A Minimum Interference Routing Algorithm

Gustavo B. Figueiredo and Nelson L. S. da Fonseca
State University of Campinas
Brazil
Email: {gustavo, nfonseca}@ic.unicamp.br

José A. Suruagy Monteiro
Salvador University
Brazil
Email: suruagy@unifacs.br

***Abstract*— Minimum Interference Routing is instrumental to MPLS Traffic Engineering under realistic assumptions of unknown traffic demand. This work presents a new algorithm for minimum interference routing, called Light Minimum Interference Routing (LMIR). This algorithm introduces a new approach for critical link identification that reduces the computational complexity. Results, derived via simulation, show that LMIR is precise and has indeed a low computational complexity.**

I. Introduction

The ability to dynamically create Label Switched Paths (LSPs) at low operational costs makes MultiProtocol Label Switching (MPLS) a fundamental tool for traffic engineering. Route selection, dimensioning and traffic partitioning among LSPs are the major challenges for MPLS based traffic engineering.

The selection of paths which satisfy multiple independent QoS requirements is an NP-complete problem. Therefore, the solution of QoS based routing problems involves the adoption of heuristics to find feasible paths in real time.

Path selection for a given source-destination (SD) pair traffic may follow different criteria. Minimum interference routing algorithms have attempted to reduce blocking probabilities by finding edges which minimizes the maximum flow reduction between other SD pairs.

The central idea behind these algorithms is that the larger the available maximum flow is, the smaller is the blocking probability of requests for an SD pair. Therefore, minimum interference routing algorithms try to route a bandwidth request into paths which minimizes the maximum flow reduction between others SD pairs [1]. This is an NP-hard problem, and heuristics have been proposed to deal with this problem.

This paper presents a new heuristic, called *Light Minimum Interference Routing* (LMIR), which tries to optimize resource utilization with low computational complexity. As others existing minimum interference routing algorithms, LMIR is an on-line algorithm which do not make any assumption about the network traffic. Its major advantage in comparison to previous approaches is the low computational complexity which leads to greater scalability.

The rest of this paper is structured as follows. Section II introduces LMIR and analyze its asymptotical complexity. Section III presents numerical examples. Finally, conclusions are presented in Section IV.

II. The Light Minimum Interference Routing Algorithm

Existing minimum interference routing algorithm are based on max-flow type of solution. The most common maxflow computation method is *Ford-Fulkerson*'s [2], and its mostly known implementation, *Edmonds-Karp* algorithm, uses a breadth search to find the augmenting paths. Its complexity is $O(V \cdot E^2)$. For networks such as the Internet, the average degree is around 3.5 [3], i.e., the number of edges is significantly larger than the number of vertices, which makes *Edmonds-Karp*'s algorithm perform near the worst case (dense graphs).

The fastest maxflow algorithm is Goldberg's algorithm [4] with complexity $O(\min(V^{2/3}, E^{1/2}) \cdot E \cdot \log(V^2/E) \cdot \log(U))$ for networks with $|V|$ vertices, $|E|$ edges with capacities in the interval [1,U] [4]. Even though Goldberg's algorithm leads to a significant reduction in computational complexity, other approaches that avoid the use of maxflow can be used to reduce even more the complexity of minimum interference algorithms.

In this section, a new minimum interference routing algorithm, called *Light Minimum Interference Routing (LMIR)*, that does not involve the execution of maxflow algorithms, is presented.

LMIR assumes the existence of signaling protocols (e.g., RSVP-TE or CR-LDP) responsible for the updating of the information about residual bandwidth, $R(l)$, and topology changes. LMIR attempts to find the K paths with the lowest capacities among all SD pairs in order to determine the critical edges. LMIR neither assumes any knowledge about future requests nor any statistical traffic profile.

LMIR seeks the paths which minimize the interference of SD pairs, generating a balanced utilization of resources. Upon receiving a request for bw bandwidth units for a SD pair (s, d), $ri(s, d, bw)$, LMIR finds the K critical paths containing at least K critical edges for the other SD pairs. Note that a given lowest capacity path may contain more than one critical edge. This search for critical edges is performed by finding the K paths with the lowest capacities, since the path flow is limited by the minimum capacity edge.

After finding K paths with the lowest capacities, weights, computed according to Equation (1), are assigned to all of their edges. In this way, weights inversely proportional to

the residual capacities are assigned to the critical path edges. Lastly, Dijkstra's algorithm using $w(l)$ as weights is executed for selecting non-critical edges or edges with low criticality. The LMIR algorithm is presented in Algorithm 1.

Algorithm 1 LMIR

INPUT

A residual graph $G = (V, E)$ and $ri(s, d, bw)$, a request for bw bandwidth units between pair (s, d).

OUTPUT

A path (route) connecting s to d with bw bandwidth units.

LMIR

1: Find the paths with the K lowest capacities $\forall(s', d') \in \delta$.
2: Compute the weights according to equation (1) for all the edges belonging to the paths found.
3: Eliminate all the edges with residual capacities less than bw.
4: Run *Dijkstra*'s algorithm using $w(l)$ as the weights.
5: Create an LSP connecting s to d and update the edge capacities.

Edge weights are computed as

$$w(l) = \sum_{(s',d') \in P} \frac{f_G^{(s',d')}}{c(u,v) \cdot R(l)}, \qquad l \in E \tag{1}$$

where $R(l)$ is link l residual capacity, $f_G^{(s',d')}$ is the flow of a least capacity path between s' and d' and $c(u, v)$ is the capacity of the edge (u, v).

The first step of LMIR is the search for the paths with the lowest capacities (where δ is the set of SD pairs). This is done by a variation of Dijkstra's algorithm, called "LowestCapacities" (Algorithm 2).

Algorithm 2 LowestCapacities

1: **for all** $(v \in |V|)$ **do**
2: $D[v], di[v] = \infty$
3: $\pi[v] = NIL$
4: D[s], di[s] = 0.0
5: $Q \leftarrow s$
6: **while** (Q) **do**
7: $u \leftarrow extract\_min(Q)$
8: **for all** $v \in ADJ[u]$ **do**
9: $\gamma = min[c(u, v), D[u]]$
10: **if** $[(\gamma < D[v]) \vee ((\gamma == D[v]) \wedge (di[u] < di[v]))]$ **then**
11: $D[v] = \gamma$
12: $di[v] = di[u] + 1$
13: $\pi[v] = u$
14: $Q \leftarrow Q \bigcup v$

In LowestCapacities, v represents any network vertex, D is a vector which contains the minimum capacity found in any path from s to v. Vector π contains v's predecessor vertex and vector di stores the distance from s to v in number of edges. Q represents the list of vertices which adjacencies were not yet visited. Function *extract_min(Q)* returns the element $u \in Q$ for which value $D[u]$ is the lowest one and $ADJ[v]$ represents the adjacency list of vertex v.

The complexity analysis of LMIR is as follows. Step 6 of the *LowestCapacities* algorithm is executed $|V|$ times, since all vertices are visited. Step 8 is also executed $|V|$ times resulting in $O(V^2)$ complexity. Moreover, *extract_min(Q)* has complexity $O(E)$ in the worst case, when the queue of vertices with adjacencies not yet visited is implemented as a linked list. Therefore, LMIR complexity is $O(V^2 + E) = O(V^2)$ [2]. Since each pair executes this algorithm K times, the complexity of identifying critical edges is $O(K \cdot V^2) = O(V^2)$. The other minimum interference algorithms, evaluated in this paper, have $O(\min(V^{2/3}, E^{1/2}) \cdot E \cdot \log(V^2/E) \cdot \log(U))$ complexity in the critical edges identification step.

In order to show that the complexity of *LowestCapacities* is lower than the complexity of Goldberg et al.'s, which is the fastest maxflow algorithm known, the complexity analysis of these algorithms are derived for dense and sparse graphs.

For dense graphs it is assumed that $|E| = |V|^2$. Therefore, Goldberg's algorithm has complexity

$$O((V^{2/3}) \cdot V^2 \cdot \log(V^2/V^2) \cdot \log(U))$$

while *LowestCapacities* has complexity $O(V^2)$. Assuming that the term $\log(V^2/V^2)$ has omitted other terms; since the complexity cannot be zero, and $\log(V^2/V^2) \cdot \log(U) > 1$, it follows that

$$\text{Goldberg's } \textit{maxflow} = O(V^{2/3} \cdot V^2 = V^{8/3}), \text{ and}$$

$$(V^{8/3} > V^2) = \text{LowestCapacities.}$$

Hence, $V^2 = O(V^{8/3}) \Rightarrow$ LowestCapacities $= O($Goldberg's *maxflow*$)$ □

For sparse graphs it is assumed that $|E| = |V|$. In this case, Q can be implemented as a heap [2]. Therefore,

$$\textit{LowestCapacities} = O(V \cdot \log V),$$

since step *extract_min(Q)* has complexity $O(\log V)$. Besides, assuming that $\log(V^2/E) \cdot \log(U) > 1$, taking the first two factors of $O(\min(V^{2/3}, E^{1/2}) \cdot E \cdot \log(V^2/E) \cdot \log(U))$, it can be shown that

$$\text{Goldberg's maxflow} = V^{1/2} \cdot V, \text{ and}$$

$$\textit{LowestCapacities} = O(V \cdot \log V).$$

Therefore, discarding factor V in both algorithms and taking the log

$$\textit{LowestCapacities} = \log(\log V), \text{ and}$$

$$\text{Goldberg's maxflow} = \log V^{1/2} = 1/2 \cdot \log V.$$

Hence, $\log(\log(V)) = O(1/2 \cdot \log V) \Rightarrow$ *LowestCapacities* $=$ $O($Goldberg's maxflow$)$□

These results show that for both dense and sparse graphs, the critical edges detection in LMIR is less expensive than in the algorithms that use maxflow.

The proper choice of K is of paramount importance for the achievement of good performance, as will be explained in the next section. This happens because when the algorithm searches for the K least capacity paths for a given source-destination pair, it tries to find out the critical edges, which in the best case correspond to the minimal cut edges. So, an inadequate value of K can lead to a large number of rejections. Although it is not possible to establish an optimum value, Theorem 1 establishes an upper bound for K.

Theorem 1: Let $K(\theta)$ be an upper bound to K, $c(u,v)$ the capacity of the edge (u,v), $\omega^{(u,v)}$ the flow of the least capacity path between u and v, $\theta_{(G)}^{(u,v)}$ the maxflow between u and v in G, we have:

$$K(\theta) = d(u), \text{ if } \theta^{(u,v)} = \sum_{\forall i | (u,i) \in E(G)} c(u,i) \qquad (2)$$

or

$$K(\theta) = d(v), \text{ if } \theta^{(u,v)} = \sum_{\forall i | (i,v) \in E(G)} c(i,v) \qquad (3)$$

or

$$K(\theta) = \left\lceil \frac{\theta^{(u,v)}}{\omega^{(u,v)}} \right\rceil . \qquad (4)$$

Proof: See appendix. ■

Note that to compute $K(\theta)$ it is necessary to execute a max-flow algorithm once whereas for the other non-interfering algorithms a max-flow algorithm should be computed at every request.

Based on numerous simulation experiments pursued for the evaluation of LMIR, it was observed that the best performance and precision are obtained when $K = K(\theta)$ for $K < 5$ and K is in the interval $[5, 8)$ for $K(\theta) \geq 5$.

III. Numerical Examples

Simulation experiments were carried out to verify the precision and performance of LMIR. Experiments with small and large networks were pursued. Small networks were used to compare to previously published results for both the *Minimum Interference Routing Algorithm*, MIRA [1] and the algorithm presented in [5], WSC. The network topology as well as SD pairs are shown in Figure 1. SD pairs are the only network input and output traffic. Lighter links have capacity of 1200 bandwidth units, whilst the darker ones have 4800 bandwidth units capacity, representing OC-12 and OC-48 rates, respectively.

Links are bi-directional, i.e., they represent two links with the same capacity in opposite directions. Requests were randomly generated using the uniform distribution in the interval $[1, 4]$.

Two other algorithms that do not take interference into account were also investigated so that the benefits of the non-interfering approach can be highlighted. The Minimum Hop Algorithm (MHA) is an implementation of Dijkstra's algorithm while the Widest Shortest Path Algorithm (WSP) finds the maximum capacity path in the network. If there is more than one path with the same capacity, WSP returns the path with the least number of edges.

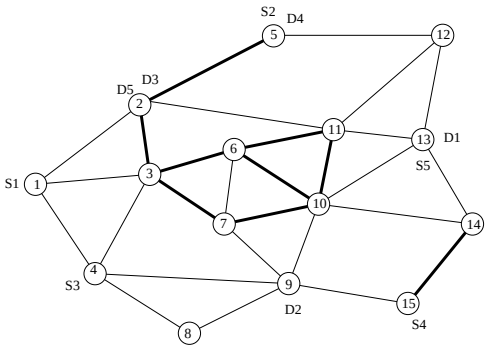

Fig. 1. Topology used in the simulations.

It is assumed that LSPs have long lives, i.e., once accepted its resources are kept until the end of the experiments. 8,000 requests were randomly generated among the five SD pairs shown before.

Figure 2 shows the reduction of total available bandwidth as the number of accepted requests. It can be seen that minimum interfering routing algorithms produce the lowest reduction rate for the total flow. WSP and MHA produce steeply reductions since they use critical edges indiscriminately, causing the reduction of the maxflow of several SD pairs, which demonstrates the importance of the minimum interference approach. In this example, LMIR produces a flow reduction rate smaller than the reduction produced by WSC and it is equal to the one produced by MIRA up to 3,000 requests. From this point on, LMIR has decreased slightly more accentuated than both of them until 5,000 requests. After that, the behavior of all the algorithms are about the same, almost linear.

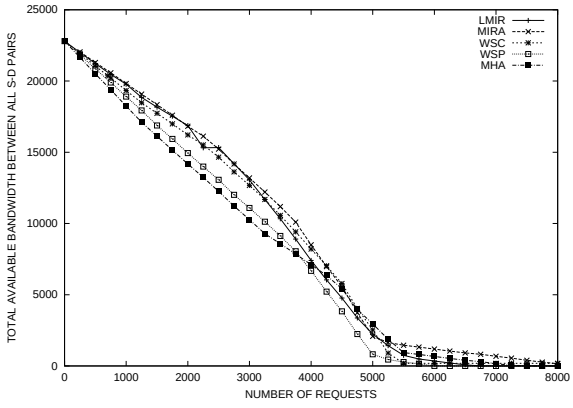

Fig. 2. Total bandwidth among all network source-destination pairs.

Figure 3 shows the number of requests rejected as a function of the total number of requests which arrived to the network. MHA is the algorithm that rejects the highest number of requests. MHA always chooses the shortest path, saturating

TABLE I

PARAMETERS USED IN GRAPHS RANDOM GENERATION.

	$\|V\|$	Type	α	β	L	$\|E\|$	$K(\theta)$
1	30	Dense	0.4	0.4	141	222	13
2	30	Sparse	0.2	0.2	141	118	7
3	40	Dense	0.4	0.4	141	354	13
4	40	Sparse	0.2	0.2	141	170	3
5	50	Dense	0.4	0.4	141	544	15
6	50	Sparse	0.2	0.2	141	302	5

rapidly the edges that belong to this path. WSP starts rejecting connections around 5,000 requests. However, because it does not take into consideration edge criticality, it can choose paths that have critical edges from various pairs causing their saturation. LMIR and MIRA produce the lowest number of rejected requests.

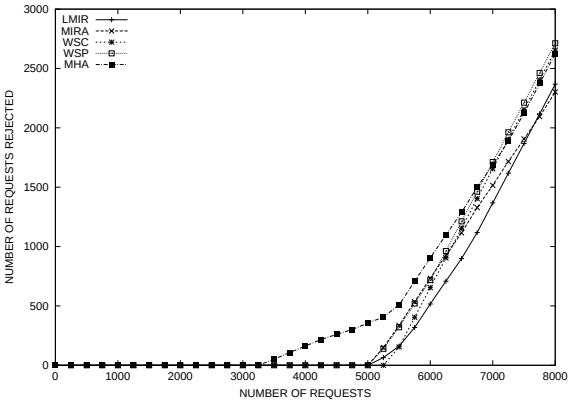

Fig. 3. Number of requests rejected.

The time spent running LMIR is 2.2 second lower than the time to run WSC and MIRA for $K = 1$ and is the same for $K = 8$. WSP and MHA have execution times smaller than the non-interfering algorithms. However, they cause an excessive number of rejections. The evaluation of the execution time of non-interfering algorithm as well the choice of the K value are presented next.

Large networks with 30, 40, and 50 vertices were also used in the experiments. Both sparse and dense networks were investigated. The topology of these networks were randomly created using Waxman's method [3], [6] and the parameters used are summarized in Table I, where $\alpha, \beta \leq 1$ are model parameters.

Bidirectional edge capacities, the number of requests and the bandwidth per request were generated as in the previous example. 30,000 requests were randomly generated among the five source-destination pairs.

Results for networks with 50 nodes (Table I) are presented here. Results for the other networks are omitted due to space limitation and they imply in similar conclusions.

The 50 vertices dense networks have 544 edges which implies in a large number of paths. In order to obtain an overload scenario, all edges were assigned capacities of 1,200 bandwidth units.

It can be seen in Figure 4 that the behavior of WSC, MIRA, and LMIR are quite similar up to 20,000 requests. Only after this point MIRA achieves a slightly smaller reduction in the maximum flow as compared to WSC and LMIR algorithms.

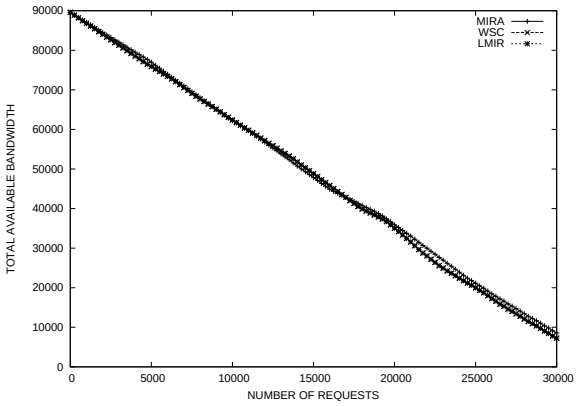

Fig. 4. Total available bandwidth (50 vertices dense networks).

Figure 5 presents the total number of rejections. Note that the first request rejected by LMIR occurs only after 17,500 requests, while the first reject request by WSC and by MIRA occur around 16,000 requests. MIRA and WSC reject a larger number of requests than LMIR in the interval $[16,000; 25,000]$. However, after that, the three algorithms behaves alike and at the end of the experiment they present a very close number of rejected connections.

Figure 6 illustrates the fact that the quality of LMIR depends on the choice of K. As can be seen the best performance was achieved for $K = 7$ ($K \in [5, 8]$).

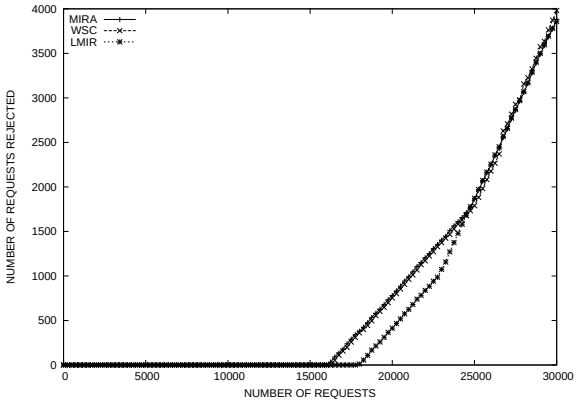

Fig. 5. Number of rejected requests (50 vertices dense networks).

The decrease in the available bandwidth was analyzed for all pairs and no interference was observed.

Figure 7 shows the reduction of the maximum flow in sparse networks with 50 vertices. It can be clearly seen a smaller reduction in the maximum flow achieved by LMIR with respect to WSC and MIRA. After 4,000 requests approximately, WSC presents a larger reduction than MIRA and LMIR, which

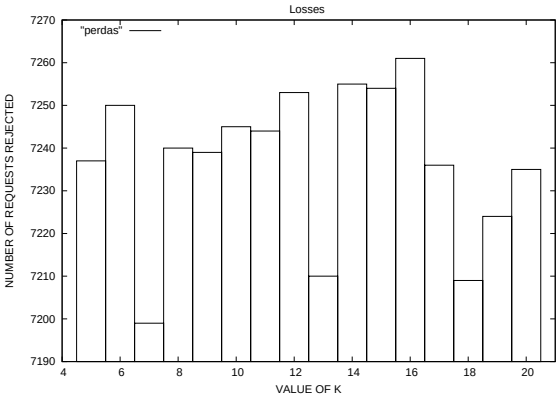

Fig. 6. Number of losses as a function of K (50 vertices dense networks).

behave in a similar way up to 6,000 requests. At this point one can see a slower reduction in the available bandwidth, part of LMIR up to 14,000 requests when the maximum flow becomes null.

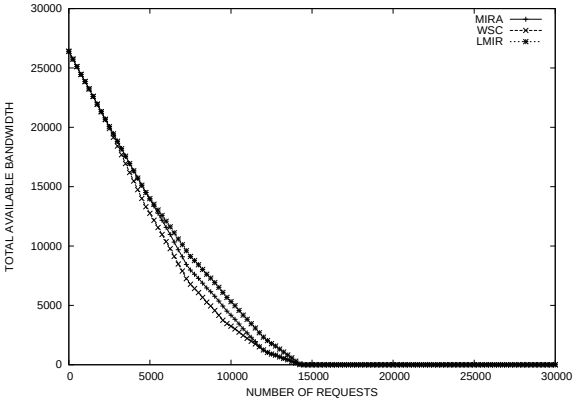

Fig. 7. Total available bandwidth (50 vertices sparse networks).

Figure 8 shows the number of requests rejected by WSC, MIRA and LMIR algorithms. It can be seen that the first requests are rejected right after 5,000 requests. The algorithms exhibit a very close behavior in the interval $[5,000; 10,000]$. LMIR in this experiment rejects a smaller number of requests (approximately 19,500). The second best algorithm in this experiment was MIRA with 20,000 rejected requests, followed by WSC with a little less than 20,500 rejections.

The decrease in the available bandwidth for all SD pairs was recorded and again no interfering was observed.

Table II presents the algorithms execution times for the networks used in the experiments. The values were obtained using Linux's *time* command, in Intel Celeron machines with a 1.2 GHz clock, and 256 MB of RAM. In the table, results are presented results for LMIR for K between 5 and 10. After that, the execution time difference between LMIR and the others becomes greater than 40 seconds. It can be seen that K should be chosen in the interval $[5, 8)$.

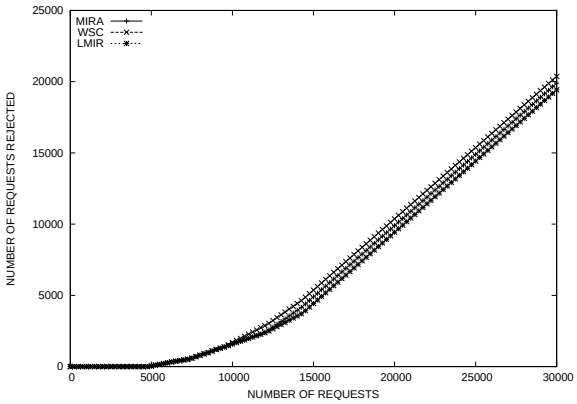

Fig. 8. Number of rejected requests (50 vertices sparse networks).

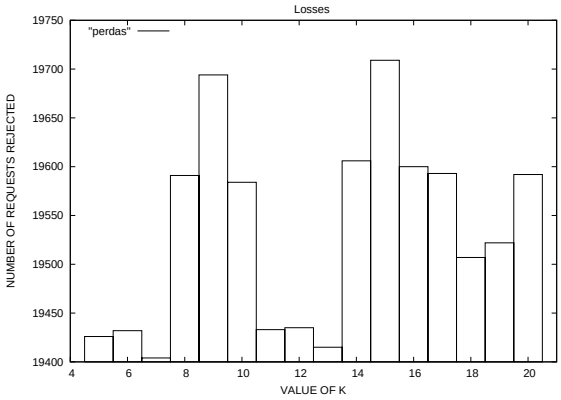

Fig. 9. Number of losses as a function of K (50 vertices sparse networks).

IV. Conclusions

In this paper, a new minimum interference routing algorithm which does not use maxflow for critical edges detection was presented. Modifications in Dijkstra's algorithm were pursued in order to find the paths with lowest capacities. After finding K critical edges, weights are assigned to the edges and an algorithm to find the shortest path is applied.

Results obtained via simulation point out that the LMIR produces similar outcomes for dense large networks than WSC and MIRA and slightly better performance for sparse networks.

TABLE II
EXECUTION TIMES FOR 30,000 REQUESTS

	50 (D)	50 (S)	40 (D)	40 (S)	30 (D)	30 (S)
MIRA	3m24.97s	2m50.48s	2m25.53s	2m52.19s	12m2.37s	2m54.30s
WSC	3m20.41s	2m52.49s	2m27.35s	2m52.87s	2m5.84s	2m34.61s
K=5	2m5.70s	2m23.50s	2m8.19s	1m49.87s	2m0.36s	2m19.61s
K=6	2m23.48s	2m31.34s	2m19.67s	2m13.73s	2m3.73s	2m31.44s
K=7	2m27.64s	2m43.83s	2m36.94s	2m21.71s	2m9.45s	2m34.12s
K=8	2m30.27s	2m47.92s	2m39.94s	2m54.20s	2m16.37s	2m52.71s
K=9	3m4.96s	2m53.27s	3m0.13s	3m3.33s	2m26.43s	3m6.29s
K=10	3m28.76s	3m2.53s	3m3.19s	3m18.70s	2m37.87s	3m18.67s

ACKNOWLEDGMENT

This work was partially sponsored by CNPq.

REFERENCES

[1] M. S. Kodialam and T. V. Lakshman, "Minimum Interference Routing with Applications to MPLS Traffi c Engineering," in *INFOCOM (2)*, 2000, pp. 884–893.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. MIT Press and McGraw Hill, 1990.

[3] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *IEEE Infocom*, vol. 2. San Francisco, CA: IEEE, March 1996, pp. 594–602.

[4] A. V. G. e Satish Rao, "Beyond the fbw decomposition barrier," in *J. ACM 45 (5)*, 1998, pp. 783–797.

[5] X. S. B. Wang and C. Chen, "A New Bandwidth Guaranteed Routing Algorithm for MPLS Traffi c Engineering," in *Proceedings of IEEE International Conference on Communications*, 2002, pp. 1001–1005.

[6] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, 1988.

[7] D. B. West, *Introduction to Graph Theory*, P. Hall, Ed. Prentice Hall, 1996.

APPENDIX

Definition [7] 1: Let $G = (V, E)$ be a graph and let $V(G)$ and $E(G)$ be its vertex and edge sets, respectively. A **disconnecting set** is a set $F \subseteq E(G)$ such that $G-F$ has more than one component. A graph is said to be k-edge-connected if every disconnecting set has at least k edges. The edge-connectivity, $\kappa'(G)$, is the minimum size of a disconnecting set.

Theorem 2: Let $\lambda(u, v)$ be the number of u, v-internally disjoint paths (or just disjoint paths) then

$$\lambda(u, v) \leq min\{d(u), d(v)\},$$

where $d(u)$ and $d(v)$ are the degrees of u and v, respectively.

Proof: (**Outline**) Let P and P' be two internally disjoint paths between u and v. Now, suppose by contradiction that, $\lambda(u, v) > \min\{d(u), d(v)\}$. If $\lambda(u, v) > \min\{d(u), d(v)\}$, at least one edge incident to u or to v was counted in two internally disjoint paths, P and P'. Thus, if P and P' have an edge in common, they aren't internally disjoint, contradicting the hypothesis. ■

Theorem (Menger-1927) 3: Let u and v be two vertices of a graph G and $(u, v) \notin E(G)$. Then, $\max(\kappa'(G)) = \min(\lambda(u, v))$.

Proof: See [7]. ■

Although Menger's Theorem suggests an upper bound for K, critical edges may not be identified when the number of disjoint paths is lower than the number of critical edges. Few changes may be done in G in order to use this theorem. Given a graph G with N integer capacities, we form a graph G' splitting each edge of capacity x into x edges with the same endpoints and capacity 1 unit of flow.

Thus, a new theorem relating maximum flow and Menger's theorem (Theorem 3) is presented:

Theorem 4:

$$\lambda(G') \geq \theta^{uv} = \min cap(S, T) \geq \kappa'(G').$$

Where $\lambda(G')$ is the number of internally disjoint paths between u and v, θ^{uv} is the maxflow between u and v, $cap(S, T)$ represents the capacity of a cut (S,T) such that $u \in S$ and $v \in T$, and $\kappa'(G)$ is the minimum size of a disconnecting set of G.

Proof: See [7]. ■

Theorem 5: Let $f_G^{(u,v)}$ be a flow of a least capacity path between u and v in G, thus $n = f_G^{(u,v)} = \sum_{i=1}^{n} f_{G'}^{(u,v)}$.

Proof: In G' each edge has capacity of 1 unit of flow. Thus, in order to send n units of flow, we have to use n distinct paths. Furthermore, the capacity of a path in G is limited by the flow of the least capacity edge in that path. Since G' is built from G, using a R capacity edge in G is equivalent to using R edges with the same endpoints in G'. Thus the equality holds. ■

Theorem 6: Let $K(\theta)$ be an upper bound to K, $f_G^{(u,v)}$ be a flow of a least capacity path between u and v in G, $\omega_{(G)}^{(u,v)}$ be the flow of the least capacity path between u and v in G, and $\theta_{(G)}^{(u,v)}$ be the maxflow between u and v in G, we have:

$$K(\theta) = d(u), \text{ if } \theta^{(u,v)} = \sum_{\forall i | (u,i) \in E(G)} c(u, i) \tag{5}$$

or

$$K(\theta) = d(v), \text{ if } \theta^{(u,v)} = \sum_{\forall i | (i,v) \in E(G)} c(i, v) \tag{6}$$

or

$$K(\theta) = \left\lceil \frac{\theta^{(u,v)}}{\omega^{(u,v)}} \right\rceil \tag{7}$$

Proof:

Case 1:

$$\theta^{(u,v)} = \sum_{\forall i | (u,i) \in E(G)} c(u, i).$$

The Max-flow Min-cut theorem [7] implies that the maximum network flow is bounded by the minimum capacity of a source/sink cut. So, if

$$\theta^{(u,v)} = \sum_{\forall i : (u,i) \in E(G)} c(u, i)$$

the maximum network flow is bounded by the capacities of the edges incident to u.

Case 2:

$$\theta^{(u,v)} = \sum_{\forall i : (i,v) \in E(G)} c(i, v)$$

Using a reasoning analogous to case 1.

Case 3:

Let

$$n = \theta_{(G')}^{(u,v)} = \theta_{(G)}^{(u,v)}$$

it is known that all paths have capacity 1 in G'. Therefore,

$$n = \theta_{(G)}^{(u,v)} = n \cdot f_{G'}^{(u,v)} = \sum_{i=1}^{n} f_{G'}^{(u,v)}$$

making ω and $K(\theta)$ such that $\omega \cdot K(\theta) \leq n$ we have that

$$n = \theta_{(G)}^{(u,v)} \geq K(\theta) \cdot \sum_{i=1}^{\omega} f_{G'}^{(u,v)}. \tag{8}$$

However, using Theorem 5 in Equation 8 the following holds

$$K(\theta) \cdot \omega_{(G)}^{(u,v)} \leq \theta_{(G)}^{(u,v)}. \tag{9}$$

Since in G both $\theta_{(G)}^{(u,v)}$ and $\omega_{(G)}^{(u,v)}$ are known, we can define $K(\theta)$ as

$$K(\theta) = \left\lceil \frac{\theta^{(u,v)}}{\omega^{(u,v)}} \right\rceil.$$

■