Revista Eletrônica de Iniciação Científica em Computação, 2025, 23:1
ISSN: 3085-8461 • doi: 10.5753/reic.2025.6081

Artigo de Pesquisa/Research Paper

Um Algoritmo para *Fast-ReRoute* com *Backtracking* e Baseado em Avaliação de Fluxo Máximo

A Fast-ReRoute Algorithm with Backtracking and Based on Maximum Flow Evaluation

Leon Okida [Universidade Federal do Paraná | *laog19@inf.ufpr.br*]
Elias Procópio Duarte Jr. [Universidade Federal do Paraná | *elias@inf.ufpr.br*]

Departamento de Informática, Universidade Federal do Paraná, Av. Cel. Francisco H. dos Santos, 100, Jardim das Américas, Curitiba, PR, 81530-000, Brasil.

Resumo. Este trabalho apresenta o algoritmo *MaxFlowRouting*, para *Fast-ReRoute* (FRR) com *backtracking*. Para cada destino, o roteador mantém rotas alternativas além da rota principal, que são utlizadas quando a rota principal falha. O algoritmo utiliza a avaliação de fluxo máximo para computar rotas que possuem mais opções de alternativas. Se *não* há rota alternativa funcional, o algoritmo faz *backtracking* para o roteador anterior. Foram obtidos resultados extensivos de simulação para diversas topologias e métricas. Este artigo inclui resultados para o número de rotas alternativas e quantidade de *backtracks* feitos em grafos de mundo pequeno, grafos de conexão preferencial e quatro topologias da Internet: RNP (Brasil), Internet2 (EUA), Géant (Europe) and Wide (Japan).

Abstract. This paper presents the MaxFlowRouting algorithm, a *Fast-ReRoute* (FRR) strategy with backtracking. For each destination, a router maintains alternative routes alongside the primary route. FRR actives an alternative route, after the primary route fails. The MaxFlowRouting algorithm employs maximum flow evaluation to compute routes that have more alternative route options that can be activated in case of failure. If there is no working alternative route, the algorithm backtracks to the previous router. Extensive simulation results have been computed for multiple topologies and metrics. This paper presents an evaluation of the number of alternative routes and the number of backtracks for small world graphs, preferential attachment graphs, and four Internet topologies: RNP (Brazil), Internet2 (USA), Géant (Europe) and Wide (Japan).

Palavras-chave: Roteamento, Tolerância a Falhas, Fast-ReRoute (FRR), Internet, Avaliação de Fluxo Máximo

Keywords: Routing, Fault Tolerance, Fast-ReRoute (FRR), Internet, Maximum Flow Evaluation

Recebido/Received: 29 May 2025 • **Aceito/Accepted:** 24 June 2025 • **Publicado/Published:** 09 July 2025

1 Introdução

Organizações e indivíduos estão cada vez mais dependentes da Internet. Entretanto, a rede pode apresentar bastante instabilidade em algumas instâncias [Bischof *et al.*, 2023; Duarte Jr *et al.*, 2010]. Para aumentar a resiliência da rede, é fundamental adotar estratégias de roteamento tolerante a falhas [Liu *et al.*, 2024; Duarte Jr *et al.*, 2004]. Com isso, a rede se recupera rapidamente da ocorrência de uma falha em sua infraestrutura [Duarte and Musicante, 1999]. O *Fast-ReRoute* (FRR) [Chiesa *et al.*, 2021] é uma estratégia proativa de reparação de falhas, em que uma rota alternativa é empregada quando a rota primária falha. O FRR foi adotado em diversos protocolos da Internet, como o *Internet Protocol* (IP) [Shand and Bryant, 2010], o *Multi-Protocol Label Switching* (MPLS) [Pan *et al.*, 2005] e o *Open Shortest Path First* (OSPF) [Filsfils *et al.*, 2012].

Sem FRR, é necessário aguardar a reconvergência das tabelas de roteamento para a nova topologia da rede, o que pode representar uma latência significativa [Nassu *et al.*, 2007]. Além disso, pacotes enviados ao destino durante o intervalo de reconvergência serão descartados. O FRR computa rotas alternativas que são armazenadas na tabela de roteamento junto da rota principal. Assim, quando um roteador detecta uma falha, a rota alternativa pode ser ativada imediatamente. Entretanto, a efetividade do FRR depende da existência de rotas alternativas que possam contornar o ponto em que ocorreu a falha.

Neste trabalho, é proposto o algoritmo *MaxFlowRouting*, baseado em avaliação do fluxo máximo [Cormen *et al.*, 2022; Schroeder *et al.*, 2004; Okida, 2024], para a seleção de rotas robustas. Como o fluxo máximo é igual à quantidade de caminhos disjuntos em arestas, a avaliação de fluxo máximo encontra, implicitamente, rotas com maior conectividade possível considerando a topologia da rede. Consequentemente, é maior o número de rotas alternativas que podem ser utilizadas na ocorrência de uma falha.

A Figura 1 ilustra como um roteador executando o algo-

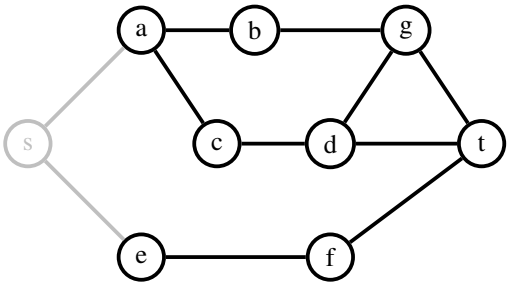

Figura 1. Exemplo de escolha do próximo *hop* baseada em avaliação do fluxo máximo.

ritmo *MaxFlowRouting* escolhe o próximo *hop*. Na figura, a origem s precisa enviar uma mensagem ao destino t. O algoritmo está sendo executado pelo vértice s. Existem duas arestas adjacentes a ele: (s, a) e (s, e). Isso significa que o próximo *hop* será para a ou para e. Para fazer essa escolha, o algoritmo inicialmente gera um subgrafo removendo o vértice que está executando o algoritmo. Consequentemente, as arestas adjacentes a esse vértice são removidas também. Na figura, o subgrafo é formado pelas linhas e vértices escuros.

Para avaliar a opção pela aresta (s, a), o algoritmo computa os critérios de caminho mínimo e de fluxo máximo entre a e t no subgrafo. Ele obtém, nesse caso, o caminho mínimo de tamanho 3 e o fluxo máximo de valor 2. Fazendo a mesma avaliação para a opção pela aresta (s, e), obtém-se o caminho mínimo de tamanho 2 e o fluxo máximo de valor 1. Isso significa que existem dois caminhos disjuntos entre a e t, enquanto apenas um entre e e t. O algoritmo utiliza pesos para definir a influência de cada critério na decisão do próximo *hop*. Quando o foco do algoritmo é tolerância a falhas, o critério de fluxo máximo deve ter um peso maior. No exemplo, o vértice a tem preferência maior como o próximo *hop* por ter um valor de fluxo máximo do que o vértice e.

Destaca-se que em [Schroeder and Duarte Jr, 2007] é descrito um algoritmo de roteamento que também utiliza fluxo máximo, entretanto aquele *não* é um algoritmo de reroteamento rápido: cada mensagem carrega toda a rota já percorrida, e quando é recebida por um roteador, este atualiza a topologia removendo todos os vértices já percorridos (e todas as suas arestas adjacentes) e recalcula o próximo passo executando o algoritmo de fluxo máximo para cada mensagem processada. O algoritmo apresentado no presente artigo reduz o custo ao executar o fluxo máximo apenas para pré-computar rotas alternativas.

A estratégia de FRR proposta também utiliza *backtracking*. Se a rota alternativa também falha, o pacote pode retornar para o roteador anterior para que ele tente outra rota. Assim, o roteamento tem maiores chances de funcionar, mesmo que as tabelas de roteamento não reflitam a topologia real da rede. Basta que nenhum roteador visitado anteriormente falhe e que exista pelo menos uma rota funcional entre a origem e o destino.

O algoritmo proposto foi avaliado através de simulação e comparado com uma versão do FRR que calcula rotas com o algoritmo de Dijkstra. Foram obtidos resultados de simulação para diversas topologias e métricas. Neste artigo são apresentados resultados para grafos de mundo pequeno de Watts-Strogatz [Watts and Strogatz, 1998], grafos de conexão preferencial de Barabási-Albert [Albert and Barabási, 2002] e *backbones* da Internet, como a RNP [1] do Brasil, a Internet2 [2] dos E.U.A., a Géant [3] da Europa e a Wide [4] do Japão. Resultados mostram que o algoritmo *MaxFlowRouting* computou rotas com até 1,54 vezes mais rotas alternativas por vértice.

O restante deste trabalho está organizado como segue. A Seção 2 apresenta o algoritmo *MaxFlowRouting* e a estratégia de FRR propostos neste trabalho. A Seção 3 apresenta os resultados de avaliação. Por fim, a Seção 4 conclui o trabalho.

[1] https://www.rnp.br/en/ipe-network
[2] https://internet2.edu/services/layer-1
[3] https://network.geant.org
[4] https://two.wide.ad.jp

2 O Algoritmo *MaxFlowRouting*

Esta seção apresenta o algoritmo *MaxFlowRouting* para roteamento tolerante a falhas. Um roteador que executa o algoritmo mantém o grafo $G = (V, E)$ como uma representação local da topologia da rede. Não é necessário que o grafo G reflita exatamente a topologia real da rede. Quando um roteador detecta uma mudança na topologia, ele deve disseminar essa informação para os demais roteadores. A tabela de roteamento e o grafo G são atualizados pelo roteador quando este recebe uma informação de mudança na topologia. Cada roteador executa o Algoritmo 1 (*MaxFlowRouting*) para gerar a tabela de roteamento para todos os destinos possíveis na rede. O algoritmo utiliza a avaliação de fluxo máximo e o algoritmo de Dijkstra para decidir qual deve ser o próximo *hop* da rota.

O fluxo máximo é equivalente ao problema do corte mínimo [Cohen *et al*., 2012, 2017; Maske *et al*., 2020]. O corte mínimo m entre dois vértices u e v de um grafo G é o conjunto de arestas C, tal que a remoção de todas as arestas de C em G remove todos os caminhos possíveis entre u e v, desconectando-os. A cardinalidade do corte C, denotada por $|C|$ é o número de arestas em C. Um corte C é chamado de corte mínimo se, para cada corte C' entre o mesmo par de vértices u e v no grafo G, $|C| \leq |C'|$. Para cada par de vértices no grafo G, o fluxo máximo e o corte mínimo possuem a mesma cardinalidade e podem ser computados pelos mesmos algoritmos.

Neste trabalho, considera-se que a capacidade de fluxo de todas as arestas é 1. Assim, a avaliação de fluxo máximo encontra, implicitamente, rotas com mais conectividade e que possuem mais opções de caminhos alternativos em caso de falha. Isso ocorre nesse caso porque o fluxo máximo entre dois vértices indica o número de caminhos disjuntos em arestas entre eles. Consequentemente, obtém-se um número maior de caminhos alternativos entre os vértices [Cohen *et al*., 2011].

Um roteador seleciona o próximo *hop* ordenando todos os seus vizinhos com base em uma função avaliadora $\Gamma(G', i, d)$. A função é executada por cada roteador s para cada vizinho i considerando um destino d. Essa função utiliza o grafo $G' = (V', E')$ onde $V' = V - s$ e $E' = E - (s, j), \forall j | (s, j) \in E$. A função $\Gamma(G', i, d)$ retorna um valor numérico para cada vértice i adjacente a s considerando o destino d, de forma que valores altos são considerados melhores que valores baixos, como descrito a seguir.

A função $\Gamma(G', i, d)$ é computada levando em conta o fluxo máximo e o tamanho da rota, sendo calculada pela expressão $\Gamma(G', i, d) = w_1 c_1(G', i, d) + w_2 c_2(G', i, d)$. Essa expressão leva em conta os critérios de valor do fluxo máximo entre i e d no grafo G' (critério c_1) e de tamanho da rota de tamanho mínimo entre i e d no grafo G' (critério c_2). Cada critério é multiplicado por um peso (respectivamente w_1 e w_2), que determina como o critério influencia o resultado da função $\Gamma(G', i, d)$. Então, dado um vértice s, todos os seus vizinhos i que possuem caminho para o destino d são ordenados de modo decrescente de acordo com o resultado da função $\Gamma(G', i, d)$ como candidatos a próximo *hop*. Os vértices i de maior valor de $\Gamma(G', i, d)$ são escolhidos primeiro para rotear um pacote de s para o destino d. Assim, o peso correspondente ao critério de fluxo máximo deve ser positivo, pois o objetivo é maximizá-

lo, enquanto o peso correspondente ao critério de caminho mínimo deve ser negativo, pois deseja-se minimizá-lo.

O Algoritmo 1 (*MaxFlowRouting*) é executado por cada roteador s, que inicialmente se remove do grafo G da topologia conhecida, gerando o grafo G'. Após isso, para cada destino possível d e para cada vizinho i, s computa o fluxo máximo e o tamanho do menor caminho entre i e d. Feito isso, a função $\Gamma(G', i, d)$ é computada utilizando pesos para os critérios de fluxo máximo e distância. Para cada destino (entrada na tabela de roteamento), existe uma lista de *CandidatosAProximoHop*, que é inicializada vazia. Um vizinho i é incluído em *CandidatosAProximoHop* se possui um caminho para o destino d em G'. Finalmente, os vizinhos em *CandidatosAProximoHop* são ordenados de acordo com a função avaliadora $\Gamma(G', i, d)$.

A tabela de roteamento gerada pelo algoritmo *MaxFlowRouting* é utilizada como base para o roteamento de pacotes. O Algoritmo 2 de FRR utiliza a tabela de roteamento para selecionar o próximo *hop* dado o destino do pacote.

O Algoritmo 2 (*Fast-ReRoute* com *Backtracking*) é executado pelo roteador s para encaminhar um pacote enviado pela origem o para o destino d. Primeiro, o roteador s checa se ele é vizinho do roteador d e, caso seja, envia o pacote diretamente a d. Um algoritmo de roteamento com *backtracking* tem uma demanda extra: é necessário que cada pacote carregue a sequência de vértices pelos quais já passou. Para compreender esta necessidade, basta pensar que sem esta informação um pacote pode facilmente entrar em loop na rede. Desta forma, cada pacote pct carrega consigo a lista sequencial dos vértices pelos quais passou, chamada de $pct.NodosVisitados$. O roteador s adiciona a si mesmo a $pct.NodosVisitados$ caso não esteja na lista. Caso o roteador já esteja na lista, um ciclo é detectado, o que ocorre quando o pacote é enviado por *backtracking* por um vizinho que não tinha rota disponível para o destino. Após isso, o roteador s seleciona o próximo *hop* para encaminhar o pacote. O vizinho de melhor classificação na entrada da tabela de roteamento para o destino d que não esteja em $pct.NodosVisitados$ é selecionado para ser o próximo *hop*. Caso não exista um vizinho disponível para isso, o roteador s retorna o pacote para o roteador r de onde recebeu o pacote, isto é, faz *backtracking*. Caso não exista r antes de s, então s é própria a origem o do pacote e não existe rota disponível entre o e d.

Algorithm 1 Algoritmo *MaxFlowRouting*: Geração da tabela do roteador s

```
G' ← G − {s}
for all destino d em G' do
    CandidatosAProximoHop ← lista vazia
    for all vizinho i adjacente a s em G do
        if não há caminho entre i a d em G' then
            ignora i
        else
            adiciona i a CandidatosAProximoHop
            computa Γ(G', i, d)
        end if
    end for
    if CandidatosAProximoHop possui mais de um elemento then
        ordena CandidatosAProximoHop de acordo com Γ(G', i, d)
    end if
    TabelaDeRoteamento[d] ← CandidatosAProximoHop
end for
```

Algorithm 2 Algoritmo *Fast-ReRoute* com *Backtracking* no roteador s.

```
if destino d é adjacente a s em G then
    encaminha pacote pct para d
else if pct.NodosVisitados não contém o roteador s then
    adiciona s a pct.NodosVisitados
    if TabelaDeRoteamento[d] contém roteadores que não estão em pct.NodosVisitados then
        escolhe o próximo hop i de acordo com TabelaDeRoteamento[d], tal que i ∉ pct.NodosVisitados
        envia pacote pct para i
    else
        {Não é possível encaminhar o pacote, deve fazer backtracking}
        if pct.NodosVisitados contém pelo menos um roteador r antes de s then
            envia pct de volta para r
        else
            {Não é possível fazer backtracking, pois está na origem o}
            retorna Erro: Não há rota disponível entre o e d
        end if
    end if
end if
```

2.1 Prova de Corretude do Algoritmo de FRR com *Backtracking*

Esta seção apresenta a prova de que o algoritmo de FRR com *backtracking* consegue rotear um pacote do nodo s para o nodo t pela rede representada pelo grafo G se nenhum nodo visitado anteriormente falha e se existe pelo menos uma rota funcional da origem até o destino.

Teorema 2.1. Considere o grafo $G = (V, E)$ representando a topologia da rede e dois vértices selecionados arbitrariamente $s, t \in V$, representando a origem e o destino de um pacote p. O algoritmo de FRR com *backtracking* consegue rotear o pacote com sucesso entre s e t se nenhum vértice já visitado falha e se existe pelo menos uma rota funcional entre s e t em G.

Prova. O nodo executando o algoritmo de FRR com *backtracking* tenta enviar o pacote através do nodo vizinho de melhor avaliação na entrada da tabela de roteamento correspondente ao destino t. Este nodo, por sua vez, também faz isso. Esse processo se repete até que o pacote chegue ao destino. Caso um nodo j receba o pacote e detecte que não possui uma rota funcional para o destino, ele pode retornar o pacote ao nodo i anterior de onde recebeu o pacote. O nodo i tenta enviar o pacote através do próximo nodo k de melhor avaliação na entrada da tabela de roteamento correspondente ao destino. O processo se repete: se não há rota funcional entre k e t, o pacote será novamente retornado para i. Após i tentar todas as suas opções de próximo *hop* sem sucesso, ele retorna para o nodo anterior de onde recebeu o pacote. Se nenhum nodo já visitado anteriormente falhar, esse processo pode acontecer, sucessivamente, até o pacote ser retornado para a origem s, que também tentará todas as suas alternativas de próximo *hop*. Assim, se existe uma rota funcional entre a origem e o destino na rede, ela será encontrada após um tempo e utilizada para rotear o pacote com sucesso para o destino t. □

3 Avaliação

Esta seção faz inicialmente uma descrição dos resultados apresentados no trabalho original, dos quais um pequeno extrato é apresentado no presente artigo.

No Trabalho de Conclusão de Curso (TCC) [Okida, 2024] sobre o qual este artigo se baseia são apresentados resultados de simulação em diversas topologias e métricas para a avaliação do algoritmo *MaxFlowRouting* através de dois grandes experimentos. O primeiro deles compara as rotas produzidas pelo algoritmo *MaxFlowRouting* com as rotas produzidas por uma versão do algoritmo de FRR com *backtracking* na qual as rotas são calculadas com o algoritmo de Dijkstra. O segundo experimento avalia o comportamento do FRR após a ocorrência de uma falha. Os experimentos foram realizados com programas desenvolvidos com a linguagem *Python* e a biblioteca *NetworkX* [Hagberg *et al*., 2008]. Nos dois experimentos, o algoritmo *Push-Relabel* é utilizado para a avaliação do fluxo máximo. O algoritmo *MaxFlowRouting* é executado nos experimentos considerando 3 pares diferentes de pesos para os critérios de Fluxo Máximo (FM) e Caminho Mínimo (CM). Os pares escolhidos foram: FM = 2 e CM = -5, que faz com que o algoritmo produza rotas de tamanhos menores; FM = 5 e CM = -5, que equilibra a influência dos dois critérios; e FM = 5 e CM = -1, que favorece a produção de rotas com maior conectividade.

Foram obtidos resultados para 3 tipos de grafos aleatórios. O primeiro deles é o grafo aleatório de conectividade variável de Erdos-Renyi [Erdos and Rényi, 1960], os grafos de conexão preferencial de Barabási-Albert [Albert and Barabási, 2002] e os grafos de mundo pequeno de Watts-Strogatz [Watts and Strogatz, 1998]. Os algoritmos foram também executados em grafos representando topologias reais da Internet. Foram selecionadas topologias importantes dos E.U.A [Internet2, 2025], da Europa [GÉANT, 2025], do Brasil [RNP, 2025] e do Japão [WIDE Project, 2025].

O primeiro experimento, que compara as rotas produzidas pelo algoritmo *MaxFlowRouting* com as produzidas pelo algoritmo de FRR que computa rotas com o algoritmo de Dijkstra, leva em consideração 3 métricas. A primeira métrica é o tamanho médio das rotas. A segunda métrica é a soma dos graus de todos os vértices da rota. A terceira métrica é o número médio de rotas alternativas disponíveis por vértice de cada rota. Uma rota alternativa é uma rota disjunta da rota original produzida pelo algoritmo de roteamento.

Como resultado do experimento, observa-se que as rotas computadas pelo *MaxFlowRouting* possuem mais opções de rotas alternativas por vértice do que as rotas produzidas pelo algoritmo alternativo. Também é constatado que número de rotas alternativas por vértice cresce conforme o tamanho do grafo aumenta. No entanto, a diferença entre os resultados dos algoritmos diminui conforme a conectividade do grafo aumenta. Considerando a métrica de média da soma dos graus dos vértices da rota, o algoritmo *MaxFlowRouting* produz rotas com maior conectividade. No entanto, à medida que a conectividade do grafo aumenta, a diferença entre a conectividade das rotas geradas pelos dois algoritmos diminui. O tamanho das rotas produzidas pelo *MaxFlowRouting* é levemente maior do que o da alternativa. A diferença nos tamanhos das rotas produzidas pelos diferentes algoritmos diminui em grafos de maior conectividade.

No segundo experimento, o algoritmo *MaxFlowRouting* é comparado com uma alternativa que usa Dijkstra para calcular as rotas. Existe uma versão da alternativa sem FRR e outra com FRR. Para simular uma falha em algum vértice da rota principal, inicialmente a rota é computada sem a ocorrência de falhas. Após isso, escolhe-se um vértice aleatório (com exceção da origem ou do destino) para introduzir uma falha e cada algoritmo refaz o roteamento considerando a falha no vértice escolhido. Foram utilizadas 3 métricas para comparar o comportamento dos algoritmos. A primeira delas é o tamanho médio da rota sem a ocorrência da falha. A segunda é o tamanho médio da rota com a ocorrência da falha. A terceira métrica é a quantidade média de *backtracks* que ocorrem. Essa métrica é computada com o uso de um contador no algoritmo de roteamento, que é inicializado com 0. O contador é incrementado quando um vértice esgota todas as suas possibilidades de roteamento e precisa retornar para um vértice anterior. No caso do roteamento sem FRR, são realizados *backtracks* do ponto onde ocorreu a falha até a origem da rota, pois ele não faz uso de rotas alternativas.

Em geral, é observada uma ocorrência maior de *backtracks* na execução da alternativa sem o uso do FRR. À medida que o tamanho dos grafos cresceu, a quantidade de *backtracks* em todos os algoritmos de roteamento diminui. Entretanto, observou-se uma alta variabilidade na ocorrência média de *backtracks* na execução do algoritmo *MaxFlowRouting* em diferentes topologias. É observado também que a ocorrência de uma falha faz com que as rotas produzidas por todos os algoritmos tenham tamanho médio maior em comparação com o cenário sem a ocorrência de falhas. A exceção ocorre quando a falha impossibilita o roteamento.

A Subseção 3.1 apresenta resultados obtidos no primeiro experimento, que compara as rotas produzidas pelos algoritmos para grafos de conexão preferencial de Barabási-Albert, para grafos de mundo pequeno de Watts-Strogatz e para grafos que representam topologias reais da Internet. A Subseção 3.2 apresenta um extrato dos resultados obtidos no segundo experimento, que compara o comportamento dos algoritmos com a ocorrência de uma falha durante o roteamento, para os mesmos grafos.

3.1 Comparação das Rotas Produzidas pelos Algoritmos

A Subsubseção 3.1.1 apresenta os resultados do experimento que compara as rotas produzidas pelos algoritmos para os grafos de conexão preferencial. A Subsubseção 3.1.2 apresenta os resultados do experimento para os grafos de mundo pequeno de Watts-Strogatz. A Subsubseção 3.1.3 apresenta os resultados do experimento para as topologias reais da Internet. Os resultados são apresentados considerando as métricas de tamanho médio das rotas, soma de graus dos vértices média e número de rotas alternativas por vértice em ambas as subsubseções.

3.1.1 Grafos de Conexão Preferencial

Os grafos de conexão preferencial de Barabási-Albert [Albert and Barabási, 2002] são grafos de N vértices, gerados de forma iterativa pela adição de vértices com m arestas, que se conectam preferencialmente a vértices de maior grau. No ex-

perimento, foram utilizados grafos de tamanho $N = 100, 150$ e 200 com vértices com $m = 3$ arestas. Foram computadas rotas entre todos os vértices do grafo que não são vizinhos entre si. As métricas foram calculadas somente considerando casos em que as rotas produzidas por cada algoritmo são diferentes. Os resultados obtidos são apresentados na Tabela 1.

A Figura 2 mostra que o algoritmo *MaxFlowRouting* produz rotas com uma quantidade média de rotas alternativas por vértice maior do que o algoritmo de Dijkstra. Entretanto, o par de pesos FM = 5 e CM = -1 fez com que o *MaxFlowRouting* produzisse resultados com alta variabilidade conforme N varia, partindo de 17,88 rotas alternativas por vértice em grafos de tamanho $N = 100$, indo para 22,94 em grafos de tamanho $N = 150$, mas caindo para 20,05 em grafos de tamanho $N = 200$. Nos grafos de tamanho 150, o algoritmo com esse par de pesos produziu rotas com 1,54 vezes mais rotas alternativas por vértice do que as rotas computadas pelo algoritmo de Dijkstra.

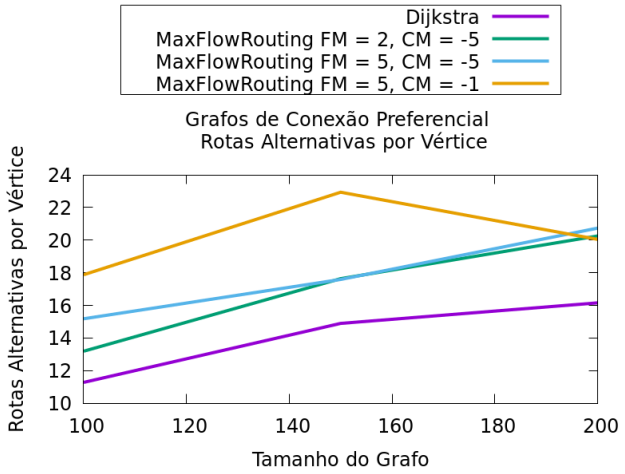

Figura 2. Comparação da quantidade média de rotas alternativas por vértice. Roxo: Dijkstra. Verde: *MaxFlowRouting* FM = 2, CM = -5. Azul: *MaxFlowRouting* FM = 5, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -1.

A Figura 3 mostra que, em média, a soma dos graus dos vértices das rotas produzidas pelo algoritmo *MaxFlowRouting* é maior do que a das rotas produzidas pelo algoritmo de Dijkstra. Em todos os algoritmos, o valor da soma cresceu conforme N aumentou. O algoritmo *MaxFlowRouting* com o par de pesos FM=5 e CM=-1 produziu as rotas com a maior soma de graus dos vértices. Nos grafos de tamanho 150, esse algoritmo produziu rotas que, em média, possuem uma soma de graus dos vértices 1,43 vezes maior do que as rotas produzidas pelo algoritmo de Dijkstra.

A Figura 4 mostra que o algoritmo *MaxFlowRouting* com os pesos FM = 2 e CM = -5 produz rotas de tamanho médio pouco maior do que as rotas produzidas pelo algoritmo de Dijkstra. Já o *MaxFlowRouting* com os pesos FM = 5 e CM = -1 apresentou um aumento significativo no tamanho das rotas conforme o tamanho N do grafo aumentou. Nos grafos de tamanho 150, porém, o tamanho das rotas produzidas por esse algoritmo foi apenas 1,06 vezes maior do que as rotas computadas pelo algoritmo de Dijkstra.

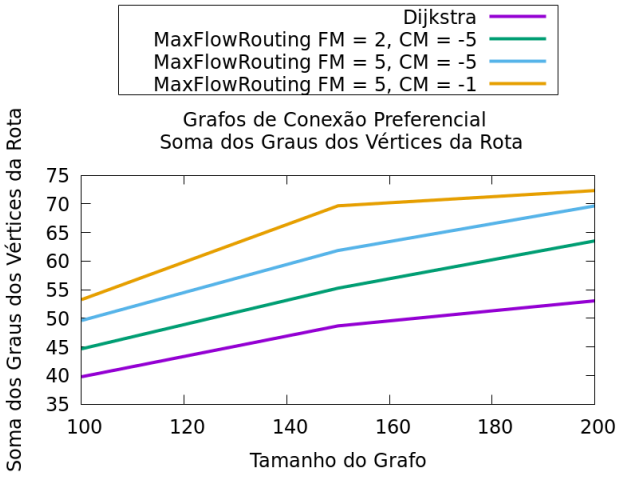

Figura 3. Comparação da média da soma dos graus dos vértices das rotas produzidas. Roxo: Dijkstra. Verde: *MaxFlowRouting* FM = 2, CM = -5. Azul: *MaxFlowRouting* FM = 5, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -1.

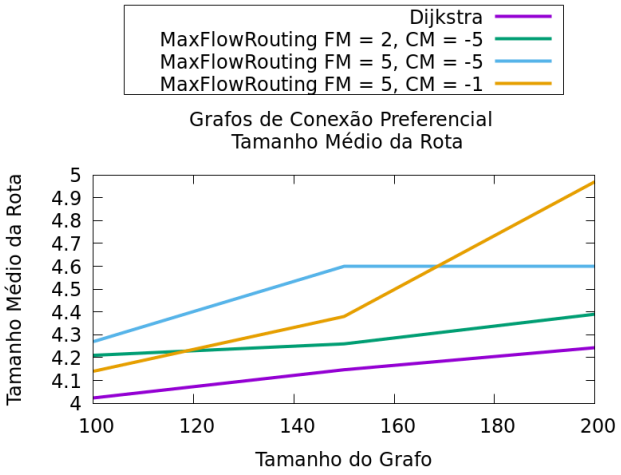

Figura 4. Comparação do tamanho médio das rotas produzidas. Roxo: Dijkstra. Verde: *MaxFlowRouting* FM = 2, CM = -5. Azul: *MaxFlowRouting* FM = 5, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -1.

3.1.2 Grafos de Mundo Pequeno

Os grafos de mundo pequeno de Watts-Strogatz [Watts and Strogatz, 1998] são grafos de N vértices, gerados a partir de uma topologia em forma de anel em que cada vértice é conectado a seus k vizinhos mais próximos. Com uma probabilidade p, cada aresta (u, v) entre vizinhos u e v é substituída por uma aresta (u, t), em que t é um outro vértice do grafo, selecionado aleatoriamente. No experimento, foram utilizados grafos de tamanho $N = 100, 150$ e 200, com vértices com $k = 4$ vizinhos e probabilidade $p = 0,4$ das arestas serem substituídas. Foram computadas rotas entre todos os vértices do grafo que não possuem aresta entre si. As métricas foram calculadas considerando casos em que as rotas produzidas por cada algoritmo diferem. Os resultados obtidos com o experimento estão apresentados na Tabela 2.

A Figura 5 mostra que o número médio de rotas alternativas por vértice das rotas produzidas pelo *MaxFlowRouting* ultrapassa o valor obtido pelas rotas computadas pelo algoritmo de FRR que utiliza o algoritmo de Dijkstra para computar rotas. Entretanto, é possível notar uma alta variabilidade nos valores conforme N varia. As rotas computadas pelo algoritmo *MaxFlowRouting* com pesos FM=5 e CM=-1

Tabela 1. Resultados do Primeiro Experimento com Grafos de Conexão Preferencial

N	Algoritmo	Tam. Médio	Soma de Graus	Rotas Alternativas
100	Dijkstra	4,02	39,11	11,28
100	*MaxFlowRouting* com 2,-5	4,21	44,66	13,19
100	*MaxFlowRouting* com 5,-5	4,27	49,61	15,18
100	*MaxFlowRouting* com 5,-1	4,14	53,26	17,88
150	Dijkstra	4,15	48,67	14,90
150	*MaxFlowRouting* com 2,-5	4,26	55,27	17,64
150	*MaxFlowRouting* com 5,-5	4,60	61,86	17,59
150	*MaxFlowRouting* com 5,-1	4,38	69,67	22,94
200	Dijkstra	4,24	53,06	16,16
200	*MaxFlowRouting* com 2,-5	4,39	63,53	20,26
200	*MaxFlowRouting* com 5,-5	4,60	69,65	20,74
200	*MaxFlowRouting* com 5,-1	4,97	72,33	20,05

Tabela 2. Resultados do Primeiro Experimento com Grafos de Mundo Pequeno

N	Algoritmo	Tam. Médio	Soma de Graus	Rotas Alternativas
100	Dijkstra	5,17	22,61	2,48
100	*MaxFlowRouting* com 2,-5	5,61	26,41	3,06
100	*MaxFlowRouting* com 5,-5	6,07	28,50	2,96
100	*MaxFlowRouting* com 5,-1	10,76	54,91	3,37
150	Dijkstra	5,54	24,26	2,49
150	*MaxFlowRouting* com 2,-5	5,96	27,11	2,80
150	*MaxFlowRouting* com 5,-5	6,47	31,77	3,22
150	*MaxFlowRouting* com 5,-1	13,98	70,97	3,28
200	Dijkstra	5,84	25,36	2,43
200	*MaxFlowRouting* com 2,-5	6,16	28,57	2,91
200	*MaxFlowRouting* com 5,-5	7,08	34,08	3,05
200	*MaxFlowRouting* com 5,-1	15,46	74,51	2,94

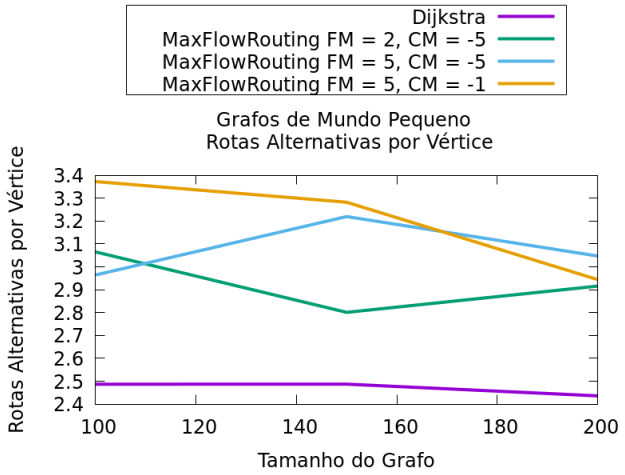

Figura 5. Comparação da quantidade média de rotas alternativas por vértice. Roxo: Dijkstra. Verde: *MaxFlowRouting* FM = 2, CM = -5. Azul: *MaxFlowRouting* FM = 5, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -1.

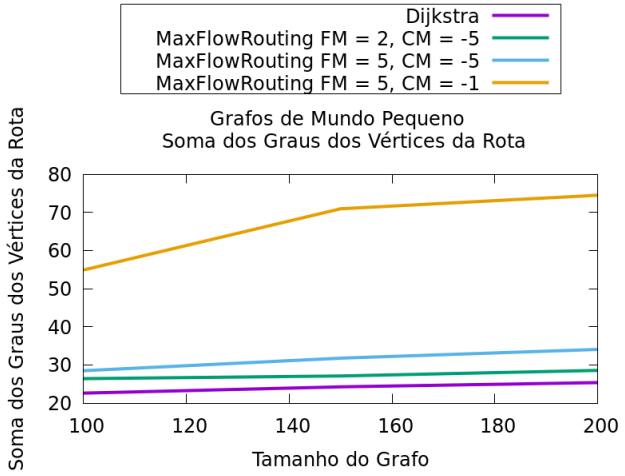

Figura 6. Comparação da média da soma dos graus dos vértices das rotas produzidas. Roxo: Dijkstra. Verde: *MaxFlowRouting* FM = 2, CM = -5. Azul: *MaxFlowRouting* FM = 5, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -1.

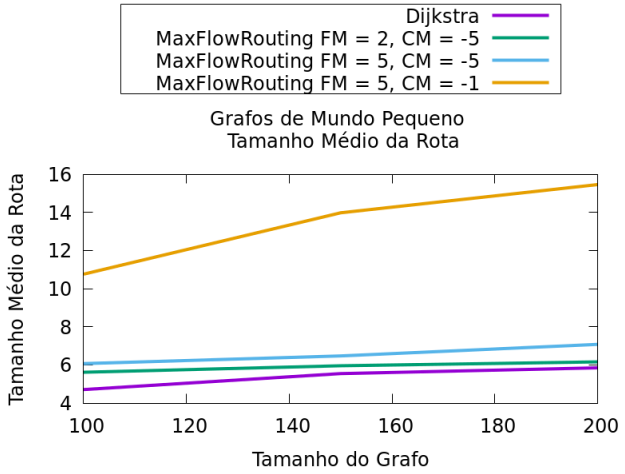

Figura 7. Comparação do tamanho médio das rotas produzidas. Roxo: Dijkstra. Verde: *MaxFlowRouting* FM = 2, CM = -5. Azul: *MaxFlowRouting* FM = 5, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -1.

apresentaram uma média de 3,37 rotas alternativas por vértice em $N = 100$, caindo para 3,28 em $N = 150$ e ainda mais para 2,94 em $N = 200$.

A Figura 6 mostra que a maior média de soma de graus dos vértices ocorre em rotas computadas pelo algoritmo *MaxFlowRouting* com o par de pesos FM = 5 e CM = -1. Isso é esperado, pois esse par de pesos favorece a conectividade nas rotas. Com outros pares de pesos, o *MaxFlowRouting* também mostrou uma vantagem comparado ao algoritmo que utiliza Dijkstra para computar rotas, considerando essa métrica.

A Figura 7 mostra que as rotas computadas pelo *MaxFlowRouting* com pesos FM = 2 e CM = -5 possuem um tamanho médio menor comparadas às rotas produzidas pelo algoritmo com os outros pares de pesos (FM = 5 e CM = -5 e FM = 5 e CM = -1). No melhor caso, as rotas computadas pelo *MaxFlowRouting* foram um pouco maiores do que as rotas computadas com o uso do algoritmo de Dijkstra, que computa rotas com o menor tamanho possível.

3.1.3 Topologias Reais da Internet

Os experimentos foram executados em grafos representando as seguintes topologias reais da Internet: Internet2 [Internet2, 2025], com $N = 54$ vértices; Géant [GÉANT, 2025], com $N = 44$ vértices; RNP [RNP, 2025], com $N = 28$ vértices; e Wide [WIDE Project, 2025], com $N = 14$ vértices. Foram computadas rotas entre todos os vértices do grafo que não possuem aresta entre si. As métricas foram calculadas somente considerando casos em que as rotas produzidas por cada algoritmo diferem. Os resultados experimentais estão apresentados na Tabela 3.

A Figura 8 mostra que o algoritmo *MaxFlowRouting* produz rotas com mais rotas alternativas por vértice com os 3 pares de pesos do que o algoritmo de FRR que computa rotas com o algoritmo de Dijkstra. É possível ver também que a variação nos resultados obtidos pelo *MaxFlowRouting* com os diferentes pares de pesos é pequena. A quantidade de rotas alternativas por vértice varia significativamente de uma topologia para outra.

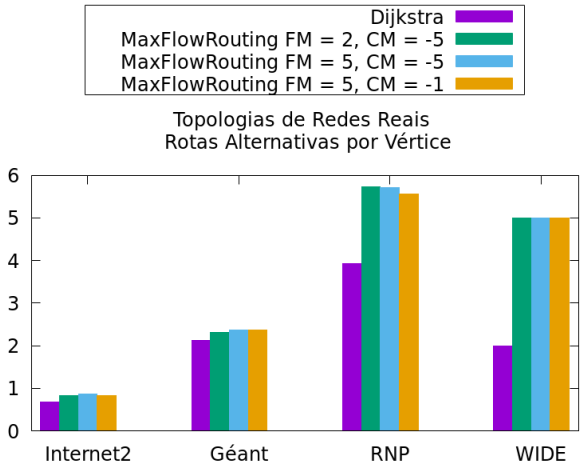

Figura 8. Comparação da quantidade média de rotas alternativas por vértice. Da esquerda para a direita: Internet2, Géant, RNP, Wide. Roxo: Dijkstra. Azul: *MaxFlowRouting* FM = 2, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -5. Amarelo: *MaxFlowRouting* FM = 5, CM = -1.

Como apresentado na Figura 9, o algoritmo *MaxFlowRouting* computa rotas com a maior média de soma de graus dos vértices com o par de pesos FM = 5 e CM = -1. O algoritmo *MaxFlowRouting* com outros pares de pesos também ultrapassa o algoritmo que utiliza Dijkstra para computar rotas nesse quesito. Isso é notável nos resultados observados

Tabela 3. Resultados do Primeiro Experimento com Grafos Representando Topologias Reais da Internet

Topologia	Algoritmo	Tam. Médio	Soma de Graus	Rotas Alternativas
Internet2	Dijkstra	9,32	24,74	0,68
Internet2	*MaxFlowRouting* com 2,-5	9,72	26,76	0,83
Internet2	*MaxFlowRouting* com 5,-5	9,49	26,38	0,87
Internet2	*MaxFlowRouting* com 5,-1	10,42	28,70	0,83
Géant	Dijkstra	6,37	23,94	2,14
Géant	*MaxFlowRouting* com 2,-5	6,46	25,16	2,32
Géant	*MaxFlowRouting* com 5,-5	6,77	26,42	2,36
Géant	*MaxFlowRouting* com 5,-1	7,61	30,11	2,37
RNP	Dijkstra	4,39	21,58	3,93
RNP	*MaxFlowRouting* com 2,-5	4,44	25,79	5,73
RNP	*MaxFlowRouting* com 5,-5	4,42	25,54	5,71
RNP	*MaxFlowRouting* com 5,-1	4,74	27,02	5,57
WIDE	Dijkstra	3,44	12,22	2,00
WIDE	*MaxFlowRouting* com 2,-5	3,66	17,66	5,00
WIDE	*MaxFlowRouting* com 5,-5	3,83	18,83	5,00
WIDE	*MaxFlowRouting* com 5,-1	3,83	18,83	5,00

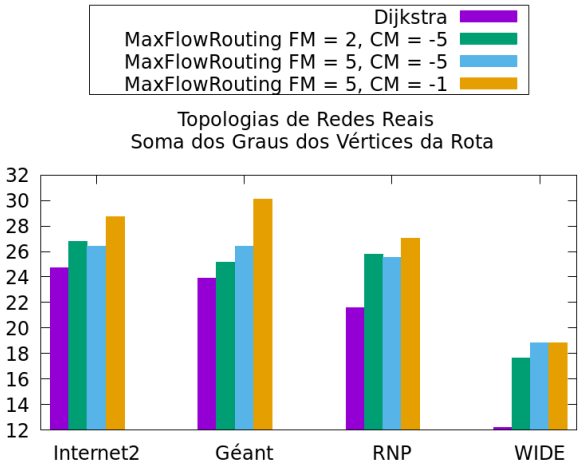

Figura 9. Comparação da média da soma dos graus dos vértices das rotas produzidas. Da esquerda para a direita: Internet2, Géant, RNP, Wide. Roxo: Dijkstra. Azul: *MaxFlowRouting* FM = 2, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -5. Amarelo: *MaxFlowRouting* FM = 5, CM = -1.

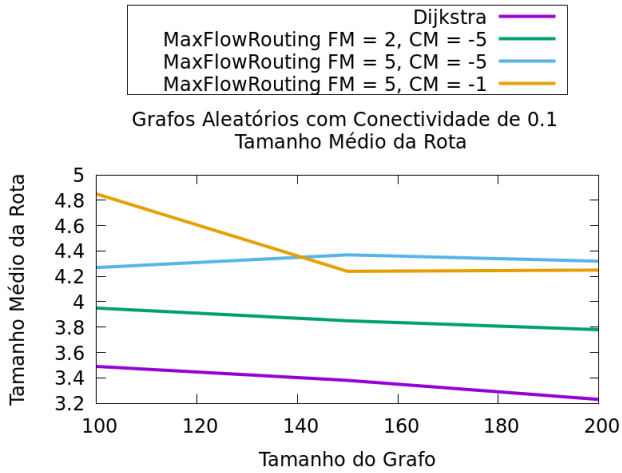

Figura 10. Comparação do tamanho médio das rotas produzidas. Da esquerda para a direita: Internet2, Géant, RNP, Wide. Roxo: Dijkstra. Azul: *MaxFlowRouting* FM = 2, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -5. Amarelo: *MaxFlowRouting* FM = 5, CM = -1.

na topologia Wide.

A Figura 10 mostra que as rotas computadas pelo algoritmo *MaxFlowRouting* com o par de pesos FM = 2 e CM = -5 são menores quando comparadas às rotas produzidas pelo algoritmo com os outros pares de pesos em quase todas as topologias. As rotas produzidas com esse par de pesos são pouco maiores do que as rotas computadas com o uso do algoritmo de Dijkstra, que produz as rotas com o menor tamanho possível. Na topologia Internet2, o par de pesos FM = 5 e CM = -5 fez com que o algoritmo tivesse o melhor desempenho nesse quesito comparado aos outros pares de pesos.

3.2 Comparação do Comportamento dos Algoritmos na Ocorrência de Falhas

A Subsubseção 3.2.1 apresenta os resultados do experimento que compara como os algoritmos se comportam ao se depararem com uma falha na rota principal para os grafos de conexão preferencial de Barabási-Albert. A Subsubseção 3.2.2 apresenta os resultados do experimento para grafos de mundo pequeno de Watts-Strogatz. A Subsubseção 3.2.3 apresenta os resultados do experimento para as topologias reais da Internet. Os resultados são apresentados considerando as métricas de tamanho médio das rotas sem ocorrência de falha, métricas de tamanho médio das rotas com ocorrência de falha e quantidade média de *backtracks*.

3.2.1 Grafos de Conexão Preferencial

No segundo experimento, foram utilizados grafos de conexão preferencial de Barabási-Albert [Albert and Barabási, 2002] de tamanho N, cujos vértices possuem m arestas. O experimento foi executado com grafos de tamanho $N = 100, 150$ e 200 cujos vértices possuem $m = 3$ arestas. Para cada valor de N, foram gerados 6 grafos diferentes. Para cada um deles, cada algoritmo de roteamento computou rotas entre 6 pares de vértices não-vizinhos selecionados aleatoriamente. Os resultados experimentais obtidos são apresentados na Tabela 4.

A Figura 11 mostra que em média, o algoritmo de Dijks-

Tabela 4. Resultados do Segundo Experimento com Grafos de Conexão Preferencial

N	Algoritmo	*Backtracks*	Tam. Médio Sem Falha	Tam. Médio com Falha
100	Dijkstra sem FRR	0,42	3,75	4,22
100	Dijkstra com FRR	0,00	3,75	4,44
100	*MaxFlowRouting* com 2,-5	0,00	3,75	4,44
100	*MaxFlowRouting* com 5,-5	0,03	3,75	5,22
100	*MaxFlowRouting* com 5,-1	0,03	3,75	5,67
150	Dijkstra sem FRR	0,39	3,78	4,22
150	Dijkstra com FRR	0,03	3,78	4,58
150	*MaxFlowRouting* com 2,-5	0,03	3,92	4,83
150	*MaxFlowRouting* com 5,-5	0,03	4,56	4,97
150	*MaxFlowRouting* com 5,-1	0,03	4,64	5,61
200	Dijkstra sem FRR	0,58	4,14	4,53
200	Dijkstra com FRR	0,00	4,14	4,92
200	*MaxFlowRouting* com 2,-5	0,00	4,19	4,94
200	*MaxFlowRouting* com 5,-5	0,00	4,25	5,39
200	*MaxFlowRouting* com 5,-1	0,00	4,25	5,22

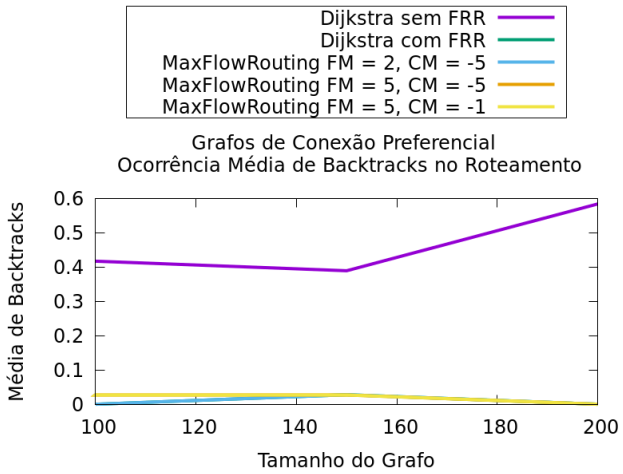

Figura 11. Comparação da ocorrência média de *backtracks* nos algoritmos de roteamento. Roxo: Dijkstra sem FRR. Verde: Dijkstra com FRR. Azul: *MaxFlowRouting* FM = 2, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -5. Amarelo: *MaxFlowRouting* FM = 5, CM = -1.

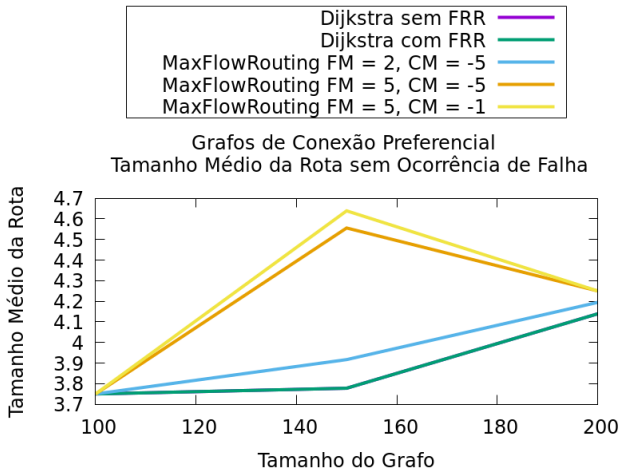

Figura 12. Comparação do tamanho médio das rotas produzidas sem a ocorrência de falha. Roxo: Dijkstra sem FRR. Verde: Dijkstra com FRR. Azul: *MaxFlowRouting* FM = 2, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -5. Amarelo: *MaxFlowRouting* FM = 5, CM = -1.

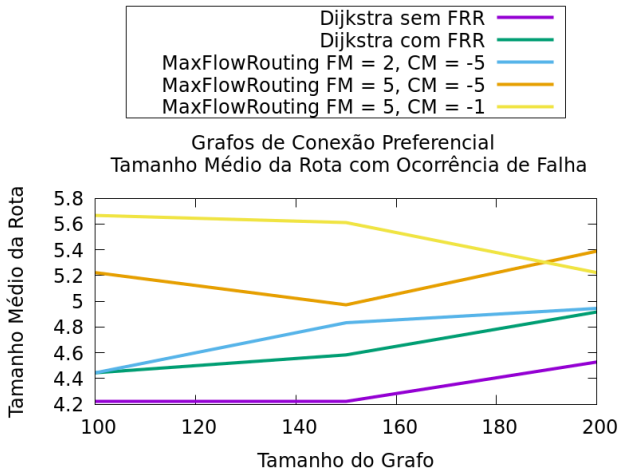

Figura 13. Comparação do tamanho médio das rotas produzidas com a ocorrência de uma falha (direita). Roxo: Dijkstra sem FRR. Verde: Dijkstra com FRR. Azul: *MaxFlowRouting* FM = 2, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -5. Amarelo: *MaxFlowRouting* FM = 5, CM = -1.

tra sem FRR precisou fazer mais *backtracks* do que os outros algoritmos de roteamento. Considerando essa métrica, houve pouca diferença nos resultados produzidos pelos algoritmos que utilizam FRR.

Considerando as métricas de tamanho médio das rotas nos cenários sem e com ocorrência de falha, as Figuras 12 e 13 mostra que o algoritmo *MaxFlowRouting* com pesos FM = 2 e CM = -5 produziu rotas com tamanho médio pouco maior do que o algoritmo de Dijkstra com FRR. O algoritmo de Dijkstra sem FRR produziu as rotas de menor tamanho médio nos dois cenários.

3.2.2 *Grafos de Mundo Pequeno*

No segundo experimento, foram utilizados grafos de mundo pequeno de Watts-Strogatz [Watts and Strogatz, 1998] de tamanho N, cujos vértices possuem k vizinhos iniciais. Cada vizinho inicial de um vértice é substituído com probabilidade p por outro vértice do grafo selecionado aleatoriamente. Os experimentos foram executados com grafos de tamanho $N = 100, 150$ e 200 cujos vértices possuem $k = 4$ arestas e com uma probabilidade de substituição de arestas $p = 0,4$. Para

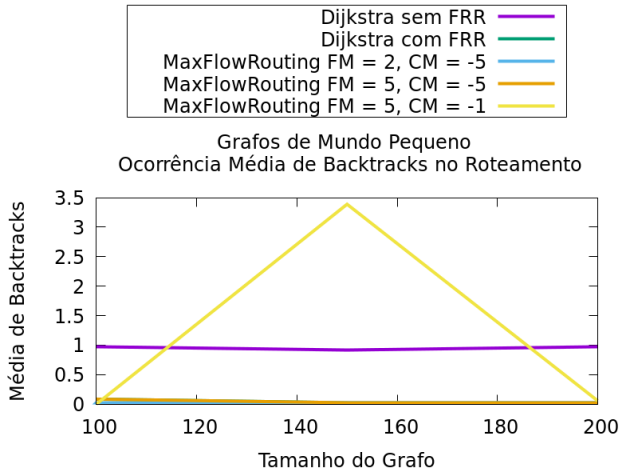

Figura 14. Comparação da ocorrência média de *backtracks* nos algoritmos de roteamento. Roxo: Dijkstra sem FRR. Verde: Dijkstra com FRR. Azul: *MaxFlowRouting* FM = 2, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -5. Amarelo: *MaxFlowRouting* FM = 5, CM = -1.

cada valor de N, 6 grafos diferentes foram gerados. Em cada um deles, cada algoritmo de roteamento produziu rotas entre 6 pares de vértices não-adjacentes selecionados aleatoriamente. Os resultados do experimento realizado são apresentados na Tabela 5.

Como mostrado na Figura 14, é notável que o algoritmo *MaxFlowRouting* com pesos FM=5 e CM=-1 apresentou alta variabilidade considerando a métrica de ocorrência média de *backtracks* no roteamento. Nos grafos de tamanho 150, o algoritmo fez, em média, 3,39 *backtracks* por roteamento, comparado a 0,00 nos grafos de tamanho 100 e a 0,06 nos grafos de tamanho 200. Nos demais casos, o algoritmo de Dijkstra sem o uso de FRR teve uma média maior de ocorrência de *backtracks* do que os algoritmos que utilizam FRR. Além disso, nos grafos de tamanho 100, o *MaxFlowRouting* fez 62,5% menos *backtracks* que o algoritmo de Dijkstra com FRR.

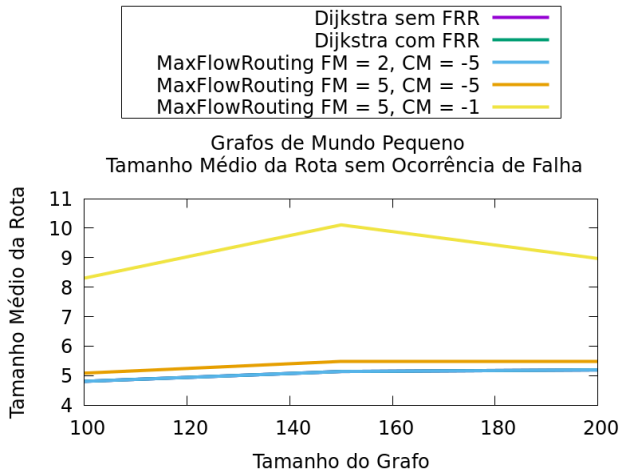

Figura 15. Comparação do tamanho médio das rotas produzidas sem a ocorrência de falha. Roxo: Dijkstra sem FRR. Verde: Dijkstra com FRR. Azul: *MaxFlowRouting* FM = 2, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -5. Amarelo: *MaxFlowRouting* FM = 5, CM = -1.

As Figuras 15 e 16 mostram que as rotas produzidas no cenário com falha possuem tamanho médio maior que no cenário sem falha. O algoritmo *MaxFlowRouting* com pesos FM=5 e CM=-1 produziu rotas com tamanho médio

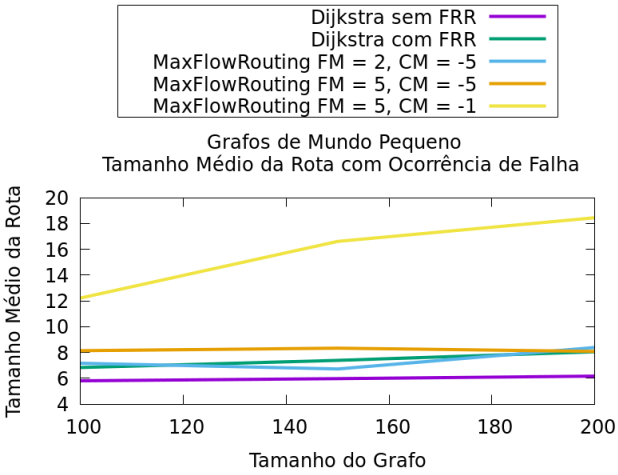

Figura 16. Comparação do tamanho médio das rotas produzidas com a ocorrência de uma falha (abaixo). Roxo: Dijkstra sem FRR. Verde: Dijkstra com FRR. Azul: *MaxFlowRouting* FM = 2, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -5. Amarelo: *MaxFlowRouting* FM = 5, CM = -1.

notavelmente maior do que as rotas produzidas pelos demais algoritmos. Além disso, o *MaxFlowRouting* com o par de pesos FM=2 e CM=-5 produziu rotas de tamanho médio com pouca diferença em comparação às rotas produzidas pelo algoritmo de FRR que usa o algoritmo de Dijkstra.

3.2.3 Topologias Reais da Internet

O segundo experimento também foi executado em grafos representando as seguintes topologias reais da Internet: Internet2 [Internet2, 2025], com $N = 54$ vértices; Géant [GÉANT, 2025], com $N = 44$ vértices; RNP [RNP, 2025], com $N = 28$ vértices; e Wide [WIDE Project, 2025], com $N = 14$ vértices. Em cada topologia, cada algoritmo de roteamento computou rotas entre 30 pares de vértices que não possuem aresta entre si. Os resultados estão apresentados na Tabela 6.

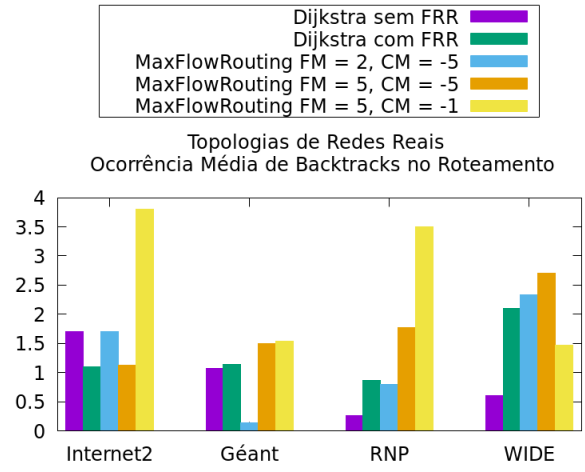

Figura 17. Comparação da ocorrência média de *backtracks* nos algoritmos de roteamento. Da esquerda para a direita: Internet2, Géant, RNP e Wide. Roxo: Dijkstra sem FRR. Verde: Dijkstra com FRR. Azul: *MaxFlowRouting* FM = 2, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -5. Amarelo: *MaxFlowRouting* FM = 5, CM = -1.

A Figura 17 mostra que houve bastante variabilidade na ocorrência de *backtracks* em cada algoritmo de roteamento nas diferentes topologias. O algoritmo *MaxFlowRouting* com pesos FM=5 e CM=-1 apresentou valores relativamente altos nas topologias Internet2 e RNP, enquanto apresentou o menor

Tabela 5. Resultados do Segundo Experimento com Grafos de Mundo Pequeno

N	Algoritmo	*Backtracks*	Tam. Médio Sem Falha	Tam. Médio com Falha
100	Dijkstra sem FRR	0,97	4,81	5,81
100	Dijkstra com FRR	0,08	4,81	6,83
100	*MaxFlowRouting* com 2,-5	0,03	4,81	7,17
100	*MaxFlowRouting* com 5,-5	0,08	5,08	8,14
100	*MaxFlowRouting* com 5,-1	0,00	8,31	12,22
150	Dijkstra sem FRR	0,92	5,14	5,97
150	Dijkstra com FRR	0,03	5,14	7,39
150	*MaxFlowRouting* com 2,-5	0,03	5,14	6,72
150	*MaxFlowRouting* com 5,-5	0,03	5,47	8,33
150	*MaxFlowRouting* com 5,-1	3,39	10,11	16,61
200	Dijkstra sem FRR	0,97	5,19	6,17
200	Dijkstra com FRR	0,03	5,19	8,06
200	*MaxFlowRouting* com 2,-5	0,03	5,19	8,39
200	*MaxFlowRouting* com 5,-5	0,03	5,47	8,08
200	*MaxFlowRouting* com 5,-1	0,06	8,97	18,44

Tabela 6. Resultados do Segundo Experimento com Grafos Representando Topologias Reais da Internet

Topologia	Algoritmo	*Backtracks*	Tam. Médio Sem Falha	Tam. Médio com Falha
Internet2	Dijkstra sem FRR	1,70	6,63	9,50
Internet2	Dijkstra com FRR	1,10	6,63	10,80
Internet2	*MaxFlowRouting* com 2,-5	1,70	6,63	11,37
Internet2	*MaxFlowRouting* com 5,-5	1,13	6,63	10,53
Internet2	*MaxFlowRouting* com 5,-1	3,80	8,13	14,03
Géant	Dijkstra sem FRR	1,07	4,97	6,47
Géant	Dijkstra com FRR	1,13	4,97	7,53
Géant	*MaxFlowRouting* com 2,-5	0,13	4,97	7,33
Géant	*MaxFlowRouting* com 5,-5	1,50	5,17	8,40
Géant	*MaxFlowRouting* com 5,-1	1,53	5,97	8,43
RNP	Dijkstra sem FRR	0,27	4,00	5,43
RNP	Dijkstra com FRR	0,87	4,00	6,17
RNP	*MaxFlowRouting* com 2,-5	0,80	4,00	6,23
RNP	*MaxFlowRouting* com 5,-5	1,77	4,00	6,30
RNP	*MaxFlowRouting* com 5,-1	3,50	4,23	7,30
Wide	Dijkstra sem FRR	0,60	3,47	2,37
Wide	Dijkstra com FRR	2,10	3,47	2,67
Wide	*MaxFlowRouting* com 2,-5	2,33	3,47	2,20
Wide	*MaxFlowRouting* com 5,-5	2,70	3,50	2,13
Wide	*MaxFlowRouting* com 5,-1	1,47	3,50	2,63

número médio de *backtracks* na topologia Wide dentre os algoritmos que utilizam FRR.

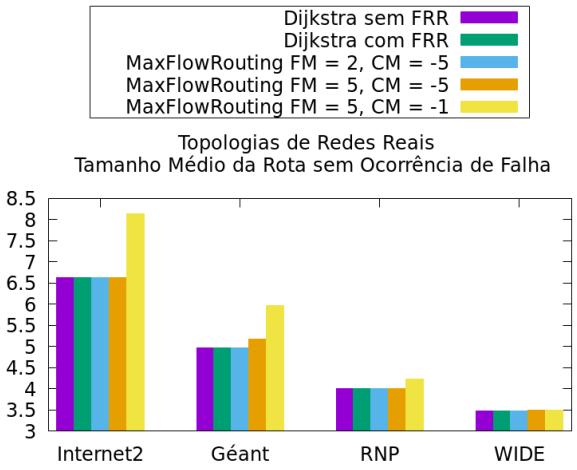

Figura 18. Comparação do tamanho das rotas produzidas em cenários sem falhas. Da esquerda para a direita: Internet2, Géant, RNP e Wide. Roxo: Dijkstra sem FRR. Verde: Dijkstra com FRR. Azul: *MaxFlowRouting* FM = 2, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -5. Amarelo: *MaxFlowRouting* FM = 5, CM = -1.

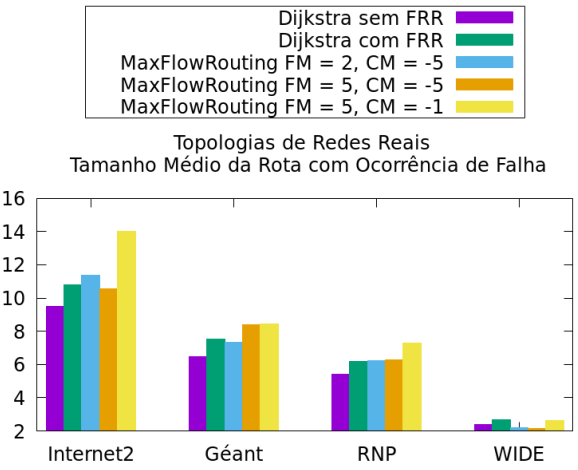

Figura 19. Comparação do tamanho médio das rotas depois de uma falha. Da esquerda para a direita: Internet2, Géant, RNP e Wide. Roxo: Dijkstra sem FRR. Verde: Dijkstra com FRR. Azul: *MaxFlowRouting* FM = 2, CM = -5. Marrom: *MaxFlowRouting* FM = 5, CM = -5. Amarelo: *MaxFlowRouting* FM = 5, CM = -1.

Nas Figuras 18 e 19, é possível ver que o algoritmo *MaxFlowRouting* com o par de pesos FM=5 e CM=-1 produziu rotas com o maior tamanho médio nos cenários sem e com a ocorrência de falha. O *MaxFlowRouting* com os outros pares de pesos produziu rotas com pouca diferença de tamanho com relação ao algoritmo de FRR que utiliza Dijkstra para computar rotas. O algoritmo de roteamento sem FRR que utiliza o algoritmo de Dijkstra para produzir rotas produziu as rotas de menor tamanho nos cenários sem e com a ocorrência de falha. Na topologia Wide, a ocorrência de falha tornou o roteamento impossível em alguns casos e, por isso, a média do tamanho das rotas com falha é menor do que a média do tamanho sem falha.

4 Conclusão

Este trabalho apresentou o algoritmo *MaxFlowRouting* para roteamento tolerante a falhas. O algoritmo atua no contexto do *Fast-ReRoute*, que é uma estratégia proativa de reparação de falhas. No FRR, uma rota alternativa é empregada por um roteador quando este detecta uma falha na rota primária. O *MaxFlowRouting* utiliza a avaliação de fluxo máximo para encontrar rotas com uma quantidade maior de rotas alternativas além de *backtracking*. Além disso, foi proposta uma estratégia de backtracking: sempre que não houver alternativa o pacota volta ao roteador anterior que o processou. Foi apresentada uma avaliação baseada em simulação. Os experimentos foram feitos em grafos de pequeno mundo, conexão preferencial e em topologias reais da Internet. Os resultados dos experimentos mostram que o *MaxFlowRouting* produz rotas com mais opções de alternativas do que o algoritmo de Dijkstra. No melhor caso, o algoritmo proposto computou rotas com até 1,54 vezes mais rotas alternativas por vértice.

Durante a elaboração do presente trabalho percebeu-se a lacuna de um *framework* em Teoria dos Grafos para suportar nossa visão de FRR, em que buscamos rotas mais conectadas. Trabalhos futuros vão desde a especificação do *framework* até a especificação de um protocolo de roteamento TCP/IP baseado no algoritmo *MaxFlowRouting*.

Declarações complementares

Financiamento

Este trabalho foi parcialmente financiado pelo CNPq, projeto número 308959/2020-5.

Contribuições dos autores

LO e ED contribuíram para a concepção deste estudo. LO implementou todo o software e realizou os experimentos. Todos os autores leram e aprovaram o manuscrito final.

Conflitos de interesse

Os autores declaram que não têm nenhum conflito de interesses.

Disponibilidade de dados e materiais

Os conjuntos de dados e softwares gerados e analisados durante o estudo atual estão disponíveis em https://gitlab.c3sl.ufpr.br/laog19/fluxo-maximo.

Referências

Albert, R. and Barabási, A. (2002). Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47. DOI: 10.1103/RevModPhys.74.47.

Bischof, Z. S., Pitcher, K., Carisimo, E., Meng, A., Bezerra Nunes, R., Padmanabhan, R., Roberts, M. E., Snoeren, A. C., and Dainotti, A. (2023). Destination unreachable: Characterizing internet outages and shutdowns. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 608–621. DOI: 10.1145/3603269.3604883.

Chiesa, M., Kamisiński, A., Rak, J., Rétvári, G., and Schmid, S. (2021). A survey of fast-recovery mechanisms in packet-switched networks. *IEEE Communications Surveys & Tutorials*, 23(2):1253–1301. DOI: 10.1109/COMST.2021.3063980.

Cohen, J., Duarte Jr, E., and Schroeder, J. (2011). Connectivity criteria for ranking network nodes. In *Complex*

Networks: Second International Workshop, CompleNet 2010, Rio de Janeiro, Brazil, October 13-15, 2010, Revised Selected Papers, pages 35–45. Springer. DOI: 10.1007/978-3-642-25501-4_4.

Cohen, J., Rodrigues, L., and Duarte Jr, E. (2012). A parallel implementation of gomory-hu's cut tree algorithm. In *2012 IEEE 24th International Symposium on Computer Architecture and High Performance Computing*, pages 124–131. IEEE. DOI: 10.1109/SBAC-PAD.2012.37.

Cohen, J., Rodrigues, L., and Duarte Jr, E. (2017). Parallel cut tree algorithms. *Journal of Parallel and Distributed Computing*, 109:1–14. DOI: 10.1016/j.jpdc.2017.04.007.

Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2022). *Introduction to Algorithms, fourth edition*. MIT Press.

Duarte, E. and Musicante, M. A. (1999). Formal specification of snmp mib's using action semantics: The routing proxy case study. In *The 6th IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 417–430. DOI: 10.1109/INM.1999.770698.

Duarte Jr, E., Garrett, T., Bona, L., Carmo, R., and Züge, A. (2010). Finding stable cliques of PlanetLab nodes. In *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, pages 317–322. IEEE. DOI: 10.1109/DSN.2010.5544300.

Duarte Jr, E., Santini, R., and Cohen, J. (2004). Delivering packets during the routing convergence latency interval through highly connected detours. In *International Conference on Dependable Systems and Networks (DSN), 2004*, pages 495–504. IEEE. DOI: 10.1109/DSN.2004.1311919.

Erdos, P. and Rényi, A. (1960). On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1):17–60. DOI: 10.1515/9781400841356.38.

Filsfils, C., Francois, P., Shand, M., Decraene, B., Uttaro, J., Leymann, N., and Horneffer, M. (2012). Loop-free alternate (LFA) applicability in service provider (SP) networks. RFC 6571. DOI: 10.17487/RFC6571.

GÉANT (2025). GÉANT network. Disponível em: `https://network.geant.org/`. Acessado em 28/03/2025.

Hagberg, A., Schult, D., and Swart, P. (2008). Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA. DOI: 10.25080/TCWV9851.

Internet2 (2025). Layer 1 service. Disponível em: `https://internet2.edu/services/layer-1/`. Acessado em 28/03/2025.

Liu, K., Fan, W., Xiao, F., Mao, H., Huang, H., and Zhao, Y. (2024). Secure paths based trustworthy fault-tolerant routing in data center networks. *Concurrency and Computation: Practice and Experience*, 36(23):e8229. DOI: 10.1002/cpe.8229.

Maske, C., Cohen, J., and Duarte Jr, E. (2020). Speeding up the gomory-hu parallel cut tree algorithm with efficient graph contractions. *Algorithmica*, 82(6):1601–1615. DOI: 10.1007/s00453-019-00658-6.

Nassu, B., Nanya, T., and Duarte Jr, E. (2007). Topology discovery in dynamic and decentralized networks with mobile agents and swarm intelligence. In *Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007)*, pages 685–690. IEEE. DOI: 10.1109/ISDA.2007.13.

Okida, L. (2024). Um Algoritmo Para Fast-ReRoute Utilizando Avaliação de Fluxo Máximo e Backtracking. Trabalho de Conclusão de Curso (TCC), BCC, UFPR. DOI: 10.13140/RG.2.2.28994.08646.

Pan, P., Swallow, G., and Atlas, A. (2005). Fast reroute extensions to RSVP-TE for LSP tunnels. RFC 4090. DOI: 10.17487/RFC4090.

RNP (2025). Ipê network. Disponível em: `https://www.rnp.br/sistema-rnp/infraestrutura-para-pesquisa/evolucao-da-rede-ipe/`. Acessado em 28/03/2025.

Schroeder, J. and Duarte Jr, E. (2007). Fault-tolerant dynamic routing based on maximum flow evaluation. In *Latin-American Symposium on Dependable Computing*, pages 7–24. Springer. DOI: 10.1007/978-3-540-75294-3_3.

Schroeder, J., Guedes, A., and Duarte Jr, E. (2004). Computing the minimum cut and maximum flow of undirected graphs. *Technical report, Federal University of Paraná*. Disponível em: `https://www.inf.ufpr.br/elias/papers/2004/RT_DINF003_2004.pdf`.

Shand, M. and Bryant, S. (2010). IP fast reroute framework. RFC 5714. DOI: 10.17487/RFC5714.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442. DOI: 10.1038/30918.

WIDE Project (2025). WIDE internet official site. Disponível em: `https://two.wide.ad.jp/`. Acessado em 28/03/2025.