Um Algoritmo para Roteamento com Interferência Mínima

Gustavo B. Figueiredo[1]*, Nelson L. Saldanha da Fonseca[1] , José A. Suruagy Monteiro[2]

[1]Instituto de Computação – Universidade Estadual de Campinas
Caixa Postal 6176 – 13083-970 Campinas, SP

[2]NUPERC – Universidade Salvador
R. Ponciano de Oliveira, 126 – 41950-275 Salvador, Ba

gustavo.figueiredo@ic.unicamp.br, nfonseca@ic.unicamp.br, suruagy@unifacs.br

Resumo. *O roteamento com interferência mínima é de suma importância para o processo de Engenharia de Tráfego na Internet com MPLS, dada a necessidade de se atender a demanda por estabelecimento de Label Switched Paths (LSP's). Este trabalho apresenta um algoritmo de roteamento com interferência mínima denominado Light Minimum Interference Routing (LMIR). Diferentemente dos algoritmos de interferência mínima existentes, o LMIR usa uma variação do algoritmo de Dijkstra para a determinação das arestas críticas. Resultados obtidos através de simulações, indicam que a baixa complexidade computacional do LMIR não interfere na precisão do algoritmo.*

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

1. Introdução

O processo de Engenharia de Tráfego desempenha um importante papel no oferecimento de Qualidade de Serviço na Internet. O aumento da largura de banda dos enlaces da Internet não elimina por completo a ocorrência de congestionamentos causados pela distribuição não balanceada do tráfego na rede [Banerjee and Sidhu, 2002].

Por ser uma convergência do paradigma de encaminhamento orientado à conexão e do modelo sem conexão utilizado nas redes IP, a tecnologia *MultiProtocol Label Switching* (MPLS) tem sido apontada como fundamental no processo de Engenharia de Tráfego. Dentre as vantages do MPLS, menciona-se a capacidade de criar dinamicamente LSP's (*Label Switched Paths*) explícitos a baixo custo operacional. O dimensionamento e o particionamento do tráfego entre múltiplos LPS's são os principais desafios da Engenharia de Tráfego baseada no MPLS. A essência da Engenharia de Tráfego é o mapeamento dos fluxos em uma estrutura física, sendo a seleção dos caminhos o cerne deste processo [Banerjee and Sidhu, 2002].

A escolha de caminhos que satisfaçam múltiplos requisitos independentes de QoS é um problema NP-completo, ou seja, não existe (ou não se conhece) algoritmo polinomial para resolvê-lo. A solução para problemas de roteamento baseado em QoS na

*Trabalho parcialmente financiado pelo CNPq, processo 300064/95-0

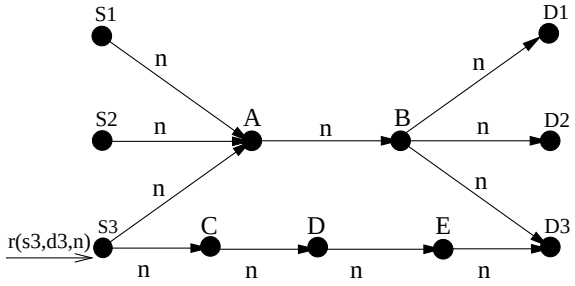

Figura 1: Interferência: exemplo

Internet passa pela adoção de heurísticas que encontrem caminhos factíveis em tempo real.

A escolha de um caminho para o tráfego entre um par origem-destino pode seguir critérios diversos. Os algoritmos de interferência mínima o fazem de forma a minimizar a probabilidade de bloqueio das (futuras) requisições para estabelecimento de caminhos. A minimização da probabilidade de bloqueio é realizada através da minimização da redução do fluxo máximo entre outros pares origem-destino. A redução no fluxo máximo entre pares origem-destino devido ao roteamento de uma conexão específica entre um outro par origem-destino é chamada interferência.

Diversos trabalhos vêm sendo propostos para resolver o problema de roteamento com interferência mínima. O presente trabalho apresenta uma nova heurística para o problema de roteamento com interferência mínima, denominada *Light Minimum Interference Routing* (LMIR), que otimiza a utilização dos recursos da rede com baixa complexidade computacional. Assim como os algoritmos de interferência mínima existentes, o LMIR é um algoritmo *online* e independente dos padrões de tráfego. Sua principal vantagem em relação às soluções anteriores é a baixa complexidade computacional, possibilitando maior escalabilidade.

O restante do artigo está organizado da seguinte maneira: a seção 2, apresenta algoritmos existentes de roteamento com interferência mínima. A seção 3, descreve o *LMIR* e analisa sua complexidade assintótica. A seção 4, exemplifica o algoritmo LMIR e conclusões são apresentadas na seção 5.

2. Algoritmos de roteamento com interferência mínima

Os algoritmos de roteamento com interferência mínima objetivam reduzir a probabilidade de bloqueio da rede, encontrando caminhos que minimizem a redução do fluxo máximo entre outros pares origem-destino.

A idéia central dos algoritmos de interferência mínima é que quanto maior for o fluxo máximo disponível entre um par origem-destino menor será a probabilidade de bloqueio das requisições desse par. Assim, os algoritmos de interferência mínima tentam rotear uma requisição de uso de banda passante por caminhos que maximizem o fluxo máximo disponível entre outros pares origem-destino [Kodialam and Lakshman, 2000], de forma a poder acomodar futuras requisições. Este é um problema NP-Difícil. Existem, porém, heurísticas precisas para este problema. A Figura 1 ilustra o conceito de interferência mínima. Após uma requisição de um caminho entre *S3-D3*, com ¨*n*¨ unidades de banda, o algoritmo tenta escolher o melhor caminho. Se o caminho escolhido for *S3 - A - B - D3*, então todas as futuras requisições entre os pares *S1-D1* e *S2-D2* são perdidas. Assim, o melhor caminho é *S3 - C - D - E - D3*, mesmo sendo mais longo que *S3 - A - B - D3*.

O primeiro algoritmo publicado que introduziu a terminologia "interferência mínima", foi o *Minimum Interference Routing Algorithm - MIRA* [Kodialam and Lakshman, 2000]. Este algoritmo assume que os pares origem-destino são conhecidos, bem como a topologia. Além disso, ele assume a existência de protocolos de sinalização responsáveis pela difusão das informações sobre topologia, banda residual e rotas explícitas.

O fluxo máximo disponível entre um par origem-destino é conhecido como *maxflow* e associado a ele está um conjunto de corte mínimo [Cormen et al., 1990]. O *maxflow* de um par origem-destino diminui quando a capacidade de uma aresta que compõe um conjunto de corte mínimo diminui. Sempre que uma nova requisição é aceita entre um par origem-destino, o fluxo máximo desse par diminui já que o fluxo obrigatoriamente passa por uma aresta de corte. A idéia do algoritmo *MIRA* é evitar rotear as requisições por arestas que pertencem ao conjunto de corte mínimo de outros pares *origem-destino*.

O algoritmo *MIRA* pode ser descrito como segue. Seja um grafo *G=(V,E)*, com $|V|$ vértices e $|E|$ arestas. Seja *B* o conjunto das capacidades residuais dos enlaces e seja *r(a,b,bw)* uma requisição de *¨bw¨* unidades de banda entre o par (a, b). O algoritmo *MIRA* retorna uma rota entre (a, b) com capacidade *¨bw¨*.

O algoritmo possui três etapas. A primeira compreende a computação do *maxflow* entre todos os pares $(s, d) \in P \backslash (a, b)$. A segunda etapa inclui o cálculo das arestas críticas e a atribuição dos pesos às mesmas. Os pesos são calculados segundo a expressão:

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

onde α_{sd} é o peso do par (s, d) e C_{sd} é o conjunto de arestas críticas. O uso de peso para pares origem-destino pode ser especialmente útil quando se quer atribuir prioridades entre os mesmos. A terceira etapa do algoritmo é a execução do algoritmo de *Dijkstra*, usando $w(l)$ como peso das arestas.

Um outro algoritmo de interferência mínima foi proposto por Wang, Su e Chen [Su and Chen, 2002]. Ele é baseado no algoritmo *MIRA* e tem as mesmas premissas básicas, ou seja, a existência de protocolos especializados na difusão das informações de estado e conhecimento da topologia da rede. Suas principais diferenças estão na detecção da criticalidade das arestas e na função de custo.

Segundo Wang et. al [Su and Chen, 2002], o algoritmo *MIRA* pode falhar pois está centrado na detecção de enlaces críticos somente para um único par origem-destino. Assim, ele não pode detectar a criticalidade de um enlace para um grupo de pares, rejeitando muitas requisições em topologias específicas como em grafos com vértices concentradores ou vértices distribuidores. Por essas razões, os autores propuseram uma maneira de identificar o impacto relativo à utilização dos enlaces nas futuras requisições. A idéia é calcular a contribuição do enlace em relação ao *maxflow* entre um par origem-destino, evitando os enlaces de menor contribuição. Os pesos utilizados nas arestas são calculados por

$$w(l) = \sum_{(s',d') \in P} \frac{f_l^{s'd'}}{\theta^{s'd'} \cdot R(l)}, \qquad l \in E \tag{2}$$

onde $R(l)$ é a capacidade residual do enlace l, $\theta^{s'd'}$ representa o fluxo máximo entre o par $(s'd')$ e $f_l^{s'd'}$ é a contribuição da aresta l no fluxo máximo entre o par $(s'd')$.

Uma vez computados os pesos, as arestas que têm capacidade residual inferior à banda requisitada são eliminadas e o algoritmo de *Dijkstra* é executado usando $w(l)$ como peso [Su and Chen, 2002] .

A detecção das arestas críticas é, sem dúvida, a tarefa crucial dos dois algoritmos apresentados nesta seção pois é o que determina a sua eficiência. Os dois algoritmos apresentados utilizam o *maxflow* para a detecção das arestas críticas. O algoritmo *MIRA* [Kodialam and Lakshman, 2000] utiliza o conjunto de corte mínimo associado ao *maxflow* para realizar esta detecção, ao passo que o algoritmo de Wang, Su e Chen apresentado em [Su and Chen, 2002] baseia-se na contribuição de cada enlace para realizar a determinação de criticalidade dos mesmos.

A necessidade de execução do *maxflow* majora a complexidade de ambos os algoritmos mencionados, sendo uma tarefa custosa e proibitiva em redes de médio/grande porte. O método mais comum para cálculo do *maxflow* é o método de *Ford-Fulkerson* [Cormen et al., 1990]. O algoritmo de *Edmonds-Karp* é a sua implementação mais conhecida e usa uma busca em largura no grafo para descobrir os caminhos aumentantes. Sua complexidade é $O(V \cdot E^2)$.

Em redes como a Internet, o grau médio (*average degree*) chega a ser igual a 3,5 [Zegura et al., 1996]. Em outras palavras, o número de arestas é significativamente maior que o de vértices, o que leva o algoritmo de *Edmonds-Karp* a trabalhar próximo do pior caso (grafos densos). O algoritmo de *maxflow* mais rápido é o de Goldberg et. al [Goldberg and Rao, 1998]. Ele tem complexidade igual a $O(\min(V^{2/3}, E^{1/2}) \cdot E \cdot \log(V^2/E) \cdot \log(U))$ em uma rede com $|V|$ vértices, $|E|$ arestas e enlaces com capacidades no intervalo [1,U] [Goldberg and Rao, 1998]. Apesar do algoritmo de Goldberg apresentar uma redução significativa na complexidade do cálculo do *maxflow*, outras abordagens que evitem o uso de algoritmos de *maxflow* podem ser utilizadas para reduzir ainda mais a complexidade dos algoritmos de interferência mínima, tal como é feito neste artigo.

Além do cálculo do *maxflow*, os algoritmos de interferência mínima demandam também o tempo para determinação da criticalidade das arestas, alocação dos pesos e mais $O(V^2 + E)$ para execução do algoritmo de *Dijkstra*. Contudo, a complexidade computacional dessas etapas ainda é menor que a do algoritmo do *maxflow*.

3. O algoritmo *Light Minimum Interference Routing*

Como mencionado anteriormente, os algoritmos de *maxflow* podem causar um impacto significativo na complexidade dos algoritmos de roteamento com interferência mínima. Introduz-se nesta seção, um novo algoritmo denominado *Light Minimum Interference Routing (LMIR)* para roteamento com interferência mínima que não utiliza algoritmos *maxflow*.

O LMIR assume a existência de protocolos de sinalização (ex. RSVP-TE ou CR-LDP) responsáveis pela atualização das informações de banda residual dos enlaces, *R(l)*, e mudanças de topologia ou falhas nos links. Ele tenta solucionar o problema através de uma abordagem diferente, buscando encontrar os *K* caminhos com menor capacidade entre todos os pares origem-destino para então determinar as arestas críticas. É importante ressaltar que o LMIR não assume nenhum conhecimento sobre requisições futuras nem dos perfis estatísticos do tráfego.

A idéia é que o caminho entre um par origem-destino com menor capacidade, contém a aresta de menor capacidade, ou seja, a aresta crítica. Dessa forma, encontrar os *K* piores caminhos significa encontrar as *K* arestas críticas entre o par origem-destino em questão. Assim, o algoritmo LMIR busca as *K* arestas críticas entre cada par origem-

destino e procura evitá-las.

O algoritmo LMIR tenta encontrar rotas que minimizem a interferência entre os pares origem-destino da rede, gerando uma utilização balanceada dos recursos da mesma. No algoritmo LMIR, ao chegar uma requisição de bw unidades de banda passante para um par origem-destino *(s,d), ri(s,d,bw),* as *K* arestas críticas entre os outros pares origem-destino devem ser encontradas. Esta busca por arestas críticas é feita, encontrando-se os *K* caminhos com menor capacidade. Dado que o fluxo de um caminho é limitado pela aresta de menor capacidade, os *K* caminhos em questão contêm as *K* arestas críticas.

Note que para se alcançar uma utilização balanceada dos caminhos entre um par origem-destino e maximizar o fluxo, as arestas críticas entre esse par devem ser evitadas se houver mais de um caminho entre a origem e o destino. Dessa forma, se a utilização da aresta crítica (u, v) não for evitada, o caminho P ao qual (u, v) pertence pode ser saturado e provocar uma utilização desbalanceada mesmo que existam outros caminhos disponíveis na rede. Além disso, se (u, v) pertencer ao conjunto de corte mínimo, ela deverá ser evitada pois dentre as arestas que reduzem o fluxo máximo, ela tende a ser saturada mais rapidamente.

Depois de encontrados os *K* caminhos de menor capacidade, todas as arestas que os compõem recebem pesos calculados de acordo com a equação (2). A utilização destes pesos permite que as arestas dos caminhos críticos recebam pesos inversamente proporcionais à sua capacidade residual.

A última etapa do algoritmo é a execução do algoritmo de *Dijkstra* usando $w(l)$ como peso. Esta etapa permite a seleção de arestas não críticas ou arestas com baixa criticalidade. A seguir o algoritmo LMIR é apresentado. No algoritmo, δ representa o conjunto dos pares origem-destino.

Algoritmo 1 LMIR

ENTRADA

Um grafo residual $G = (V, E)$ e $ri(s, d, bw)$, uma requisição de bw unidades de banda entre o par s, d.

SAIDA

Rota entre s e d com bw unidades de banda

LMIR

1: Encontre os ¨*K*¨ caminhos com menor capacidade $\forall(s', d') \in \delta$.
2: Calcule os pesos das arestas pertencentes aos caminhos encontrados, de acordo com a equação (2).
3: Elimine as arestas com capacidade residual menor do que *bw*.
4: Execute o algoritmo de *Dijkstra* usando $w(l)$ como peso.
5: Estabeleça o LSP entre *s* e *d* e atualize a capacidade das arestas.

O passo 1 do algoritmo LMIR é uma busca por caminhos de menor capacidade, onde δ é o conjunto de todos os pares origem-destino. O algoritmo usado para fazer isso é uma variação do algoritmo de *Dijkstra,* denominada ¨MenorCapacidade¨. Este algoritmo é mostrado a seguir:

No algoritmo MenorCapacidade v é um vértice qualquer da rede, D é um vetor que contém a capacidade mínima encontrada no caminho entre s e v. O vetor π contém o vértice predecessor de v e o vetor di indica a distância em número de arestas entre s e v. Q representa a lista de vértices cuja adjacência ainda não foi visitada, a função *extract_min(Q)* retorna o elemento $u \in Q$ cujo valor $D[u]$ é o menor e ADJ representa a lista de adjacência do vértice v.

Algoritmo 2 MenorCapacidade

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

Assim como nos algoritmos de interferência apresentados na seção 2, a identificação das arestas críticas é a tarefa que mais impacta na complexidade computacional do algoritmo LMIR. A atribuição dos pesos e o algoritmo de *Dijkstra* impactam da mesma forma a complexidade de todos os algoritmos de roteamento com interferência mínima em questão. Portanto, a diferença entre as complexidades dos três algoritmos de roteamento com interferência mínima apresentados neste artigo é devida à forma como as arestas críticas são identificadas. Como já mencionado, o algoritmo LMIR usa o algoritmo MenorCapacidade, para encontrar as arestas críticas. Uma análise da complexidade deste algoritmo é apresentada a seguir.

O passo 2 do algoritmo **MenorCapacidade** é executado $|V|$ vezes, já que todos os vértices são visitados. O passo 5 é executado $|V|$ vezes resultando em uma complexidade de $O(V^2)$. Além disso, o passo *extract_min(Q)* em MenorCapacidade tem complexidade igual a $O(E)$ no pior caso, se a fila Q de vértices cuja adjacência ainda não foi visitada for implementada como uma lista encadeada. Assim, a complexidade total do algoritmo é de $O(V^2 + E) = O(V^2)$ [Cormen et al., 1990]. Como cada par executa este algoritmo *K* vezes, a complexidade para identificar arestas críticas seria $O(K \cdot V^2)$ = $O(V^2)$, enquanto que os outros algoritmos de interferência mínima apresentam, na etapa de identificação de arestas críticas, complexidade igual a $O(\min(V^{2/3}, E^{1/2}) \cdot E \cdot \log(V^2/E) \cdot \log(U))$.

Para mostrar que a complexidade do MenorCapacidade é inferior à do *maxflow* de Goldberg et al, que é o algoritmo de *maxflow* mais rápido, a análise da complexidade de ambos será feita em separado para grafos densos e esparsos.

Em grafos densos, pode-se assumir $|E| = |V|^2$ assim, o algoritmo de Goldberg et al. ficaria igual a

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

enquanto que a do MenorCapacidade tem complexidade igual a $O(V^2)$. Assumindo que o termo $\log(V^2/V^2)$ possui termos omitidos, já que a complexidade não pode ser zero, e que $\log(V^2/V^2) \cdot \log(U)) > 1$, tem-se que

$$\textit{maxflow} \text{ de Goldberg et al.} = V^{2/3} \cdot V^2 = V^{8/3} > V^2 = \text{MenorCapacidade}.$$

Assim, $V^2 = O(V^{8/3}) \Rightarrow$ MenorCapacidade $= O($*maxflow* de Goldberg et al.$)$ □

Em grafos esparsos pode-se assumir que $|E| = |V|$. Neste tipo de grafos, *Q* pode

ser implementada como um *heap* [Cormen et al., 1990]. Assim, tem-se

$$\text{MenorCapacidade } = O(V \cdot \log V),$$

já que o passo *extract_min(Q)* tem complexidade igual a $O(\log V)$. Além disso, assumindo que $\log(V^2/E) \cdot \log(U)) > 1$, tomando os dois primeiros fatores de $O(\min(V^{2/3}, E^{1/2}) \cdot E \cdot \log(V^2/E) \cdot \log(U))$, tem-se

$$\text{Goldberg et al. } = V^{1/2} \cdot V \text{ e MenorCapacidade } = O(V \cdot \log V).$$

Assim, descartando o fator V, presente nos dois algoritmos e aplicando $\log$ tem-se

$$\text{MenorCapacidade } = \log(\log V) e \text{Goldberg et al. } = \log V^{1/2} = 1/2 \cdot \log V.$$

Então temos,

$$\log(\log(V)) = O(1/2 \cdot \log V) \Rightarrow \text{MenorCapacidade} = O(\text{Goldberg et al.})$$

□

O que mostra que tanto em grafos densos quanto em grafos esparsos, a detecção de arestas críticas no LMIR é menos custosa que nos algoritmos que usam *maxflow*.

4. Exemplos numéricos

Foram realizados experimentos de simulação a fim de se verificar a precisão e o desempenho do algoritmo proposto. A topologia utilizada é mostrada na Figura 2. Esta topologia foi utilizada em [Su and Chen, 2002] e em [Kodialam and Lakshman, 2000]. Os pares origem-destino $(S1, D1), (S2, D2), (S3, D3), (S4, D4), (S5, D5)$ são mostrados na figura e são os únicos pontos de entrada e saída de tráfego da rede. Os enlaces mais claros têm capacidade de 1200 unidades de banda e os enlaces mais escuros têm 4800 unidades. Estes valores simbolizam a taxa do OC-12 e do OC-48, respectivamente.

Cada enlace na figura é bidirecional, ou seja, representa dois enlaces de igual capacidade e sentidos opostos. As requisições entre os pares foram geradas aleatoriamente e a quantidade de banda requisitada é uniformemente distribuída no intervalo $[1, 4]$.

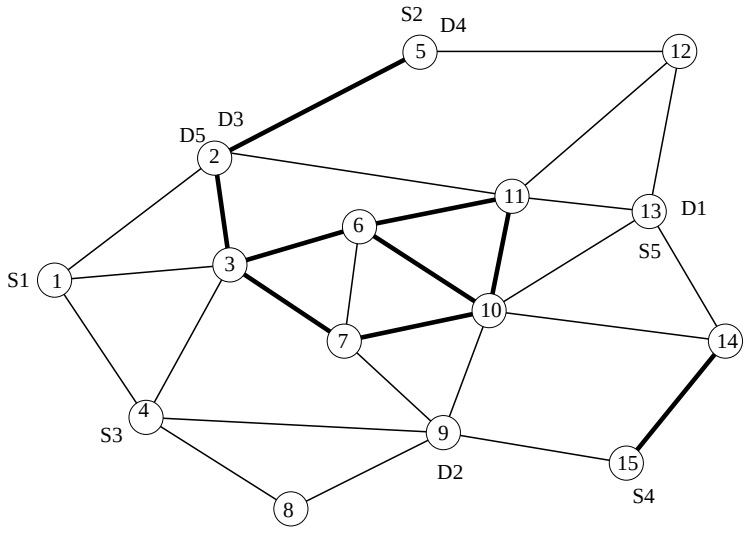

Figura 2: Topologia usada nas simulações

Nas legendas das figuras apresentadas a seguir, os algoritmos de interferência mínima serão representados por ICC [Su and Chen, 2002] MIRA

[Kodialam and Lakshman, 2000] LMIR. Além dos algoritmos de interferência mínima apresentados, mais dois algoritmos foram utilizados nas simulações para que comparações fossem feitas também com algoritmos que não levam em consideração o fenômeno da interferência. O *Minimum Hop Algorithm (MHA)* é uma implementação do algoritmo de*Dijkstra* e o *Widest Shortest Path Algorithm (WSP)* encontra o caminho com maior capacidade da rede. Se mais de um caminho com mesma capacidade existir, o algoritmo WSP devolve o caminho com o menor número de arestas.

Nestes experimentos, assume-se que os LSPs têm vida loga, ou seja, uma vez aceitos, eles ocupam os recursos até o final dos experimentos. Foram geradas aleatoriamente 8.000 requisições entre os cinco pares já mencionados.

Sabe-se que quanto menor for a taxa de diminuição do fluxo máximo, menor será a probabilidade de bloqueio. A Figura 3 mostra a diminuição do fluxo máximo da rede em função do número de chegadas. Pode-se ver que a taxa de diminuição do fluxo total da rede é menor para os algoritmos de interferência mínima. Os algoritmos WSP e MHA têm uma redução mais acentuada pois utilizam arestas críticas indiscriminadamente, causando a redução do *maxflow* entre diversos pares origem-destino o que demonstra a importância dos algoritmos de interferência mínima. O LMIR tem uma taxa de diminuição do fluxo máximo inferior à do ICC e igual à do MIRA até as 3.000 requisições. A partir deste ponto, tem um decaimento levemente mais acentuado que os dois até as 5.000 requisições. A partir das 5.000 requisições, o comportamento de todos os algoritmos torna-se bem parecido, quase linear.

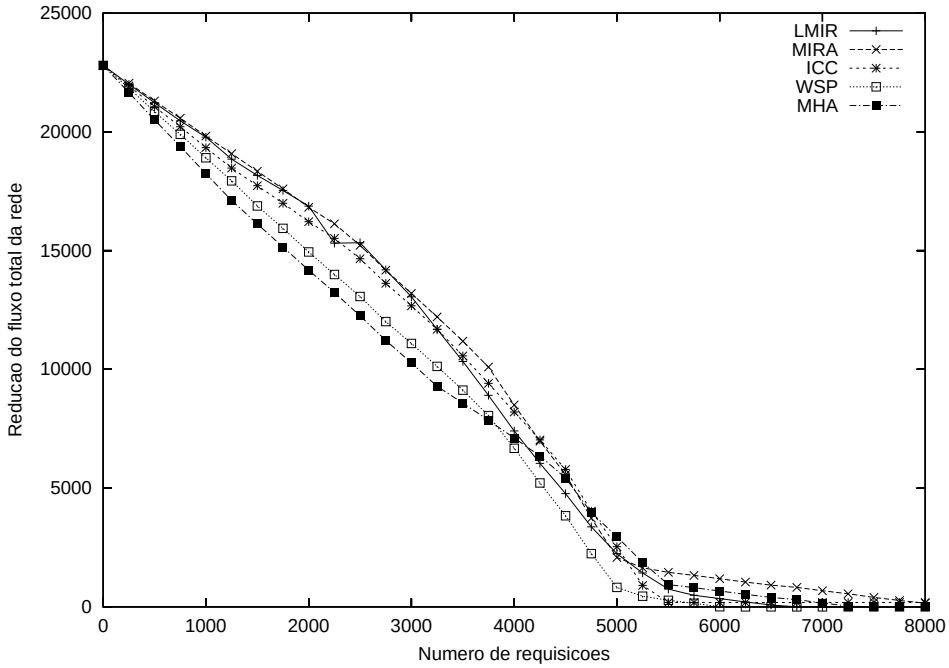

Figura 3: Largura de banda total entre todos os pares origem-destino da rede.

A Figura 4 mostra o número de requisições rejeitadas como função do número de requisições que chegam no sistema. O MHA é o algoritmo que rejeita o maior número de conexões para um menor número de requisições demandadas, o que se verifica logo após 3.000 requisições. Isso acontece porque ele sempre escolhe o caminho mais curto, saturando rapidamente as arestas que fazem parte deste caminho. O WSP começa a rejeitar conexões perto das 5.000 conexões. Entretanto, por não levar em consideração a criticalidade das arestas, ele pode escolher caminhos que possuem arestas críticas de vários pares e provocando a saturação das mesmas. Entre os algoritmos de interferência mínima, o LMIR e o MIRA são os algoritmos que apresentam um menor número de conexões re-

jeitadas.

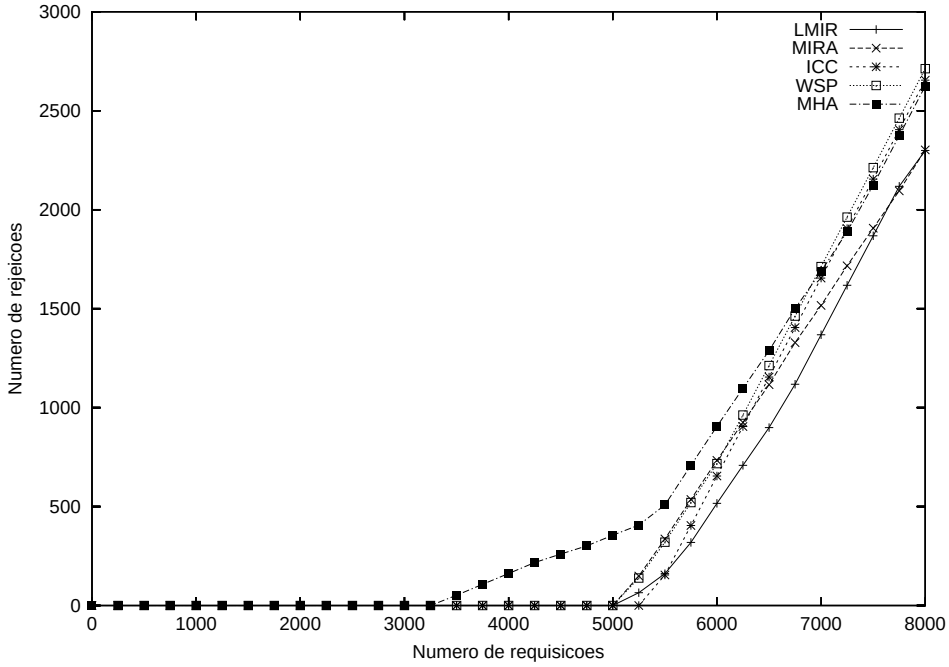

Figura 4: Número de rejeições

Conforme mencionado, interferência é a diminuição do fluxo de um par origem-destino devido ao roteamento de requisições entre outros pares. A Figura 5 mostra um experimento realizado para avaliar a interferência sofrida pelo par $(S1, D1)$ quando requisições entre outros pares são roteadas na rede. As requisições de conexão chegam aleatoriamente para os pares $(S2, D2), (S3, D3), (S4, D4)$ e $(S5, D5)$. Pretende-se verificar a influência destas requisições no fluxo de $(S1, D1)$.

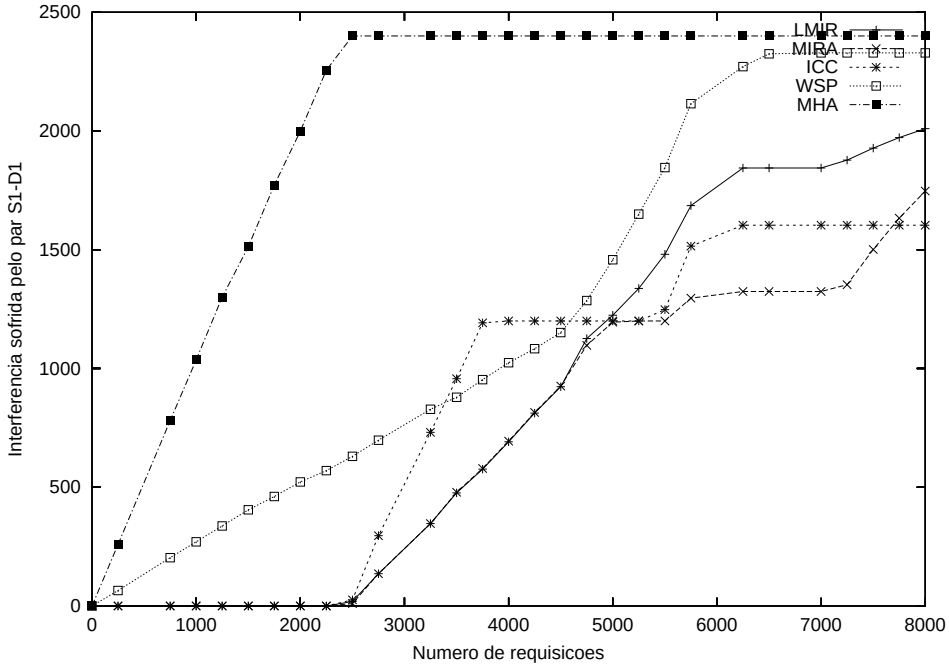

Figura 5: Interferência sofrida pelo par S1-D1

A Figura 5 mostra a interferência no par $(S1, D1)$ quando conexões são roteadas entre os outros pares. É fácil ver que os algoritmos MHA e WSP, que não levam em consideração o fenômeno de interferência, provocam uma diminuição do *maxflow* entre

Tabela 1: Tempos de Execução para 8.000 requisições

	ICC	MIRA	LMIR k=1	LMIR k=2	LMIR k=3	LMIR k=4	LMIR k=5	LMIR k=6
Tempo	7.23s	7.24s	5.017s	5.67s	6.54s	6.94s	7.01s	7.14s

$(S1, D1)$ antes mesmo de 1.000 conexões. O MHA por usar sempre o menor caminho, provoca a saturação logo após as 2.000 conexões. A partir daí o caminho torna-se saturado e não existe mais diminuição. O WSP, muito embora provoque interferência desde as primeiras conexões, demora mais para saturar do que o MHA. Isto acontece porque ele escolhe, à medida que um caminho vai saturando, outros caminhos com maior capacidade. A maior acentuação da interferência quando se usa o algoritmo WSP é percebida entre as 4.500 e 6.500, onde existe uma estagnação da curva até as 8.000 requisições.

Entre os algoritmos de interferência mínima, a interferência só é percebida após a chegada de 2.500 requisições, ou seja, não existe redução *maxflow* entre $(S1, D1)$ até esta chegada. A partir desta chegada, o algoritmo LMIR consegue um desempenho melhor, juntamente com o algoritmo MIRA, até bem próximo da chegada de 5.000 requisições. Mesmo causando uma diminuição acentuada no *maxflow* de $(S1, D1)$ logo após as 2.500 requisições, o algoritmo ICC consegue o melhor desempenho a longo prazo pois consegue uma maior estabilização do que o algoritmo LMIR e o algoritmo MIRA. O LMIR e o MIRA, têm um desempenho muito parecido até próximo das 5.000 requisições, o que talvez possa ser explicado pelo fato de que até as 5.000 requisições, as *K* arestas críticas vistas pelo algoritmo LMIR formem conjuntos de corte mínimo. Assim sendo, as arestas críticas seriam as mesmas para os dois algoritmos. Depois das 5.000 requisições o algoritmo MIRA começa a se estabilizar encontrando caminhos que não afetem $(S1, D1)$ e o algoritmo LMIR permanece causando interferência, mesmo que muito branda até as 8.000 requisições. Ao final das 8.000 requisições, ainda existe uma grande distância entre o LMIR que foi o algoritmo de interferência mínima causador de maior interferência e o WSP que é o algoritmo sem levar em consideração interferência de melhor desempenho. Note que neste experimento, só é levada em consideração a interferência no par $(S1, D1)$. É importante ressaltar que o LMIR consegue um melhor desempenho que o ICC quando a interferência entre todos os pares é levada em consideração, o que foi visto na Figura 3.

Como visto na Seção 3, o algoritmo LMIR encontra os *K* caminhos de menor capacidade para cada par. A escolha do valor deste parâmetro pode alterar o desempenho do algoritmo. A Figura 6 mostra o número de conexões rejeitadas em função de *K*. Pode-ser ver que o número mínimo de conexões rejeitadas, nesta topologia, foi alcançado com $K = 5$. Depois disso, os caminhos começam e se sobrepor e a identificação das arestas críticas passa a ser uma tarefa difícil.

A Tabela 1 mostra o tempo de execução dos algoritmos com 8.000 requisições entre todos os pares. Os valores foram obtidos utilizando o comando *time* do Linux, usando máquinas Intel Celeron 1.2 GHz, 256 MB de RAM.

Como era de se esperar, pode-se perceber que o tempo gasto na execução do algoritmo LMIR é menor que o tempo de execução do algoritmo ICC e do algoritmo MIRA. Além disso, o algoritmo LMIR tem a vantagem de conseguir oferecer um compromisso velocidade *X* precisão a depender da escolha do parâmetro *K*. A partir de $K = 7$, na topologia apresentada, os caminhos começam a se sobrepor e a detecção das arestas críticas torna-se mais difícil, deteriorando o desempenho do algoritmo. Os algoritmos WSP e MHA têm seu tempo de execução em torno de 4 segundos. Muito embora mais rápidos, estes algoritmos por não levarem em consideração o fenômeno da interferência, provocam um número excessivo de rejeições e por essas razões foram omitidos.

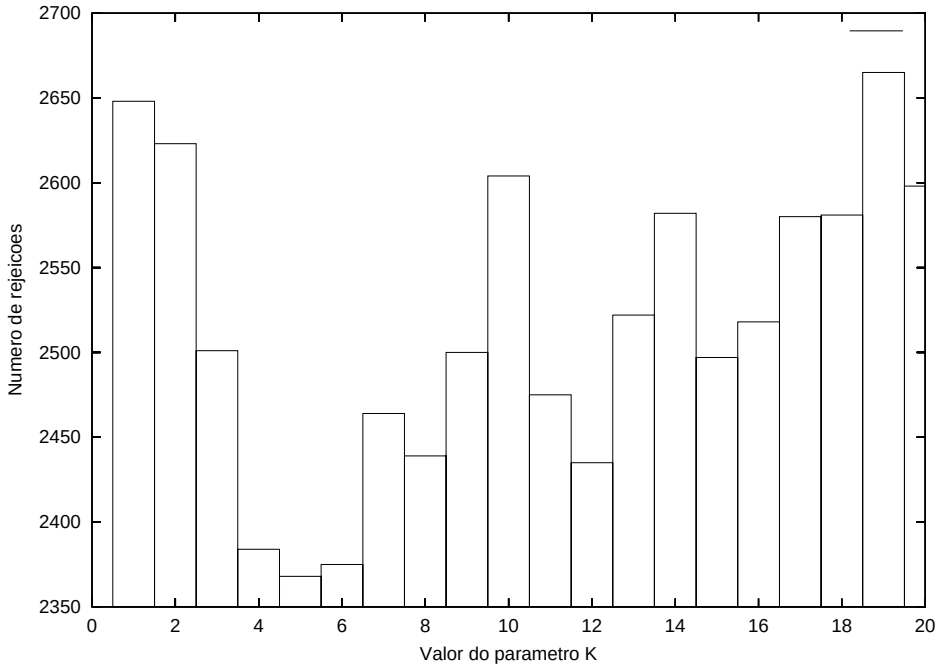

Figura 6: Influência do parâmetro ¨K¨ no número de conexões rejeitadas

5. Conclusões

Entre os principais desafios da Engenharia de Tráfego na Internet, usando MPLS, está o roteamento adequado dos LSPs de forma a maximizar o número de conexões atendidas. Pela falta de informações sobre demandas futuras, o roteamento deve ser realizado de forma que as conexões sejam atendidas sem detrimento do atendimento das futuras requisições. Assim, o roteamento com interferência mínima, assume papel primordial no processo de Engenharia de Tráfego.

Neste trabalho, um novo algoritmo de roteamento com interferência mínima foi introduzido. Este algoritmo, diferentemente dos algoritmos de roteamento com interferência mínima existentes, não executa algoritmos de *maxflow* para detecção das arestas críticas. Em substituição ao algoritmo de *maxflow*, modificações foram feitas no algoritmo de *Dijkstra* para que caminhos com menor capacidade fossem encontrados. Após encontradas *K* arestas críticas (arestas que limitam o fluxo dos menores caminhos), pesos são atribuídos as arestas e um algoritmo para encontrar o menor caminho é executado.

Resultados obtidos através de simulações mostraram que o algoritmo LMIR tem precisão semelhante à dos algoritmos de roteamento com inteferência mínima. No entanto, o algoritmo LMIR apresenta menor complexidade computacional do que tais algoritmos.

Referências

Banerjee, G. and Sidhu, D. (2002). Comparative analysis of path computation techniques for MPLS traffic engineering. *Computer Networks*, 40(1):149–165.

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

Goldberg, A. V. and Rao, S. (1998). Beyond the flow decomposition barrier. In *J. ACM 45 (5)*, pages 783–797.

Kodialam, M. S. and Lakshman, T. V. (2000). Minimum interference routing with applications to MPLS traffic engineering. In *INFOCOM (2)*, pages 884–893.

Su, B. W. X. and Chen, C. P. (2002). A new bandwidth guaranteed routing algorithm for MPLS traffic engineering. In *Proceedings of IEEE International Conference on Communications*.

Zegura, E. W., Calvert, K. L., and Bhattacharjee, S. (1996). How to model an internetwork. In *IEEE Infocom*, volume 2, pages 594–602, San Francisco, CA. IEEE.