2001.

[14] L. Guo and I. Matta, "Search space reduction in QoS routing," *Comput. Networks*, vol. 41, no. 1, pp. 73–88, 2003.

[15] Y. H. S. Wan and Y. Yang, "Approach for multiple constraints based qos routing problem of network," *Hybrid Intelligent Systems, International Conference on*, vol. 2, pp. 66–69, 2009.

[16] W. L.-j. S. Li-juan and W. Ru-chuan, "Ant colony algorithm for solving qos routing problem," *Wuhan University Journal of Natural Sciences*, vol. 7, pp. 449–453, 2004.

[17] P. Van Mieghem and F. A. Kuipers, "On the complexity of QoS routing," *Computer Communications*, vol. 26, no. 4, pp. 376–387, 2003.

[18] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, pp. 712–716, 1971.

[19] M. M. J. Hershberger and S. Suri, "Finding the k shortest simple paths: A new algorithm and its implementation," *ACM Transactions on Algorithms*, vol. 3, 2007.

[20] B. L. Fox, "More on kth shortest paths," *Commun. ACM*, vol. 18, no. 5, p. 279, 1975.

[21] D. Eppstein, "Finding the k shortest paths," in *FOCS*, 1994, pp. 154–165.

[22] W. Hoffman and R. Pavley, "A method for the solution of the nth best path problem," *J. ACM*, vol. 6, no. 4, pp. 506–514, 1959.

[23] N. Katoh, T. Ibaraki, and H. Mine, "An efficient algorithm for $K$ shortest simple paths," *Networks*, vol. 12, no. 4, pp. 411–427, 1982.

[24] E. L. Lawler, "A procedure for computing the $K$ best solutions to discrete optimization problems and its application to the shortest path problem," *Management Science*, vol. 18, pp. 401–405, 1972.

[25] A. Zimolka, "Design of survivable optical networks by mathematical optimization," Ph.D. dissertation, Technische University Berlin, 2007.