A MINIMUM INTERFERENCE ROUTING ALGORITHM[1]

Gustavo Bittencourt Figueiredo

gustavo@ic.unicamp.br

Instituto de Computação, Universidade Estadual de Campinas (UNICAMP)

Avenida Albert Einstein, 1251 - Caixa Postal 13084-971 Campinas-SP, Brasil

Tel: +55 (19) 3788-5878 FAX: +55 (19) 3788-5847

Advisor: Nelson L. S. da Fonseca

nfonseca@ic.unicamp.br

Instituto de Computação, Universidade Estadual de Campinas (UNICAMP)

Avenida Albert Einstein, 1251 - Caixa Postal 13084-971 Campinas-SP, Brasil

Tel: +55 (19) 3788-5878 FAX: +55 (19) 3788-5847

Co-Advisor: José A. S. Monteiro

suruagy@unifacs.br

Universidade Salvador (UNIFACS)

R. Ponciano de Oliveira, 126 - Rio Vermelho CEP 41.950-275 Salvador-BA, Brasil

TEL +55 (071) 330-4663 FAX: +55 (071) 330-4666

[1]This work was partially sponsored by the Brazilian National Council for Research (CNPq).

A Minimum Interference Routing Algorithm

Gustavo B. Figueiredo Nelson L. Saldanha da Fonseca

Institute of Computing

State University of Campinas

Brazil

Email: {gustavo, nfonseca}@ic.unicamp.br

José A. Suruagy Monteiro

NUPERC

Salvador University

Brazil

Email: suruagy@unifacs.br

Abstract

The problems of LSP routing and capacity provisioning in MPLS networks were investigated in [1]. A new minimum interference routing algorithm as well as a new mechanism of dynamic sizing of LSPs in MPLS networks were proposed. Howerver, due to space limitations, only the routing algorithm will be presented in this work. Informations regarding dynamic sizing of LSs can be found in [1], [2].

Minimum Interference Routing algorithms aim at reducing 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.

I. Introduction

In spite of the increase of Internet link capacity, congestion is still common. One of the main reasons for this is an unbalanced use of resources and Traffi c Engineering if often employed to balance this use by mapping flows into network links. MultiProtocol Label Switching (MPLS) is the key to Traffi c Engineering, since it promotes path establishment, thus guaranteeing the Quality of Service (QoS) required by networks flows.

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 consequence, new alternative algorithms, among them the minimum interference routing algorithms, have been proposed to promote a more balanced utilization of resources. The criteria for selecting a path adopted by these algorithms, consider those paths which will result in the least impact on the rejection of future

requests. However, such criteria, with multiple independent QoS requirements, lead to NP-complete problems [3], [4]. As a consequence, various heuristic algorithms have been proposed to deal with such criteria.

Most of minimum interference algorithms are based on the use of a maximum flow type of algorithm to identify critical links, since these links, which belong to the minimum cut set of this pair of nodes, are what is responsible for a reduction in the maximum flow between a 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 [1], [5], which guarantees minimum interference without the need for the execution of repeated max-flow algorithms. Instead, this algorithm uses a modified version of Dijkstra's algorithm, which is less computationally complex than max-flow algorithms, yet produces 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 II introduces the definition of the minimum interference routing problem and describes existing algorithms. Section III introduces the Light Minimum Interference Routing (LMIR) algorithm. Section IV evaluates the performance of the algorithm proposed via simulations. Finally, Section V presents the conclusions.

II. Minimum Interference Routing Algorithms

The central idea behind existing 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 (SD) pairs. The aim is to increase the number of available paths which can accommodate future LSP establishment requests.

Prior to the presentation of the minimum interference problem itself, certain relationships are defined. These are determined as follows: Let the graph $G = (V, E)$ represent a network, where V is the set of vertices (or nodes) and E the set of edges (or links). The flow is the amount of data transmitted between two nodes of G with a positive capacity, $c(u, v)$ assigned to each link $l = (u, v)$ to denote the maximum amount of flow that can pass through (u, v). Each link has a certain residual capacity, which is defined as $c_r(u, v) = c(u, v) - f(u, v)$ where $f(u, v)$ is the amount of flow passing through (u, v). The residual graph, $G_f(V, E_f)$, is composed of the set of links, E_f, for which residual capacities are greater than zero.

A subset of the nodes, denoted by P, involves the ingress/egress nodes which are the end points of the LSPs. The network topology needs to be known to determine the route between an source-destination pair and it is assumed that a link state routing protocol is in place to diffuse the network topology, as well as a routing database for the storing of the values of residual link capacities.

Each request arrival is specified by three variables: S, D and bw, where S denotes the source node, D the egress destination node, and bw the bandwidth required to carry the traffic between S and D.

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 restriction that flow splitting is prohibited.

Two major algorithms are presented in this section: the Minimum Interference Routing Algorithm (MIRA) and *Wang, Su and Chen's* algorithm (WSC).

A. The Minimum Interference Routing Algorithm

The maximum flow of an SD pair is reduced whenever the available bandwidth of an link belonging to the minimum cut set is reduced. The Minimum Interference Routing Algorithm (MIRA) [6] avoids links belonging to the min-cut set of other SD pairs when establishing an LSP between a pair of nodes. MIRA assumes that the network topology and the residual bandwidth of the links are known at the time of an LSP establishment.

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

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

where α_{sd} is the weight of an (s,d) pair and C_{sd} is its set of critical links. The third step of the algorithm involves the execution of the Dijkstra algorithm using $w(u,v)$ as weights.

Kodialam and Lakshman [6] showed 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, thus, limiting the rejection of many future requests. Evidence of the impact of a reduction in maximum flow on blocking probability was presented in their seminal paper [6], which shows that their algorithm produces lower blocking probability than do algorithms which do not take interference into account.

B. The Wang, Su and Chen Algorithm

The algorithm proposed by *Wang, Su and Chen* [7] 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. [7], the major drawback of MIRA is that critical links identification focuses on a single SD pair and does not detect the critical nature of an 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 just one 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

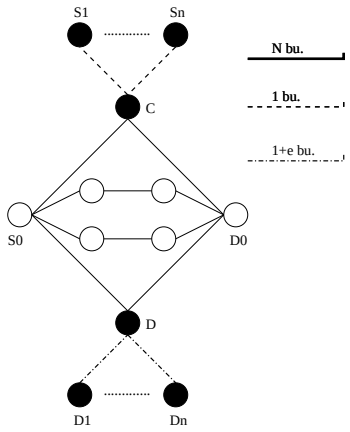

Fig. 1. Graph with concentrating vertices.

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

In order to avoid paths through concentrating vertices, Wang et al., proposed an algorithm with a weight function that takes the link contribution to the maximum flow into consideration. 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 max-flow contribution. In this case, the link belongs to the min cut set. Higher weights values are, then, assigned to those links.

In their algorithm, link weights are computed as

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

where:

- $c_r(u,v)$ is the residual link capacity,
- $\theta^{s'd'}$ represents the maximum flow between pair (s',d'),
- $f_{(u,v)}^{s'd'}$ is the contribution of the link to the maximum flow of (s',d').

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 [7]. Results derived via simulation indicate that the algorithm of Wang et al. produces slightly fewer rejections than does MIRA although to avoids denials resulting from either concentrating and distributing vertices.

The crucial step in relation to the efficiency of these algorithms is the use of max-flow algorithms in the identification of critical links. MIRA [6] uses maximum flow in conjunction with the minimum cut set to identify

critical links whereas the WSC algorithm [7] detects critical links from their contribution to the maximum flow. Executing such max-flow algorithms, however, is prohibitively expensive for large networks. The *Edmonds-Karp* algorithm, which is the most well known implementation of the *Ford-Fulkerson*'s max-flow computation method [8], 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 [9], the number of links is significantly greater than that of vertices, so that the *Edmonds-Karp* algorithm performs very inefficiently in dense graphs.

Goldberg's algorithm, which is the fastest known max-flow 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] [10]. Other approaches that do not use max-flow computation are thus necessary to reduce the complexity of minimum interference algorithms.

III. Light Minimum Interference Routing Algorithm

In this section, the Light Minimum Interference Routing (LMIR) algorithm is presented. The main characteristic 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 max-flow 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 SD 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 in the LMIR algorithm involves the identification of K least capacity paths with the use of the LowestCapacities algorithm, a variation of 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 involving 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). As nodes are removed from the queue using procedure *extract_min(Q)* and their adjacent nodes processed the algorithm progresses.

For each step of the LowestCapacities algorithm, the metric 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) and is either smaller than the value of $F[v]$ or is the same, but located at distance at least one hop less 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.

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: $Q \leftarrow Q \bigcup v$

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

$$w(u, v) = \sum_{(s',d') \in P} \frac{f_G^{(s',d')}}{c_r(u, v)}, \qquad (u, v) \in E_r, \tag{3}$$

where $c_r(u, v)$ is the residual capacity of link (u, v) and $f_G^{(s',d')}$ is the flow between s' and d', the path with the lowest capacity.

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

The complexity of the LMIR algorithm can be analyzed as follows. Step 6 of *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 $O(E_f)$ in the worst possible case, as would happen

Algorithm 2 LMIR

INPUT

A residual graph $G_r = (V, E_r)$ and $r(s, d, bw)$ and 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 (3) for all the 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 weights.
5: Create an LSP connecting s to d and update the capacity of the links.

when the queue of vertices with non-visited adjacent vertices is implemented as a linked list. Thus, the complexity of the LMIR algorithm is, $O(V^2 + E_f) = O(V^2)$ [8].

Since each pair requires K executions of the LowestCapacities algorithm, the complexity of the LMIR algorithm for identifying critical links is $O(K \cdot V^2) = O(V^2)$, whereas in the other existing minimum interference algorithms the complexity involves the performance of a max-flow algorithm which has a complexity of $O(\min(V^{2/3}, E_f^{1/2}) \cdot E \cdot \log(V^2/E_f) \cdot \log(U))$. Thus, the following theorem can then be affirmed:

Theorem 1: The computational complexity of the LMIR algorithm is upperbounded by the complexity of the MIRA algorithm.

Proof: See Appendix I. ■

The number of least capacity paths considered by the LMIR algorithm has a tremendous impact 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:

$$K(\theta) = d(u), \text{ if } \theta^{(u,v)} = \sum_{\forall i | (u,i) \in E(G)} c(u, i), \tag{4}$$

or

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

or

$$K(\theta) = \left\lceil \frac{\theta^{(u,v)}}{\omega^{(u,v)}} \right\rceil. \tag{6}$$

where: $\omega_{(G)}^{(u,v)}$ is the flow along the path of least capacity between u and v in G, and $\theta_{(G)}^{(u,v)}$is the maximum flow between them.

Proof:

See Appendix II ■

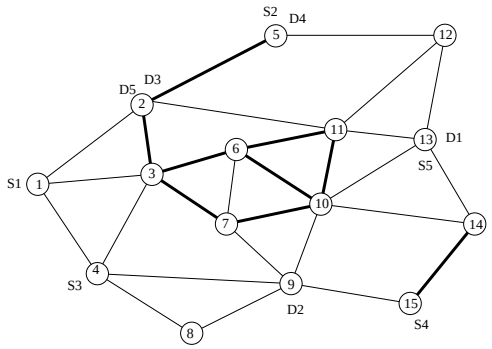

Fig. 2. Topology used in simulations for small networks.

IV. A COMPARISON BETWEEN THE LMIR ALGORITHM AND OTHER EXISTING MINIMUM INTERFERENCE ALGORITHMS

Simulation experiments were used 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 can be assessed. MHA is actually the Dijkstra algorithm; the WSP algorithm identifies either the path with maximum capacity in the network or that with the least number of links.

Both, large and small networks were considered in this study. The small network used in both [6] and [7], was utilized so that a comparison could be made to previously published results, while the topology for the large networks was randomly generated.

The topology used in the small network experiments as well as the source-destination pairs are shown in Figure 2. The lighter bi-directional links represent a capacity of 1,200 bandwidth units each, while darker ones have capacity of 4,800 bandwidth units each, corresponding to transmission rates of OC-12 and OC-48, respectively. Eight thousand requests among the five SD pairs were randomly generated using a uniform distribution in the interval $[1, 4]$ with long-lived LSPs assumed so that once accepted, resources are held until the end of the experiments.

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 WSP and MHA algrithms, since the latter two do not consider the criticality of link in choosing a paths.

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

The importance of the non-interference approach can be seen in the large reduction resulting from the application of the WSP and MHA algorithms. This reduction is a direct consequence of the unnecessary allocation of critical links, greatly increasing the impact on the maximum flow.

Figure 4 shows the number of requests rejected as a function of the number of requests arriving. The WSP

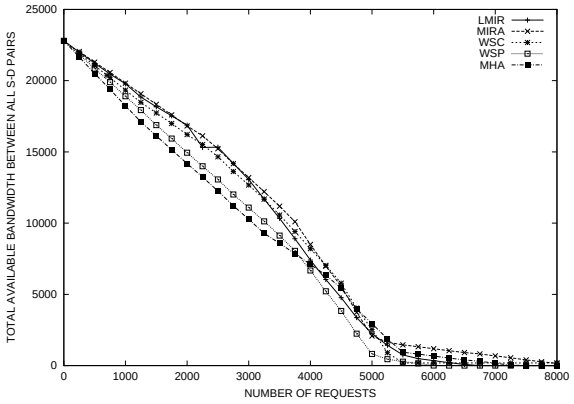

Fig. 3. Total bandwidth for all network source-destination pairs.

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 the lowest number of requests rejected, which proves the benefi ts of the non-interference approach.

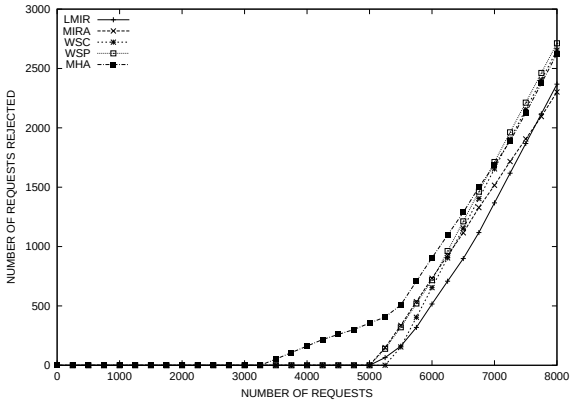

Fig. 4. Number of rejections.

The two previous examples have illustrated the performance of different algorithms in relation to the total flow in a network. Another interesting question is how these algorithms influence the flow of an individual SD 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 the other SD pairs. It can be observed that the use of the MHA and WSP algorithms leads to a reduction in the maximum flow as soon as the fi rst request arrives. Network saturation when using the MHA algorithm, is reached immediately after the arrival of the $2,000^{th}$ request. From there on, the path is saturated. The WSP algorithm takes a somewhat longer period to reach saturation since when one path is approaching saturation it chooses other high-capacity.

The minimum interference algorithms do not provide any interference under low-load conditions (up to the arrival of the $2,500^{th}$ request), but both the LMIR and MIRA algorithms produce less interference than the WSC

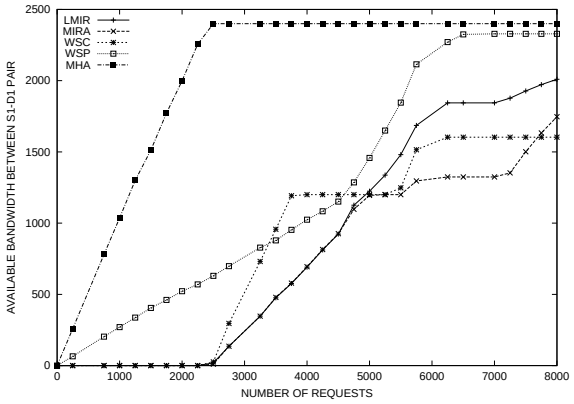

Fig. 5. Effects on pair $(S1, D1)$.

algorithm for as many as 5.000 requests, although in the long term the use of the WSC algorithm results in a less of a reduction in the maximum flow of the pair than do the LMIR and MIRA algorithms which produce similar interference up to the $5,000^{th}$ request since they choose the same set of critical links. After the arrival of the $5,000^{th}$ request, however, 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 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 values to them.. Figure 6 shows the number of requests rejected as a function of the choice of K. This shows that the minimum number of requests rejected are when $K = 5$. Increasing the value of K does not lead to any improvement, yet makes the identification of critical links difficult.

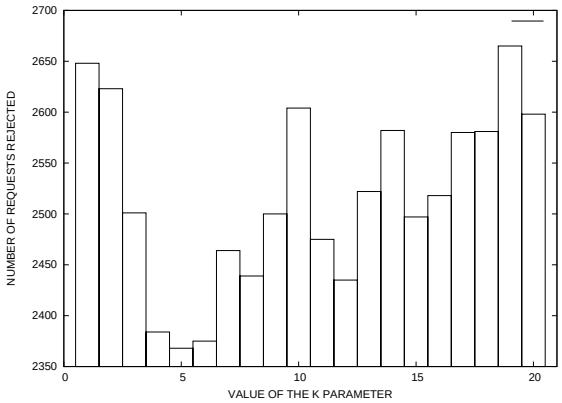

Fig. 6. Influence of parameter K in the number of rejected connections.

For the speed of performance of an even large number of requests, the algorithms were tested for 8,000 requests measured with the Linux *time* command in an Intel Celeron machines with a 1.2 GHz clock, and 256 MB of RAM. Table I shows the values measured.

TABLE I

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 aproximately 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 to 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, the faster is the execution time. However, the results tend to differ signifi cantly from the optimum. Selecting a larger number of least capacity paths than the optimum only increases the execution time.

Large networks simulations were also conducted with the LMIR algorithm using networks with 30, 40, and 50 vertices. For these large networks, both sparse and dense, only minimum interference algorithms were tested. Networks are modeled as non-oriented graphs in which each link has a non-negative capacity. For this simulations, network topologies were created using Waxman's method [9], [11]. In this method, the probability of the existence of link between u and v is given by

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

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 choice for each bandwidth size (1,200 or 4,800) was made.

LSPs are again assumed to be long-lived and 30,000 requests were and only generated among the fi ve source-destination pairs with the bandwidth requested uniformly distributed in the interval $[1, 4]$. Scenarios with low between 0 and $10,000$, mean between $10,000$ and $20,000$ and high between $20,000$ and $30,000$ loads were considered.

A. Dense 30-vertex networks

Figure 7 shows the maximum flow reduction for the selected source-destination pairs for dense 30-vertex networks, with all algorithms evaluated performing at a similar level. This performance worsens under high loads, 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 having available capacities are subject to allocation. The difference

in the performance of the algorithms can be attributed to the way they identify critical links, which leads to different approachs to saturation.

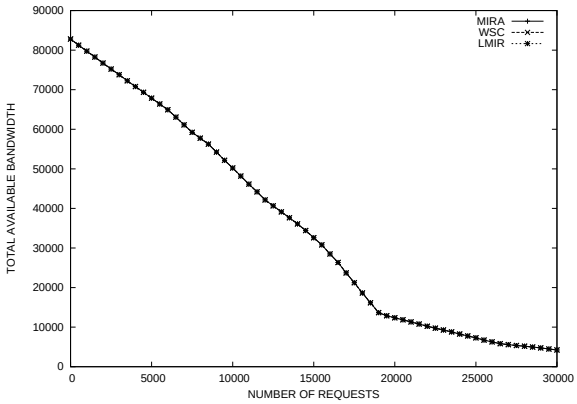

Fig. 7. Total bandwidth (dense 30-vertex networks).

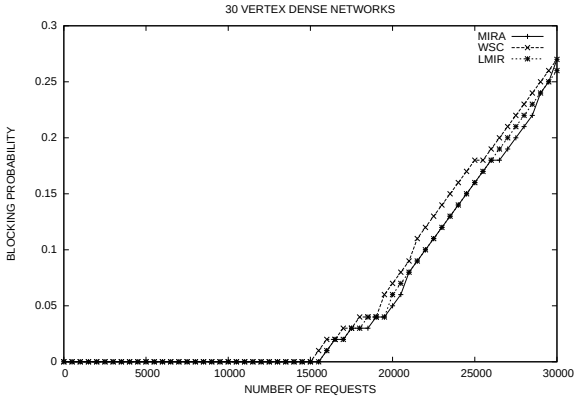

Fig. 8. Blocking Probability (dense 30-vertex networks).

B. 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.

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.

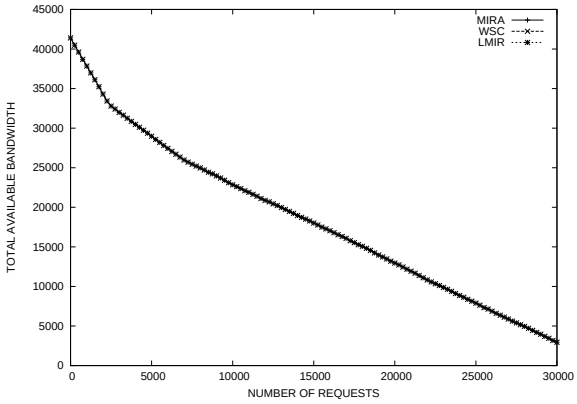

Fig. 9. Total bandwidth (sparse 30-vertex networks).

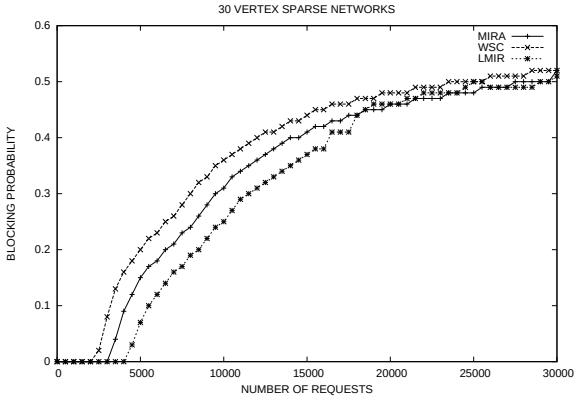

Fig. 10. Blocking Probability (sparse 30-vertex networks).

C. Dense 50-vertex networks

The dense 50-vertex networks used here have 544 links which implies on a large number of paths to choose from. As a consequence, saturation was not observed in the simulation experiments. To achieve saturation in such networks, all of the links are 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 algirihms, and for high loads their performance were almost indistinguishable.

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

D. Sparse 50-vertex networks

Sparse networks with 50 vertices (Figure 13)are subject to a slower reduction in the maximum flow when the LMIR algorithm was used than when the WSC and MIRA. With 30 nodes, saturation is reached rapidly due to the

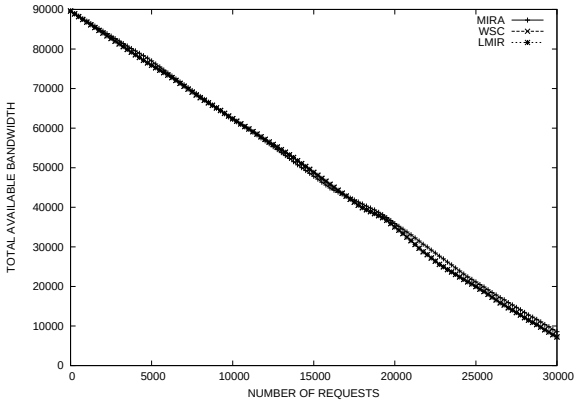

Fig. 11. Total bandwidth (dense 50-vertex networks).

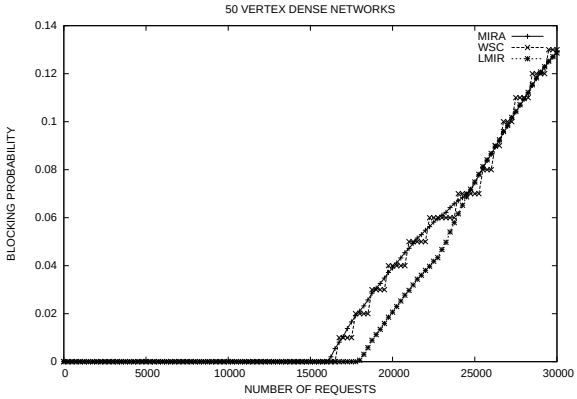

Fig. 12. Blocking Probability (50 vertices dense networks).

saturation of critical links. This rapid saturation 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 rejection. The lowest blocking probability is produced by the LMIR algorithm.

The relative gain in execution time for the minimum interference algorithms studied here was determined and is shown in Table II. The highest value (produced by either the MIRA or WSC algorithms) is used as a standard (0% gain). These results show that the LMIR algorithm is the fastest, with reductions of up to 39% when five least capacity paths were involved, although this performance deteriorates for a larger number of paths. In all the simulations, the most propitious was five as this resulted in the fastest execution time while producing blocking probability values similar to those obtained using the MIRA and WSC algorithm.

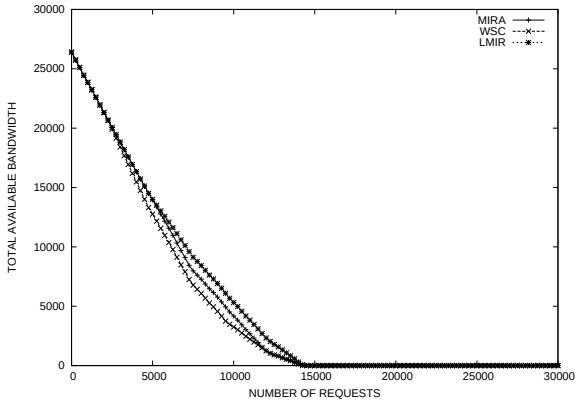

Fig. 13. Total bandwidth (sparse 50-vertex networks).

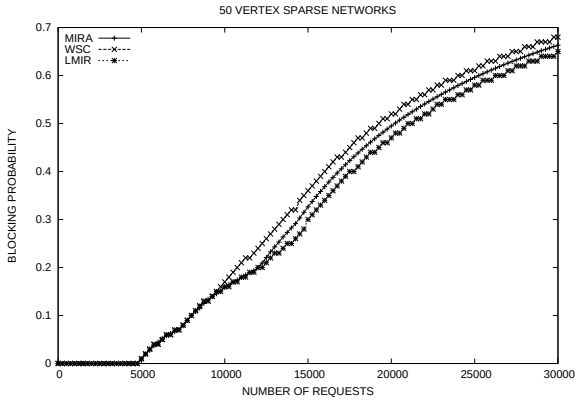

Fig. 14. Number of requests rejected (sparse 50-vertex networks).

V. Conclusions

In this paper, a new minimum interference routing algorithm which does not use max-flow algorithms for critical link identification, was presented. This LMIR algorithm uses a modified Dijkstra algorithm to identify paths with the least capacity and these paths are 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 impact on its performance. The best results in simulation experiments were obtained when this was limited to five.

The results obtained using the LMIR algorithm are similar to those obtained with the MIRA and WSC algorithms. Moreover, for large dense networks, it produced the lowest probability of blocking as well as minimal reductions in maximum flow. Furthermore, the LMIR algorithm involves some 39% less computational time than the MIRA and WSC algirithms. 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.

Acknowledgment

This work was partially sponsored by the Brazilian National Council for Research (CNPq).

TABLE II

RELATIVE EXECUTION TIME GAINS FOR INDIVIDUAL CONNECTIONS

	50 (D)	50 (E)	40 (D)	40 (E)	30 (D)	30 (E)
Standard	6.836×10^{-3}	5.683×10^{-3}	4.851×10^{-3}	5.740×10^{-3}	4.079×10^{-3}	6.477×10^{-3}
MIRA	0.00	1%	1%	0.00	0.00	0.00
WSC	2%	0.0	0.00	0%	2%	20%
LMIR K =5	39%	17%	13%	36%	2%	28%
LMIR K =6	30%	12%	5%	23%	-1%	22%
LMIR K =7	28%	5%	-7%	18%	-6%	19%
LMIR K =8	27%	3%	-9%	-1%	-11%	11%
LMIR K =9	10%	0%	-22%	-6%	-20%	4%
LMIR K =10	2%	-6%	-24%	-6%	-29%	-2%

REFERENCES

[1] G. B. Figueiredo, "Algoritmos de Roteamento com Interferência Mínima," Master's thesis, Instituto de Computaç ˜ao - UNICAMP. http://www.ic.unicamp.br/˜ gustavo/dissertacao, 2003.

[2] G. B. Figueiredo, J. A. S. Monteiro, N. L. S. da Fonseca, and A. A. A. Rocha, "Dynamic Sizing of Label Switched Paths in MPLS Networks," in *IEEE International Telecommunications Symposium*, 2002, pp. 593–598.

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

[4] S. Chen and K. Nahrstedt, "QoS Routing for Next-Generation Networks," *IEEE Network*, Dezembro 1998.

[5] G. B. Figueiredo, N. L. S. da Fonseca, and J. A. S. Monteiro, "A Minimum Interference Routing Algorithm," in *To appear in proceedings of IEEE International Conference on Communications, Paris -France*, 2004.

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

[7] B. Wang, X. Su, 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.

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

[9] 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.

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

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

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

APPENDIX I

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 also the same a 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 max-flow algorithm known.

The proof is carried out for both sparse and dense graphs. For **dense graphs** it is assumed that $|E_f| = |V|^2$. 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

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

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

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

For **sparse graphs** it is assumed that $|E_f| = |V|$. In this case, Q (the queue of non visited nodes) can be implemented as a heap [8]. 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 $\log(V^2/E_f) \cdot \log(U) > 1$, using the first two factors of $O(\min(V^{2/3}, E_f^{1/2}) \cdot E_f \cdot \log(V^2/E_f) \cdot \log(U))$, it can be shown that

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

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

Discarding V in both algorithms and taking the log gives:

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

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

Hence, $\log(\log(V)) = O(1/2 \cdot \log V) \Rightarrow$ *LowestCapacities algorithm* $= O($Goldberg's max-flow$)$.$\square$

This shows that the critical links detection with the LMIR algorithm has a lower computational complexity than does any other minimum interference algorithm using max-flow algorithms. ■

APPENDIX II

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

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

or

$$K(\theta) = d(v), \text{ if } \theta^{(u,v)} = \sum_{\forall i | (i,v) \in E(G)} c(i,v), \tag{8}$$

or

$$K(\theta) = \left\lceil \frac{\theta^{(u,v)}}{\omega^{(u,v)}} \right\rceil. \tag{9}$$

where: $\omega_{(G)}^{(u,v)}$ is the flow along the path of least capacity between u and v in G, and $\theta_{(G)}^{(u,v)}$ is the maximum flow between u and v.

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 disconnected graphs. Before showing the proof for the lower bound value, some 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 [12].

Definition 2: Two paths from u to v are internally disjointed if they have no common internal vertices (edges) [12].

Theorem 3: Let $\lambda(u, v)$ be the number of u, v-internally disjointed paths (or just disjointed 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: Let P and P' be two internally disjointed paths between u and v. Now, suppose by contradiction that $\lambda(u,v) > \min\{d(u), d(v)\}$. If $\lambda(u,v) > \min\{d(u), d(v)\}$, this implies that at least one edge incident to u or v was included in both of the internally disjointed paths, P and P'. However, if P and P' have an edge in common they are not 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 [12] uses these concepts of disconnecting sets and disjointed paths:

Theorem (Menger-1927) 4: 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 [12]. ■

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 15 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 with a capacity one.

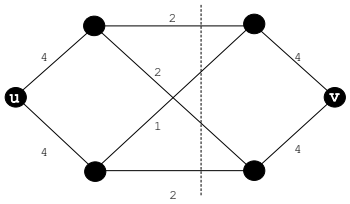

Fig. 15. Number of disjoint paths less than the number of edges in the min cut set.

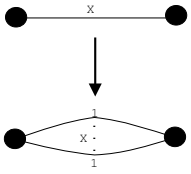

Fig. 16. Transformation in the original network.

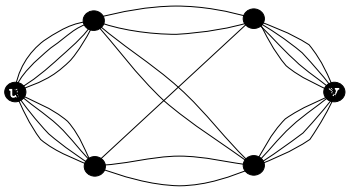

Fig. 17. Transformed network for Figure 15.

Thus, a new theorem relating maximum flow and Menger's theorem (Theorem 4) is presented:

Theorem 5:

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

where $\lambda(G')$ is the number of internally disjointed paths between u and v, θ^{uv} is the maximum flow 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 [12]. ■

Theorem 6: Let $f_G^{(u,v)}$ be a flow passing through a path of least capacity 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 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:

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

The Max-flow Min-cut theorem [12] implies that the maximum network flow is bounded by the minimum capacity of a source/sink cut. Thus 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:

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

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

Case 3:

Let

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

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

$$n = \theta^{(u,v)}_{(G)} = n \cdot f^{(u,v)}_{G'} = \sum_{i=1}^{n} f^{(u,v)}_{G'},$$

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

$$n = \theta^{(u,v)}_{(G)} \geq K(\theta) \cdot \sum_{i=1}^{\omega} f^{(u,v)}_{G'}. \tag{10}$$

The application of Theorem 6 in Equation 10 gives the following:

$$K(\theta) \cdot \omega^{(u,v)}_{(G)} \leq \theta^{(u,v)}_{(G)}. \tag{11}$$

Since in G both $\theta^{(u,v)}_{(G)}$ and $\omega^{(u,v)}_{(G)}$ are known, we can defi ne $K(\theta)$ as

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

■