Heuristics for Multicriteria Routing Problem

Alia BELLABAS[1, 3], Miklos MOLNAR[1, 3], Samer LAHOUD[1, 2]

[1]*Institut de Recherche en Informatique et Systèmes Aléatoires IRISA*
Campus de Beaulieu, 35042 Rennes, FRANCE

[2]*Rennes1 university, Rennes, France*

[3]*Insitut National des Sciences Appliquées INSA*
Rennes, FRANCE

{alia.bellabas, miklos.molnar, samer.lahoud}@irisa.fr

Abstract— **The applications in current networks require more and more quality of services. Hence, a routing algorithm has to satisfy several constraints such as delay, bandwidth or jitter. This is called multicriteria routing algorithm. Since the multicriteria routing is an NP-Hard problem, we propose heuristics that calculate quickly paths that satisfy Quality of service QoS constraints between a source node and a destination node. Several solutions exist in the literature; the most efficient algrithm is SAMCRA (Self Adaptive Multiple Constraints Routing Algorithm) which was proposed by Van Mieghem et al. in 2001. SAMCRA is an exact unicast algorithm for multiple constraints which has a high complexity. In our study, we examine the possibility to replace SAMCRA by an optimized *k* shortest paths algorithm. The simulation results show that applying such algorithm reduces significantly the complexity of the multicriteria routing algorithm, and gives efficient solutions.**

I. INTRODUCTION

Recently, routing problems have become increasingly important given the emergence of applications that require guarantees on a range of QoS parameters such as delay, cost, bandwidth, loss rate, jitter, etc. These new challenges have led to multicriteria routing problem. Several algorithms are proposed to resolve multicriteria routing problems. Some of these algorithms consider two main metrics: cost and delay [1][2].Two approaches exist for these algorithms. A first approach considers the multicriteria routing problem as a mono-objective optimization one that minimizes the cost under the delay constraint. A second approach uses a multi-objective formulation: to solve this class of problems, some proposed works use meta-heuristics such as genetic algorithms [2], taboo search [3], or ant colonies [4]. Other works propose exact multi-objective algorithms. The most efficient one is SAMCRA which was proposed by Van Mieghem et al. in [5]. SAMCRA is an exact multicriteria routing algorithm. However, the major drawback of SAMCRA is its complexity [6]. In this paper, we examine the possibility to replace SAMCRA by simple heuristics that reduce efficiently execution time and return satisfactory solutions.

In the following, we first give a formal definition of the multicriteria routing problem. In Section III, we present SAMCRA algorithm. In Section V, we give an overview of different algorithms for calculating the k shortest paths existing in the literature. We then detail the operation of Yen's algorithm. In section VII, the performance of our heuristics is investigated through a large number of simulations.

II. PROBLEM FORMULATION

A communication network is modelled as a valuated graph $G(N, E)$, where N is the set of nodes and E the set of links. Let consider a pair of a source node s and a destination node d. For the multicriteria routing problem, the Constraints are given by the vector $\vec{L} = (L_1, L_2, \ldots, L_m)$ where m is the number of metrics to consider. For each link e is associated a weight vector $\vec{w}(e) = (w_1^e, w_2^e, \ldots, w_m^e)$.

In network, the metrics can either be additive (such as delay and cost), concave (typically available bandwidth), or multiplicative (like loss rate). Concave metrics are treated by omitting all links that violate the constraints, and then try to find, in the residual graph, a path between the source node s and the destination node d. On the other hand, the multiplicative metrics can be translated to additive metrics using the logarithm. Thus, the most general metrics and most difficult to satisfy are the additive ones. In the following, we only consider additive metrics.

The weight of a path p corresponding to a metric i is equal to the sum of weights of its links for this metric:

$$l_i(p) = \sum_{e \in p} w_i^e$$

The multicriteria routing problem was formulated in two different ways [6].

Definition 1. MCP problem

The Multi-Constraint Path (MCP) problem *(P)* consists to find a feasible path p that satisfies all constraints L_i:

$$(P)\text{: } l_i(p) \leq \vec{L_i} \text{ for } 1 \leq i \leq m$$

Definition 2. MCOP problem

Considering a length function l (e.g. $l(p) = \frac{1}{m}\sum_{i=1}^{m}\frac{l_i(p)}{L_i}$), the Multi-Constraint Optimal Path (MCOP) problem *(P*)* consists to find the smallest length path within the set of feasible paths $feasible\_paths(s, d)$ between source s and destination d:

$$(P^*)\text{: } min_{p \in feasible\_paths(s,d)} l(p)$$

III. SAMCRA ALGORITHM

SAMCRA [7] is an exact unicast multicriteria routing algorithm that resolves the *MCOP* problem basing on two principles: non-linear length and dominance of paths.

Definition 3. Non-linear length
SAMCRA uses a non-linear length function l to calculate the paths. Let consider a path P with k links: $P = \{e_1, e_2, \dots, e_k\}$
The length of p proposed is:

$$l(P) = max_{1\le i\le m}\left(\frac{\sum_{j=1}^{k} w_i^{e_j}}{L_i}\right)$$

Definition 4. Dominance
A path p_2 is said to be dominated by a path p_1 if $l_i(P_1) \le l_i(P_2)$ for each metric i, and $l_j(P_1) < l_j(P_2)$ for at least one metric *j*.

SAMCRA returns, for a nodes pair *(s, d)*, the path with the smallest non-linear length that satisfies all the constraints, if such path exists. SAMCRA begins by exploring the neighbours of the source *s*, then goes to the neighbour with the smallest non-linear length, and explores its neighbours. Thus, SAMCRA explores all nodes from the source *s*, and prunes dominated paths. SAMCRA stops when the destination *d* is selected as the node corresponding to the shortest non-dominated path. Thus, all paths those can improve the length of the best path found until now are already covered.

IV. OUR CONTRIBUTION

It has been proved that multicriteria routing problem is *NP*-hard [16]. SAMCRA is an exact algorithm with a high complexity [6][1]. Multicriteria routing problems do not necessarily looking for the optimal solution, but a feasible one. Therefore, we propose heuristics based on a modified *k* shortest paths algorithm to calculate more quickly feasible paths. The idea of applying such an algorithm is that the shortest paths may be feasible. In our study, we propose the modification of a known algorithm (see Yen algoirithm in section V.B.), to calculate paths between two nodes in increasing order of their lengths, until a path that satisfies all constraints is found. These heuristics resolve the *MCP* problem but don't guarantee to find an optimal solution. However, the heuristics return satisfactory solutions, and the simulation results presented in Section VII confirm this idea.

V. THE K SHORTEST PATHS ALGORITHMS

A. RELATED WORKS

The shortest paths (e.g. in number of hops) are fundamental in many areas of computer science, operation research and engineering. Most complex applications of the shortest path problem, however, require more than just the calculation of a single shortest path.

[1] [6]: In this paper, authors study the case when SAMCRA achieves the worst case.

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 increasing order of length are sought. The k shortest paths problem in which paths can contain loops turns out to be significantly easier. An algorithm with the complexity of $O(|E| + klog|N|)$ has been known since 1975[8]; a recent improvement by Eppstein [9] achieves the optimal complexity of $O(|E| + |N|\log |N| + k)$. But, the problem of determining the k shortest paths without loops (loopless paths) has proved to be more challenging. The problem was first examined by Hoffman and Pavley[10]. For undirected graphs, the most efficient algorithm was proposed by Katoh et al. [11] which has the complexity of $O\big(k(|E| + |N|\log |N|)\big)$. For the most general case, the best known algorithm is that proposed by Yen in [12], and generalized by Lawler in [13]. This algorithm has achieved the complexity of $O\big(k|N|(|E| + |N|\log |N|)\big)$.

B. YEN'S ALGORITHM

Yen's algorithm belongs to the deviation algorithms class. The deviation of a path from a set of paths is explained by the example shown in Fig1.

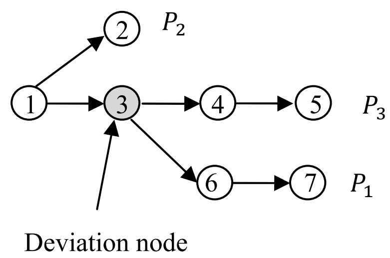

Fig1. Paths' deviation

Let consider three paths P_1, P_2 and P_3. It should be noticed that these paths do not contain repeated nodes, therefore they are loopless paths. P_1 deviates from P_2 in *node 1*. Moreover, P_3 coincides with P_1 from *node 1* until *node 3*, and coincides with P_2 only in *node 1*. Therefore, P_3 deviates from $\{P_2, P_3\}$ in *node 3*. *Node 3* is called deviation node.
The pseudo-code of Yen's algorithm is described in Fig2. The algorithm begins by calculating, for the chosen pair of nodes, the shortest path *P** (line 1). The algorithm performs k iterations. At the i^{th} iteration, the algorithm gets the shortest path stored in the candidates set *D*. This path is the i^{th} shortest path from source to destination. Then, it calculates the deviation node of this path form all the *(i-1)* paths in *X*. To avoid recalculating already computed paths, the algorithm removes nodes and links as explained in lines 9 and 10. In the residual graph, the shortest path *P'* between deviation node and destination is calculated. Then, *P'* is concatenated with the sub-path of the i^{th} shortest path from source to deviation node. The new constructed path is saved in the set of candidates *D*. All deleted nodes and links are restored (line 14), and the algorithm comes to the successor of deviation node in the i^{th} shortest path (line 15).

Yen's algorithm (G, s, t, k)

1. P^* ← Dijkstra (s, t)
2. D ← {P^*} //set of candidates
3. X ← {} // set of the k shortest paths
4. For i from 1 to k
5. P ← the shortest path in D
6. v ← the deviation node (P, X)
7. X ← X+{P}
8.   While $v!=t$
9.   remove all nodes of P form s to v
10.   remove all output links of v that belong to X
11. P' ← Dijkstra (v, t)
12.   concatenate P' with the sub-path of P from s to v
13. D ← D+{P'}
14.   restore all removed nodes and links
15. v ← successor(v, P)

Fig2. Yen's algorithm pseudo-code

VI. Proposed Algorithm and Approaches

In this paper, we propose to replace SAMCRA by a modified Yen's algorithm for computing the k shortest paths. This algorithm has a complexity much smaller than SAMCRA's one. However, the calculation of k shortest paths uses a single metric. Therefore, to apply the proposed algorithm, we translate the multicriteria problem to a monocriterion one. For this, we propose two approaches:

- **the hop count approach:** In this approach, the algorithm searches the shortest paths considering the number of hops that satisfy all the constraints
- **the metrics linearization approach:** one of the most common approaches to the metrics linearization is that proposed by Jaffe [14]. This approach considers the weight of a link e given by: $w(e) = \sum_{i=1,m} \alpha_i w_i^e$, where w_i^e is the weight of e corresponding to the i^{th} metric. To calculate the parameters α_i, we propose to calculate the m shortest paths $P_1, P_2, \dots, P_m$ corresponding to the m metrics. The parameters α_i are given by:

$$\alpha_i = \frac{\sum_{e \in P_i} w_i^e}{L_i}$$

Furthermore, for a pair of nodes, and for a given integer k, Yen's algorithm calculates the k shortest paths. In our case, the proposed algorithm looks for the first feasible path or stops at a maximal number of calculated paths *k_max* given as an input parameter.

The meta-code of the metrics linearization approach is given in Fig3. The algorithm begins by calculating, the shortest paths corresponding to each metric. At line 2, the parameters α_i associated to the weights of each metric are calculated. At line 3, we associate for each link e, a new mixed weight w'. Lines 5 to 10 illustrate how to calculate the next shortest path. At line 6, the i^{th} shortest path is calculated by applying the same principle as in Yen's algorithm (Fig2). At line 8, the algorithm checks the feasibility of the calculated path after initial weights are restored. If this path is feasible, the algorithm stops, otherwise the next shortest path has to be calculated.

Metrics linearization approach (G, s, t, k)

1. Calculate $P_1^*, P_2^*, \dots P_m^*$ the m shortest paths between s and t for the m métrics
2. $\alpha_i = \frac{\sum_{e \in P_i^*} w_i^e}{L_i}$: fraction between linear length of P_i^* and the constraint L_i
3. $w'(e) = \sum_{i=1}^{m} \alpha_i w_i$: for each link e of G
4. i ← 0, found_solution ← false
5. While ((found_solution ≠ true) and (i < *k_max*))
6.   Calculate the i^{th} shortest path P_i between s and t
7.   Restore the initial weights of links
8.   if P_i is feasible {found_solution ← true, return P_i }
9.   else {i ← i+1}
10.   put back the new weights w' to the links e

Fig3. Metrics linearization meta-code

VII. Performance Analysis

In this paper, we use a topology that represents an operator network with 50 nodes and 88 links [15]. For each link, we generate two uncorrelated weights using a uniform distribution on the interval [1,100]. We make this draw 100 times. For each weight draw, we choose 100 random source-destination pairs, and for each source-destination pair, we apply the three algorithms: SAMCRA, Hops count approach and Metrics linearization approach. The constraint vector L is drawn using a uniform distribution and takes its values in intervals which are predetermined for each weight distribution, and for each source-destination pair. Fig4 illustrates the draw of the constraints L_i. Let consider P_1 and P_2 two paths that minimize respectively the metrics w_1 and w_2. The gray rectangle represents the constraints space. These constraints are called strict constraints. We use 15 constraints intervals that run diagonally from the strictest constraints (rectangle A) to the less strict ones (rectangle B).

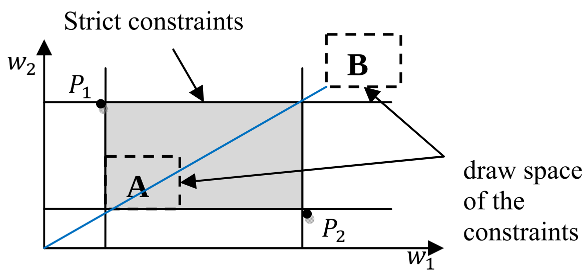

Fig4. Draw of constraints

For proposed approaches, we fix the maximum number of paths that can be calculated to three ($k \leq 3$). To evaluate the performance of the three algorithms, we analyze:

- success rate: number of satisfied requests divided by the number of generated requests
- absolute complexity: average number of *visits of nodes* [2] for all generated requests
- relative complexity: average number of visits of nodes for satisfied requests
- non-linear length of paths: used by SAMCRA (see Section III)
- average length of paths: the average sum of weights for each metric(see example MCOP problem section II)
- the number of calculated paths before finding a feasible solution: this test is only significant for our heuristics.

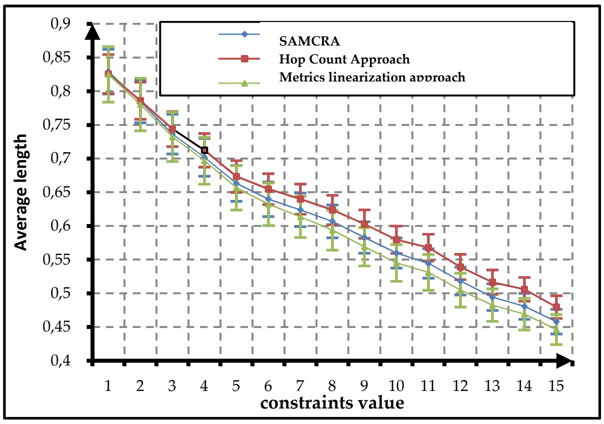

Fig5. Average length of found solutions

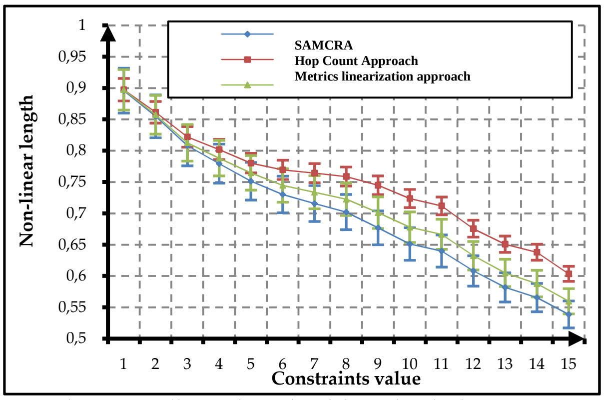

Fig6. Non-linear length of found solutions

In Fig6, the solutions found by our two approaches: hop count and metrics linearization are worse than those found by SAMCRA with 6.67% and 2.39% respectively. However, we notice that in Fig5, for average length, the solutions found by SAMCRA are worse than those found by metrics linearization approach with (1.59%), and very close to those found by the hop count approach. Indeed, the linearization of the metrics on a link involves the calculation of paths based on both two metrics, unlike SAMCRA which considers only the longest length, so do not take into account the variance between the values of the metrics. Moreover, the solutions found when the constraints are strict are very similar for the three algorithms. Indeed, the strict constraints reduce the number of feasible solutions; therefore the solutions found by our two approaches are close to optimal solutions found by SAMCRA. For less strict constraints, SAMCRA find better solutions than the other two approaches. Notice that, in order to do not bias the results, we consider only the solutions that are found by the three algorithms.

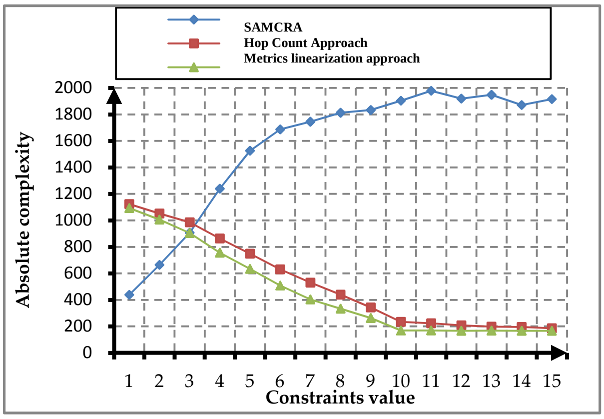

Fig7. Absolute complexity

Relative complexity is calculated only if a solution is found by the three algorithms. In Fig8, we note that the complexity of SAMCRA is significantly larger than the other two algorithms. Indeed, when the constraints are not strict, SAMCRA has more paths to explore before returning the best solution, while both approaches stop at the first feasible path they find. This reduces the execution time of our approaches. In Fig7, the absolute complexity of both proposed approaches is greater than SAMCRA's one for strict constraints. Indeed, when the constraints are strict, SAMCRA prunes faster not feasible requests, while both approaches explore the maximal number of paths (fixed to three) without finding solutions. When constraints become less strict; the two proposed approaches find feasible solutions before calculating the three paths, which reduces their complexity.

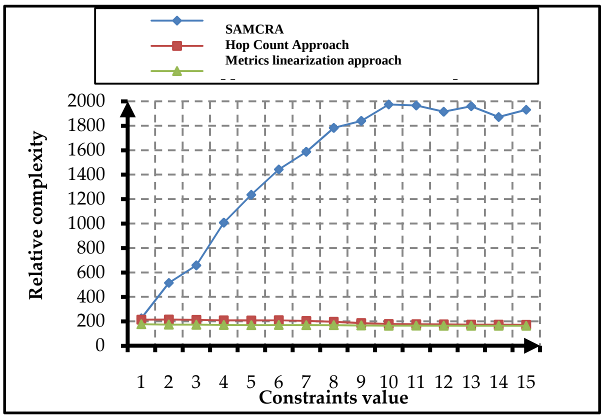

Fig8. Relative complexity

[2] One visit of node represents a fundamental operation.

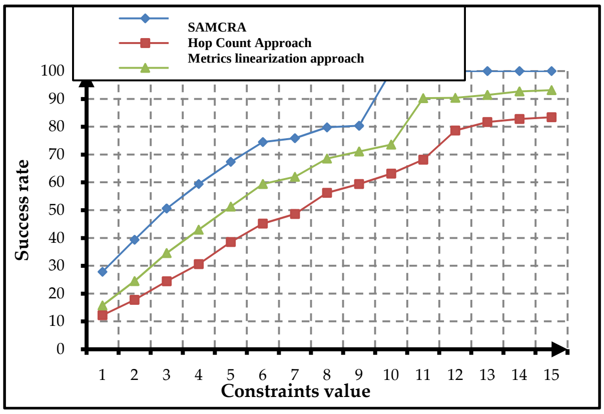

Fig9. Success rate

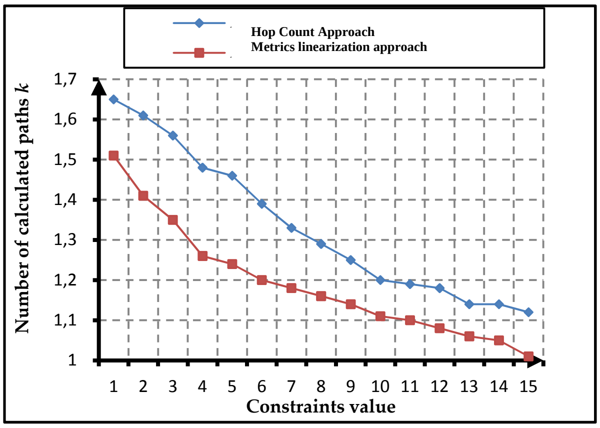

Fig10. Number of calculated paths

In Fig9, we note that the success rate of both approaches: hop count and metrics linearization approach the success rate of SAMCRA by 16% and 10% respectively. When constraints become not strict, SAMCRA achieves a success rate of 100%, the metrics linearization approach a success rate of 94%, and hop count approach a success rate of 84%. These success rates are very satisfactory.

Fig10 shows that the number of calculated paths before finding a feasible solution is less than two. This number decreases when the constraints become less strict; this explains the choice to set the number of paths to calculate at 3.

Simulation results show that, for constraints those are not strict, the two proposed approaches return solutions close to the optimum and reduce the complexity up to 10 times, making their application for Multicriteria unicast routing problems very promising.

VII. Conclusions

To solve Multicriteria routing problem, we proposed an algorithm obtained by modifying Yen's algorithm, an efficient algorithm for computing the *k* shortest paths. Two approaches have been proposed to translate the routing problem from multicriteria problem to monocriterion one. Then, we applied the proposed approaches to the arising problem. The simulation results show that the two proposed approaches provide satisfactory solutions with success rates close to the optimum found by the exact algorithm SAMCRA. Moreover, the proposed approaches calculate in average 1.43 paths before finding a feasible solution. This reduces significantly their execution time.

To resolve the multicast multicriteria routing problem, MAMCA (Multicast Adaptive Multiple Constraints Routing Algorithm) was introduced in [5]. MAMCRA uses in its first step SAMCRA, and thus inherits its complexity. Our aim will be then to include the proposed approaches in MAMCRA, and study their performance. Also, it is possible to improve these approaches by including a non-feasibility detection mechanism, so we can further reduce the complexity of the proposed algorithms.

References

[1] A. Puri, S. Tripakis, , “Algorithms for the Multi-constrained Routing Problem”, Algorithm Theory, pp. 97-112, 2002.

[2] S. Yussof, O. Hong See, “Finding Multi-Constrained Path Using Genetic Algorithm”, 2007 IEEE International Conference on Telecommunications and Malaysia International Conference on Communications, pp. 713-718, Malaysia, 14-16 May 2007.

[3] H. Lin, D. Xue-Wu, X. Jin, “Multi-Constrained Routing Based on Tabu Search”, 2007 IEEE International Conference on Control and Automation, pp. 157-161, May 30-June 1, Guangzhou, China .

[4] H. Wang, Z. Shi, A. Ge, C. Yu, “An optimized ant colony algorithm based on the gradual changing orientation factor for multi-constraint QoS routing”, Computer Communications, (32): 586-593, 2009.

[5] F. Kuipers, P. Van Mieghem, “MAMCRA: a constrained-based multicast routing algorithm”, Computer communications, (25): 802-811, 2002.

[6] P. Van Mieghem, F. A. Kuipers, “On the complexity of QoS Routing”, Computer Communications, 26: 376-387, 2003.

[7] P. Van Mieghem, H. De Neve, F. A. Kuipers, “Hop-by-hop Quality of service routing”, Computer Networks, 37(3-4): 407-423, October 2001.

[8] B. L. Fox. “*k*-th shortest paths and applications to the probabilistic networks”. In RSA/TIMS Joint National Mtg., 23: B263, 1975.

[9] D. Eppstein. Finding the k shortest paths. SIAM J. Computing, 28(2): 652–673, 1998.

[10] W. Hoffman and R. Pavley. A method for the solution of the Nth best path problem. journal of the ACM, 6: 506–514, 1959.

[11] N. Katoh, T. Ibaraki, and H. Mine. An efficient algorithm for k shortest simple paths. Networks, 12:411–427, 1982

[12] J. Y. Yen. Finding the K shortest loopless paths in a network. Management Science, Vol. 17, pp. 712–716, 1971.

[13] 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.

[14] J. M. Jeffe, “Algorithms for finding paths with multiple constaints”, Networks, 4: 95-115, 1984.

[15] A. Zimolka, “Design of Survivable Optical Networks by Mathematical Optimization”, thesis on mathematic, Berlin University, 2007.

[16] Z. Wang, J. Crowcroft, “Quality-of-Service Routing for Supporting Multimedia Applications ”, IEEE Journal of Selected Areas in Communications, 14(7): 1228-1234, 1996.