A Minimum Interference Routing Algorithm with Reduced Computational Complexity

Gustavo B. Figueiredo, [a] Nelson L. S. da Fonseca [a]
José A. S. Monteiro [b]

[a] *Institute of Computing, State University of Campinas, Brazil*
[b] *NUPERC, Salvador University, Brazil*

Abstract

Minimum Interference Routing algorithms are designed to reduce rejections of future requests for the establishment of Label Switched Paths (LSPs) but make no assumption about specific patterns of arrival request. This paper introduces a novel minimum interference routing algorithm, Light Minimum Interference Routing (LMIR), which is based on a new approach to the identification of critical links. This approach reduces the computational complexity involved in finding a path for the establishment of an LSP. The LMIR is shown to have the same precision as existing algorithms but with less computational complexity.

Key words: Traffic Engineering; MPLS; Minimum Interference Routing.

1 Introduction

In spite of the increase in Internet link capacity, congestion is still common. One of the main reasons for this is an unbalanced use of resources, and Traffic Engineering is often employed to balance this use by mapping flows onto network links. MultiProtocol Label Switching (MPLS) [1] is of paramount importance to Traffic Engineering [2,3], since it promotes path establishment, thus guaranteeing the Quality of Service (QoS) required by networks flows [4].

The use of traditional Internet routing algorithms based on the shortest path in MPLS networks leads to a rapid saturation of network links. As a conse-

Email addresses: gustavo@ic.unicamp.br (Gustavo B. Figueiredo,), nfonseca@ic.unicamp.br (Nelson L. S. da Fonseca), suruagy@unifacs.br (José A. S. Monteiro).

quence, new alternative algorithms, among them minimum interference routing algorithms, have been proposed to promote a more balanced utilization of resources. The criterion for path selection, adopted by these algorithms, considers those paths which will result in the least impact on the rejection of future requests. However, such a criterion, with multiple independent QoS requirements, leads to NP-complete problems [5–7]. As a consequence, various heuristic algorithms have been proposed to deal with this criterion [8–13].

Most minimum interference algorithms are based on the use of the maximum flow to identify critical links, since these links, which belong to the minimum cut set of a pair of nodes, are what is responsible for a reduction in the maximum flow between that pair of nodes. In such algorithms, a max-flow algorithm is typically executed for all ingress/egress node pairs of a network domain for each LSP establishment request, although this increases the computational complexity of these algorithms.

This paper introduces a new heuristic, the *Light Minimum Interference Routing* (LMIR) algorithm [14], which guarantees minimum interference, but without the need for the execution of repeated maxflow algorithms. Instead, this algorithm uses a modified version of Dijkstra's algorithm, which is less computationally complex than maxflow algorithms, while producing similar results. The LMIR algorithm is also independent of the pattern of arrival of LSP establishment requests.

The rest of this paper is structured as follows. Section 2 introduces the definition of the minimum interference routing problem and describes existing algorithms. Section 3 introduces the Light Minimum Interference Routing (LMIR) algorithm. Section 4 evaluates the performance of the algorithm proposed via simulations. Finally, Section 5 presents the conclusions.

2 Minimum Interference Routing Algorithms

Prior to the presentation of the minimum interference problem itself, certain relationships must be defined. Table 1 summarizes the mathematical symbols used in the paper.

Let the graph $G = (V, E)$ represent a network, where $V(G)$ is the set of vertices (or nodes) and $E(G)$ the set of edges (or links). The flow, $f(u, v)$, is the amount of data transmitted between nodes u and v through a link. The capacity of a link connecting a pair of nodes, $c(u, v)$, is the maximum amount of flow that can pass through this link. The residual capacity of a link, $c_r(u, v)$, is the amount of data that, when added to the current flow in a link, results in the maximum flow allowed in that link. The graph $G_r(V, E_r)$, composed of the

nodes of the graph G and the set of links E_r with non-null residual capacity, constitutes a residual graph.

The capacity of a path is the flow that can be accommodated by the links composing that path. Thus, the least capacity link of a path determines the capacity of the entire path. Moreover, the link of least capacity between the pair of nodes s and d belongs to the path of least capacity between these two nodes. This path of least capacity is represented by $f^i_{(s,d)}$ with $f^0_{(s,d)}$ being the actual path of least capacity, $f^1_{(s,d)}$ being that of the next smallest capacity and so on.

The path with the i^{th} smallest capacity is denoted by $f^i_{(s,d)}$, i.e., $f^0_{(s,d)}$ is the least capacity path between $s-d$, $f^1_{(s,d)}$ is the path with the second smallest capacity and so on. The maximum flow ($\theta^{(s,d)}$) is the largest amount of flow that can go through any path connecting source and destination nodes in G.

The problem to be solved by minimum interference routing algorithms is the choice of a path for the establishment of an LSP which minimizes the reduction in maximum flow of other source-destination (S-D) pairs. The aim is to increase the number of paths which will be available to accommodate future requests for LSP establishment. The minimum interference routing problem can be stated as follow:

"Find a path for the establishment of an LSP from a source node to a destination node such that each link along the path has a residual capacity value of at least the requested amount of bandwidth. The chosen path should minimize the reduction of the maximum network flow, with the condition that flow splitting is prohibited".

Four major algorithms have been proposed for dealing with this minimum interference routing and will be presented in this section: the Minimum Interference Routing Algorithm (MIRA), *Wang, Su and Chen's* algorithm (WSC), the Residual Network and Link Capacity (RNLC) algorithm and *Kumar, Kuri, and Kumar's* algorithm.

2.1 The Minimum Interference Routing Algorithm

The maximum flow of an S-D pair is reduced whenever the available bandwidth of a link belonging to the minimum cut set is reduced. The Minimum Interference Routing Algorithm (MIRA) [15] avoids links belonging to the mincut set of other S-D pairs when establishing an LSP between a pair of nodes. It also assumes that the network topology and the residual bandwidth of the links are known at the time of LSP establishment.

Table 1
Table of used symbols

Symbol	Meaning
G	Graph representing the network.
V(G)	Set of vertices of G.
E(G)	Set of links of G.
$l = (u, v)$	Link connecting u to v.
$c(u, v)$	Capacity of the link (u, v), i.e., the maximum flow that can pass through the link (u, v).
$f(u, v)$	Flow transmitted through the link (u, v).
$c_r(u, v)$	Residual capacity of the link (u, v), i.e., the difference between the capacity of the link (u, v) and the amount of flow being transmitted over it.
P	Set of source-destination pairs of G.
$\theta^{(s,d)}$	Maximum flow between s and d, i.e., maximum amount of flow that can be transmitted from a source node s to a destination node d.
$f^0_{(s,d)}$	Amount of flow along the least capacity path in G between the source-destination pair (s, d).
$f^i_{(s,d)}$	Amount of flow along the *i-th* least capacity path between the source-destination pair (s, d).
$w(u, v)$	Weight of the link (u, v).
$r(s, d, bw)$	Request of bw bandwidth units from s to d.
α_{sd}	Weight of the source-destination pair (s, d) .
C_{sd}	Set of critical links (Specific to the MIRA algorithm).
$f^{s'd'}_{(u,v)}$	The contribution of the link (u, v) to the maximum flow of the source-destination pair (s', d') (Specific to the WSC algorithm).
N_c	$\sum_{(u,v)\in E_r} c_r(u, v)$ represents the sum of all residual capacities in G
β	Determines whether or not the algorithm for the shortest path should be used (Specific to the RNLC algorithm).
$K(\theta)$	Upper bound for the number of least capacity paths to be identified by the LMIR algorithm.
$\lambda(s, d)$	Number of internally disjoint paths between s and d, i.e. , the number of paths which have no link, except the initial and final ones, in common.
$dg(v)$	Degree of a vertex v ,i.e, number of links incident to v in G.
$\kappa'(G)$	Minimum size of a disconnecting set of G, i.e, the smallest number of links which disconnect G by their removal.
$C(S, D)$	Capacity of a cut $S - D \vert s \in S \wedge d \in D$, representing the sum of all edges belonging to that cut.

MIRA consists of three steps, the first involving the computation of the maximum flow between all S-D pairs, and the second involving the identification of critical links and the assignment of weights representing priorities in relation to other S-D pairs. These weights are computed according to the following equation:

$$w(u,v) = \sum_{(s,d)|(u,v)\in C_{sd}} \alpha_{sd}, \qquad \forall (u,v) \in E, \tag{1}$$

The third step of the algorithm involves the execution of the Dijkstra algorithm using $w(u,v)$ as weights.

Kodialam and Lakshman [15] have shown that the number of rejections resulting from the use of MIRA is lower than when such minimum interference criteria are not used. The improvement is due mainly to the avoidance of links which would reduce the maximum flow of other pairs as this limits the rejection of future requests. Evidence of the impact of a reduction in maximum flow on blocking probability was presented in their seminal paper [15], which showed that their algorithm produces lower blocking probability than do algorithms which do not take interference into account.

2.2 The Wang, Su and Chen Algorithm

The algorithm proposed by *Wang, Su and Chen* (WSC) [16] is based on MIRA but adopts a different criterion for the identification of critical links in the computation of weights.

According to Wang et al. [16], the major drawback of MIRA is that critical link identification focuses on a single S-D pair and does not detect the critical nature of a link for a group of pairs. Such a drawback can potentially lead to the denial of various requests, especially in networks with concentrating vertices.

In Figure 1, a graph with concentrating vertices is presented. Suppose that $n+1$ requests arrive to the pairs (S_0, D_0), (S_1, D_1),..., (S_n, D_n). The first request demands an LSP with n bandwidth units, while all the rest require LSPs with a single bandwidth unit.

When the first request arrives, MIRA computes the maximum flow and identifies the critical links for all other source-destination pairs, i.e., (S_1, D_1),..., (S_n, D_n). The critical links after the arrival of this first request and, therefore, the only ones with $w(u,v) \neq 0$ will be (S_1, C),..., (S_n, C), since none of the other links are in the minimum cut set of any source-destination pair. Thus, the weights for all the links of the four available paths connecting S_0 to D_0 will be the same. The chosen path would be the one passing through one of the black vertices (C or D in the figure), since these involve the fewest hops.

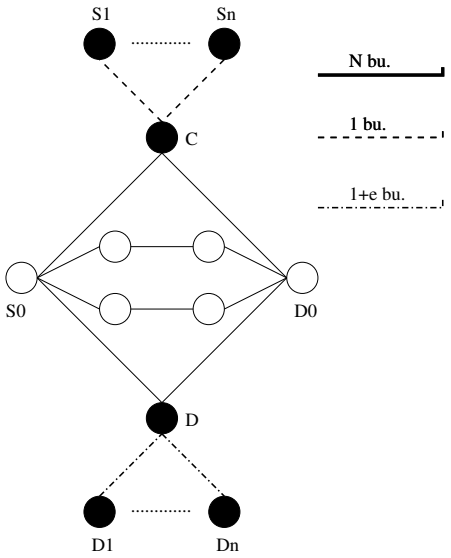

Fig. 1. Graph with concentrating vertices.

Once an LSP using this path is established, its residual link capacities are updated to 0, and no subsequent request will be accepted.

In order to avoid the use of paths containing concentrating vertices, Wang et al., proposed an algorithm with a weight function that takes into consideration the contribution of a link to the maximum flow. There are two cases to be considered. In the first, the link contribution to the maximum flow is less than its residual capacity, which means that the link does not belong to the minimum cut set and will, therefore, receive smaller weighting. In the second case, the link contribution is equal to its maxflow contribution. In this case, the link belongs to the min cut set. Higher weighting are assigned to these links.

In their algorithm, link weights are computed as

$$w(u,v) = \sum_{(s',d') \in P} \frac{f_{(u,v)}^{s'd'}}{\theta^{(s',d')} \cdot c_r(u,v)}, \qquad (u,v) \in E, \tag{2}$$

Links with residual capacities smaller than the requested bandwidth are eliminated and the Dijkstra algorithm is executed using weights ($w(u,v)$) as link costs [16]. Results derived via simulation indicate that the algorithm of Wang et al. produces slightly fewer rejections than does MIRA, although it avoids denials resulting from both concentrating and distributing vertices.

The crucial step in relation to the efficiency of these algorithms is the use of maxflow algorithms for the identification of critical links. MIRA [15] uses maximum flow in conjunction with the minimum cut set to identify critical

links, whereas the WSC algorithm [16] detects critical links from their contribution to the maximum flow. Executing such maxflow algorithms, however, is prohibitively expensive for large networks. The *Edmonds-Karp* algorithm, which is the most well known implementation of the *Ford-Fulkerson* maxflow computation method [17], uses a breadth search to find the augmenting paths. It has a complexity of $O(V \cdot E^2)$. For networks such as the Internet, which has an average node degree of about 3.5 [18], the number of links is significantly greater than that of vertices, and the *Edmonds-Karp* algorithm performs very inefficiently in such dense graphs.

Goldberg's algorithm, which is the fastest known maxflow algorithm, still involves a prohibitively high complexity for problems involving large networks. It has a complexity of $O(\min(V^{2/3}, E^{1/2}) \cdot E \cdot \log(V^2/E) \cdot \log(U))$ for networks with $|V|$ vertices and $|E|$ links, with capacities in the interval [1,U] [19]. Other approaches that do not use maxflow computation are necessary to reduce the complexity of minimum interference algorithms.

2.3 The Residual Network and Link Capacity Algorithm

Hendling et al. [20] proposed an algorithm for dynamic routing of LSP's with a guaranteed bandwidth. This *Residual Network and Link Capacity* (RNLC) algorithm applies a penalty function given by:

$$w(u, v) = \frac{N_c}{c_r(u, v)} + \beta, \tag{3}$$

The value of β determines whether or not the algorithm for the shortest path should be used. If β is high, the algorithm implements shortest path routing; otherwise, the link weight variance is increased and the residual capacities are reflected in link weight values.

If a network is underloaded, the links with small residual capacities are more highly penalized than those with larger capacities. In this case, paths with greater residual capacities are chosen. When the links have approximately the same weights, however, the algorithm behaves like a shortest path algorithm. RNLC presents a low computational complexity, but it fails to take into consideration interference in the maximum flow of the other S-D pairs.

2.4 Kumar, Kuri and Kumar's Algorithm

The two-step algorithm proposed by *Kumar, Kuri, and Kumar* [21] guarantees the requested bandwidth. The first step, executed at any time the network

topology changes, computes all the paths between source-destination pairs and, for each pair, identifies a set of links, weighted to reflect sharing with other source-destination pairs, for use in flow routing with higher weights assigned to the shortest paths. The proposed weight function is:

$$w(u,v) = \sum_{sd|(u,v)\in S_{sd}} \alpha_{sd}, \tag{4}$$

where S_{sd} represents the set of links between s and d.

The second step is executed upon the arrival of a new request. Link weights are then updated according to the available residual capacity of each link. The proposed updating equation is

$$w(u,v) = \frac{w(u,v) - I_{(u,v)\in S_{sd}}\alpha_{sd}}{c_r(u,v)}, \tag{5}$$

where I can assume values of either 0 or 1. If the link (u,v) is in the link set of S_{sd}, the value is 1; otherwise it is 0. Dijkstra's algorithm is then executed.

Although the Kumar, Kuri and Kumar algorithm accepts a relatively large number of LSPs, the execution time can be prohibitively long since all paths between S-D pairs must be computed with each topology change. The procedure is particularly problematic in dense networks involving frequent topology changes.

3 Light Minimum Interference Routing Algorithm

In this section, the Light Minimum Interference Routing (LMIR) algorithm is presented. The main advantage of this algorithm is that its computational complexity is lower than that of other existing minimum interference algorithms. This reduced computational complexity is achieved by avoiding numerous executions of maxflow algorithms for the identification of critical links.

The central idea of the LMIR algorithm is the selection of links for the establishment of LSPs which exert the least impact on the maximum flow between other S-D pairs by identifying the links with the smallest available capacity as critical links. Such links are close to saturation and are likely to be included in any minimum cut set of the network. To identify critical links, the LMIR algorithm identifies the paths of least capacity, since all critical links belong to those paths, although a single least-capacity path may involve more than one critical link.

The first step of the LMIR algorithm involves the identification of K least capacity paths with the use of the LowestCapacities algorithm, a variation of

the Dijkstra algorithm. This LowestCapacities algorithm compares the flow at a node v with the smaller of the two values represented by the possible flow of the link (u, v) and the actual flow at the node u. The algorithm stores the information about the distance between a source node and any other node as a function of the number of hops between these nodes ($di[]$), as well as the minimum capacity of all paths between that source node and all other nodes ($F[]$), and the precedence vector involved in reaching a specific destination ($\pi[]$).

Initially, all non-source nodes are attributed an infinitely large value for both flow and distance, whereas these values are considered to be zero at the source node; the node prior to the source node is not considered (Steps 2 to 4).

The first adjacent node v is then analyzed. The flow value at the node v ($F[v]$) is established to be the lower of the two values representing either 1) the current value of $F[s]$ or 2) the flow value of the link (Step 11). The distance (number of hops) is then increased by one (Step 12), with s becoming the new predecessor (Step 13). Additional adjacent nodes are queued according to increasing distance values for future adjacency analysis (Step 14). The algorithm progresses as nodes are removed from the queue using the procedure *extract_min(Q)* and their adjacent nodes processed.

For each step of the LowestCapacities algorithm, the values associated with the adjacent nodes of any specific source node are updated This updating occurs whenever the value of the flow which begins at the source node (s) and ends at the adjacent node (v) and has passed through the target node (u) is either smaller than the value of $F[v]$ or is the same, but located at distance at least one hop shorter than that between the source and the destination distance, $di[u] < di[v] - 1$. Removing the links with the least capacity from the network makes it possible to identify other paths with critical links so that paths with ever-increasing capacities will be identified.

Once the paths with least capacity are identified, weights are assigned to their edges according to the following formula(Step 2):

$$w(u,v) = \sum_{\forall (s,d) \in P : i=0}^{K-1} \frac{f^i_{(s,d)}}{c_r(u,v)}, \forall (u,v) \in f^i_{(s,d)}, \tag{6}$$

The weights assigned to the links of the paths found in Step 1 are thus inversely proportional to the residual capacity. The next step (Step 3) involves the removal of links with a residual capacity smaller than the required level (Step 3). Finally, the Dijkstra algorithm is executed using $w(u, v)$ as the weight to select non-a-less critical links (Step 4). The LMIR algorithm is presented as Algorithm 2.

Algorithm 1 LowestCapacities

1: **for all** $(v \in |V|)$ **do**
2: $F[v], di[v] = \infty$
3: $\pi[v] = NIL$
4: F[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: $\gamma = min[c(u,v), F[u]]$
10: **if** $[(\gamma < F[v]) \vee ((\gamma == F[v]) \wedge (di[u] < di[v]))]$ **then**
11: $F[v] = \gamma$
12: $di[v] = di[u] + 1$
13: $\pi[v] = u$
14: $Q \leftarrow Q \bigcup v$

Algorithm 2 LMIR

INPUT

A residual graph $G_r = (V, E_r)$ and $r(s, d, bw)$ a request for bw bandwidth units between pair (s, d).

OUTPUT

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

LMIR

1: Find K least capacity paths $\forall(s, d) \in P$ (applying LowestCapacities algorithm).
2: Compute weights according to Equation (6) for all links belonging to the paths identified.
3: Eliminate all links with residual capacity smaller than bw.
4: Execute the *Dijkstra* algorithm using $w(u, v)$ as the weights.
5: Create an LSP connecting s to d and update the capacity of the links.

Both the LowestCapacities algorithm and maxflow algorithms need to store in memory the representation of a graph either for computation of the shortest path or for the computation of the maximum flow. Thus, both algorithms require $O(V^2)$ storage space, where V is the number of nodes of the graph [22].

The computational complexity of the LMIR algorithm can be analyzed 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 a complexity of $O(V^2)$. The function *extract_min(Q)* involves a complexity of $O(E_r)$ in the worst possible case, which would happen when the queue of vertices with non-visited adjacent vertices is implemented as a linked list. Thus, the complexity of the *LowestCapacities* algorithm is, $O(V^2 + E_r) = O(V^2)$ [17].

Since each pair requires K executions of the LowestCapacities algorithm, the complexity of the LMIR algorithm for identifying critical links is $O(|P| \cdot K \cdot V^2)$ $= O(V^2)$, whereas in other existing minimum interference algorithms the com-

Table 2
Algorithm Complexity

Algorithm	Critical Link Detection	Weight computation	Dijkstra Algorithm	Total
LMIR	$O(V^2)$	$O(E_r)$	$O(V^2)$	$O(V^2 + E_r)$
MIRA	$O(MaxFlow)$	$O(E_r)$	$O(V^2)$	$O(MaxFLow + V^2 + E_r)$
WSC	$O(MaxFlow)$	$O(E_r)$	$O(V^2)$	$O(MaxFLow + V^2 + E_r)$

plexity involves the performance of a maxflow algorithm which has a complexity of $O(|P| \cdot \min(V^{2/3}, E_r^{1/2}) \cdot E_r \cdot \log(V^2/E_r) \cdot \log(U)) = O(\min(V^{2/3}, E_r^{1/2}) \cdot E_r \cdot \log(V^2/E_r) \cdot \log(U))$. The complexity of weight computation is $O(E_r)$ for LMIR, WSC and MIRA since all edges of a graph may be involved. Since the complexity of the Dijkstra algorithm is $O(V^2)$, the total complexity of the LMIR algorithm is $O(2 \cdot V^2 + E_r) = O(V^2 + E_r)$. while the complexity of WSC and MIRA is $O(\min(V^{2/3}, E_r^{1/2}) \cdot E_r \cdot \log(V^2/E_r) \cdot \log(U) + E_r)$. Table 2 [1] summarizes the computational complexity of the three algorithms.

As mentioned above, Steps 2 to 5 of the LMIR algorithm are common to WSC and MIRA. It is quite clear from Table 2 that the great advantage of LMIR is to reduce the complexity of the identification of critical links procedure which is the most costly step in all minimum interference routing algorithms. Moreover, the following theorem establishes a relationship between the complexity of LMIR and of the other algorithms:

Theorem 1 *The computational complexity of the LMIR algorithm is upper-bounded by the complexity of the MIRA and WSC algorithms.*

Proof See Appendix A.

The number of least capacity paths considered by the LMIR algorithm has an impacts on its performance. Although it is not possible to determine an optimum value for the number of least capacity paths to be identified, an upper bound for that value can be established (Theorem 2).

Theorem 2 *An upper bound, $K(\theta)$, for the number of least capacity paths to be identified by the LMIR algorithm is given by:*

$$K(\theta) = dg(s), \text{ if } \theta^{(s,d)} = \sum_{\forall i | (s,i) \in E(G)} c(s,i), \tag{7}$$

[1] In this table, $O(MaxFlow) = O(\min(V^{2/3}, E^{1/2}) \cdot E \cdot \log(V^2/E) \cdot \log(U))$ which gives the complexity of the fastest known maxflow algorithm

or

$$K(\theta) = dg(d), \ \textit{if}\ \theta^{(s,d)} = \sum_{\forall i | (i,d) \in E(G)} c(i,d), \tag{8}$$

or

$$K(\theta) = \left\lceil \frac{\theta^{(s,d)}}{f^0_{(s,d)}} \right\rceil . \tag{9}$$

where: $f^0_{(s,d)}$ is the flow along the path of least capacity between s and d, $\theta^{(s,d)}$ is the maximum flow between them and $dg(s)$ and $dg(d)$ represent the degree of nodes s and d, respectively.

Proof

See Appendix B.

4 Comparison of LMIR and other existing minimum interference algorithms

Simulation experiments were carried out to compare the performance of the LMIR algorithm with that of both the MIRA and WSC algorithms. Although the Minimum Hop Algorithm (MHA) and the Widest Shortest Path algorithm (WSP) do not take interference into account, they were also included in the simulation experiments so that the benefits of a non-interference approach could be assessed. MHA is practically the same as the Dijkstra algorithm while the WSP algorithm identifies the path with the least number of links and if there are two or more of the links, the algorithm picks that with the maximum available capacity.

The reduction in maximum flow, the blocking probability for requests and the degree of interference were used as performance indicators for comparison. A reduction in maximum flow provides a measure of the reduction in total available bandwidth [2] given by the total maxflow of all pair of nodes. Moreover, the degree of interference indicates the maxflow reduction of a specific pair.

The same seeds are used for the random number generators of all the algorithms used. Intervals with 95% of confidence were derived using the replication method.

Both, large and small networks were considered in this study. The small network with 15 vertices used in both [15] and [16] was utilized so that a com-

[2] This definition of total available bandwidth is used for comparison with [15, 16].

parison could be made with previously published results, while the topology for the large networks was randomly generated.

Simulations were conducted using networks with 30, 40, 50, 100 and 150 vertices. For these large networks, both sparse and dense, only results furnished by the minimum interference algorithms are shown. Networks are modeled as non-oriented graphs in which each link has a non-negative capacity. For these simulations, network topologies were created using Waxman's method [18, 23] in which the probability of the existence of a link between u and v is given by the following

$$P(u, v) = \alpha e^{-d/(\beta L)},$$

where $0 < \alpha, \beta \leq 1$ are model parameters, d is the Euclidean distance between u and v, and L is the maximum Euclidean distance between any two vertices of the graph. Links are bidirectional, and an equal number of choices for each bandwidth size (1,200 or 4,800) was made. For all networks, the number of least capacity paths to be searched by the LMIR algorithm was 5.

LSPs are assumed to be long-lived so that once one is accepted, resources are held until the end of the experiments. Either $8,000$ (15-vertex network) or $30,000$ (30,40 and 50-vertex networks) requests were generated among the source-destination pairs, with the bandwidth requests uniformly distributed in the interval $[1, 4]$. In large networks, scenarios with low (between 0 and $10,000$), intermediate (between $10,000$ and $20,000$) and high (between $20,000$ and $30,000$) loads were considered.

4.1 15-Vertex Network

The topology used in the 15-vertex network experiments, as well as the source-destination pairs, is shown in Figure 2. The lighter bi-directional links represent a capacity of 1,200 bandwidth units each, while the darker ones have a capacity of 4,800 bandwidth units, corresponding to transmission rates of OC-12 and OC-48, respectively.

Figure 3 shows the reduction in total bandwidth available as a function of the number of requests. It can be seen that the minimum interference routing algorithms produce lower reductions than do the WSP and the MHA algrithms, since the latter do not consider how critical a link is when choosing a path.

Up to 3,000 requests, the results of the LMIR and MIRA algorithms are equivalent, both resulting in less bandwidth reduction than found for the WSC algorithm, but between 3,000 and 5,000 requests, the decrease in reduction is greater for the LMIR algorithm.

The advantage of the non-interference approach is clear from the large re-

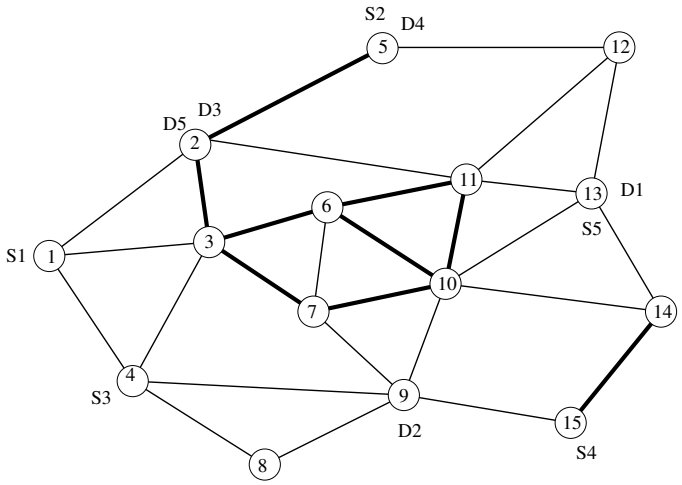

Fig. 2. Topology used for simulations of small networks.

duction resulting from the application of the WSP and MHA algorithms. This reduction is a direct consequence of the unnecessary allocation of critical links, which greatly increases the impact on the maximum flow.

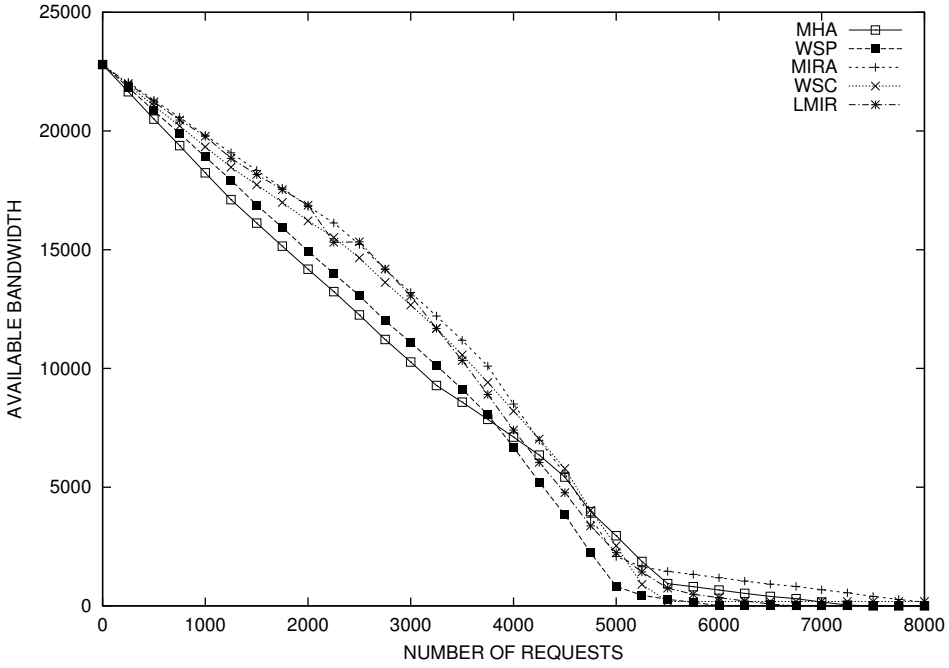

Fig. 3. Total bandwidth for all network source-destination pairs (15-vertex network).

Figure 4 shows the number of requests rejected as a function of the number of requests arriving. The WSP algorithm starts rejecting connections only after the arrival of 5,000 requests. However, since it does not take link criticality into consideration, it may choose paths that include critical links of various pairs, thus leading to saturation. The LMIR and MIRA algorithms result in a low number of rejections, thus proving the benefits of the non-interference approach.

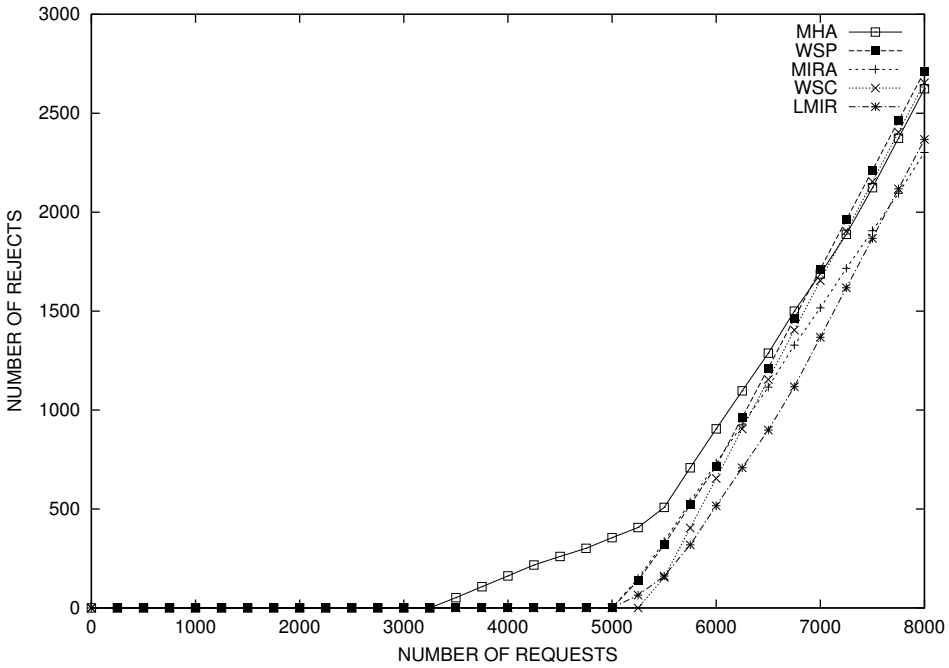

Fig. 4. Number of rejections (15-vertex network).

The two previous examples have illustrated the performances of different algorithms in relation to the total flow in a small network. Another interesting question is how these algorithms influence the flow of an individual S-D pair. To shed some light on this question, the pair $(S1, D1)$ was chosen as the object of this analysis. Figure 5 shows the reduction in maximum flow of this source-destination pair as a function of the other S-D pairs. It is clear that the use of the MHA and WSP algorithms leads to a reduction in maximum flow as soon as the first request arrives. When using the MHA algorithm network saturation is reached immediately after the arrival of the $2,000^{th}$ request and from then on, the path is saturated. The WSP algorithm takes a somewhat longer period to reach saturation, since a path approaching saturation is passed over from another with a higher capacity.

The minimum interference algorithms do not provide any interference under low-load conditions (up to the arrival of 2,500 requests), but both the LMIR and MIRA algorithms produce less interference than the WSC algorithm for as many as $5,000$ requests, although in the long run the use of the WSC algorithm results in less reduction in maximum flow for the pair than do the LMIR and MIRA algorithms. These both produce similar interference up to the $5,000^{th}$ request since they identify the same set of critical links, but after the arrival of 5,000 requests the MIRA algorithm identifies fewer paths not interfering with the maximum flow of pair $(S1, D1)$ than does the LMIR algorithm.

The choice of the number of least capacity paths to be identified in Step 2 is the key to the performance of the LMIR algorithm, since the selection of a

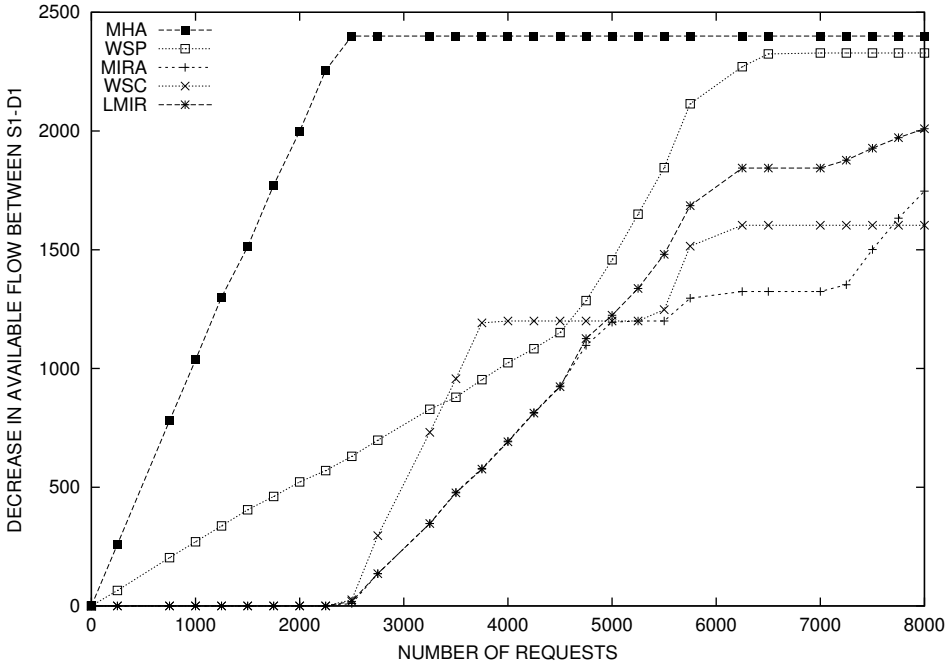

Fig. 5. Effects on pair $(S1, D1)$ (15-vertex network).

value for K which differs significantly from the number of links in the minimum cut set may lead to the misidentification of critical links and the attribution of high weight to them. Figure 6 shows the number of requests rejected as a function of the choice of K, which reveals that the minimum number of rejection is reached for $K = 5$. Increasing the value of K does not lead to any improvement; moreover, it makes the identification of critical links difficult.

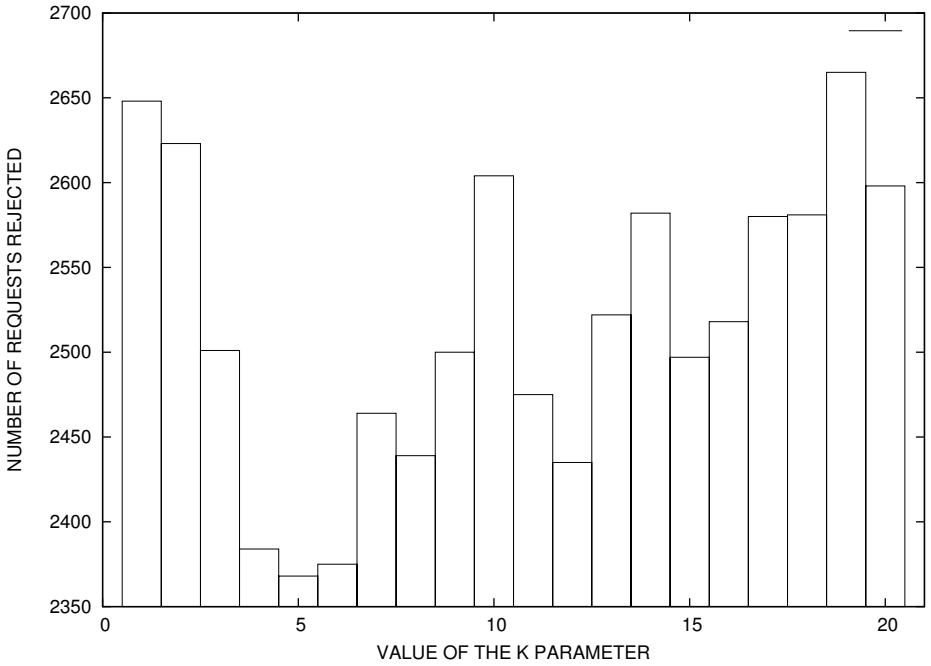

Fig. 6. Influence of the parameter K in the number of rejected connections.

In order to evaluate the execution time, the algorithms were tested for 8,000 requests measured with the Linux *time* command in an Intel Celeron machine with a 1.2 GHz clock, and 256 MB of RAM. Table 3 shows the values measured.

Table 3
Execution times for 8,000 requests.

	WSC	MIRA	LMIR $K=1$	LMIR $K=2$	LMIR $K=3$	LMIR $K=4$	LMIR $K=5$	LMIR $K=6$
`Time`	7.23s	7.24s	5.017s	5.67s	6.54s	6.94s	7.01s	7.14s

Both the WSP and MHA algorithms require execution times of approximately 4 seconds. Although these are the fastest results, the algorithms are not satisfactory because they lead to an excessive number of rejections. As expected, the time required for the LMIR algorithm is less than that for both the WSC and MIRA algorithms, although the numerical results produced are the same for all three. Actually, there is a trade off between precision and execution time involving the number of least capacity paths chosen. The lower the value of K is, the faster the execution time. However, the results tend to differ significantly from the optimum. Selecting a larger number of least capacity paths than the optimum only increases execution time.

4.2 Dense 30-vertex networks

Figure 7 shows the maximum flow reduction for the selected source-destination pairs for dense 30-vertex networks, with all three algorithms evaluated performing at a similar level. This performance worsens under high loads, however, since links belonging to the minimum cut set may be used to establish LSPs. With high loads, 70% of the requests are rejected (Figure 8). Both LMIR and MIRA algorithms block roughly the same number of requests, fewer than those blocked by the WSC algorithm. Only towards the end of the simulation does the LMIR algorithm produce slightly lower blocking probabilities than does the MIRA algorithm.

Under high loads, only a small number of paths have enough capacity to be available for allocation. The difference in the performance of the algorithms can be attributed to the way they identify critical links, which leads to different approaches to saturation.

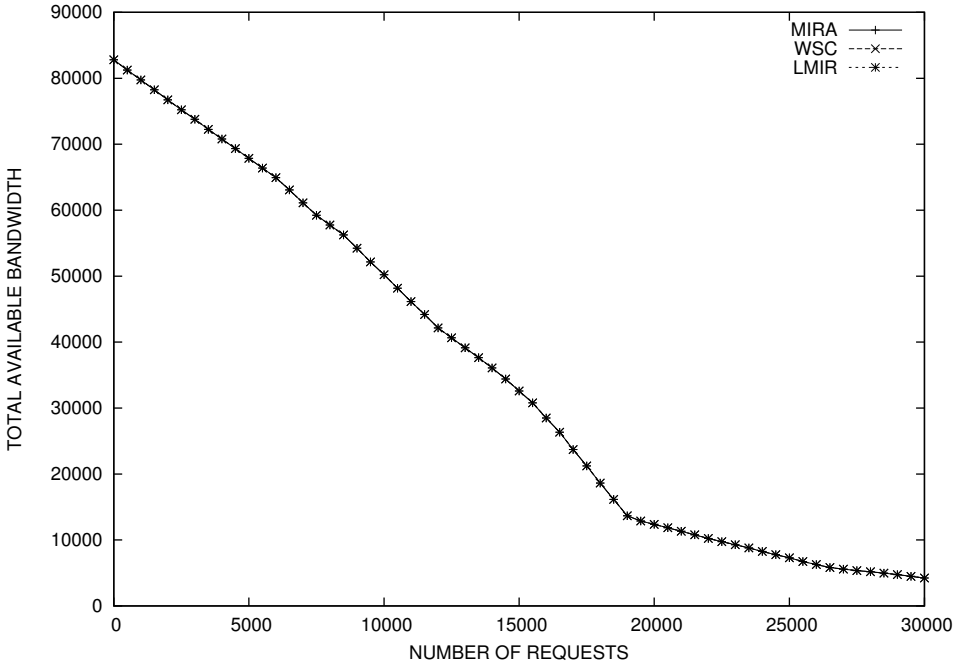

Fig. 7. Total bandwidth (dense 30-vertex networks).

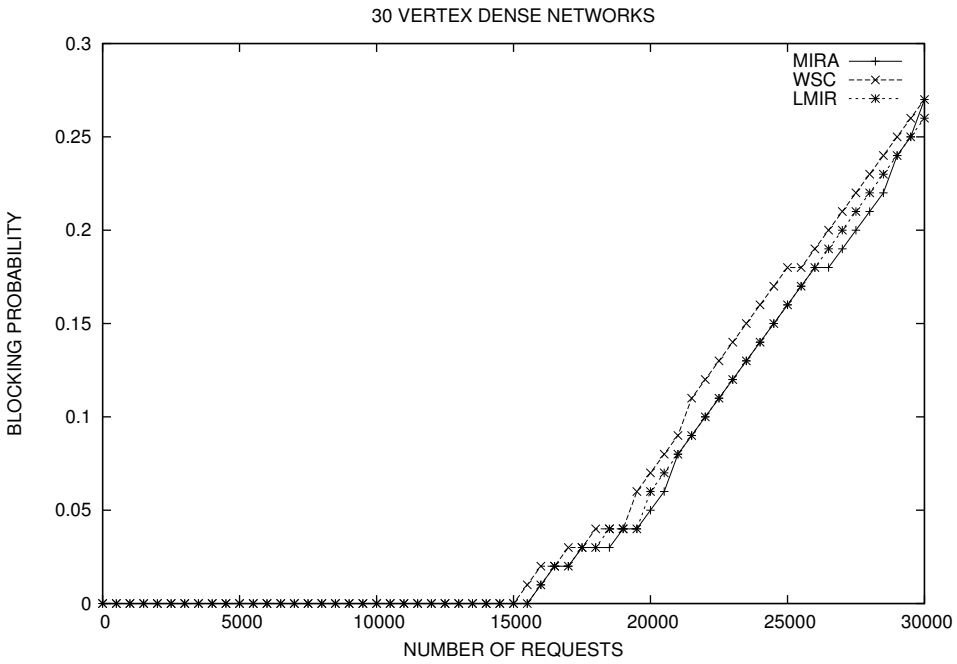

Fig. 8. Blocking Probability (dense 30-vertex networks).

4.3 Sparse 30-vertex networks

Figure 9 illustrates the maximum flow reduction between source-destination pairs for sparse networks with 30 vertices. The behavior of the three algorithms (WSC, MIRA, and LMIR) is very similar whatever the load condition.

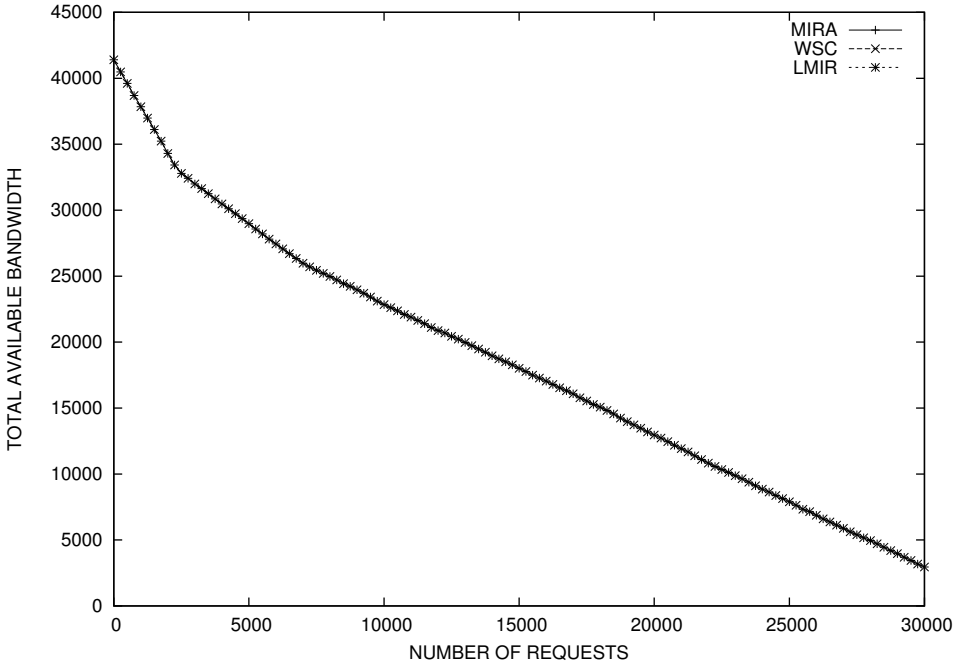

Fig. 9. Total bandwidth (sparse 30-vertex networks).

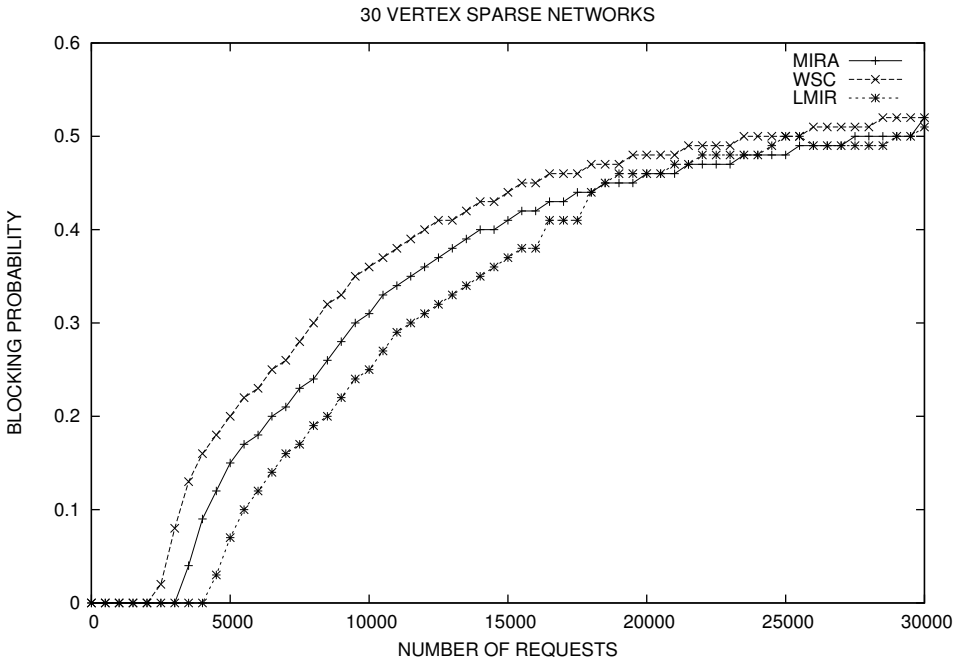

Fig. 10. Blocking Probability (sparse 30-vertex networks).

This happens because of the reduced number of paths existing for each source-destination pair, as this obviously reduces the number of alternative paths for each algorithm. Moreover, the blocking probability values (Figure 10) become almost indistinguishable under high loads. However, under low and medium loads, the LMIR algorithm produces blocking probabilities which are 0.17 and 0.1 lower than those produced by the WSC and MIRA algorithms, respectively.

4.4 Dense 50-vertex networks

The dense 50-vertex networks used here have 544 links which implies a large number of paths to choose from. As a consequence, saturation was not observed in the simulation experiments. To reach saturation in these networks, all of the links were assigned a capacity of 1,200 bandwidth units.

The behavior of the WSC, MIRA, and LMIR algorithms was found to be quite similar for up to 20,000 requests (Figure 11). For medium loads the LMIR algorithm produced the lowest probability of blocking (Figure 12), whereas for light loads the probability of blocking was null for all three algorithms, and for high loads their performance were almost indistinguishable.

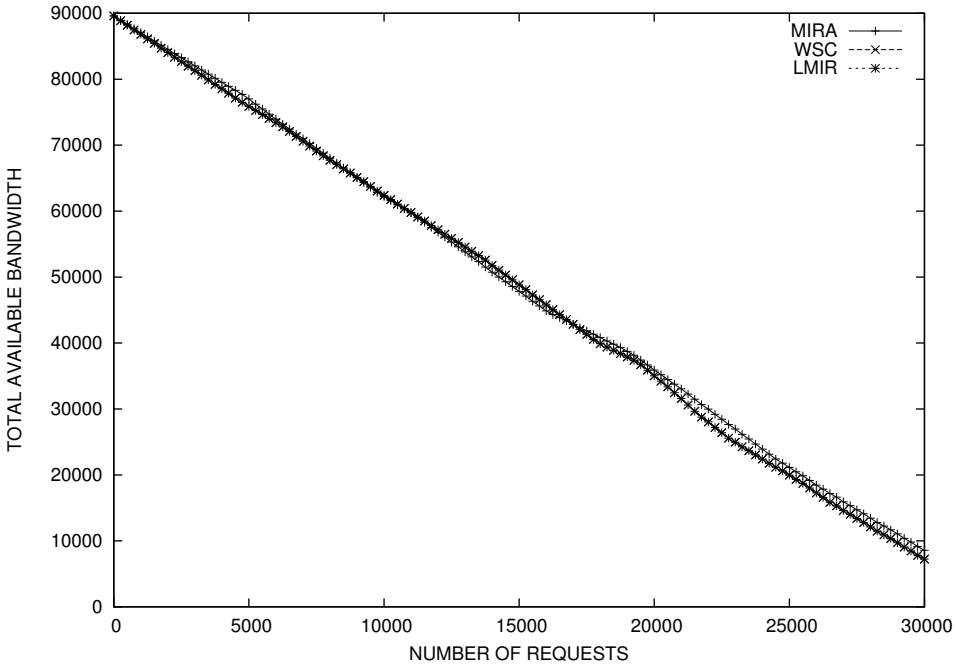

Fig. 11. Total bandwidth (dense 50-vertex networks).

The number of rejections in these dense 50-vertex networks is substantially lower than in the sparse networks or those with a smaller number of available links, due to the large number of paths available to establish LSPs.

4.5 Sparse 50-vertex networks

Sparse networks with 50 vertices are subject to a slower reduction in the maximum flow with the use of the LMIR algorithm than of the WSC and MIRA algorithms (Figure 13). With sparse 50-vertex networks, saturation is reached rapidly due to the saturation of critical links. This rapid saturation

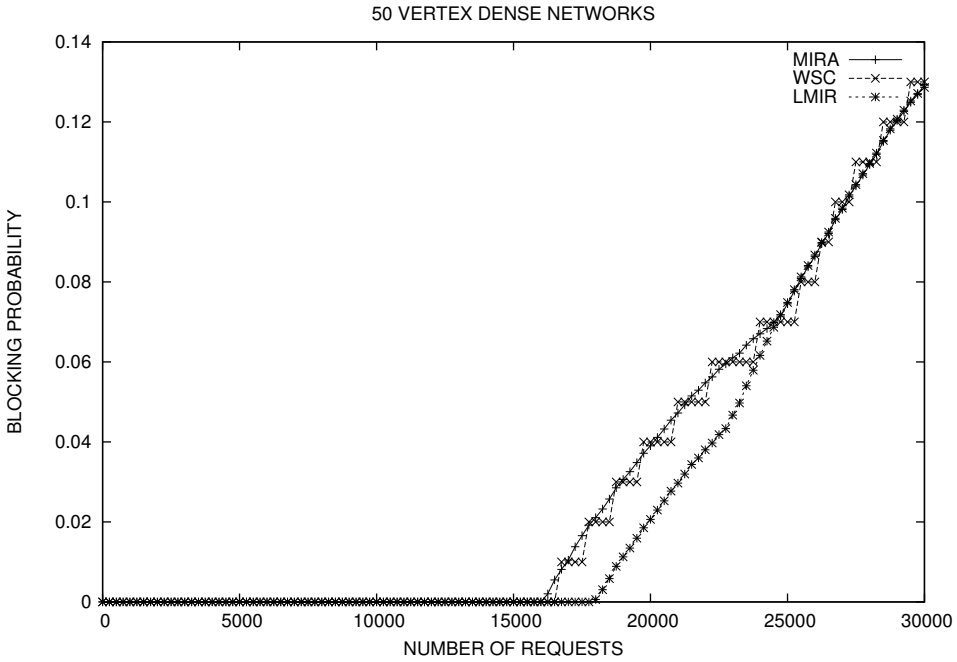

Fig. 12. Blocking Probability (50 vertices dense networks).

means that the maximum flow reduction is more intense in sparse than in dense networks with the same number of nodes.

The use of links belonging to the minimum cut set implies the absence of alternative paths and consequently leads to a high number of rejections (Figure 14). Nonetheless, the choice of paths pursued by the LMIR algorithm produces the lowest number of rejections of requests, whereas the WSC algorithm produces the highest number.

Figure 14 shows the probability of blocking in sparse networks with 50 vertices. The critical links are used intensively due to the lack of alternative paths, which leads to a high number of request rejections. The lowest blocking probability is produced by the LMIR algorithm.

4.6 100-vertex networks

Figure 15 shows the reduction in maximum flow for networks with 100 vertices and 1150 links, with the three algorithms producing the same bandwidth reduction.

Figure 16 displays the blocking probability as a function of the number of arrivals. LMIR results in the lowest blocking probability with a difference of 10% in some cases.

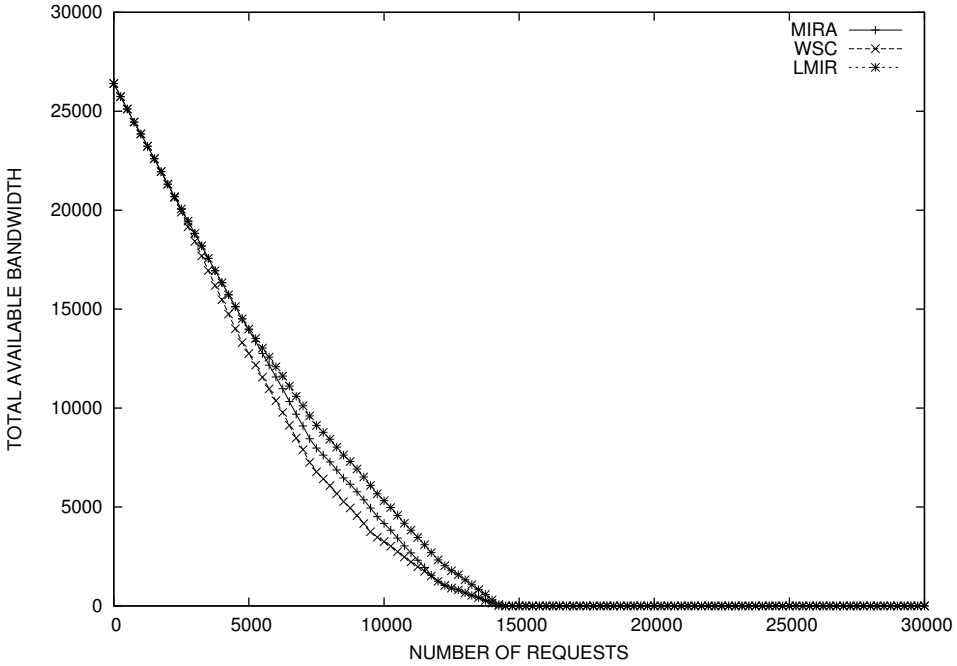

Fig. 13. Total bandwidth (sparse 50-vertex networks).

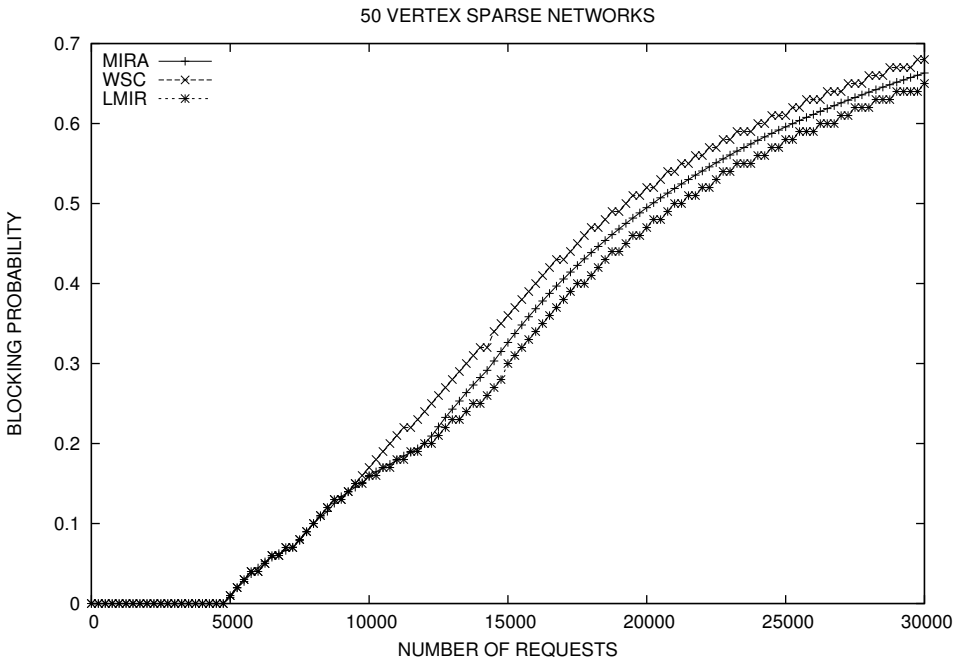

Fig. 14. Number of requests rejected (sparse 50-vertex networks).

4.7 150-vertex networks

Figure 17 shows the maximum flow reduction for networks with 150 vertices and 2484 links. As with other netwrok sizes, the reduction of the maxflow is about the same for the three algoritms. In this case, and for the interval

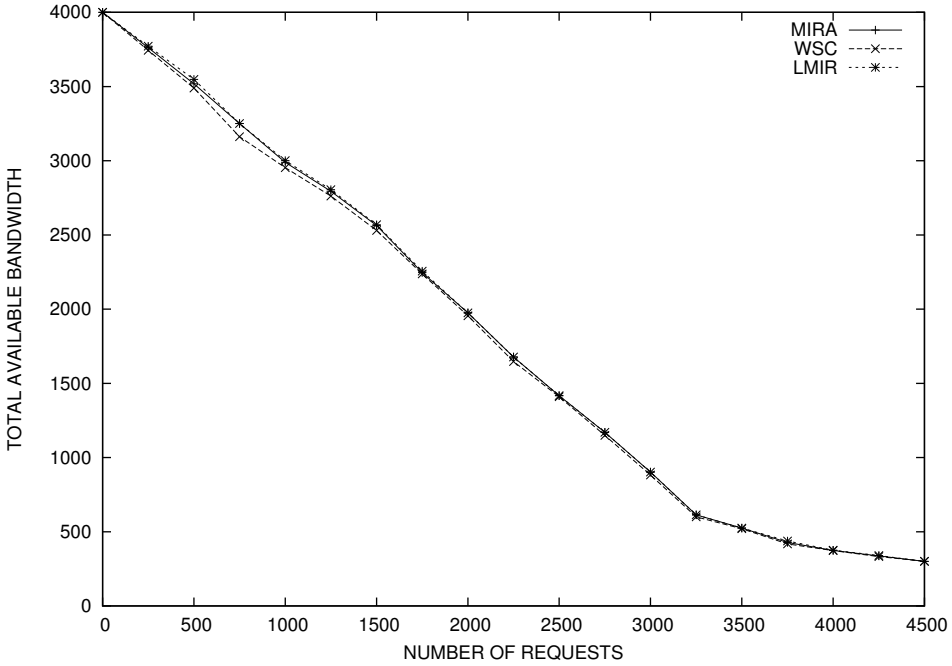

Fig. 15. Total bandwidth (100-vertex networks).

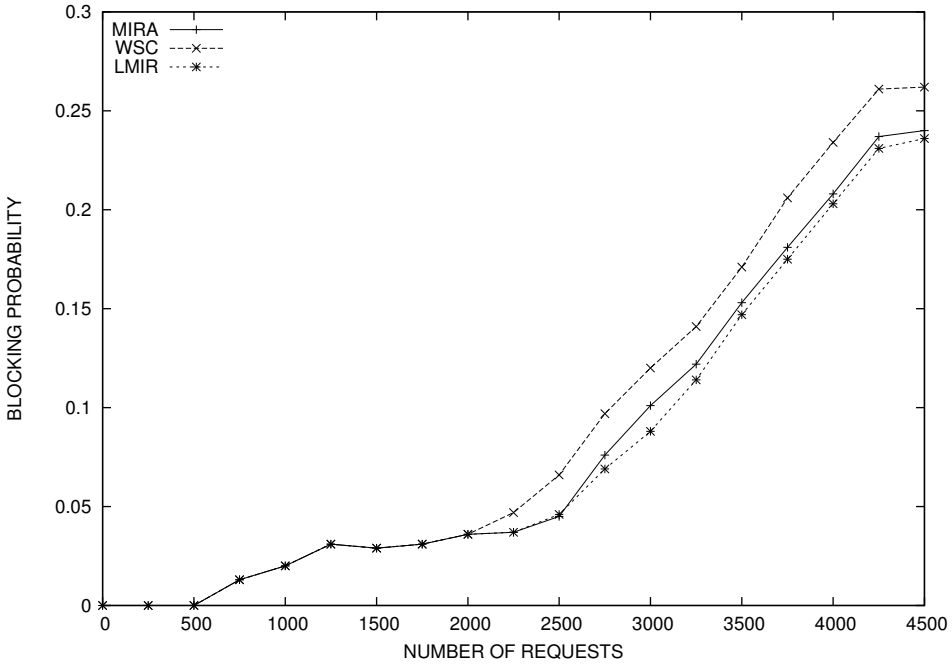

Fig. 16. Blocking Probability (100-vertex networks).

of 2,000 to 2,500 arrivals, LMIR showed a slightly lower reduction than did MIRA and WSC algorithms.

Figure 18 plots the blocking probability, with the use of LMIR leading to slightly lower blocking probability for requests in excess of 3,000.

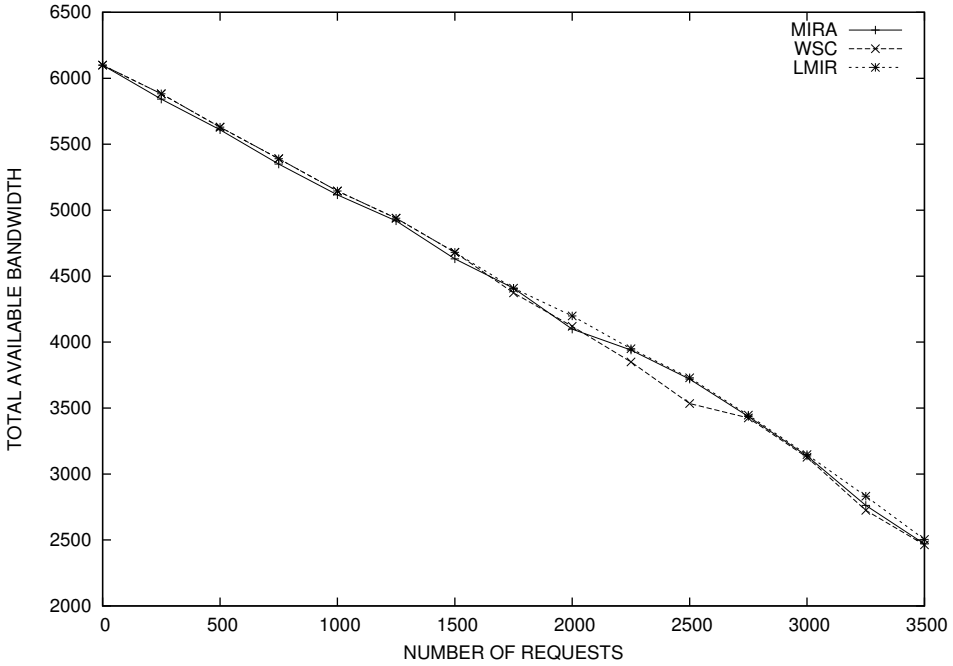

Fig. 17. Total bandwidth (150-vertex networks).

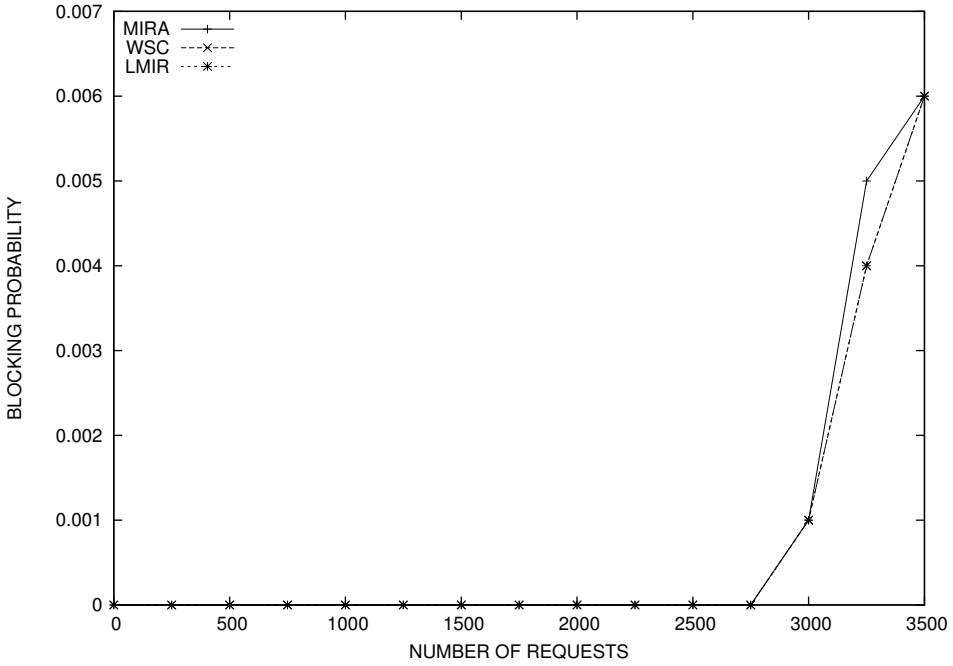

Fig. 18. Blocking Probability (150-vertex networks).

The relative execution time for the minimum interference algorithms investigated here was determined and is shown in Table 4. The highest value (produced by the MIRA and WSC algorithms) is used as a reference value. Negative values in this table mean that the execution time was less than this reference value. These results show that the LMIR algorithm is the fastest, with reductions of up to 42% when five least capacity paths are involved,

Table 4
Relative execution time for individual connections

	150	100	50 (D)	50 (E)	40 (D)	40 (E)	30 (D)	30 (E)
Reference Value	4.1×10^{-2}	1.5×10^{-1}	6.8×10^{-3}	5.7×10^{-3}	4.8×10^{-3}	5.7×10^{-3}	4.1×10^{-3}	6.5×10^{-3}
MIRA	-1%	0.00	0.00	-1%	-1%	0.00	0.00	0.00
WSC	0.0	-7.2%	-2%	0.0	0.00	0%	-2%	-20%
MHA	-86%	-85%	-76%	-79%	-82%	-85%	-84%	-88%
WSP	-79%	-78%	-68%	-69%	-75%	-77%	-77%	-80%
LMIR K =5	-42.7%	-33.7%	-39%	-17%	-13%	-36%	-2%	-28%
LMIR K =6	-36.4%	-25.7%	-30%	-12%	-5%	-23%	+1%	-22%
LMIR K =7	-31.7%	-18.8%	-28%	-5%	+7%	-18%	+6%	-19%
LMIR K =8	-27.2%	-9.1%	-27%	-3%	+9%	+1%	+11%	-11%
LMIR K =9	-18.3%	-1.0%	-10%	0%	+22%	+6%	+20%	-4%
LMIR K =10	-4.2%	+4.5%	-2%	+6%	+24%	+6%	+29%	+2%

although this performance deteriorates for a larger number of paths. For all simulations, the most efficient was five path as this resulted in the fastest execution time while producing blocking probability values similar to those obtained using the MIRA and WSC algorithms.

5 Conclusions

This paper has presented a new minimum interference routing algorithm which does not use maxflow algorithms for critical link identification. This LMIR algorithm uses a modified Dijkstra algorithm to identify the paths with the least capacity, and these paths are then used to identify the critical links. Weights are then assigned to the links of these paths, and the shortest path is identified. The number of critical paths to be identified when using the LMIR algorithm has an impact on its performance with the best results in simulation experiments obtained when this is limited to five.

The results obtained using the LMIR algorithm are similar to those obtained with the MIRA and WSC algorithms. Although, for large dense networks, it produces the lowest probability of blocking, as well as minimal reductions in maximum flow. Furthermore, the LMIR algorithm involves some 42% less computational time than do the MIRA and WSC algorithms. In general, this

novel algorithm thus seems to be the best candidate for adoption in on-line procedures for the establishment of LSP's in MPLS networks.

As future work, the use of maximum flow trees [24, 25] should be investigated to reduce the computational complexity of minimum interference routing algorithms.

Acknowledgements

We would like to thank the reviewers for their invaluable suggestions. This work was partially sponsored by the Brazilian National Council for Research (CNPq).

References

[1] Eric Rosen, Arun Viswanathan and Ross Callon, Muliprotocol Label Switching Architecture, Tech. Rep., IETF (January 2001).

[2] Daniel O. Awduche and Bijan Jabbari, Internet Traffic Engineering using Multi-Protocol Label Switching (MPLS), Computer Networks 40 (1)(2002) 111–129.

[3] Daniel O. Awduche, MPLS and Traffic Engineering in IP Networks, IEEE Communications Magazine, 37 (12)(1999), 42-47.

[4] T. Anjali, C. Scoglio, J. C. de Oliveira, I. F. Akyildiz and G. Uhl, Optimal Policy for Label Switched Path Setup in MPLS Networks, Computer Networks 39 (2) (2002) 165–183.

[5] M. Garey and D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Co., 1979.

[6] Shigang Chen and Klara Nahrstedt, An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions, IEEE Network, 6 (1998) 64-79.

[7] Shigang Chen, Routing Support for Providing Guaranteed End-to-End Quality-of-Service, PhD thesis, Engineering college – University of Illinois at Urbana-Champaign, 1999.

[8] Gargi Banerjee and Deepinder Sidhu, Comparative Analysis of Path Computation Techniques for MPLS Traffic Engineering, Computer Networks 40(1) 2002 149-165.

[9] J. Jaffe, Algorithms for Finding Paths with Multi Constraints, Networks (1984) 95-116.

[10] T. Anjali, C. Scoglio, J. C. de Oliveira, L. Chen, I. F. Akyildiz, and J. A. Smith, A New Path Selection Algorithm for MPLS Networks Based on Available Bandwidth Estimation, In: QofIS'2002, 2002.

[11] H. de Neve and P. V. Mieghem, A Multi Quality of Service Routing Algorithm for PNNI, ATM Workshop, 1998.

[12] Z. Wang and J. Crowcroft, Quality-of-Service Routing for Supporting Multimedia Applications, IEEE Journal on Selected Areas in Communications, (1996) 1228-1234.

[13] Shigang Chen and Klara Nahrstedt, On Finding Multi-Constrained Paths, In: IEEE International Conference on Communications, 1998, pp. 874-879.

[14] Gustavo B. Figueiredo, Nelson L. S. da Fonseca and Jose A. S. Monteiro, A Minimum Interference Routing Algorithm, In: IEEE International Conference on Communications, 2004.

[15] Murali S. Kodialam and T. V. Lakshman, Minimum Interference Routing with Applications to MPLS Traffic Engineering, In: IEEE INFOCOM (2), 2000, pp. 884-893.

[16] Bin Wang, Xu Su and C. L. P. Chen A New Bandwidth Guaranteed Routing Algorithm for MPLS Traffic Engineering, In: IEEE International Conference on Communications, 2002, pp. 1001-1005.

[17] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest,Introduction to Algorithms, MIT Press and McGraw Hill, 1990.

[18] Ellen W. Zegura, Kenneth L. Calvert and Samrat Bhattacharjee, How to Model an Internetwork, In: IEEE INFOCOM, 1996, pp. 594-602.

[19] Andrew V. Goldberg and Satish Rao, Beyond the Flow Decomposition Barrier, J. ACM 45 (5) (1998), 783-797.

[20] K. Hendling, G. Franzi, B. Statovci-Halimi and A. Halimi, Residual Network and Link Capacity Weighting for Efficient Traffic Engineering in MPLS Networks, In: Proceedings of ITC, 2003, pp. 51-60.

[21] Deepak Kumar, Joy Kuri and Anurag Kumar, Routing Guaranteed Bandwidth Virtual Paths with Simultaneous Maximization of Additional Flows, In: IEEE International Conference on Communications, 2003, pp. 1759-1764.

[22] Christos M. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.

[23] Bernard M. Waxman, Routing of Multipoint Connections, IEEE Journal on Selected Areas in Communications 6 (9)(1988) 1617-1622.

[24] , Marco Schneider, Flow Routing in Computer Networks, PhD thesis, The University of Texas at Austin, 1997.

[25] B. Chazelle, A Minimum Spanning Tree Algorithm with Inverse-Ackermann Type Complexity, Journal of the ACM 47 (2000) 1028-1047.

[26] Douglas B. West, Introduction to Graph Theory, Prentice Hall, 1996.

A Theorem 1

This appendix furnishes proofs for the theorems presented in the paper.

Theorem 1 *The computational complexity of the LMIR algorithm is upper bounded by the complexity of the MIRA algorithm.*

Proof

Since Steps 2 through 5 of the LMIR algorithm (see Algorithm 2) are the same as those executed by the MIRA and WSC algorithms, and since the complexity of these steps is a function of the procedure used to identify the critical links, it is sufficient to show that the complexity of the LowestCapacities algorithm is less than that of Goldberg's algorithm, the least complex maxflow algorithm known.

The proof is carried out for both sparse and dense graphs. For **dense graphs** it is assumed that $|E_r| = |V|^2$. Goldberg's algorithm has the following complexity:

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

while the *LowestCapacities* algorithm has the complexity of $O(V^2)$. Assuming that some of the terms involved in $\log(V^2/V^2)$ have been omitted (since the complexity cannot be zero), and given the fact that $\log(V^2/V^2) \cdot \log(U) > 1$, it follows that

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

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

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

For **sparse graphs** it is assumed that $|E_r| = |V|$. In this case, Q (the queue of non visited nodes) can be implemented as a heap [17]. Therefore,

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

since the step of *extract_min(Q)* evidences a complexity of $O(\log V)$. Assuming that $\log(V^2/E_r) \cdot \log(U) > 1$, using the first two factors of $O(\min(V^{2/3}, E_r^{1/2}) \cdot E_r \cdot \log(V^2/E_r) \cdot \log(U))$, it can be shown that

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

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

Discarding V in both algorithms and taking the log gives:

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

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

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

This shows that critical links detection with the LMIR algorithm involves a lower computational complexity than does any other minimum interference algorithm using maxflow algorithms.

B Theorem 2

Theorem 2 *An upper bound, $K(\theta)$, for the number of least capacity paths identified by the LMIR algorithm is given by:*

$$K(\theta) = dg(s), \text{ if } \theta^{(s,d)} = \sum_{\forall i|(s,i)\in E(G)} c(s,i),$$

or

$$K(\theta) = dg(d), \text{ if } \theta^{(s,d)} = \sum_{\forall i|(i,d)\in E(G)} c(i,d),$$

or

$$K(\theta) = \left\lceil \frac{\theta^{(s,d)}}{f^0_{(s,d)}} \right\rceil .$$

where: $f^0_{(s,d)_G}$ is the flow along the path of least capacity between s and d, $\theta^{(s,d)}$ is the maximum flow between them and $dg(s)$ and $dg(d)$ are the degree of nodes s and d respectively.

Proof To find the lower bound for the number of links in the minimum cut set, it is necessary to identify the minimum number of links which forms a disconnected graph. Before showing the proof for this lower bound value, however, certain definitions must be introduced.

Definition 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 [26].*

Definition 2 *Two paths from s to d are internally disjointed if they have no common internal vertices (edges) [26].*

Theorem 3 *Let $\lambda(s,d)$ be the number of s,d- internally disjointed paths; then*

$$\lambda(s,d) \leq min\{dg(s), dg(d)\},$$

where $dg(s)$ and $dg(d)$ are the degrees of s and d, respectively.

Proof Let P and P' be two internally disjointed paths between s and d. For the proof suppose that $\lambda(s,d) > \min\{dg(s), dg(d)\}$ which would imply that at least one edge incident to s or d has been included in both of the internally disjointed paths, P and P'. However, if P and P' had an edge in common, they would not be internally disjointed thus contradicting the hypothesis. □

The following theorem establishes a relationship between the cardinality of a set of disconnecting links and the number of disjointed paths of a graph.

Menger's theorem [26] uses these concepts of disconnecting sets and disjointed paths:

Theorem (Menger-1927) 4 *Let s and d be two vertices of a graph G and $(s,d) \notin E(G)$. Then,* $\max(\kappa'(G)) = \min(\lambda(s,d))$.

Proof See [26].

The relation between Theorem 3 and Menger's Theorem suggests an upper bound for K, since it indicates the maximum number of edges a minimum size disconnecting set can have.

However, as we shall see, this is not true. Figure .1 illustrates a graph with only two disjointed paths but with four edges in the minimum cut set. This graph can be transformed into an equivalent one to which Menger's Theorem can be applied. The transformation involves changing each edge with a capacity x into x edges each with a capacity of one.

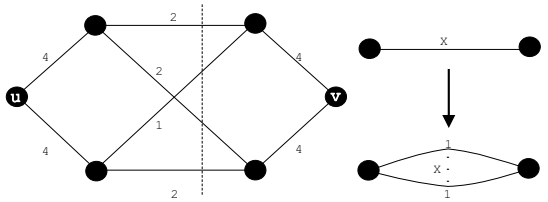

Fig. .1. Number of disjointed paths less than the number of edges in the min cut set.

Fig. .2. Transformation in the original network.

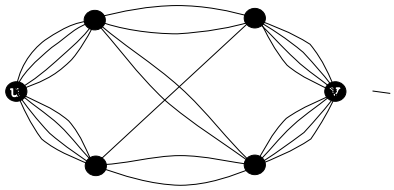

Fig. .3. Transformed network for Figure .1.

This leads to a new theorem relating maximum flow and Menger's theorem (Theorem 4):

Theorem 5

$$\lambda(s,d) \geq \theta_{(G')}^{(s,d)} = \min C(S,D) \geq \kappa'(G'),$$

where $\lambda(s,d)$ is the number of internally disjointed paths between s and d, $\theta_{(G')}^{(s,d)}$ is the maximum flow between s and d in G', $C(S,D)$ represents the capacity of a cut (S,D) such that $s \in S$ and $d \in D$, while $\kappa'(G)$ is the minimum size of a disconnecting set of G.

Proof See [26].

Theorem 6 *Let $f^0_{(s,d)_G}$ be a flow passing through a path of least capacity between s and d in G, thus $n = f^0_{(s,d)_G} = \sum_{i=1}^{n} f^0_{(s,d)_{G'}}$.*

Proof In G', each edge has a 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 edge of least capacity on that path. Since G' is built from G, including an edge with a capacity of R in G is equivalent to using R edges with the same endpoints in G'. Thus the equality holds. □

Case 1:

$$\theta^{(s,d)}_{(G)} = \sum_{\forall i|(s,i)\in E(G)} c(s,i).$$

The Maxflow Mincut theorem [26] implies that the maximum network flow is bounded by the minimum capacity of a source-sink cut. Thus if

$$\theta^{(s,d)}_{(G)} = \sum_{\forall i|(s,i)\in E(G)} c(s,i),$$

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

Case 2:

$$\theta^{(s,d)}_{(G)} = \sum_{\forall i|(i,d)\in E(G)} c(i,d).$$

This can be proved using reasoning analogous to that for Case 1.

Case 3:

Let

$$n = \theta^{(s,d)}_{(G')} = \theta^{(s,d)}_{(G)},$$

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

$$n = \theta^{(s,d)}_{(G)} = n \cdot \omega^{(s,d)}_{(G')} = \sum_{i=1}^{n} f^0_{(s,d)_{G'}},$$

if b and $K(\theta)$ are chosen so that $b \cdot K(\theta) \leq n$, then

$$n = \theta^{(s,d)}_{(G)} \geq K(\theta) \cdot \sum_{i=1}^{b} f^0_{(s,d)_{G'}}. \tag{.1}$$

The application of Theorem 6 in Equation .1 gives the following:

$$K(\theta) \cdot f^0_{(s,d)_G} \leq \theta^{(s,d)}_{(G)}. \tag{.2}$$

Since in G both $\theta^{(s,d)}_{(G)}$ and $f^0_{(s,d)_G}$ are known, we can define $K(\theta)$ for a network G:

$$K(\theta) = \left\lceil \frac{\theta^{(s,d)}}{f^0_{(s,d)}} \right\rceil.$$

□