ROGÉRIO SANTINI

ROTEAMENTO TOLERANTE A FALHAS BASEADO EM DESVIOS DE ALTA CONECTIVIDADE

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. Elias Procópio Duarte Júnior.

CURITIBA
2003

Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno Rogério Santini, avaliamos o trabalho intitulado, "*Roteamento Tolerante a Falhas Baseado em Desvios de Alta Conectividade* ", cuja defesa foi realizada no dia 07 de novembro de 2003, às quatorze horas, no Anfiteatro A do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 07 de novembro de 2003.

Prof. Dr. Elias Procópio Duarte Jr.
DINF/UFPR – Orientador

Prof. Dr. Joni da Silva Fraga
UFSC - Membro Externo

Prof. Dr. André Luiz Pires Guedes
DINF/UFPR - Membro Interno

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

Agradecimentos

Ao meu orientador, professor Elias Procópio Duarte Júnior, pela sua compreensão, paciência e total dedicação.

À minha esposa Francielen, pelo apoio incondicional e sobretudo pela paciência em todos os momentos deste trabalho.

Aos professores Jaime Cohen e André Guedes, pela ajuda constante e importante contribuição para a realização deste trabalho.

Aos meus pais, Sérgio e Marlene, e meus irmãos, Fabiano e Liliane, por sempre terem me apoiado em todos os momentos da minha caminhada escolar e acadêmica.

Aos meus amigos do Mestrado, que de alguma forma contribuíram para a realização deste trabalho.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho apresenta uma abordagem para roteamento tolerante a falhas baseada em desvios de alta conectividade. O objetivo desta abordagem é permitir que os nodos da rede continuem a se comunicar, mesmo durante o período de tempo em que o protocolo de roteamento utilizado ainda não atualizou as tabelas de rotas dos roteadores para refletir uma falha ocorrida na rede. O protocolo de roteamento BGP-4 (*Border Gateway Protocol* versão 4), utilizado no roteamento da Internet, pode levar minutos para atualizar as tabelas de rotas dos roteadores de modo a refletir uma falha ocorrida na rede, gerando potenciais perdas de pacotes e de conexão entre as aplicações que se comunicam através da rede. Para evitar que tal situação ocorra, rotas alternativas são criadas antes do protocolo de roteamento atualizar as tabelas de rotas dos roteadores, ou seja, sem a informação de qual parte da rede está falha. Desta forma, as rotas alternativas devem possuir grande probabilidade de desviar de uma falha ocorrida na rede, sem o conhecimento da localização da falha. As rotas alternativas são criadas através de nodos da rede chamados desvios, que pertencem a componentes que possuem alta conectividade. Tais componentes possuem um maior número de caminhos distintos. Desta forma, é maior a probabilidade da rota alternativa criada através de um desvio pertencente a um componente de alta conectividade ser funcional, mesmo na presença de falhas na rede. Critérios de conectividade são utilizados na seleção dos desvios. Além do algoritmo exato para cálculo dos critérios de conectividade, uma heurística também é apresentada. Grafos utilizados na obtenção dos resultados experimentais foram gerados aleatoriamente utilizando o método Waxman, que se propõe a capturar a característica de localidade existente nas redes reais. Os resultados experimentais, obtidos pela implementação dos algoritmos na linguagem Java, demonstram que a cobertura de falhas obtida pela utilização do melhor desvio, selecionado a partir dos critérios de conectividade propostos, chega a 90%. Já a cobertura de falhas obtida pela utilização dos três melhores desvios chega a 98%. Tais resultados visam comprovar que as rotas alternativas criadas através de desvios selecionados a partir dos critérios de conectividade propostos possuem boa probabilidade de desviar de uma falha ocorrida na rede, mesmo sem a informação da localização da falha.

Abstract

This work presents a novel approach for fault-tolerant routing based on highly connected detours. Routing protocols present a convergence latency for all routers to update their routing tables after a fault occurs in the network. During this time interval, which may be of up to minutes, packets may be lost before reaching their destinations. In order to allow nodes to continue communicating during the convergence latency interval, we propose the use of alternative routes that are employed to send packets that were lost. These alternative routes are chosen without the knowledge of which node or link is faulty. Furthermore, alternative routes must present a good probability of being able to bypass the faulty element. In the proposed approach, alternative routes are created through network nodes called detours, that belong to network components that have high connectivity. These components present a larger number of distinct paths, thus the probability that the alternative route created through a detour is faulty-free is higher. The connectivity criteria used to select detours are presented. An exact algorithm to compute those criteria, as well as an efficient heuristic are also presented. Experimental results were obtained with random graths generated with the Waxman method, which aims at producing Internet-like topologies. Results show that the fault coverage obtained through the usage of the best detour, computed based on the proposed connectivity criterias, is up to 90%. When the three best detours are considered, the fault coverage is up to 98%. These results confirm the effectiveness of the proposed criteria for selecting detours that avoid unknown faulty elements.

Capítulo 1

Introdução

A Internet é formada por um conjunto de entidades chamadas Sistemas Autônomos (AS - *Autonomous Systems*), conectadas arbitrariamente [1]. Um AS utiliza um protocolo de roteamento interno para rotear pacotes dentro do AS e um protocolo de roteamento externo para rotear pacotes para outros AS's [2]. Através de um protocolo de roteamento externo, um AS deve anunciar suas redes internas para outros AS's para que elas sejam visíveis na Internet. Além disso, cada AS pode definir políticas de roteamento que influenciam na escolha da melhor rota para um determinado destino.

Os protocolos de roteamento interno e externo são implementações de algoritmos de roteamento. Os algoritmos vetor de distância, vetor de caminhos e o algoritmo do caminho mínimo são exemplos de algoritmos de roteamento. O algoritmo vetor de distância também é conhecido como algoritmo de Bellman-Ford [3]. Este algoritmo utiliza a distância para alcançar uma determinada rede, calculada em número de *hops*, na escolha da melhor rota para tal rede. O número de *hops* é definido como o número de nodos ou roteadores que devem ser atravessados para alcançar uma determinada rede. O protocolo de roteamento interno RIP (*Routing Information Protocol*) [4] é uma implementação do algoritmo vetor de distância. O algoritmo vetor de caminhos é uma variação do algoritmo vetor de distância. O algoritmo vetor de caminhos utiliza um vetor de caminhos que informa quais AS's foram atravessados pelo anúncio de uma determinada rota. O vetor de caminhos é utilizado na detecção de *loops* no roteamento. O protocolo de roteamento

externo BGP (*Border Gateway Protocol*) versão 4 [2], utilizado no roteamento da Internet, é uma implementação do algoritmo vetor de caminhos. Já no algoritmo do caminho mínimo, cada roteador possui a topologia completa da rede e calcula a melhor rota para cada destino utilizando o algoritmo da árvore de caminhos mínimos de Dijkstra [5]. O protocolo OSPF (*Open Shortest Path First*) [6] é uma implementação do algoritmo vetor de caminhos.

Os protocolos de roteamento levam uma certa quantidade de tempo para atualizar as tabelas de rotas de todos os roteadores, de modo a refletir uma mudança que tenha ocorrido na topologia da rede. Este período de tempo é conhecido como *latência* do protocolo. Quando as tabelas de rotas de todos os roteadores é atualizada, é dito que o protocolo de roteamento convergiu para uma nova tabela de rotas estável, ou seja, alcançou a convergência [7]. A latência média do protocolo BGP e conseqüentemente da Internet é de 3 minutos, sendo que foram observados períodos de latência de até 15 minutos [8]. Durante o período de tempo compreendido pela latência do protocolo, alguns roteadores possuem informações inconsistentes na sua tabela de rotas, gerando potenciais perdas de pacotes e de conexão entre as aplicações que se comunicam através da rede [8].

Este trabalho apresenta uma abordagem para roteamento tolerante a falhas, que tem como objetivo permitir que os nodos continuem a se comunicar, mesmo durante o período de tempo em que as tabelas de rotas dos roteadores ainda não foram atualizadas para refletir uma falha ocorrida na rede. Os nodos da rede que comunicam-se utilizando a rota falha fazem uso de uma *rota alternativa*, que possui grande probabilidade de desviar da falha. A rota alternativa é utilizada antes do protocolo de roteamento alcançar a convergência. Portanto, a rota alternativa é utilizada sem a informação da localização da falha.

Rotas alternativas são criadas através de nodos da rede chamados *desvios*. Um *desvio* é um nodo que possui a capacidade de atuar como uma *ponte* na comunicação entre dois nodos da rede. Nodos da rede pertencentes a componentes de maior conectividade possuem preferência para serem escolhidos como desvios. O número de caminhos distintos que atravessam um componente de maior conectividade é maior, se comparado a um

componente de menor conectividade. Desta forma, é maior a probabilidade da rota alternativa, criada através de um desvio pertencente a um componente de maior conectividade, ser funcional mesmo na presença de falhas na rede.

Os critérios de conectividade *número de conectividade* $\#C(v)$ e *conjunto* $MCC(v)$ propostos em [9] são utilizados na determinação da conectividade de cada nodo da rede. A topologia da rede sem a presença de falhas, representada por um grafo não direcionado, é utilizada no cálculo dos critérios de conectividade $\#C(v)$ e $MCC(v)$. Em relação a um grafo não direcionado G, o $\#C(v)$, para $v \in G$, refere-se à conectividade de arestas do subconjunto de vértices de G que contém v e que possui a maior conectividade de arestas. Já o $MCC(v)$ refere-se ao maior subconjunto de vértices de G que contém v e que possui conectividade de arestas igual a $\#C(v)$. Os critérios de conectividade $\#C(v)$ e $MCC(v)$ são calculados utilizando uma árvore de Gomory-Hu [10] ou árvore de corte do grafo que representa a topologia da rede. O $\#C(v)$ e $MCC(v)$ podem ser obtidos em tempo linear através da árvore de corte. Porém, a construção da árvore de corte é um processo custoso, que possui ordem de complexidade $\mathcal{O}(V^2 \cdot E)$.

Para reduzir o custo do cálculo dos critérios de conectividade, uma heurística é apresentada. A cada iteração, a heurística procura por nodos pertencentes a componentes 2-aresta conexos. Tais nodos têm o seu $\#C(v)$ estimado incrementado de duas unidades. Os nodos que não pertencem a componentes 2-aresta conexos têm o seu $\#C(v)$ incrementado em uma unidade. A heurística utiliza uma árvore de busca em profundidade (DFT - *Depth First Tree*) e um conjunto de arestas chamadas arestas de cobertura (*cover edges*) para encontrar o conjunto de componentes 2-aresta conexos. Em redes esparsas a ordem de complexidade da heurística é dada por $\mathcal{O}(V + E)$. A heurística não realiza cálculo do $MCC(v)$. Porém, o $MCC(v)$ não é estritamente necessário na seleção do desvio para um par de nodos.

O algoritmo de seleção do desvio para um par de nodos a e b leva em consideração os critérios de conectividade $\#C(v)$, $MCC(v)$ e o comprimento da rota alternativa criada através do nodo candidato a desvio. O comprimento da rota alternativa é levado em consideração para evitar que nodos pertencentes a componentes altamente conexos, mas

muito distantes dos nodos a e b sejam escolhidos para desvios. Caso a heurística tenha sido utilizada no cálculo do $\#C(v)$, o critério $MCC(v)$ não é utilizado no algoritmo de seleção do melhor desvio. O algoritmo retorna uma lista ordenada contendo todos os nodos da rede, em ordem de preferência dos melhores desvios para os nodos a e b, de acordo com os critérios de conectividade propostos.

A figura 1.1 apresenta um exemplo de seleção do melhor desvio para os nodos X e Y, selecionado a partir dos critérios de conectividade propostos. Assumindo que a rota entre os nodos X e Y está falha, a melhor escolha é criar uma rota alternativa usando o nodo A como desvio. O nodo A é preferido em relação ao nodo B por pertencer a um componente de maior conectividade. Desta forma, a probabilidade da rota alternativa criada através do nodo A estar sem falhas é maior, se comparada à rota alternativa criada através do nodo B. O nodo A é preferido em relação ao nodo C pois a rota alternativa criada através do nodo A possui um comprimento inferior ao comprimento da rota alternativa criada através do nodo C.

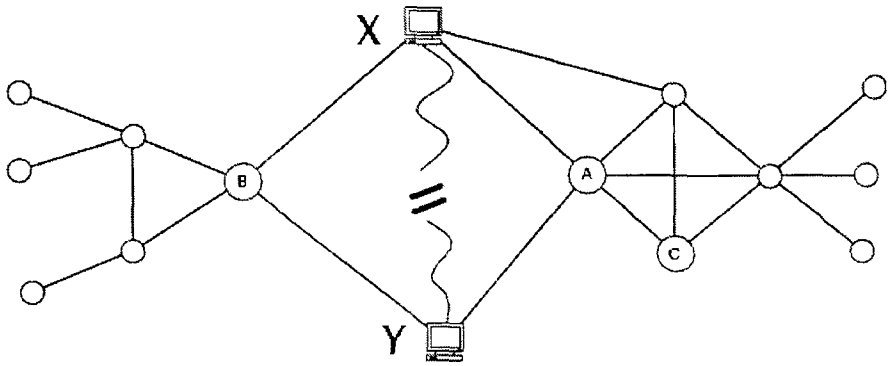


Figura 1.1: Um exemplo da seleção do melhor desvio para um par de nodos X e Y.

As redes de computadores são freqüentemente modeladas através de grafos. Além disso, grafos gerados de maneira aleatória são utilizados para modelar redes de topologia arbitrária, como a Internet. Existem vários métodos de geração aleatória de grafos para a representação de redes de computadores, classificados em três classes: *flat random*, regular e hierárquica [11]. Os métodos pertencentes à classe *flat random* constroem grafos adicionando probabilisticamente arestas a um dado conjunto de vértices. Os métodos pertencentes à classe regular geram grafos que possuem topologias conhecidas como estrelas,

anéis e hipercubos, não possuindo nenhuma característica aleatória. Já os grafos pertencentes à classe hierárquica constroem grandes topologias de forma hierárquica, através de componentes topológicos menores. Além destes métodos, existem geradores de grafos que baseiam-se nas leis de potência da Internet [12, 13]. As leis de potência representam de forma concisa várias propriedades topológicas da Internet [14]. O método Waxman, pertencente à classe *flat random*, foi o método escolhido para a geração aleatória dos grafos que foram utilizados nos experimentos realizados neste trabalho. Tal método leva a consideração a distância entre dois vértices na inclusão de uma aresta no grafo, tentando desta forma capturar a característica de localidade existente nas redes de computadores reais.

Os algoritmos que calculam os critérios de conectividade $\#C(v)$ e $MCC(v)$ através da construção da árvore de corte, o algoritmo utilizado no cálculo heurístico do $\#C(v)$ e o algoritmo de seleção do melhor desvio para um par de nodos foram implementados na linguagem Java [15]. A linguagem Java foi escolhida pela possibilidade de utilização de diversas classes já existentes. Classes Java do projeto JDigraph [16], que implementam alguns conceitos de grafos, foram utilizadas.

Para avaliar a eficiência da estratégia de seleção de desvios, foram implementados algoritmos para cálculo da cobertura de falhas quando são utilizados o primeiro, segundo e terceiro melhores desvios, selecionados a partir dos critérios de conectividade propostos. Para cada par de nodos u e v do grafo, o algoritmo de cobertura de falhas obtém o caminho mínimo p que os nodos u e v utilizam para se comunicar. O algoritmo então falha cada aresta do caminho p e verifica se a rota alternativa consegue desviar da aresta falha. Foram obtidos resultados de cobertura de falhas quando a árvore de corte é utilizada no cálculo $\#C(v)$ e $MCC(v)$, e quando a heurística é utilizada no cálculo do $\#C(v)$. Por fim, foi implementado um algoritmo que calcula a diferença entre o $\#C(v)$ médio obtido através da árvore de corte e o $\#C(v)$ médio obtido através da heurística, com o objetivo de verificar se a heurística obtém valores de $\#C(v)$ próximos aos valores obtidos através da árvore de corte.

Para a obtenção dos resultados experimentas, foram gerados grafos aleatórios utili-

zando o método Waxman, com tamanhos que variam de 10 a 100 nodos e grau médio dos nodos variando de 3 a 8. Os resultados experimentais demonstram que a heurística obtém valores de $\#C(v)$ próximos aos valores obtidos através da árvore de corte. A cobertura de falhas obtida pelo melhor desvio selecionado a partir dos critérios de conectividade propostos variou de 70% a 90%, dependendo do grau médio dos nodos do grafo. Quando os três melhores desvios foram utilizados em conjunto, a cobertura de falhas variou de 81% até 99%, dependendo do grau médio dos nodos do grafo. A cobertura de falhas obtida pelos melhores desvios selecionados utilizando o $\#C(v)$ retornado pela heurística apresentou valores similares aos valores apresentados acima. Estes resultados visam comprovar que rotas alternativas, criadas através de desvios selecionados a partir dos critérios de conectividade propostos, possuem boa probabilidade de desviar de uma falha na rede, mesmo sem a informação da localização da falha.

O restante deste trabalho está organizado da seguinte maneira. O capítulo 2 apresenta a arquitetura de roteamento da Internet. São apresentados os algoritmos de roteamento já citados e alguns protocolos de roteamento interno e externo que implementam tais algoritmos. Uma análise da latência do protocolo BGP também é apresentada. O capítulo 3 apresenta o roteamento tolerante a falhas baseado em desvios de alta conectividade. Os algoritmos utilizados no cálculo dos critérios de conectividade e na seleção do melhor desvio são apresentados. O capítulo 4 apresenta métodos de geração de grafos para a representação de topologias de redes de computadores. O capítulo 5 apresenta os resultados experimentais obtidos pela implementação da estratégia de seleção de desvios. Finalmente, o capítulo 6 apresenta as conclusões finais do trabalho.

Capítulo 2

Arquitetura de Roteamento da Internet

A Internet pode ser vista como um conjunto de entidades chamadas Sistemas Autônomos (AS - *Autonomous Systems*) conectadas arbitrariamente [1]. A definição clássica de um Sistema Autônomo (AS) é um conjunto de roteadores sob uma única administração técnica, que utilizam um protocolo de roteamento interno (IGP - *Interior Gateway Protocol*) para rotear pacotes dentro do AS, e um protocolo de roteamento externo para rotear pacotes para outros Sistemas Autônomo [2]. Cada AS é identificado por um número exclusivo de comprimento 16 bits.

Através de um protocolo de roteamento interno, os roteadores de um AS anunciam uns aos outros as redes que são internas ao AS. Um ou mais roteadores do AS anunciam tais redes para outros AS's utilizando um protocolo de roteamento externo. Estes roteadores são conhecidos como roteadores de borda. Um AS deve anunciar suas redes internas para outros AS's para permitir que tais redes sejam visíveis na Internet [3]. A figura 2.1 exemplifica a utilização dos protocolos de roteamento interno e externo em dois AS's.

Um AS define suas próprias políticas de roteamento. Através das políticas de roteamento, um AS pode especificar quais rotas são anunciadas a um determinado AS, pode definir filtros e atribuir graus de preferências para as rotas recebidas de um determinado AS [1].

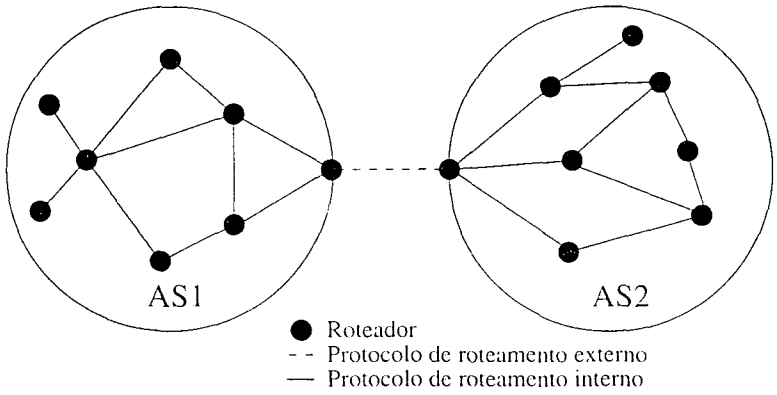


Figura 2.1: Utilização dos protocolos de roteamento interno e externo.

Os AS's são classificados como AS's de trânsito e AS's *stub*. Os AS's de trânsito formam o *backbone* da Internet e servem para interconectar AS's *stub* de forma eficiente. Um AS *stub* deve se conectar a um ou mais AS's de trânsito. Um AS *stub* que se conecta a mais de um AS de trânsito é chamado AS *multi-homed stub*. AS's *stub* também podem possuir ligações entre si. Um AS *stub* se comunica com outro AS *stub* através dos AS's de trânsito ou diretamente, caso exista uma ligação entre eles. A figura 2.2 apresenta um exemplo da interligação entre AS's *stub* e AS's de trânsito.

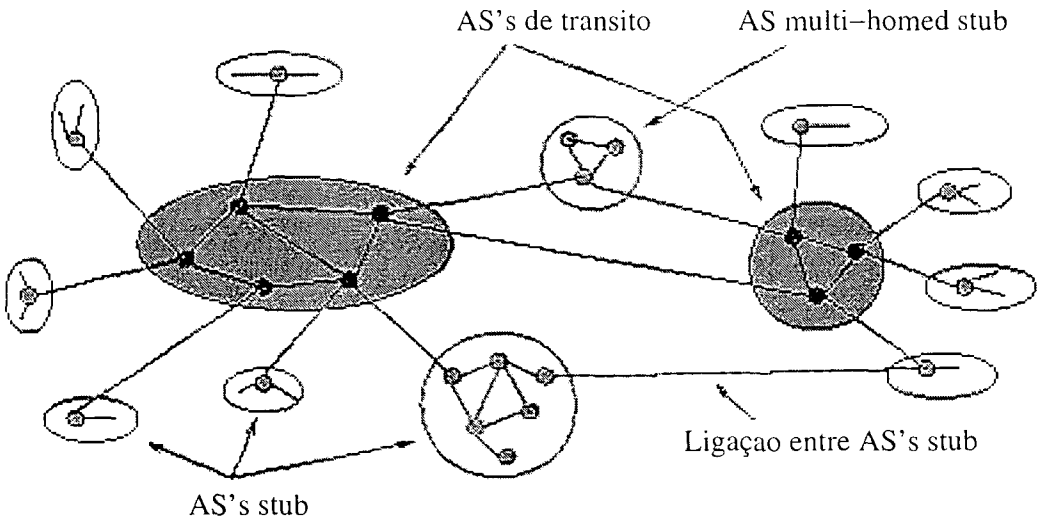


Figura 2.2: Interconexões entre transit e stub AS's.

Protocolos de roteamento implementam algoritmos de roteamento. Algoritmos de roteamento são apresentados na seção 2.1. A seção 2.2 apresenta o protocolo de roteamento

interno RIP. A seção 2.3 apresenta o protocolo de roteamento interno OSPF. A seção 2.4 apresenta o protocolo de roteamento externo BGP. Por fim, a seção 2.5 apresenta uma análise da latência do protocolo BGP.

2.1 Algoritmos de Roteamento

Os protocolos de roteamento interno e os protocolos de roteamento externo são implementações de algoritmos de roteamento. A seguir, o algoritmo vetor de distância, o algoritmo vetor de caminhos e o algoritmo do caminho mínimo são apresentados.

2.1.1 Algoritmo Vetor de Distância

O algoritmo vetor de distância também é conhecido como algoritmo de Bellman-Ford. O termo *vetor de distância* vem do fato do protocolo incluir um vetor de distância associado a cada rota sendo anunciada [17]. A distância de cada rota pode ser representada pelo número de *hops* (roteadores) que devem ser atravessados para alcançar a rede sendo anunciada.

Quando um roteador que roda o algoritmo vetor de distância é inicializado, ele preenche sua tabela de rotas com as redes que estão diretamente conectadas a ele. Cada entrada na tabela de rotas é composta por uma rede de destino e pela distância necessária para alcançar tal rede. A tabela de rotas inicial de um roteador que possui duas redes diretamente conectadas é apresentada na tabela 2.1.

Destino	**Distância**	**Rota**
Rede 1	0	Direto
Rede 2	0	Direto

Tabela 2.1: Tabela de rotas inicial de um roteador com duas redes diretamente conectadas.

Periodicamente, cada roteador anuncia sua tabela de rotas aos outros roteadores que pertencem à suas redes diretamente conectadas. Um roteador X instala uma rota recebida em um anúncio de um roteador Y se:

- O roteador Y possui a menor distância para alcançar a rede anunciada;

- O roteador X não possui uma entrada para a rede anunciada em sua tabela de rotas;
- O roteador X já possui uma entrada para a rede em sua tabela de rotas através do roteador Y, mas com uma distância diferente.

O roteador X atualiza a distância da rota recém instalada com o valor da distância recebida do roteador Y acrescido de uma unidade. A tabela 2.2 mostra o exemplo de uma tabela de rotas de um roteador X e a tabela de rotas recebida de um roteador Y.

Tabela de Rotas Roteador X

Destino	Distância	Rota
Rede 1	0	Direto
Rede 2	0	Direto
Rede 4	8	Roteador L
Rede 17	5	Roteador M
Rede 24	6	Roteador J
Rede 30	2	Roteador Q
Rede 42	2	Roteador Y

Rotas Recebidas de Y

Destino	Distância
Rede 1	2
Rede 4	**3**
Rede 17	6
Rede 21	**4**
Rede 24	5
Rede 30	1
Rede 42	**3**

Tabela 2.2: Tabela de rotas de um roteador X e rotas recebidas de um roteador Y. As entradas que serão utilizadas pelo roteador X são apresentadas em negrito.

A tabela 2.3 mostra a tabela de rotas atualizada do roteador X, depois do anúncio recebido do roteador Y.

Tabela de Rotas Roteador X

Destino	Distância	Rota
Rede 1	0	Direto
Rede 2	0	Direto
Rede 4	4	Roteador Y
Rede 17	5	Roteador M
Rede 21	5	Roteador Y
Rede 24	6	Roteador J
Rede 30	2	Roteador Q
Rede 42	4	Roteador Y

Tabela 2.3: Tabela de rotas do roteador X, após anúncio recebido do roteador Y.

Apesar de ser de fácil implementação, o algoritmo vetor de distância possui desvantagens. Quando uma rota sofre uma alteração, a informação propaga-se lentamente de um roteador para outro, permitindo que alguns roteadores possuam informações inconsistentes na sua tabela de rotas. Além disso, os anúncios realizados por um roteador

contém todas as entradas de sua tabela de rotas. Desta forma, quando a tabela de rotas possui muitas entradas, os anúncios tornam-se grandes, causando uma maior utilização de recursos de rede e de processamento dos roteadores. E finalmente, o fato de todos os roteadores da rede terem que participar do algoritmo vetor de distância pode levar à geração de uma grande quantidade de anúncios [3].

2.1.2 Algoritmo Vetor de Caminhos

O algoritmo vetor de caminhos é bem similar ao algoritmo vetor de distância. No algoritmo vetor de caminhos, um roteador anuncia a seus roteadores vizinhos todas as rotas que ele possui em sua tabela de rotas, como acontece no algoritmo vetor de distância. Porém, ao invés de utilizar um vetor de distância, é utilizado um vetor de caminhos associado a cada rota. O vetor de caminhos informa a lista dos AS's que devem ser atravessados para alcançar a rede de destino.

No roteamento externo entre AS's, nem sempre o menor caminho para alcançar uma rede de destino é o caminho escolhido. Por exemplo, um AS pode escolher uma rota mais longa para alcançar um destino, mas que passe por um AS que possua enlaces menos congestionados ou que ofereça um serviço de conexão de menor custo. Desta forma, o algoritmo vetor de distância e o algoritmo do caminho mínimo, apresentado a seguir, não podem ser utilizados no roteamento entre AS's, pois se o caminho mais curto para um destino não é sempre o caminho preferido, *loops* podem ocorrer no roteamento [7].

No algoritmo vetor de distâncias, quando um roteador recebe um anúncio de rota, ele verifica se o número de seu AS está presente no vetor de caminhos da rota sendo anunciada. Se for o caso, o roteador recusa o anúncio da rota. Senão, o roteador deve adicionar o número de seu AS no vetor de caminhos da rota, antes de anunciá-la a outro AS. Tal abordagem elimina a existência de *loops* no roteamento, mesmo quando os AS's adotam estratégias diferentes para escolher a melhor rota para um determinado destino [7].

2.1.3 Algoritmo do Caminho Mínimo

No algoritmo do caminho mínimo, os roteadores possuem um mapa que representa a topologia completa da rede. Neste mapa, os roteadores são representados por vértices e os enlaces existentes entre os roteadores são representados por arestas entre os vértices.

Cada roteador testa seus enlaces enviando mensagens a seus roteadores vizinhos. Periodicamente, os roteadores enviam informações do estado de seus enlaces para todos os outros roteadores, através de mensagens de *broadcast*. As informações de estado dos enlaces não anunciam rotas, mas somente reportam se a comunicação é possível entre dois pares de roteadores. Quando um roteador recebe uma informação de estado de enlace, ele atualiza seu mapa da rede e recalcula a melhor rota para cada destino, aplicando o algoritmo da árvore de caminhos mínimos de Dijkstra [5].

Uma das vantagens do algoritmo do caminho mínimo em relação ao algoritmo vetor de distância é o fato de cada roteador computar as melhores rotas de maneira independente, não necessitando do resultado da computação de roteadores intermediários, garantindo assim uma rápida convergência. As mensagens trocadas pelo algoritmo do caminho mínimo são pequenas pois contém apenas informações do estado dos enlaces dos roteadores e não toda a tabela de rotas, como acontece no algoritmo vetor de distância. Por estes motivos, o algoritmo do caminho mínimo é mais escalável que o algoritmo vetor de distância [3].

2.2 O Protocolo RIP

O protocolo RIP (*Routing Information Protocol*), padronizado na RFC-1058 [4] é um protocolo de roteamento interno que implementa o algoritmo vetor de distância [3].

Os roteadores que rodam o protocolo RIP realizam anúncios de rotas através de mensagens UPDATE. Uma mensagem UPDATE contém o prefixo IP da rede que está sendo anunciada e a distância para alcançar a rede, contada em número de *hops*. Quando um roteador recebe uma mensagem UPDATE, ele atualiza sua tabela de rotas aplicando o algoritmo vetor de distância. Um roteador realiza *broadcast* de mensagens UPDATE a cada 30 segundos, ou quando ocorrer alguma modificação na sua tabela de rotas. Caso

um roteador não receba mensagens UPDATE de uma determinada rota em um intervalo de 180 segundos, a rota é marcada com inválida.

O protocolo RIP marca uma rota como inválida atribuindo o valor 16 para a sua distância. Ou seja, a maior distância para uma rota válida deve ser 15 hops. Esta característica impõe restrições no tamanho e topologia das redes que utilizam o protocolo RIP, já que a distância entre dois roteadores da rede deve ser de no máximo 15 *hops*.

O algoritmo vetor de distância possui alguns problemas conhecidos como convergência lenta e contagem ao infinito. Tais problemas ocorrem porque os anúncios de rotas propagam-se lentamente através da rede. O protocolo RIP resolve estes problemas através de técnicas específicas como *split horizon* e *triggered updates* [3].

2.3 O Protocolo OSPF

O protocolo OSPF (*Open Shortest Path First*), padronizado na RFC-1583 [6], é uma implementação do algoritmo do caminho mínimo.

O protocolo OSPF permite que uma grande rede seja particionada em áreas, facilitando o gerenciamento do protocolo e permitindo um crescimento futuro da rede. Os roteadores pertencentes a uma área não têm acesso à topologia da rede de outra área.

O protocolo OSPF provê balanceamento de rotas. Se existirem múltiplas rotas de igual custo para um mesmo destino, o tráfego será dividido igualmente entre todas as rotas. Além disso, o protocolo OSPF, suporta o roteamento de subredes.

Quatro tipos de mensagem são utilizadas pelo protocolo OSPF: HELLO, DATABASE DESCRIPTION, LINK STATUS REQUEST e LINK STATUS UPDATE, descritas a seguir:

HELLO: Os roteadores periodicamente enviam mensagens HELLO em cada um dos seus enlaces, para testar se seus roteadores vizinhos estão ativos.

DATABASE DESCRIPTION: Mensagem trocada entre os roteadores para a inicialização da base de dados da topologia da rede.

LINK STATUS REQUEST: Através desta mensagem, um roteador pode solicitar informações atualizadas da topologia da rede para outro roteador.

LINK STATUS UPDATE: Através desta mensagem, um roteador informa outros roteadores do estado dos seus enlaces.

2.4 O Protocolo BGP Versão 4

O protocolo de roteamento externo utilizado na Internet nos dias atuais, para o roteamento de pacotes entre AS's é o protocolo BGP (*Border Gateway Protocol*) versão 4 [3], padronizado pela RFC-1771 [2, 18].

Neste trabalho, um roteador que roteia pacotes utilizando o protocolo BGP é chamado roteador BGP. Um AS pode possuir múltiplos roteadores BGP que comunicam-se com outros AS's. Neste caso, cada um destes roteadores BGP deve manter conexões BGP com todos os outros roteadores BGP do AS, para garantir uma visão consistente das rotas que são externas ao AS. Ou seja, todos os roteadores BGP do AS podem coerentemente selecionar uma única rota para cada rede externa ao AS. O BGP assume que o roteamento dentro do AS é realizado por um algoritmo de roteamento interno, e não faz nenhuma referência sobre quais são estes algoritmos e como eles devem ser configurados [2].

O protocolo BGP implementa o algoritmo vetor de caminhos. Desta forma, o BGP possui mecanismos que evitam a ocorrência de *loops* no roteamento, mesmo quando os AS's utilizam políticas diferentes para selecionar a melhor rota para cada destino.

O protocolo BGP é um protocolo de aplicação que roda sobre o protocolo TCP (*Transmission Control Protocol* [3]). Ou seja, o protocolo BGP roda sobre um protocolo de transporte confiável. Isto elimina a necessidade de implementação no próprio BGP de uma série de controles como fragmentação de mensagens, retransmissão e confirmação de recebimento de pacotes.

2.4.1 Sumário da Operação do Protocolo BGP

As rotas do protocolo BGP contém uma lista de atributos chamados *Path Atributes*. Os *Path Atributes* são utilizados para a detecção de *loops* no roteamento e na aplicação de políticas de roteamento definidas pelo AS.

Após o estabelecimento da conexão TCP entre dois roteadores, toda tabela de rotas BGP é transmitida. Anúncios incrementais são enviados quando ocorrem mudanças na tabela de rotas, conservando assim a banda da rede.

Mensagens de *keepalive* são enviadas periodicamente para garantir que a conexão BGP continua ativa. Mensagens de notificação são enviadas em resposta a erros ou condições especiais. Se uma conexão BGP atinge uma condição de erro, uma mensagem de notificação é enviada e a conexão é fechada.

Um roteador BGP anuncia para seus pares (outros roteadores BGP com os quais ele se comunica) em AS's vizinhos somente a melhor rota para cada destino.

2.4.2 Mensagens do Protocolo BGP

O protocolo BGP possui 4 tipos de mensagem: OPEN, UPDATE, NOTIFICATION e KEEPALIVE.

A mensagem do tipo OPEN é enviada por cada roteador que participa de uma conexão BGP após o estabelecimento da conexão TCP. As seguintes informações são encontradas na mensagem OPEN:

- Versão do protocolo BGP;
- Número do AS do roteador;
- *Hold Time*: O *Hold Time* indica o número máximo de segundos que pode transcorrer entre o recebimento de sucessivas mensagens do tipo KEEPALIVE ou UPDATE. O roteador deve escolher o menor valor para o *Hold Time* entre o valor proposto e o valor recebido do outro roteador na mensagem OPEN. O valor sugerido na RFC 1771 [2] para o valor do *Hold Time* é 90 segundos;

- Identificador: IP do roteador.

A mensagem OPEN pode ainda conter parâmetros opcionais utilizados na autenticação de uma conexão BGP.

As mensagens UPDATE são utilizadas para anunciar uma nova rota ou para retirar várias rotas não mais desejáveis. Uma mensagem UPDATE contém as seguintes informações:

- Lista de prefixos IP das rotas que devem ser retiradas, se houverem rotas para serem retiradas;
- *Network Layer Reachability Information* (NLRI): Indica o prefixo IP da rota sendo anunciada, se houver uma rota sendo anunciada.
- *Path Atributes:* Conjunto de atributos da rota sendo anunciada, se houver uma rota sendo anunciada.

Os *Path Atributes* são classificados em 4 categorias: *Well-Known mandatory* e *Well-Known discretionary*, que são atributos que devem ser conhecidos por todas as implementações do protocolo BGP, *Optional Transitive* e *Optional Non-Transitive*, que são atributos cujo implementação é opcional. Os atributos da classe *Well-Known mandatory* devem estar presentes em todo anúncio de rotas. Já os atributos *Well-Known discretionary* são opcionais. Os atributos *Optional Transitive* devem ser aceitos por todos os roteadores BGP, mesmo nos casos em que a implementação do protocolo BGP do roteador não reconhece tais atributos. Neste caso, o anúncio da rota que contém o atributo é marcado como parcial. Por fim, os atributos *Optional Non-Transitive* são ignorados por implementações do protocolo BGP que não o suportam.

Os atributos *Well-Known Mandatory* que devem estar presentes em um anúncio de rota são:

ORIGIN: neste atributo, o AS que originou o anúncio da rota indica se a rota foi obtida através de um protocolo de roteamento interno (origin=1), através do antigo

protocolo *Exterior Gateway Protocol* [19] (origin=2) ou através de outra maneira (INCOMPLETE, origin=3).

AS_PATH: indica quais AS's foram atravessados pelo anúncio da rota. Um roteador BGP deve acrescentar o número de seu AS ao atributo AS_PATH, antes de propagar o anúncio para um outro AS. O atributo AS_PATH é utilizado para evitar *loops* no roteamento.

NEXT_HOP: define o endereço IP do roteador que deve ser usado como destino para a rota sendo anunciada.

Os atributos *Well-Known Discretionary* que podem estar presentes em um anúncio de rota são:

LOCAL_PREF: atributo que deve estar contido em todo anúncio de rotas que um roteador BGP envia a outros roteadores BGP localizados no seu próprio AS. O valor deste atributo pode ser utilizado como o grau de preferência da rota. Caso um AS possua várias alternativas de rotas para um determinado prefixo IP, a rota com maior grau de preferência é escolhida para ser utilizada pelo processo de decisão do protocolo BGP [17].

ATOMIC_AGREGATE: indica que o roteador BGP está anunciando um prefixo agregado, isto é, um prefixo que é a união de vários prefixos menores.

O atributo *Optional Non-Transitive* que pode estar presente em um anúncio de rota é:

MULTI_EXIT_DISC: Quando um AS possui múltiplas conexões com um AS vizinho, ele pode utilizar o parâmetro MULTI_EXIT_DISC para indicar ao AS vizinho qual o caminho preferido de entrada para cada rota anunciada [17]. O valor deste atributo é um número inteiro e sem sinal de 4 octetos (0 a $2^{32} - 1$).

O atributo *Optional Transitive* que pode estar presente em um anúncio de rota é:

AGREGATOR: atributo que deve ser incluído em anúncios de prefixos agregados. O roteador BGP que realizou a agregação dos prefixos pode criar este atributo com o valor de seu número de AS e seu endereço IP.

Mensagens NOTIFICATION são enviadas quando erros são encontrados no processamento das mensagens BGP. Já mensagens KEEPALIVE são enviadas periodicamente para garantir que a conexão BGP continua ativa. Mensagens KEEPALIVE devem ser enviadas a uma taxa que garanta que o valor do *Hold Time* não expirará. O intervalo de tempo recomendado para o envio de sucessivas mensagens KEEPALIVE é um terço do valor do *Hold Time*, ou seja 30 segundos [17].

2.4.3 Processo de Decisão do Protocolo BGP

O processo de decisão do protocolo BGP é invocado por um roteador BGP quando o mesmo recebe uma mensagem UPDATE que:

- Anuncia uma nova rota com prefixo IP ainda não existente na tabela de rotas;
- Retira uma rota não mais desejável que estava em serviço;
- Substitui uma rota já existente por outra que possui o mesmo prefixo IP.

O resultado do processo de decisão são as rotas que serão anunciadas para todos os outros roteadores com os quais o roteador BGP que invocou o processo mantém conexões BGP.

O processo de decisão utiliza-se do cálculo do grau de preferência para definir qual a melhor rota para cada prefixo IP. O grau de preferência é calculado usando as informações contidas nos *Path Atributes* de cada rota. Quando uma rota é aprendida através de um roteador BGP pertencente ao mesmo AS, pode-se utilizar o valor do atributo LOCAL_PREF como grau de preferência. Além disso, pode-se utilizar políticas de roteamento pré-definidas para calcular o grau de preferência das rotas recebidas, seja através de um roteador pertencente ao mesmo AS, ou através de um roteador pertencente a um AS vizinho.

O processo de decisão possui três fases distintas: cálculo do grau de preferência, seleção de rotas e propagação de rotas, descritas a seguir.

Fase 1: Cálculo do Grau de Preferência

A fase 1 é iniciada quando um roteador BGP recebe uma mensagem UPDATE de um roteador BGP localizado em um AS vizinho.

Se a mensagem UPDATE recebida possuir uma rota sendo anunciada, o roteador deve repassar o anúncio para os outros roteadores BGP do seu AS, se uma das seguintes situações ocorrer:

- A rota anunciada é única para seu prefixo IP.
- O grau de preferência da rota sendo anunciada é maior que o grau de preferência de outras rotas com mesmo prefixo IP recebidas de roteadores BGP em AS's vizinhos.
- A rota possui o mesmo grau de preferência de outras rotas recebidas de roteadores BGP em AS's vizinhos, mas possui um valor menor do atributo MULTI_EXIT_DISC.
- A rota possui o mesmo grau de preferência e o mesmo valor do atributo MULTI_EXIT_DISC de outras rotas recebidas de roteadores BGP em AS's vizinhos, mas possui o menor custo para alcançar o roteador descrito no atributo NEXT_HOP.
- A rota possui o mesmo grau de preferência, o mesmo valor do atributo MULTI_EXIT_DISC, e o mesmo custo para alcançar o roteador descrito no atributo NEXT_HOP de outras rotas recebidas de roteadores BGP em AS's vizinhos, mas foi anunciada pelo roteador BGP que possui um endereço IP (identificador BGP) de menor valor.

Se a mensagem UPDATE recebida contiver prefixos IP de rotas a serem retiradas de serviço, o roteador deve realizar os procedimentos descritos a seguir para cada rota que está sendo retirada.

- Se a rota correspondente não foi previamente anunciada para outro roteador BGP, nenhuma ação é requerida.

- Se a rota correspondente foi anunciada anteriormente, e se o roteador possuir uma outra rota com o mesmo prefixo IP, tal rota substituta deve ser anunciada.

- Se a rota correspondente foi anunciada anteriormente, e se não existir outra rota com o mesmo prefixo IP para ser anunciada, o roteador deve enviar uma mensagem UPDATE com o prefixo IP da rota sendo retirada para os outros roteadores BGP aos quais ele tinha previamente anunciado a rota correspondente.

Ao final da fase 1, todos os roteadores do AS receberam uma cópia de todas as rotas válidas e devem coerentemente selecionar uma única rota para cada destino [7].

Fase 2: Seleção de Rotas

A fase 2 do processo de decisão é invocada ao término da fase 1. Esta fase é responsável pela escolha da melhor rota de todas as disponíveis para cada prefixo IP. O roteador deve descartar rotas que possuam atributo NEXT_HOP que seja inalcançável. Para cada prefixo IP, o roteador deve escolher a rota que:

- Possui o maior grau de preferência;

- Se ocorrer um empate, escolher a rota que possui o menor valor do atributo MULTI_EXIT_DISC.

- Se ocorrer um empate, escolher a rota que possui o menor custo para alcançar o endereço apresentado no atributo NEXT_HOP.

- Se ocorrer um empate, selecionar a rota que foi anunciada pelo roteador BGP de um AS vizinho que possui o endereço IP de menor valor.

- Se nenhuma rota foi anunciada por um roteador BGP em um AS vizinho, selecionar a rota que foi anunciada pelo roteador BGP interno que possui o endereço IP de menor valor.

Fase 3: Propagação de Rotas

A fase 3 do processo de decisão é responsável por disseminar as melhores rotas para os roteadores BGP localizados em um AS vizinho, de acordo com as políticas de roteamento

pré-configuradas. A fase 3 é invocada quando uma nova conexão BGP é estabelecida, quando as rotas geradas pelo AS são modificadas, ou ao término da fase 2. Toda melhor rota para um determinado prefixo IP deve ser anunciada para roteadores BGP localizados em AS's vizinhos. Também devem ser anunciados os prefixos IP das rotas que devem ser retiradas de serviço.

2.4.4 Controle do Tráfego de Roteamento

O protocolo BGP possui mecanismos para limitar o volume de tráfego de roteamento, isto é, o volume de mensagens do tipo UPDATE. Com isso, a banda de rede utilizada pelas mensagens do tipo UPDATE e a capacidade de processamento requerido pelo processo de decisão do protocolo BGP podem ser controladas.

O parâmetro *MinRouteAdvertisementInterval* determina a quantidade mínima de tempo que deve transcorrer entre sucessivos anúncios de rotas de um determinado prefixo IP, realizados por um roteador BGP. Este procedimento não é aplicado em anúncios recebidos de roteadores BGP do mesmo AS. Este procedimento também não é aplicado em anúncios que retiram de serviço rotas que não são mais desejadas. Os procedimentos listados acima não interferem na seleção de rotas, mas somente na taxa de anúncio de rotas. Se ocorrem variações na melhor rota para um prefixo IP enquanto o valor do *MinRouteAdvertisementInterval* não expira, somente a última rota selecionada é anunciada quando o valor do *MinRouteAdvertisementInterval* expirar. O valor proposto pela RFC 1771 [2] para o parâmetro *MinRouteAdvertisementInterval* é 30 segundos.

2.5 Análise da Latência do Protocolo BGP-4

Entende-se por latência de um protocolo de roteamento como sendo o tempo transcorrido desde uma mudança na topologia da rede, até que todos os roteadores que utilizam tal protocolo tenham atualizado suas tabelas de rotas de modo a refletir a mudança. Neste caso, é dito que o protocolo de roteamento convergiu para uma nova tabela de rotas estável, ou seja, alcançou a convergência. A seguir, uma análise da latência do protocolo

BGP-4 é apresentada.

O protocolo BGP, descrito com detalhes na seção 2.4, possui os parâmetros *Hold time* e *MinRouteAdvertisementInterval*. O parâmetro *Hold Time* é utilizado para detectar se uma conexão BGP continua ativa. Já o parâmetro *MinRouteAdvertisementInterval* é utilizado como um limitador para os anúncios de uma determinada rota enviados a um roteador BGP em um AS vizinho.

Os parâmetros *Hold Time* e *MinRouteAdvertisementInterval* influenciam no tempo de detecção de falhas e na latência do protocolo BGP. Um roteador BGP considera que uma conexão BGP não está mais ativa se o mesmo não receber desta conexão nenhuma mensagem do tipo UPDATE ou KEEPALIVE durante o período de tempo indicado pelo valor do *Hold Time*. Ou seja, se uma falha no enlace utilizado por uma conexão BGP ocorrer logo após um dos roteadores participantes da conexão ter recebido uma mensagem do tipo UPDATE ou KEEPALIVE, este roteador levará a quantidade de tempo indicada pelo *Hold Time* para diagnosticar a falha e atualizar a sua tabela de rotas. Neste intervalo de tempo, todos os pacotes que tem como destino uma das rotas em serviço que foram aprendidas pela conexão serão perdidos. O valor do parâmetro *Hold Time* sugerido pela RFC-1771 [2] é 90 segundos.

Já o parâmetro *MinRouteAdvertisementInterval* é utilizado como um limitador para os anúncios de uma determinada rota, que são enviados a um roteador BGP em um AS vizinho. Dois anúncios de uma mesma rota enviados a um roteador BGP em um AS vizinho devem ser separados pelo intervalo de tempo indicado pelo parâmetro *MinRouteAdvertisementInterval*. Os seja, se um roteador BGP anuncia uma rota para um roteador BGP em um AS vizinho, e logo após acontecer uma modificação nesta rota, o anúncio relativo à modificação será enviado somente após transcorrer o tempo indicado pelo parâmetro *MinRouteAdvertisementInterval*. O valor sugerido pela RFC-1771 para o parâmetro *MinRouteAdvertisementInterval* é 30 segundos. Apesar da RFC-1771 especificar que a contagem do *MinRouteAdvertisementInterval* deve ser feita de maneira independente para cada prefixo IP, muitos fabricantes de roteadores implementam somente um contador por conexão BGP [20]. Desta forma, o atraso no anúncio de uma rota

para um roteador BGP em um AS vizinho não depende somente de quando foi realizado o anúncio anterior desta mesma rota, mas sim de quando foi realizado o último anúncio de rotas para o roteador em questão.

Em [20], Craig Labovitz *et al.* apresentam um estudo do impacto da topologia e das políticas de roteamento na latência do protocolo BGP. O autor mostra que a latência ponto a ponto da Internet depende do comprimento do caminho mais longo entre um nodo de origem e um nodo de destino. Na média, cada AS acrescenta um atraso equivalente à metade do valor do parâmetro *MinRouteAdvertisementInterval* no anúncio de uma rota. Utilizando o valor sugerido pela RFC-1771 para o valor do *MinRouteAdvertisementInterval*, temos que, na média, a latência do protocolo BGP entre uma origem A e um destino B é igual a:

$$MinRouteAdvertisementInterval/2 * |p| = 15 * |p| segundos$$

onde $|p|$ é a distância, em número de AS's, do caminho mais longo entre A e B. Ainda temos que o valor limite para a latência do protocolo BGP entre uma origem A e um destino B é igual a:

$$MinRouteAdvertisementInterval * |p| = 30 * |p| segundos$$

onde $|p|$ é a distância, em número de AS's, do caminho mais longo entre A e B. Adicionando a este valor, a quantidade máxima de tempo necessária para um roteador detectar uma falha em uma de suas conexões BGP, temos que a latência do protocolo BGP entre uma origem A e um destino B pode ser igual a:

$$HoldTime + MinRouteAdvertisementInterval * |p| = 90 + 30 * |p| segundos$$

onde $|p|$ é a distância, em número de AS's, do caminho mais longo entre A e B.

Em [8], Craig Labovitz *et al.* apresenta que mudanças na topologia da Internet resultam em uma latência significativa até que o protocolo BGP alcance a convergência.

Medições realizadas em 5 ISP's comerciais durante o período de 2 anos demonstram que a latência média do protocolo BGP, e conseqüentemente da Internet é de 3 minutos, sendo que algumas mudanças na topologia apresentaram uma latência de 15 minutos. Este atraso na convergência do protocolo BGP causa perda de pacotes, perda de conexão e atrasos na comunicação fim a fim na Internet. Medições realizadas nos 5 ISP's indicam que, após acontecer uma falha na rede e durante o período compreendido pela latência do protocolo, a perda de pacotes na comunicação fim a fim foi 30 vezes superior e o tempo de resposta foi 4 vezes superior.

Os autores também apresentam que a latência do protocolo BGP é causada pelas modificações temporárias na tabela de rotas geradas durante o processo de seleção da melhor rota para um determinado prefixo IP. Quando uma falha ocorre, o protocolo BGP explora um grande número de rotas alternativas, sendo que muitas destas rotas também podem ser inválidas. Análises mostraram que, no pior caso teórico, em uma rede com N AS's completamente conectados, o protocolo BGP pode explorar todos os $N!$ possíveis caminhos antes de alcançar a convergência.

Os trabalhos apresentados acima analisam situações em que a convergência do protocolo BGP é garantida. Além disso, existem trabalhos que analisam a divergência do protocolo BGP [21, 22]. Em [21], Griffin e Wilfong mostram que é possível que um AS implemente certas políticas de roteamento que resultam em uma persistente oscilação de rotas [21]. Ou seja, podem existir situações em que, após uma falha, o protocolo BGP nunca alcance a convergência.

Capítulo 3

Roteamento Tolerante a Falhas Baseado em Desvios de Alta Conectividade

Um protocolo de roteamento leva uma certa quantidade de tempo para atualizar as tabelas de rotas de todos os roteadores, de modo a refletir uma mudança que tenha ocorrido na topologia da rede. Durante este período de tempo, alguns roteadores possuem informações inconsistentes na sua tabela de rotas, ocasionando perda de pacotes e perda de conexão entre as aplicações que se comunicam através da rede [8]. No capítulo 2 foram apresentados resultados de trabalhos que demonstram que o tempo médio de convergência do protocolo BGP, utilizado no roteamento da Internet, é de 3 minutos, sendo que foram observados tempos de convergência de até 15 minutos.

Este capítulo apresenta uma abordagem para roteamento tolerante a falhas, com o objetivo de permitir que os nodos da rede continuem a se comunicar, mesmo durante o período de tempo em que as tabelas de rotas dos roteadores ainda não foram atualizadas para refletir uma falha ocorrida na rede. A idéia principal consiste na utilização de *rotas alternativas* que passe por componentes de mais alta conectividade, possuindo assim grande probabilidade de desviar de uma falha ocorrida na rede. [9]. A rota alternativa é criada através de nodos da rede chamados *desvios*.

3.1 Utilização de Desvios para a Criação de Rotas Alternativas

Um *desvio* é um nodo que possui a capacidade de atuar como uma *ponte* na comunicação entre dois nodos. Desvios são utilizados na criação de uma rota alternativa de um nodo origem para um nodo destino que se comunicam através da rede. Uma *rota alternativa* de um nodo origem para um nodo destino é uma rota formada por duas rotas IP regulares: a primeira é a rota IP regular do nodo origem para o desvio, e a segunda é a rota IP regular do desvio para o nodo destino. Entende-se por rota IP regular de um nodo X para um nodo Y como sendo a rota IP que o nodo X utiliza para comunicar-se o nodo Y.

Um exemplo de uma rota alternativa do nodo origem A para o nodo destino B, criada através do desvio C, é apresentado na figura 3.1. No exemplo, a rota IP regular de A para B é dada por $A - E - B$. Já a rota alternativa de A para B é formada pela rota IP regular de A para C, no caso $A - C$, e pela rota IP regular de C para B, no caso $C - D - B$. Ou seja, a rota alternativa do nodo A para o nodo B, através do desvio C é dada por $A - C - D - B$. O conceito de desvio já foi apresentado em trabalhos que tratam da localização de rotas para mensagens SNMP no nível de aplicação [23], e da análise de performance de roteadores na Internet utilizando o conceito de tunelamento de pacotes TCP [24].

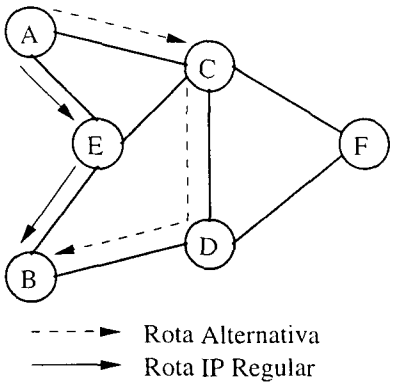


Figura 3.1: Um exemplo de rota alternativa do nodo A para o nodo B.

A rota alternativa é utilizada quando a rota IP regular de um nodo origem para um nodo destino não está funcional, ou seja, quando existe uma falha em um nodo ou enlace que faz parte da rota IP regular. Como foi apresentado anteriormente, a rota alternativa é utilizada antes do protocolo de roteamento alcançar a convergência. Ou

seja, a rota alternativa é utilizada sem a informação da localização da falha. O objetivo principal do roteamento tolerante a falhas apresentado neste trabalho é utilizar uma rota alternativa que possua grande probabilidade de desviar de uma falha ocorrida na rede, sem o conhecimento da localização da falha. Como a rota alternativa é criada através de um desvio, o problema consiste na seleção de um desvio, cuja rota alternativa possua grande probabilidade de desviar de uma falha da rede.

A seleção do desvio a ser utilizado na criação da rota alternativa é realizada levando-se em consideração critérios de conectividade apresentados em [9]. Nodos da rede pertencentes a componentes de maior conectividade possuem preferência para serem escolhidos como desvios. O número de caminhos distintos que atravessam um componente de maior conectividade é maior, se comparado a um componente de menor conectividade. Desta forma, é maior a probabilidade da rota alternativa, criada através de um desvio pertencente a um componente de maior conectividade, ser funcional mesmo na presença de falhas na rede [9].

A topologia *física* da rede *sem* a presença de falhas, modelada por um grafo não direcionado, é utilizada no cálculo dos critérios de conectividade utilizados na seleção do desvio. Além da conectividade, o comprimento da rota alternativa também é levado em consideração na seleção do desvio.

3.2 Critérios de Conectividade

Nesta seção são apresentados os critérios de conectividade propostos em [9], usados na seleção do desvio. Antes da apresentação dos critérios de conectividade, alguns conceitos preliminares da teoria dos grafos são apresentados.

3.2.1 Grafos: Conceitos Preliminares

Neste trabalho foram utilizados grafos não direcionados para modelar redes de computadores, com os vértices do grafo representando os nodos da rede e as arestas representando os enlaces existentes entre os nodos da rede. As arestas não possuem pesos, indicando que

as ligações existentes entre os nodos da rede possuem igual capacidade, sendo portanto tratadas de maneira indiferente.

Um *grafo direcionado* (ou *digrafo*) G é um par $G = (V, E)$, onde V é um conjunto finito de elementos chamados vértices e E é uma relação binária em V, chamado conjunto de arestas. É dito que uma aresta (u, v) de um digrafo G deixa o vértice u e entra no vértice v, ou é *incidente* do vértice u para o vértice v [25].

Em um *grafo não direcionado* $G = (V, E)$, o conjunto de arestas E consiste de pares não ordenados de vértices. Neste caso, uma aresta é um conjunto $\{u, v\}$ onde $u, v \in V$. Uma aresta $\{u, v\}$ é usualmente escrita como (u, v) ou (v, u) e é dito que a aresta é *incidente* em u e v [25]. Neste trabalho, um grafo direcionado é chamado de *digrafo* e um grafo não direcionado é chamado simplesmente de *grafo*. Em um grafo ou digrafo, o número de vértices é indicado por $|V|$ e o número de arestas é indicado por $|E|$.

O *grau* de um vértice v em um grafo é igual ao número de arestas incidentes a v. Em um digrafo, o *grau de saída* de um vértice v é igual ao número de arestas deixando v, e o *grau de entrada* de um vértice v é igual ao número de arestas entrando em v. O grau de um vértice em um digrafo é dado pela soma de seu grau de entrada e de seu grau de saída [26].

Um grafo $G' = (V', E')$ é um *subgrafo* de G se $V' \subseteq V$ e $E' \subseteq E$. Dado um conjunto $V' \subseteq V$, o subgrafo de G *induzido* por V' é o grafo $G' = (V', E')$, onde $E' = \{(u, v) \in E : u, v \in V'\}$. Um grafo é dito *trivial* se $|V| = 1$ [26].

Um *caminho* de *comprimento* k de um vértice u para um vértice u' em um grafo $G = (V, E)$ é uma seqüência $< v_0, v_1, ..., v_k >$ de vértices tal que $u = v_0$, $u' = v_k$, e $(v_{i-1}, v_i) \subseteq E$ para $i = 1, 2, ..., k$. O caminho contém os vértices $v_0, v_1, ..., v_k$ e as arestas $(v_0, v_1), (v_1, v_2), ..., (v_{k-1}, v_k)$. O comprimento de um caminho p, indicado por $|p|$, é igual ao número de arestas no caminho. Se existe um caminho p de u até u', é dito que u' é *alcançável* por u via p. Um *ciclo* é um caminho em que $v_0 = v_k$ e $v_0, v_1, ..., v_k$ são distintos. Um grafo *acíclico* é um grafo que não possui ciclos. Uma *árvore* é um grafo conexo acíclico [26].

Dois caminhos p e q são dito disjuntos por arestas se não possuem nenhuma aresta em

comum [26].

Um grafo G é dito *conexo* se todo vértice é alcançável por todos os outros vértices de G. Caso contrário, o grafo é dito *desconexo*. Um subgrafo conexo maximal de G é dito um *componente conexo* ou simplesmente um *componente* de G [26].

Um *corte* entre dois vértices u e v é definido como um conjunto de arestas tais que após a sua remoção, não é possível encontrar um caminho entre u e v. Um *corte mínimo* entre dois vértices $u, v \in G$, definido como $\lambda_{u,v}(G)$ é um corte entre u e v cujo número de arestas é mínimo [27]. A cardinalidade de um corte mínimo $\lambda_{u,v}(G)$ é igual ao número de arestas do corte e é definida como $|\lambda_{u,v}(G)|$.

A *conectividade de arestas* de um grafo G, definida como $\lambda(G)$, é o número mínimo de arestas cuja remoção resulta em um grafo desconexo ou trivial. Um grafo é *n-aresta conexo* se $\lambda(G) \geq n$. Analogamente, a conectividade de arestas de um subgrafo H de G, definida como $\lambda(H)$, é o número mínimo de arestas cuja remoção desconecta dois vértices $u, v \in H$. O subgrafo H é dito *n-aresta conexo* se $\lambda(H) \geq n$. Um *n-componente* de um grafo G é um subgrafo n-aresta conexo maximal [26].

A *conectividade de arestas* de um subconjunto de vértices V' de G é o número mínimo de arestas de G cuja remoção desconecta dois vértices $u, v \in V'$. Ou seja, a conectividade de arestas de um subconjunto de vértices V' de G é igual à cardinalidade do menor corte mínimo que desconecta dois vértices $u, v \in V'$.

Uma *rede de fluxo* é um digrafo $G = (V, E)$ no qual cada aresta $(u, v) \in E$ possui uma capacidade não negativa $c(u, v) \geq 0$. Se $(u, v) \notin E$, é assumido que $c(u, v) = 0$. Uma rede de fluxo possui um vértice origem s e um vértice destino t [25].

Um *fluxo* em uma rede de fluxo $G = (V, E)$ é uma função $f : V \times V \rightarrow \mathbf{R}$ que satisfaz as seguintes propriedades [25]:

Restrição de capacidade: Para todo $u, v \in V$, $f(u, v) \leqslant c(u, v)$;

Simetria: Para todo $u, v \in V$, $f(u, v) = -f(v, u)$;

Conservação de fluxo: para todo $u \in V \setminus \{s, t\}$, $\sum_{v \in V} f(u, v) = 0$.

O valor $f(u, v)$ é chamado *fluxo de rede* do vértice u para o vértice v [25].

3.2.2 Definição dos Critérios de Conectividade $\#C(v)$ e $MCC(v)$

A seguir, são definidos os critérios de conectividade $\#C(v)$ e $MCC(v)$, utilizados na seleção do desvio.

Definição 3.2.1 *O número de conectividade $\#C(v)$, $v \in G$, é definido como a conectividade de arestas de um subconjunto de vértices não unitário de G contendo v tal que a cardinalidade de qualquer corte em G que desconecta vértices deste subconjunto é maximizada [9].*

Definição 3.2.2 *Seja o conjunto $MCC(v)$ o maior subconjunto de vértices de G contendo v, tal que qualquer par de vértices deste subconjunto não pode ser desconectado por um corte em G com cardinalidade menor que $\#C(v)$ [9].*

A figura 3.2 mostra um exemplo do cálculo do $\#C(v)$ e $MCC(v)$ em um grafo G. Cada vértice v está rotulado com o seu valor de $\#C(v)$. Os conjuntos $MCC(v)$ estão identificados por círculos rotulados pelo valor $\#C(v)$ correspondente. Considere, por exemplo os vértices rotulados com o valor 4 e o subconjunto formado por tais vértices. Observa-se que a cardinalidade de qualquer corte que desconecta vértices deste subconjunto é máxima e é maior ou igual a 4. Ou seja, a conectividade de arestas de tal subconjunto de vértices é 4, e portanto o $\#C(v)$ de todos os vértices pertencentes a este subconjunto é 4. Além disso, tal subconjunto de vértices é o maior subconjunto tal que qualquer par de vértices não pode ser desconectado por um corte com cardinalidade menor que 4, sendo assim um conjunto $MCC(v)$. Considere o vértice rotulado com o valor 3 à esquerda do grafo e o subconjunto formado por tal vértice e pelos vértices rotulados com o valor 4. Observa-se que a cardinalidade de qualquer corte que desconecta vértices deste subconjunto é máxima e é maior ou igual a 3. Tal subconjunto de vértices é o maior subconjunto que não pode ser desconectado por um corte com cardinalidade menor que 3, sendo assim um conjunto $MCC(v)$.

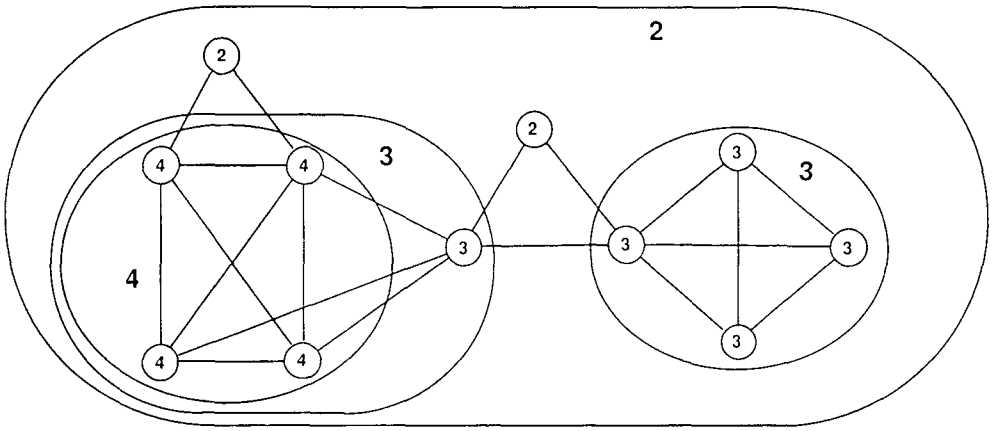


Figura 3.2: Um exemplo de números de conectividade $\#C(v)$ e conjuntos $MCC(v)$. Cada vértice v está rotulado com o seu valor $\#C(v)$. Os conjuntos $MCC(v)$ são identificados por círculos.

3.3 Cálculo dos Critérios de Conectividade Utilizando Árvores de Corte

Nesta seção são apresentados os algoritmos utilizados no cálculo dos critérios de conectividade $\#C(v)$ e $MCC(v)$.

3.3.1 Árvore de Corte e os Critérios de Conectividade $\#C(v)$ e $MCC(v)$

O seguinte lema é o ponto chave para o cálculo do critério de conectividade $\#C(v)$:

Lema 1 *Dado um grafo $G = (V, E)$ e um vértice $v \in V$, o valor do $\#C(v)$ é igual à cardinalidade máxima entre todos os cortes mínimos separando v de qualquer outro vértice de G [9].*

Prova: Considere um subconjunto de vértices não unitário V' de G contendo v e com conectividade de arestas igual a $\#C(v)$. Pela definição de conectividade de arestas de um subconjunto de vértices V' de G, $\#C(v)$ é igual à cardinalidade do menor corte mínimo que separa dois vértices de V'. Pela definição do $\#C(v)$ a cardinalidade de um corte separando v de outro vértice $u \in V'$ é maximizada. Desta forma, $\#C(v)$ é igual à cardinalidade máxima entre todos os cortes mínimos separando v de qualquer outro vértice de G. ■

O lema 1 mostra que o valor do $\#C(v)$ para $v \in V$ pode ser obtido através do cálculo do corte mínimo entre v e todos os outros vértices de V. O corte mínimo entre cada par de vértices de um grafo pode ser obtido através dos algoritmos de R.E. Gomory e T.C. HU [10] ou D. Gusfield [28]. Ambos os métodos constroem uma árvore de Gomory-Hu ou árvore de corte, definida a seguir:

Uma *árvore de corte* T de um grafo G é definida como uma árvore cujas arestas possuem pesos, e além disso:

1. Os vértices de G correspondem aos vértices da árvore T;

2. A cardinalidade de um corte mínimo entre dois vértices de G é dada pelo valor da aresta de menor peso pertencente ao caminho que liga os dois vértices correspondentes em T;

3. Um corte mínimo entre dois vértices de G é obtido removendo a aresta mencionada acima e considerando os dois conjuntos de vértices induzidos pelas duas árvores formadas pela remoção de tal aresta em T.

A figura 3.3 apresenta um exemplo da árvore de corte do grafo da figura 3.2. Através da árvore de corte, por exemplo, pode-se determinar que a cardinalidade do corte mínimo entre os vértices a e b é igual a 4 pois a aresta de menor peso no caminho que liga os vértices a e b possui peso 4. Analogamente, a cardinalidade do corte mínimo entre os vértices c e j é igual a 2 pois a aresta de menor peso no caminho que liga os vértices c e j possui peso 2.

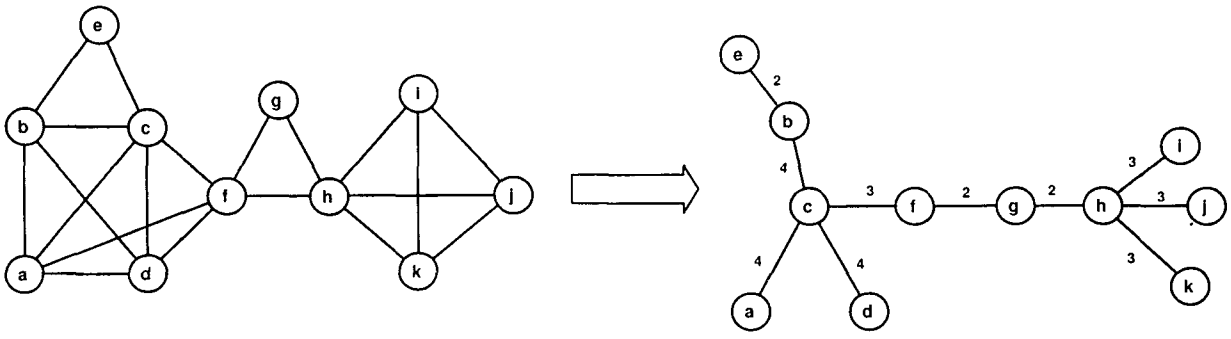


Figura 3.3: Árvore de corte do grafo da figura 3.2.

Dada uma árvore de corte T de um grafo G, o critério de conectividade $\#C(v)$ pode ser calculado utilizando o seguinte lema:

Lema 2 *Para um grafo G, uma árvore de corte T de G e um vértice v, o valor do $\#C(v)$ é igual ao valor da aresta de maior peso incidente a v em T [9].*

Prova: Pela definição, a cardinalidade de um corte mínimo separando v de outro vértice u corresponde ao valor da aresta de menor peso no caminho de v até u em T. Este valor deve ser menor ou igual ao peso de uma aresta incidente a v em T. Então a cardinalidade máxima de qualquer corte mínimo separando v de outro vértice deve ser igual ao peso de alguma aresta de T incidente a v, mais precisamente, ao peso da aresta de maior valor [9]. ■

O critério de conectividade $MCC(v)$ também podem ser calculado através da árvore de corte, utilizando o seguinte lema:

Lema 3 *Considere uma árvore de corte T de um grafo G e um vértice v. Aqueles vértices que são alcançáveis na árvore de corte a partir de v, utilizando somente arestas que possuem peso igual ou superior ao valor do $\#C(v)$, pertencem ao conjunto $MCC(v)$ [9].*

Prova: Considere um vértice $v \in G$ e uma árvore de corte T de G. Após percorrer a árvore T a partir de v e atravessando arestas que possuem peso maior ou igual ao valor de $\#C(v)$, todo vértice não alcançado pode ser separado de v por um corte de tamanho menor que o valor do $\#C(v)$. O que significa que o subconjunto de vértices construído é máximo e é o maior subconjunto de vértices que pode ser encontrado. Desta forma, tal subconjunto de vértices é o conjunto $MCC(v)$ [9]. ■

Como um exemplo do cálculo do conjunto $MCC(v)$, considere as figuras 3.3 e 3.2. Para o vértice b temos que $\#C(b) = 4$. Percorrendo a árvore de corte da figura 3.3 a partir de b, e utilizando somente arestas que possuem peso maior ou igual a 4, é possível alcançar os vértices c, a, e d. Ou seja, $MCC(b) = \{a, b, c, d\}$.

3.3.2 Algoritmos Utilizados na Geração da Árvore de Corte

O algoritmo de Gusfield, apresentado na figura 3.4, foi utilizado na geração da árvore de corte T de um grafo G. Uma *árvore orientada* é utilizada na geração da árvore de corte. Uma *árvore orientada* é um digrafo acíclico, onde o grau de saída de cada vértice é igual a 1, exceto o vértice raiz, que possui grau de saída igual a zero. No algoritmo, $p[v]$ refere-se ao vértice para o qual a aresta que sai de v está direcionada. Por exemplo, caso a árvore orientada possua a aresta (u, v), então $p[u] = v$.

```
GeraÁrvoredeCorte (Grafo G)
(1) Numerar seqüencialmente os vértices de G com os valores 1 a n;
(2) Construir uma árvore orientada T, tal que p[v] = 1 para
    v = 2, 3, ... ,n;
(3) Para i = 2 até n faça:
(4)     Seja j = p[i];
(5)     Encontrar um corte mínimo entre os vértices i e j no grafo G;
(6)     Para cada vértice x tal que p[x] = j (exceto i), faça:
           // para todos os vértices que apontam para j
(7)        Se o corte mínimo entre i e j não desconecta os vértices i e x,
           então p[x] = i;                  // vértice x aponta para i
(8)        O valor da aresta (i,j) recebe a cardinalidade do corte mínimo
           entre i e j;
(9) Retirar a orientação das arestas e retornar a árvore T.
```

Figura 3.4: Algoritmo de Gusfield utilizado na geração da árvore de corte.

A figura 3.5 apresenta um exemplo de geração da árvore de corte de um subgrafo do grafo apresentado na figura 3.2. Inicialmente todos os vértices da árvore possuem arestas orientadas para o vértice e, como pode ser observado na figura 3.5(b). Na primeira rodada do algoritmo, apresentada na figura 3.5(c), é encontrado um corte mínimo entre os vértices b e e. Os vértices a, c, d, f são alcançáveis por b após o corte. Desta forma, tais vértices passam a apontar para o vértice b. Como a cardinalidade do corte mínimo é igual a 2, a aresta (b, e) recebe peso 2. O algoritmo é executado para o restante dos vértices da árvore orientada. Ao final, o direcionamento das arestas é retirado e a árvore de corte é retornada, como apresentado na figura 3.5(h).

O corte mínimo entre dois vértices é necessário na linha (5) do algoritmo de Gusfield, descrito na figura 3.4. De acordo com o teorema de Fork-Fulkerson [29], temos que em

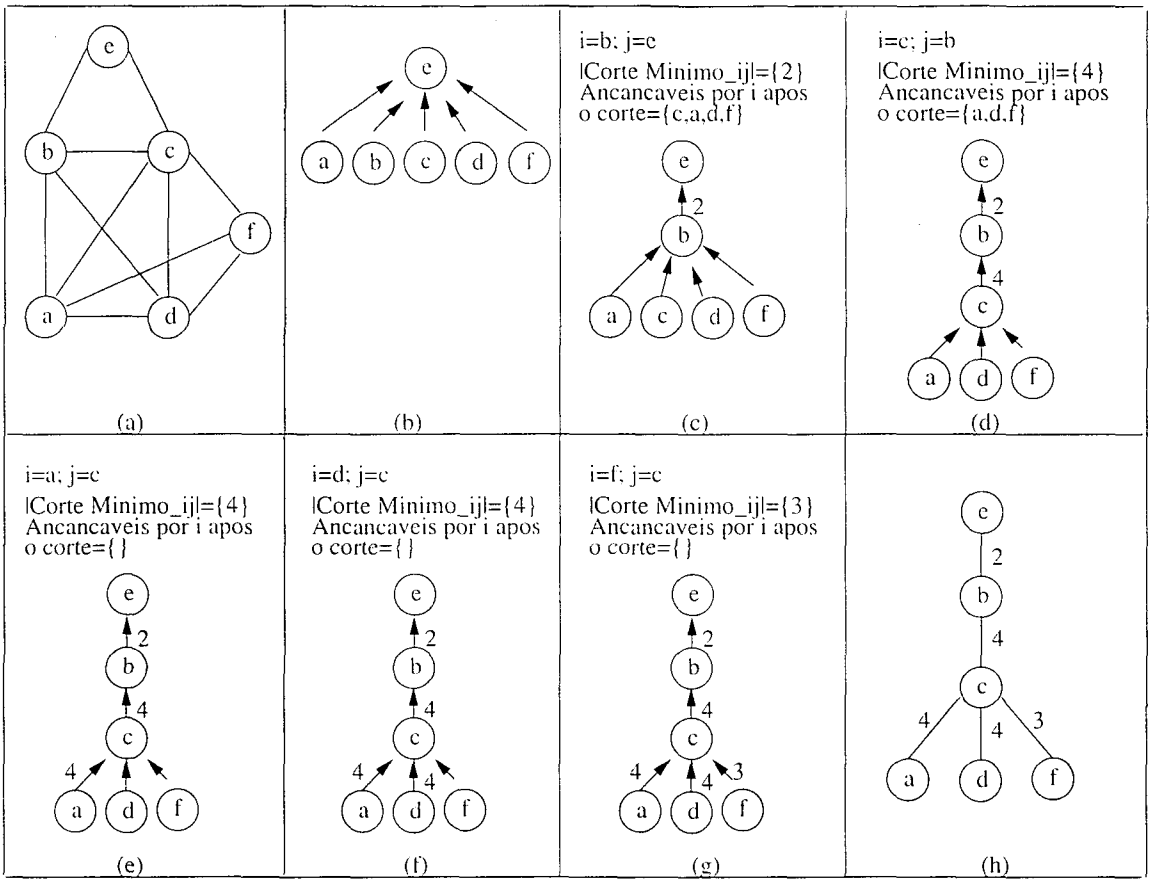


Figura 3.5: Um exemplo de geração de uma árvore de corte.

uma rede de fluxo N, o valor do fluxo máximo entre dois vértices u e v é igual à capacidade do corte mínimo entre u e v [26]. Fluxos dizem respeito a grafos direcionados. Um grafo não direcionado G pode ser transformado em um digrafo G' substituindo cada aresta (u, v) de G por duas arestas (u, v) e (v, u) em G'. Neste caso, temos que um corte entre um par de vértices u e v é mínimo em G se e somente se ele é mínimo em G' [30]. Como as arestas de G' não possuem pesos, G' pode ser visto como uma rede de fluxo em que $c(u, v) = 1$, para toda aresta $(u, v) \in G'$. Neste caso, a capacidade do corte mínimo entre um par de vértices $u, v \in G'$ é igual ao número de arestas do corte.

O algoritmo descrito na figura 3.6 [31] foi utilizado para obter o corte mínimo entre dois vértices s e t de um grafo G. Inicialmente o grafo G é transformado em um digrafo G', de acordo com o que foi apresentado no parágrafo anterior. O algoritmo é uma simplificação do algoritmo de Ford-Fulkerson descrito em [25]. Ao invés de trabalhar explicitamente com fluxos e capacidades residuais, o algoritmo opta pela remoção ou inclusão de arestas. A cada iteração, o algoritmo tenta encontrar, utilizando busca em profundidade a partir

de s, um caminho p entre os vértices s e t. Para cada aresta $(u, v) \in p$, o algoritmo remove a aresta (u, v) caso exista a aresta (v, u). Ou seja, caso não exista fluxo do vértice v para o vértice u para ser cancelado, então é criado um fluxo do vértice u para o vértice v. Como $c(u, v) = 1$, a aresta (u, v) não pode mais ser utilizada em um caminho entre s e t e por isso é removida. No entanto, se a aresta (v, u) não existir, isto indica que já existe um fluxo do vértice v para o vértice u e que tal fluxo pode ser cancelado pelo fluxo a ser criado do vértice u para o vértice v. Desta forma, a aresta (v, u) é incluída novamente no grafo.

A cardinalidade do corte mínimo entre s e t é incrementada a cada iteração do algoritmo, ou seja, a cada novo caminho encontrado entre os vértices s e t. O algoritmo retorna a cardinalidade do corte mínimo e os vértices visitados na última busca em profundidade partindo de s. Tais vértices correspondem aos vértices que são alcançáveis a partir de s, após a remoção das arestas do corte mínimo encontrado.

```
CorteMínimo (grafo G, vértice s, vértice t)
(1) Gerar um digrafo G' a partir do grafo G;
(2) CorteMínimo <- 0;
(3) Encontrar, utilizando busca em profundidade, um caminho p entre s e t
    em G'. Se existir tal caminho faça:
(4)     CorteMínimo <- CorteMínimo + 1;
(5)     Para cada aresta (u,v) no caminho p faça:
(6)                     Se existir a aresta (v,u), remova do grafo a
                        aresta (u,v);
(7)                     Senão, crie a aresta (v,u);
(8)     Retornar ao passo 3
(9) Retornar CorteMínimo e a lista dos vértices visitados na última
    busca em profundidade.
```

Figura 3.6: Algoritmo para cálculo do corte mínimo entre dois vértices de um grafo.

A ordem de complexidade da busca em profundidade utilizada na linha (3) do algoritmo do corte mínimo entre dois vértices (figura 3.6) é $\mathcal{O}(E)$, onde E é o número de arestas do grafo [25]. Ou seja, cada iteração do algoritmo possui complexidade $\mathcal{O}(E)$. A cada iteração, a cardinalidade do corte mínimo entre os vértices s e t é incrementada em uma unidade. Ou seja, a ordem de complexidade do algoritmo é dada por $\mathcal{O}(E \cdot |\lambda_{u,v}|)$, onde $|\lambda_{u,v}|$ é a cardinalidade do corte mínimo entre s e t. No entanto, temos

que $|\lambda_{u,v}| \leq min\{grau(s), grau(t)\} \leq V - 1$ [26], onde V é o número de vértices do grafo. Desta forma, a ordem de complexidade do algoritmo é dada por $\mathcal{O}(V \cdot E)$. O algoritmo de Ford-Fulkerson foi escolhido para calcular o corte mínimo entre dois vértices por ser de simples implementação, e apresentar um bom tempo de resposta para grafos em que a capacidade das arestas é dada por um valor inteiro e a cardinalidade do corte mínimo é pequena [25].

O algoritmo de Gusfield (figura 3.4), utilizado na construção da árvore de corte, executa o algoritmo do corte mínimo $V - 1$ vezes. Desta forma, a ordem de complexidade do algoritmo de construção da árvore de corte é dada por $\mathcal{O}(V^2 \cdot E)$.

De acordo com o lema 2, temos que, utilizando uma árvore de corte T de um grafo G, o $\#C(v)$ para $v \in G$ pode ser obtido em tempo linear analisando as arestas incidentes a v em T. Além disso, o conjunto $MCC(v)$, de acordo com o lema 3, também pode ser obtido em tempo linear, através de uma busca em profundidade em T partindo de v. Ou seja, os cálculos do $\#C(v)$ e do $MCC(v)$ dependem da construção da árvore de corte, que possui ordem de complexidade igual a $\mathcal{O}(V^2 \cdot E)$. Para evitar a ordem de complexidade da construção da árvore de corte, na próxima seção é apresentada uma heurística para o cálculo do $\#C(v)$. O algoritmo de cálculo do $\#C(v)$ e do $MCC(v)$ utilizando a árvore de corte será chamado de *algoritmo exato*. Já o algoritmo de cálculo do $\#C(v)$ utilizando a heurística será chamado de *algoritmo heurístico*.

3.4 Uma Heurística para o Cálculo do $\#C(v)$

A heurística para o cálculo do $\#C(v)$, proposta em [9], é baseada em um algoritmo descrito em [32], utilizado em outro problema relacionado à conectividade em grafos. Em cada passo, a heurística procura encontrar um conjunto de componentes 2-aresta conexos. Os vértices pertencentes a tais componentes têm a estimativa de seu $\#C(v)$ incrementada em duas unidades. Já a estimativa do $\#C(v)$ dos vértices não pertencentes a componentes 2-aresta conexos é incrementada em uma unidade.

O algoritmo da heurística utiliza uma árvore de busca em profundidade (*DFT - Depth First Tree*) de um grafo G e um conjunto de arestas chamadas arestas de cobertura (*cover*

edges), que são arestas de retorno, para encontrar o conjunto de componentes 2-aresta conexos. Uma *aresta de cobertura* é uma aresta $(v_k, v_0) \in G$ e $(v_k, v_0) \notin DFT$, tal que $< v_0, v_1, ..., v_k, v_0 >$ é um ciclo em G e $(v_{i-1}, vi) \in DFT$, $i = 1, ..., k$. Neste caso, é dito que a aresta de cobertura (v_k, v_0) *cobre* as arestas (v_{i-1}, vi), $i = 1, ..., k$.

As arestas de cobertura são encontradas utilizando a seguinte heurística: para cada aresta não coberta (u, v), encontre a aresta de cobertura (x, y) que *cobre* a aresta (u, v) e que retorne o mais próximo possível da raiz da árvore. Desta forma, a aresta de cobertura tende a cobrir uma maior parte da árvore. Após a tentativa de cobertura de todas as arestas da DFT, o valor estimado $\#C(v)$ é incrementado da seguinte maneira. Vértices v pertencentes a um ciclo no grafo formado pela $DFT + \{arestas\ de\ cobertura\}$ fazem parte de um componente 2-aresta conexo e têm o valor estimado de seu $\#C(v)$ incrementado de 2 unidades. Já vértices v que não pertencem a um ciclo no grafo formado pela $DFT + \{arestas\ de\ cobertura\}$ têm o valor estimado de seu $\#C(v)$ incrementado de 1 unidade. Após o incremento das estimativas do $\#C(v)$, as arestas da DFT e as arestas de cobertura são removidas de G e novas DFT's são criadas. Quando o grau de um vértice atinge zero, o vértice também é removido de G. O algoritmo termina quando todas as arestas são removidas de G. O algoritmo utilizado no cálculo heurístico do $\#C(v)$ é apresentado na figura 3.7.

A árvore DFT utilizada na linha (7) do algoritmo é construída através de uma busca em profundidade. Desta forma, a ordem de complexidade da construção da DFT é $\mathcal{O}(E')$, onde E' é o número de arestas do componente H. As arestas de cobertura podem ser encontradas em tempo proporcional à construção da DFT, utilizando uma implementação recursiva. Já o laço da linha (10) é executado para cada vértice v do componente H, possuindo ordem de complexidade de $\mathcal{O}(V')$, onde V' é o número de vértices do componente H. Como o laço da linha (6) é executado para todo componente H de G, temos que a ordem de complexidade de tal laço é $\mathcal{O}(V + E)$, onde V é o número de vértices de G e E é o número de arestas de G. A cada iteração do laço da linha (5) são removidas aproximadamente V arestas. Ou seja, o laço da linha (5) é executado aproximadamente (E/V), vezes no pior caso. Desta forma, a ordem de complexidade do algoritmo completo, no pior

```
    CalculoHeuristico#C(v)(Grafo G)
(1)    Para cada vértice v em G faça:
(2)       #C[v] = 0;
(3)       Se grau(v) = 0 então:
(4)          Remova v de G;
(5)    Enquanto o conjunto de arestas não for vazio faça:
(6)       Para cada componente conexo H de G faça:
(7)          T = uma DFT de H;
(8)          Para cada aresta não coberta 'e' de T faça:
(9)             Encontre uma aresta de cobertura de 'e', se existir;
(10)         Para cada vértice v de H faça:
(11)            Se v pertence a um ciclo em (T + arestas de cobertura)
(12)            então #C[v] = #C[v] + 2;
(13)            senão #C[v] = #C[v] + 1;
(14)         Remova as arestas de T e as arestas de cobertura de G;
(15)         Remova os vértices v tais que grau(v)=0;
    Retorne #C();
```

Figura 3.7: Algoritmo utilizado no cálculo heurístico do $\#C(v)$.

caso, é aproximadamente $\mathcal{O}((E/V) \cdot (V + E))$. Porém, grafos esparsos são mais próximos das topologias utilizadas na prática. Portanto, a complexidade esperada do algoritmo é linear e é dada por $\mathcal{O}(V + E)$, pois $\mathcal{O}(E/V) = \mathcal{O}(1)$ [9].

Esta heurística não calcula um valor estimado para o critério de conectividade $MCC(v)$. Porém o algoritmo de seleção do melhor desvio, apresentado na próxima seção, utiliza o critério de conectividade $MCC(v)$ apenas como um critério de desempate. Desta forma, o cálculo do critério de conectividade $MCC(v)$ pode ser suprimido. A heurística apresenta a vantagem de realizar o cálculo estimado do $\#C(v)$ de todo $v \in G$ em tempo linear.

Um exemplo do cálculo do número de conectividade estimado $\widetilde{\#}C(v)$ dos vértices do grafo da figura 3.2, utilizando a heurística proposta, é apresentado na figura 3.8. Os vértices da figura estão rotulados com o valor de seu $\widetilde{\#}C(v)$. Inicialmente, todos os vértices do grafo possuem $\widetilde{\#}C(v) = 0$, como é apresentado na figura 3.8(a). O vértice no canto inferior esquerdo da figura 3.8(a) é a raiz da primeira DFT. As arestas pertencentes à DFT são representadas por arestas direcionadas, indicando o caminho percorrido pelo algoritmo de construção da DFT. As arestas de coberturas são representadas por arestas tracejadas e direcionadas, indicando arestas de retorno. Ao final da primeira iteração do algoritmo, todos os vértices da figura 3.8(a) pertencem a um ciclo em $DFT + \{arestas\ de\ cobertura\}$

e portanto têm seu $\widetilde{\#}C(v)$ incrementado em duas unidades. As arestas da DFT e as arestas de cobertura são então removidas do grafo da figura 3.8(a), resultando no grafo da figura 3.8(b). Os vértices que possuem grau zero também são removidos do grafo, mas serão mantidos na figura 3.8(b) representados por círculos tracejados. Novas $DTF's$ e arestas de cobertura são encontradas nos componentes da figura 3.8(b). Todo vértice da figura 3.8(b) pertencente a um ciclo em $DFT + \{arestas\ de\ cobertura\}$ tem seu $\widetilde{\#}C(v)$ incrementado em duas unidades. Caso contrário, o vértice tem seu $\widetilde{\#}C(v)$ incrementado em uma unidade. Todas as arestas da figura 3.8(b) são removidas e o algoritmo termina. A figura 3.8(c) apresenta os vértices do grafo rotulados com o valor de seu $\widetilde{\#}C(v)$ retornado pela heurística.

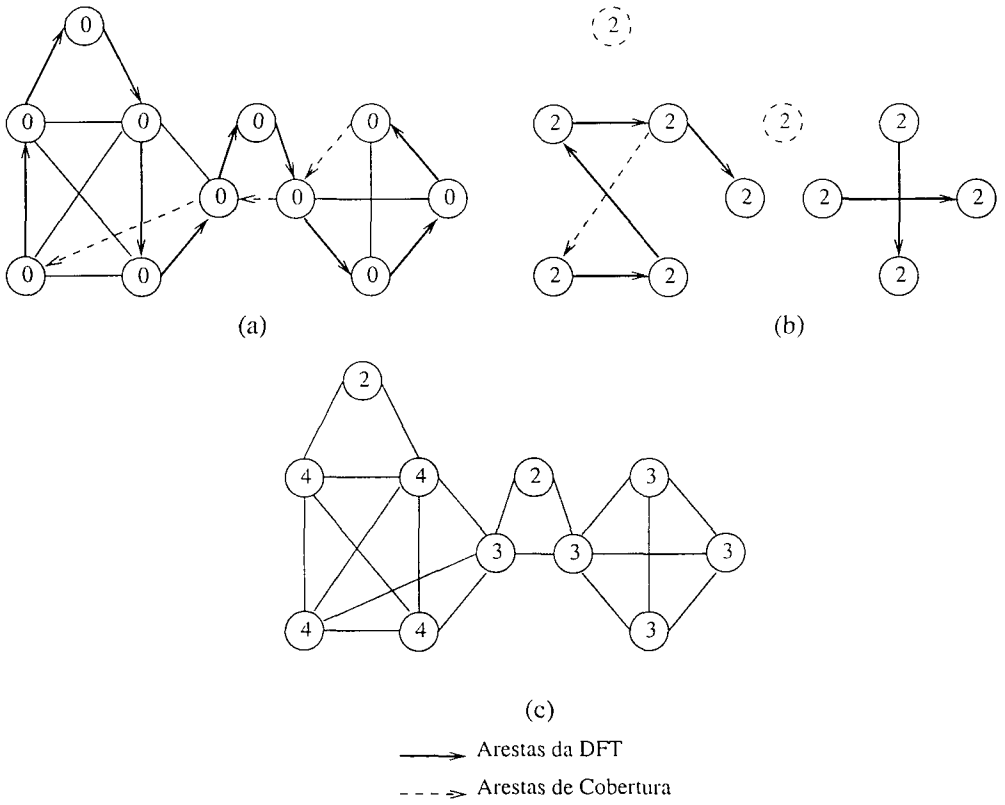


Figura 3.8: Um exemplo do cálculo do número de conectividade estimado $\widetilde{\#}C(v)$ utilizando a heurística.

O lema 4 mostra que o número de conectividade aproximado $\widetilde{\#}C(v)$, é sempre menor ou igual ao número de conectividade exato $\#C(v)$.

Lema 4 *Para cada vértice $v \in G$, o $\widetilde{\#}C(v)$ estimado retornado pela heurística é menor ou igual ao $\#C(v)$ exato [9].*

Prova: Considere um par de vértices u e v tal que os vértices u e v foram desconectados na última iteração da heurística e $grau(v) = 0$. Quando as arestas removidas em uma iteração da heurística formam dois caminhos disjuntos por arestas entre u e v, o $\widetilde{\#}C(v)$ é incrementado em 2 unidades. Da mesma forma, quando as arestas removidas em uma iteração da heurística formam somente um caminho entre u e v, o $\widetilde{\#}C(v)$ é incrementado em uma unidade. Portanto, o conjunto de arestas removidas até a última iteração contém no mínimo $\widetilde{\#}C(v)$ caminhos disjuntos por arestas entre u e v. No entanto temos que o número máximo de caminhos disjuntos por arestas entre dois vértices x e y é igual à cardinalidade do corte mínimo que separa os vértices x e y [33]. Como $\#C(v)$é igual à cardinalidade máxima entre todos os cortes mínimos separando v de qualquer outro vértice de G temos que $\widetilde{\#}C(v) \leq |\lambda_{u,v}(G)| \leq \#C(v)$. ■

3.5 Algoritmo para a Seleção do Melhor Desvio

Esta seção apresenta o algoritmo utilizado na seleção do melhor desvio para um par de vértices a e b, de acordo com os critérios de conectividade $\#C(v)$ e $MCC(v)$. As seguintes definições são utilizadas no algoritmo de seleção do melhor desvio:

$new - length(v, a, b)$: indica o comprimento da rota alternativa do vértice a para o vértice b, usando o vértice v como desvio.

$d - neighborhood(a, b)$: indica o conjunto de vértices v, tal que a rota alternativa do vértice a para o vértice b, usando o vértice v como desvio possui comprimento menor ou igual a d. Ou seja, $d - neighborhood(a, b) = \{v \in V, v \neq a, b | new - length(v, a, b) \leq d\}$.

O algoritmo procura por desvios dentro de uma vizinhança restrita. O tamanho da vizinhança é incrementado a cada iteração do algoritmo. O objetivo desta limitação é evitar que vértices pertencentes a componente altamente conexos, mas muito distantes dos vértices a e b sejam escolhidos como desvios. O incremento do tamanho da vizinhança é proporcional ao comprimento da rota regular existente entre os vértices a e b.

O número de conectividade $\#C(v)$ é utilizado na seleção do melhor desvio. Empates são resolvidos utilizando o tamanho do conjunto $MCC(v)$, caso o algoritmo exato esteja sendo utilizado para cálculo do $\#C(v)$ e $MCC(v)$, e utilizando o comprimento da rota alternativa entre a e b criada através do desvio. O algoritmo retorna uma lista ordenada por preferência dos melhores desvios para um par de vértices a e b. A lista contém todos os vértices do grafo.

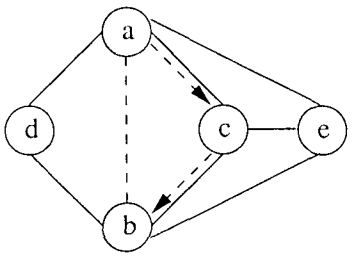
```
    EncontreMelhorDesvio(Grafo G=(V,E), vértice a, vértice b)
(1)     L <- Lista Vazia;
(2)     Para cada vértice v de V faça:
(3)        Encontre o número #C(v), e o tamanho do componente MCC(v);
(4)        Encontre new-length(v,a,b);
(5)     P <- rota IP regular usada pelo vértice a para comunicar-se o
        vértice b;
(6)     d <- |P| * 2;
(7)     Enquanto |L| <= |V|-2 faça
(8)        Ordene os vértices v pertencentes à V-L e à
           d-neighborhood(a,b) em ordem não-crescente usando a chave
           <#C(v),|MCC(v)|,-new-length(v,a,b)>;
(9)        Insira os vértices ordenados no final de L;
(10)       d = d + |P|; // a vizinhança é incrementada para alcançar
                        // mais vértices.
(11)    Retorne L;
```

Figura 3.9: Algoritmo utilizado na seleção do melhor desvio.

O algoritmo utilizado na seleção do melhor desvio entre um par de vértices a e b de um grafo $G = (V, E)$ é apresentado na figura 3.9. Inicialmente na linha (1) é criada uma lista vazia L que irá conter todos os vértices de G ordenados pelos critérios de conectividade propostos. Na linha (3), os valores $\#C(v)$ e $MCC(v)$ são calculados para cada vértice $v \in V$. Caso seja utilizado o algoritmo exato, o $\#C(v)$ e o $MCC(v)$ podem ser calculados. Caso seja utilizada a heurística, somente o $\#C(v)$ pode ser calculado. Na linha (4) é realizado o cálculo do comprimento das rotas alternativas do vértice a para o vértice b, utilizando cada vértice de G como desvio. Na linha (5), p é a rota IP regular entre os vértices a e b e na linha (6), a variável d é inicializada com o dobro do comprimento da rota p. A variável d será utilizada para definir uma vizinhança incremental. O laço da linha (7) irá executar até que todos os vértices do grafo sejam ordenados. Na linha (8),

os vértices pertencentes à $d - neighborhood(a, b)$ que ainda não foram inseridos em L são ordenados inicialmente considerando o seu número de conectividade $\#C(v)$, depois o tamanho de seu conjunto $MCC(v)$, caso o $MCC(v)$ tenha sido calculado, e finalmente o comprimento da rota alternativa. Tais vértices são inseridos no final da lista L na linha (9). Na linha 10, a vizinhança é incrementada para alcançar mais possíveis desvios.

A figura 3.10 apresenta um exemplo da seleção do melhor desvio para os vértices a e b. Os valores do $\#C(v)$ e $MCC(v)$ de cada vértice do grafo são apresentados na figura. O comprimento da rota alternativa do vértice a para o vértice b, usando o vértice c, d ou e como desvio, é igual a 2. Como o comprimento da rota IP regular de a para b é igual a 1, a variável d é igual a 2 e $2 - neighborhood(a, b) = \{c, d, e\}$. Ordenando os vértices $\{c, d, e\}$ pelo $\#C(v)$, $MCC(v)$ e depois pelo comprimento da rota alternativa, temos que o melhor desvio é o vértice c, que possui $\#C(v)$=3.



#C(a)=4 MCC(a)=2 new−length(c,a,b)=2 Vertices Ordenados=c,e,d
#C(b)=4 MCC(b)=2 new−length(d,a,b)=2 Desvio escolhido=c
#C(c)=3 MCC(c)=4 new−length(e,a,b)=2
#C(d)=2 MCC(d)=5 d=2*1=2
#C(e)=3 MCC(e)=4 2−neighborhood(a,b)={c,d,e}

Figura 3.10: Um exemplo de seleção do melhor desvio.

O valor *new-length(v, a, b)* para cada $v \in V$ pode ser calculado executando duas vezes o algoritmo de Dijkstra, uma vez tendo o vértice a como origem e outra vez tendo o vértice b como origem. O algoritmo de Dijkstra possui ordem de complexidade de $\mathcal{O}(E \cdot \log(V))$, assumindo que o grafo é esparso [33].

A ordem de complexidade do algoritmo completo depende de como os valores $\#C(v)$ e $MCC(v)$ estão sendo calculados. Na seção 3.3.2, foi apresentado que a ordem de complexidade do algoritmo exato para cálculo do $\#C(v)$ e $MCC(v)$ depende da construção da árvore de corte, que possui ordem complexidade de $\mathcal{O}(V^2 \cdot E)$. Desta forma, a ordem

de complexidade do algoritmo completo é dada por $\mathcal{O}(V^2 \cdot E) + \mathcal{O}(E \cdot \log(V))$. Já a ordem de complexidade do algoritmo heurístico para cálculo do $\#C(v)$ é de $\mathcal{O}(V + E)$, assumindo que a rede é esparsa. Desta forma, a ordem de complexidade do algoritmo completo quando a heurística é utilizada é de $\mathcal{O}(V + E) + \mathcal{O}(E \cdot \log(V))$, assumindo que a rede é esparsa.

Capítulo 4

Utilização de Grafos para a Representação de Topologias de Redes

Grafos são comumente utilizados na modelagem de topologias de redes de computadores. Além disso, grafos gerados de maneira aleatória são utilizados para modelar redes de topologia arbitrária, como a Internet. Entretanto, é tarefa difícil gerar grafos que possuam características topológicas idênticas às redes que estão sendo estudadas. Topologias regulares como estrelas, árvores e anéis, bem como topologias conhecidas como os *backbones* ARPAnet e RNP2 [34] podem refletir somente o passado ou apenas parte da topologia atual das redes em estudo. Já topologias geradas aleatoriamente podem não refletir nem o passado, presente ou futuro de tais redes [11].

Existem vários trabalhos relacionados à geração de grafos que possuam características topológicas idênticas às existentes na interconexão das redes reais [11, 14, 12, 13, 35]. Estes trabalhos descrevem e comparam vários métodos de geração de grafos aleatórios.

4.1 Métodos de Geração de Grafos Aleatórios

Os métodos de geração de grafos aleatórios podem ser classificados em três classes: *flat random*, regular e hierárquica [11], descritas a seguir.

Os métodos pertencentes à classe *flat random* constroem grafos adicionando probabilisticamente arestas a um dado conjunto de vértices. O conjunto de vértices é distribuído em um plano, e uma aresta é adicionada a cada par de vértices de acordo com uma função de probabilidade. Fazem parte desta classe os seguintes métodos:

Método *random* puro: neste método, a probabilidade de existência de uma aresta entre dois vértices é dada por um valor fixo p. Tal método é comumente utilizado por causa de sua simplicidade.

Método Waxman [35]: neste método, a probabilidade de existir uma aresta entre dois vértices varia de acordo com a distância existente entre tais vértices, tentando assim capturar a característica de localidade existente nas redes reais. A probabilidade de existir uma aresta entre os vértices u e v é dada pela seguinte função de probabilidade:

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

onde $0 < \alpha, \beta \leq 1$, d é a distância Euclideana [36] entre os vértices u e v (distância existente entre os vértices u e v no plano Euclideano), e L é a distância Euclideana máxima existente entre qualquer par de vértices. Um aumento em α aumenta o número de arestas no grafo, enquanto um aumento em β aumenta a proporção de arestas longas sobre arestas curtas.

Método exponencial [11]: este método também pretende capturar a característica de localidade existente nas redes reais relacionando a probabilidade de existência de uma aresta entre dois vértices com a distância existente entre os vértices. A probabilidade de existir uma aresta entre os vértices u e v é dada pela função:

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

onde $0 < \alpha$, d é a distância Euclideana [36] entre os vértices u e v, e L é a distância Euclideana máxima existente entre qualquer par de vértices. A probabilidade de existência de uma aresta entre dois vértices aproxima-se de zero quando a distância

entre tais vértices aproxima-se de L.

Método da localidade [11]: este método divide as potenciais arestas em grupos baseados na distância Euclideana e atribui uma probabilidade fixa de existência de arestas a cada grupo. Por exemplo, caso existam dois grupos, o parâmetro r determina o limite dos grupos da seguinte maneira:

$$P(u,v) = \begin{cases} a & \text{se } d < r \\ b & \text{se } d \geq r \end{cases}$$

Em todos os métodos pertencentes à classe *flat random*, a probabilidade de geração de um grafo conexo diminui com o aumento do número de vértices do grafo. Tal característica dificulta e torna pouco eficiente o processo de geração de grandes grafos conexos que possuam uma baixa quantidade de arestas. Caso o método não gere um grafo conexo, pode-se aplicar técnicas de reparo para torná-lo conexo. Porém, os grafos reparados não serão inteiramente aleatórios, pois sofreram a influência de um processo de reparação.

Os métodos pertencentes à classe regular geram grafos que possuem topologias conhecidas como estrelas, anéis e hipercubos, não possuindo nenhuma característica aleatória.

Já os métodos pertencentes à classe hierárquica constroem grandes topologias através de componentes topológicos menores, capturando assim a hierarquia presente na interconexão de redes de computadores [11]. Fazem parte desta classe os seguintes métodos:

Método *N-Level* [11]: constrói grafos substituindo vértices individuais de um grafo inicial por novos grafos. Iniciando com um grafo conexo, cada vértice do grafo é substituído por um novo grafo conexo. O grafo inicial e os grafos intermediários são gerados por algum método pertencente à classe *flat random*.

Método *Transit-Stub* [11]: produz grafos hierárquicos interconectando grafos intermediários que representam AS's de trânsito e AS's *stub*. Tal método tenta capturar com maior exatidão a estrutura hierárquica existente na Internet.

No capítulo 2, foi visto que a Internet é formada por um conjunto de entidades chamadas AS's, que são classificados como AS's de trânsito e AS's *stub*. No método *Transit-Stub*,

inicialmente um grafo conexo é gerado através de um método pertencente à classe *flat random*, sendo que cada vértice do grafo representa um AS de trânsito. Cada vértice é então substituído por outro grafo conexo representando o *backbone* de um AS de trânsito. Em cada vértice pertencente a um AS de trânsito são conectados novos grafos conexos que são os AS's *stub*. Por fim, conectividades extras são adicionadas na forma de arestas entre pares de vértices, um pertencente a um AS de trânsito e um pertencente a um AS *stub*, ou ambos pertencentes a AS's *stub* diferentes. O número de vértices do grafo, a distribuição dos vértices entre AS's de trânsito e AS's *stub* e o número de arestas adicionais são controlados por parâmetros do método.

O gerador de topologias GT-ITM (*Georgia Tech Internet Topology Models*) [37] implementa os métodos de geração de topologias das classes *flat random* e hierárquica.

4.2 Análise dos métodos de geração de grafos aleatórios

Em [11], E. Zegura *et al.* estudaram o problema da geração de grafos aleatórios que possuam propriedades topológicas semelhantes às existentes nas redes reais, realizando uma comparação quantitativa dos métodos de geração de grafos apresentados anteriormente.

Para realizar a comparação dos diversos métodos de geração de grafos aleatórios, os autores utilizaram um conjunto de métricas baseadas em propriedades topológicas de grafos. Para um grafo com n vértices e m arestas, as seguintes propriedades topológicas foram utilizadas como métricas:

Grau médio dos vértices: $(2m/n)$.

Diâmetro: O diâmetro de um grafo é o comprimento do caminho mínimo mais longo entre dois vértices quaisquer [36]. Foram consideradas duas variações da métrica diâmetro: diâmetro de *hop* (*hop diameter*) e diâmetro de comprimento (*length diameter*). O diâmetro de *hop* é igual ao comprimento do caminho mínimo mais longo entre dois vértices quaisquer, onde o caminho mínimo é calculado utilizando o número de *hops* como métrica. O diâmetro de comprimento é igual ao comprimento do caminho mínimo mais longo entre dois vértices quaisquer, onde o caminho

mínimo é calculado utilizando a distância Euclideana como métrica.

Número de componentes biconexos: O número de componentes biconexos é uma medida do grau de conectividade ou de redundância das arestas em um grafo. Geralmente, uma menor quantidade de componentes biconexos corresponde a um maior número de caminhos entre vértices no grafo.

A metodologia para a comparação dos diversos métodos consistiu em fixar o número de vértices e o número médio de arestas dos grafos que foram gerados. Foi investigado como os valores das métricas variam de acordo com a escolha dos parâmetros dos métodos. Após esta análise, definiu-se quais parâmetros seriam utilizados em cada método. Para cada método, foram gerados 100 grafos conexos contendo 100 vértices.

As seguintes conclusões foram obtidas após a comparação dos grafos gerados pelos diversos métodos:

- O método *random* puro produziu grafos com diâmetro de comprimento bem superior aos outros métodos, pois não leva em consideração a distância Euclideana no cálculo da probabilidade de existência de uma aresta entre dois vértices. Ou seja, o método *random* puro tende a gerar arestas mais longas, não capturando a característica de localidade existente na interconexão das redes de computadores. Após o método *random* puro, o método da classe *flat random* que gerou grafos com maior diâmetro de comprimento foi o método exponencial, seguido pelo método da localidade e pelo método Waxman.

- Os métodos hierárquicos diferem significativamente dos métodos da classe *flat random* no número de componentes biconexos, sendo que os grafos gerados por métodos hierárquicos apresentaram quase duas vezes mais componentes biconexos que os grafos gerados por métodos da classe *flat random*. Os grafos hierárquicos também tendem a possuir um diâmetro de *hop* maior e um diâmetro de comprimento menor. Tal característica deve-se à estrutura hierárquica utilizada pelos métodos, onde existe a tendência das arestas serem curtas, resultando em longos caminhos baseados no número de *hops* e em curtos caminhos baseados na distância Euclideana.

- Entre os métodos hierárquicos, o *Transit-Stub* tende a gerar grafos com um maior número de componentes biconexos e com um maior diâmetro de comprimento que o método *N-Level*. Neste método, os AS's de trânsito são utilizados para separar os AS's *stub* levando a um maior número de componentes biconexos e a maiores caminhos baseados na distância Euclideana.

- Os métodos hierárquicos são capazes de gerar eficientemente grandes grafos conexos que possuem um grau médio dos vértices baixo, ao contrário dos métodos da classe *flat random*. Foi identificado que não é prático gerar grafos conexos através de métodos da classe *flat random* que possuam tamanho moderado (n = 1500) e um grau médio dos vértices realístico (menor que 6), sem a necessidade de utilização de métodos de reparação.

4.3 A Topologia da Internet e as Leis de Potência

Em [14], M. Faloutsos *et al.* apresentaram leis de potência para várias propriedades topológicas da Internet. Tais leis foram observadas em três mapas estáticos da Internet em nível de Sistemas Autônomos (cada vértice representa um AS) datados de novembro de 1997, abril de 1998 e dezembro de 1998, período em que a Internet apresentou um crescimento de 45%.

As leis de potências representam de forma concisa várias propriedades topológicas da Internet como a distribuição dos graus dos vértices, o número total de arestas, a conectividade e distância entre os vértices e o diâmetro efetivo. Tais leis ajudam a entender melhor as características da topologia da Internet, possibilitando o desenvolvimento de protocolos de roteamento mais eficientes, a criação de topologias artificiais mais precisas para serem utilizadas em simulações e a realização de estimativas sobre a topologia da Internet no futuro.

Em [12], C. Jin *et al.* apresentam um gerador de topologias que é baseado na conectividade dos AS's da Internet chamado INET (*Internet Topology Generator*) [38]. O INET se propõe a gerar topologia que possuem características idênticas às observadas na Inter-

net. Utilizando as leis de potências da Internet descritas em [14], os autores compararam os grafos produzidos pelo INET com grafos produzidos por outros geradores de topologia como o GT-ITM [37]. Em [13], J. Winick *et al.* apresentam a versão 3 do gerador de topologias INET (INET-3.0) [38], com uma série de melhorias. Os autores realizaram comparações dos grafos gerados pelo INET-3.0 com a versão anterior do INET e comprovam que as topologias geradas pelo INET-3.0 possuem características topológicas mais próximas às características topológicas da Internet.

Capítulo 5

Resultados Experimentais

Este capítulo apresenta os resultados experimentais obtidos pela implementação da estratégia de seleção de desvios descrita no capítulo 3. Os algoritmos que calculam os critérios de conectividade $\#C(v)$ e $MCC(v)$ através da construção da árvore de corte, o algoritmo utilizado no cálculo heurístico do $\#C(v)$ e o algoritmo de seleção do melhor desvio para um par de nodos foram implementados na linguagem Java [15]. A linguagem Java foi escolhida pela possibilidade de utilização de diversas classes já existentes. Classes Java do projeto JDigraph [16], que implementam alguns conceitos de grafos, foram utilizadas. Além disso, para verificar a eficiência da estratégia de seleção de desvios, foram implementados algoritmos para cálculo da cobertura de falhas quando o primeiro, segundo e terceiro melhores desvios são utilizados. Também foi implementado um algoritmo que calcula a diferença entre o $\#C(v)$ médio obtido através da árvore de corte e o $\#C(v)$ médio obtido através da heurística, com o objetivo de verificar se a heurística obtém valores de $\#C(v)$ próximos aos valores obtidos através da árvore de corte. As classes implementadas são de domínio público e estão disponíveis em [39].

5.1 Metodologia de geração de grafos

O método Waxman foi escolhido para a geração de grafos aleatórios, dentre os métodos apresentados no capítulo 4. Optou-se pela escolha do método Waxman pois o mesmo propõe-se a capturar a característica de localidade existente nas redes reais. Os grafos

gerados tiveram o seu tamanho limitados a 100 nodos, pois o cálculo da cobertura de falhas apresentado a seguir é computacionalmente custoso. Os geradores de topologias INET e INET-3.0, que baseiam-se nas leis de potência da Internet, geram grafos com número de nodos superior a 3000 e por isso não podem ser utilizados. Os grafos gerados de menor tamanho possuem 10 nodos, tornando inviável a utilização de grafos pertencentes à classe hierárquica, que constrói grafos maiores utilizando grafos menores.

O pacote de domínio público GT-ITM [37] foi utilizado na geração dos grafos aleatórios que foram utilizados na obtenção dos resultados experimentais. O pacote GT-ITM implementa os seguintes métodos de geração de grafos, descritos com detalhes no capítulo 4: método *random* puro, método Waxman, método exponencial, método da localidade, método *N-Level* e método *Transit-Stub*.

O programa `itm` do pacote GT-ITM é o programa responsável pela geração dos grafos. O `itm` necessita de um arquivo como argumento para ser executado. Tal arquivo deve conter os parâmetros requeridos pelo método de geração escolhido. Um exemplo do arquivo contendo os parâmetros requeridos pelo método Waxman é apresentado a seguir:

```
geo 1 182
50 50 1 0.191952 1.3247
```

Na primeira linha do exemplo acima, `geo` indica que o método escolhido pertence à classe *flat random*, `1` indica que somente um grafo será gerado e `182` é a *semente* que será utilizada na geração aleatória do grafo. Na segunda linha, `50` indica o número de nodos do grafo, `50` indica a dimensão do espaço onde os nodos serão distribuídos (ou seja, os nodos serão distribuídos em um espaço de 50x50), `1` indica que o método Waxman será utilizado na geração do grafo, 0.191952 indica o parâmetro α do método Waxman e 1.3247 indica o parâmetro β do método Waxman.

Foram utilizadas 3 seqüências de números aleatórios para os parâmetros *semente*, α e β, de modo a garantir que os grafos foram gerados de maneira aleatória. As seqüências de números aleatórios foram geradas utilizando as funções `rand()` e `srand()` da linguagem de programação `awk` [40]. Foram geradas seqüências de números reais para os parâmetros

α e β. Já a seqüencia gerada para o parâmetro *semente* é composta por números inteiros. O *script* `rand_seed.sh`, implementado para a geração de números inteiros aleatórios, e o *script* `rand.sh`, implementado para a geração de números reais aleatórios estão descritos no anexo 1. Tais *scripts* necessitam de 4 argumentos: quantidade de números que serão gerados, semente inicial, limite inferior para os valores gerados e limite superior para os valores gerados.

Para cada tamanho de grafo foram geradas 3 seqüências de 1000 números aleatórios para serem utilizadas nos parâmetros α, β e *semente* do método Waxman. Os argumentos utilizados nos *scripts* `rand.sh` e `rand_seed.sh` para a geração dos números aleatórios estão descritos na tabela 5.1.

Num. Nodos	α			β			**semente**		
	Semente	Menor Valor	Maior Valor	Semente	Menor Valor	Maior Valor	Semente	Menor Valor	Maior Valor
10	7	0.1	1	3	0.2	9	5	1	1000
20	17	0.1	1	13	0.2	9	11	1	1000
30	29	0.1	1	23	0.2	6	19	1	1000
40	41	0.1	1	37	0.2	2	31	1	1000
50	51	0.1	1	47	0.15	1.5	43	1	1000
60	71	0.1	1	59	0.15	1.5	53	1	1000
70	83	0.1	1	79	0.15	1	73	1	1000
100	81	0.1	1	89	0.15	1	87	1	1000

Tabela 5.1: Argumentos utilizados na geração das seqüências de números aleatórios.

Foram gerados grafos com 10, 20, 30, 40, 50, 60, 70 e 100 nodos. Para cada tamanho, foram gerados 250 grafos distribuídos da seguinte maneira:

- 50 grafos com grau médio dos nodos pertencente ao intervalo $[3, 4)$;
- 50 grafos com grau médio dos nodos pertencente ao intervalo $[4, 5)$;
- 50 grafos com grau médio dos nodos pertencente ao intervalo $[5, 6)$;
- 50 grafos com grau médio dos nodos pertencente ao intervalo $[6, 7)$;
- 50 grafos com grau médio dos nodos pertencente ao intervalo $[7, 8]$;

Em uma árvore, o grau médio dos nodos é dado por $2(V-1)/V \leq 2$. Ou seja, o grau médio dos nodos de uma árvore é ligeiramente inferior a 2. Desta forma, optou-se pela escolha de grafos que possuem grau médio dos nodos maior ou igual a 3 pelo fato dos mesmos possuírem uma certa quantidade arestas pertencentes a componentes biconexos. Já grafos que possuem grau médio dos nodos iguais a 8 possuem uma grande conectividade, não sendo necessário analisar grafos que possuem grau médio dos nodos maior.

Para a geração de grafos utilizando o método Waxman e seguindo as definições apresentadas acima, foi desenvolvido o *script* `gera_grafos.sh`, descrito no anexo 1. O *script* `gera_grafo.sh` deve ser utilizado com os seguintes parâmetros, na ordem apresentada a seguir:

1. Número de nodos dos grafos que serão gerados;
2. Número de grafos pertencentes a cada intervalo de grau médio dos nodos, conforme foi descrito anteriormente. Por exemplo, se este parâmetro possuir o valor 50, serão gerados 50 grafos para cada intervalo de grau médio dos nodos, totalizando 250 grafos;
3. Arquivo com os valores que serão utilizados no parâmetro α do método Waxman. Tal arquivo deve ter sido gerado pelo *script* `rand.sh`;
4. Arquivo com os valores que serão utilizados no parâmetro β do método Waxman. Tal arquivo deve ter sido gerado pelo *script* `rand.sh`;
5. Arquivo com os valores que serão utilizados no parâmetro *semente* do programa `itm`. Tal arquivo deve ter sido gerado pelo *script* `rand_seed.sh`;
6. Diretório onde serão colocados os arquivos contendo os grafos gerados;

A cada iteração, o *script* `gera_grafo.sh` gera um grafo utilizando o *i-ésimo* valor do arquivo que possui os valores do parâmetro α, o *i-ésimo* valor do arquivo que possui os valores do parâmetro β e o *i-ésimo* valor do arquivo que possui os valores do parâmetro

semente. Caso o grafo gerado não se enquadre nas restrições apresentadas acima, o mesmo é descartado. O *script* `gera_grafo` termina após a geração de todos os grafos ou até o esgotamento dos valores de α, β ou *semente*.

Além dos programas e *scripts* já descritos, também foi utilizado o programa `sgb2alt` do pacote GT-ITM e implementado o *script* `alt2graph.sh`, para a conversão de formato dos arquivos de grafos gerados. O *script* `alt2graph.sh` está descrito no anexo 1.

5.2 Cálculo da Cobertura de Falhas e Avaliação da Heurística

A cobertura de falhas obtida pela utilização do algoritmo de seleção de desvios em um grafo qualquer foi calculada da seguinte maneira. Inicialmente foi obtido o caminho mínimo entre cada par de nodos do grafo, com o intuito de modelar uma rede real, onde os nodos comunicam-se através do menor caminho.

Para cada par de nodos A e B do grafo, os seguintes procedimentos foram executados:

- Obteve-se a lista ordenada dos melhores desvios tendo o nodo A como origem e o nodo B como destino;
- Para cada aresta (x, y) que não é de corte pertencente ao caminho entre A e B, os seguintes procedimentos foram executados:
  - A aresta (x, y) é marcada como falha e o contador de falhas na rede é incrementado;
  - Verifica-se se a rota alternativa através do melhor desvio $D1$ contém a aresta (x, y). Caso a rota alternativa não contenha a aresta (x, y), significa que a mesma é funcional e o contador das falhas cobertas pelo melhor desvio é incrementado. A rota alternativa é composta pela rota regular entre A e $D1$ e pela rota regular entre $D1$ e B;
  - Repete-se o mesmo procedimento para o segundo melhor desvio;
  - Repete-se o mesmo procedimento para o terceiro melhor desvio;

A falha de uma aresta de corte não foi realizada, pois não existiria outro caminho no grafo entre os nodos A e B. Nestes casos, a cobertura de falhas utilizando qualquer desvio é impossível.

Após a execução dos procedimentos acima descritos para cada par de nodos do grafo, a cobertura de falhas utilizando o melhor desvio é dada por:

$$\frac{falhas\ recuperadas\ pelo\ melhor\ desvio}{falhas\ na\ rede}$$

O cálculo da cobertura de falhas utilizando o segundo e terceiro melhores desvios é análogo.

Além do cálculo da cobertura de falhas utilizando o melhor desvio, os cálculos da cobertura de falhas utilizando o segundo e terceiro melhores desvios foram realizados com o intuito de confirmar se realmente o algoritmo implementado retorna a ordem correta dos nodos que possuem a melhor capacidade de desviar de uma falha na rede.

Caso o nodo $D1$ utilizado como desvio pertença ao menor caminho entre os nodos A e B, existe a possibilidade da rota alternativa formada pela rota regular entre os nodos A e $D1$ e pela rota regular entre os nodos $D1$ e B ser idêntica à rota regular entre os nodos A e B. Nestes casos, o nodo $D1$ utilizado como desvio não consegue recuperar as falhas das arestas existentes no caminho entre A e B. Para avaliar o impacto da escolha de um nodo pertencente ao caminho como desvio, também foram realizados cálculos de cobertura de falhas com os três melhores desvios não pertencentes ao caminho utilizado pelos nodos comunicantes, conforme descrição anterior. Além disso, foram coletadas informações da percentagem de vezes em que a rota alternativa criada através de um desvio pertencente ao caminho é idêntica à rota regular.

Além dos cálculos da cobertura de falhas utilizando desvios retornados pelo algoritmo exato, foram realizados cálculos da cobertura de falhas utilizando desvios retornados pela heurística. Também foi calculada a diferença entre o $\#C(v)$ médio retornado pelo algoritmo exato e o $\#C(v)$ médio retornado pela heurística, para verificar se a heurística retorna valores próximos aos valores retornados pelo algoritmo exato do $\#C(v)$. Os nodos selecionados para desvios pelo algoritmo exato serão chamados *desvios exatos*. Ana-

logamente, os nodos selecionados para desvios pela heurística serão chamados *desvios heurísticos.*

Ao final, as seguintes informações são obtidas para cada grafo:

- Cobertura de falhas utilizando o melhor desvio;
- Cobertura de falhas utilizando o segundo melhor desvio;
- Cobertura de falhas utilizando o terceiro melhor desvio;
- Cobertura de falhas acumulada utilizando o melhor desvio;
- Cobertura de falhas acumulada utilizando o primeiro e segundo melhores desvios;
- Cobertura de falhas acumulada utilizando o primeiro, segundo e terceiro melhores desvios;
- Cobertura de falhas utilizando o melhor desvio não pertencente ao caminho;
- Cobertura de falhas utilizando o segundo melhor desvio não pertencente ao caminho;
- Cobertura de falhas utilizando o terceiro melhor desvio não pertencente ao caminho;
- Cobertura de falhas acumulada utilizando o melhor desvio não pertencente ao caminho;
- Cobertura de falhas acumulada utilizando o primeiro e segundo melhores desvios não pertencentes ao caminho;
- Cobertura de falhas acumulada utilizando o primeiro, segundo e terceiro melhores desvios não pertencentes ao caminho;
- Cobertura de falhas utilizando o melhor desvio heurístico;
- Cobertura de falhas utilizando o segundo melhor desvio heurístico;
- Cobertura de falhas utilizando o terceiro melhor desvio heurístico;
- Cobertura de falhas acumulada utilizando o melhor desvio heurístico;

- Cobertura de falhas acumulada utilizando o primeiro e segundo melhores desvios heurísticos;
- Cobertura de falhas acumulada utilizando o primeiro, segundo e terceiro melhores desvios heurísticos;
- Cobertura de falhas utilizando o melhor desvio heurístico não pertencente ao caminho;
- Cobertura de falhas utilizando o segundo melhor desvio heurístico não pertencente ao caminho;
- Cobertura de falhas utilizando o terceiro melhor desvio heurístico não pertencente ao caminho;
- Cobertura de falhas acumulada utilizando o melhor desvio heurístico não pertencente ao caminho;
- Cobertura de falhas acumulada utilizando o primeiro e segundo melhores desvios heurísticos não pertencentes ao caminho;
- Cobertura de falhas acumulada utilizando o primeiro, segundo e terceiro melhores desvios heurísticos não pertencentes ao caminho;
- Diferença entre o $\#C(v)$ médio retornado pelo algoritmo exato e o $\#C(v)$ médio retornado pela heurística.

5.3 Apresentação dos Resultados Experimentais

A seguir, são apresentados os resultados dos experimentos descritos com detalhes na seção anterior. Foram obtidos resultados de experimentos realizados com grafos que possuem 10, 20, 30, 40, 50, 60, 70 e 100 nodos. Foram analisados 250 grafos de cada tamanho, sendo que no total foram analisados 2000 grafos com grau médio dos nodos variando de 3 a 8.

5.3.1 Avaliação dos Valores de $\#C(v)$ Retornados pela Heurística

Conforme descrito na seção anterior, a diferença entre o $\#C(v)$ médio retornado pelo algoritmo exato e o $\#C(v)$ médio retornado pela heurística foi obtida para cada grafo analisado. O $\#C(v)$ médio retornado pelo algoritmo exato será chamado de $\#C(v)$ médio exato. Analogamente, o $\#C(v)$ médio retornado pela heurística será chamado de $\#C(v)$ médio heurístico. A diferença média entre o $\#C(v)$ médio exato e o $\#C(v)$ médio heurístico é apresentada na tabela 5.2.

Num. Nodos	**Diferença**					
	Grau médio dos nodos no intervalo					Geral
	$[3,4)$	$[4,5)$	$[5,6)$	$[6,7)$	$[7,8]$	
10	0.526	0.730	0.886	1.036	1.160	0.868
20	0.648	0.892	1.029	1.150	1.265	0.997
30	0.739	0.925	1.063	1.180	1.382	1.058
40	0.691	0.929	1.137	1.269	1.413	1.088
50	0.723	0.950	1.113	1.296	1.432	1.103
60	0.715	0.948	1.147	1.262	1.470	1.108
70	0.712	0.945	1.157	1.287	1.458	1.112
100	0.717	0.948	1.235	1.293	1.457	1.130

Tabela 5.2: Diferença média entre o $\#C(v)$ médio exato e o $\#C(v)$ médio heurístico.

Observa-se na tabela 5.2 que a diferença média entre o $\#C(v)$ médio exato e $\#C(v)$ médio heurístico tende a ser maior com o crescimento do número de nodos do grafo. Analisando cada linha da tabela separadamente, observa-se que a diferença média entre o $\#C(v)$ médio exato e $\#C(v)$ médio heurístico aumenta com o crescimento do grau médio dos nodos.

A quantidade de arestas de um grafo aumenta com o crescimento do grau médio dos nodos. Desta forma, o valor médio do $\#C(v)$ exato tende a aumentar com o crescimento do grau médio dos nodos do grafo. Portanto, para uma melhor comparação da variação da diferença média entre o $\#C(v)$ médio exato e o $\#C(v)$ médio heurístico, de acordo com a variação do grau médio dos nodos do grafo, as diferenças obtidas foram normalizadas utilizando o grau médio dos nodos. Os resultados da normalização estão descritos na tabela 5.3.

Analisando isoladamente cada linha da tabela 5.3, após a normalização não se observa

Num. Nodos	Diferença / grau médio dos nodos					
	Grau médio dos nodos no intervalo					Geral
	$[3,4)$	$[4,5)$	$[5,6)$	$[6,7)$	$[7,8]$	
10	0.153	0.166	0.163	0.163	0.154	0.160
20	0.186	0.199	0.189	0.179	0.170	0.185
30	0.207	0.206	0.197	0.184	0.184	0.196
40	0.200	0.207	0.205	0.195	0.190	0.200
50	0.207	0.211	0.206	0.201	0.192	0.204
60	0.205	0.214	0.210	0.196	0.197	0.204
70	0.207	0.210	0.212	0.198	0.196	0.205
100	0.199	0.216	0.218	0.202	0.194	0.206

Tabela 5.3: Diferença média entre o $\#C(v)$ médio exato e o $\#C(v)$ médio heurístico, normalizada pelo grau médio dos nodos.

um crescimento da diferença média entre o $\#C(v)$ médio exato e o $\#C(v)$ médio heurístico de acordo com o crescimento do grau médio dos nodos. Pelo contrário, não foi possível identificar uma tendência para a variação da diferença média entre os $\#C(v)$'s de acordo com o crescimento do grau médio dos nodos.

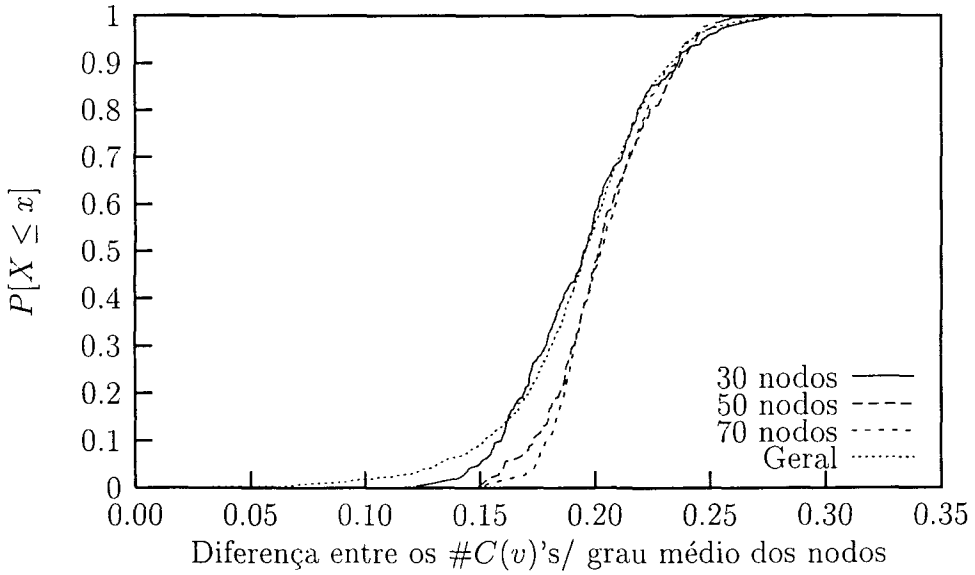


Figura 5.1: CDF da diferença do $\#C(v)$ médio exato e do $\#C(v)$ médio heurístico, normalizada pelo grau médio dos nodos.

A figura 5.1 apresenta a função de distribuição acumulada (CDF - *Cumulative Distribution Function*) da diferença do $\#C(v)$ médio exato e do $\#C(v)$ médio heurístico, após a normalização. Foram traçadas funções CDF de alguns tamanhos de grafos, além da função CDF geral que contém os dados de todos os grafos analisados. As funções CDF

dos tamanhos de grafos que não foram traçadas na figura apresentam curvas similares às que foram traçadas e por isso foram omitidas. Observa-se que em 90% dos casos, a diferença entre os $\#C(v)$'s após a normalização é inferior a 0.237. Observa-se também que as 4 curvas traçadas são bantante parecidas, a partir do ponto em que as funções CDF atingem 0.9. Ou seja, pode-se concluir que a partir deste ponto, a quantidade de nodos nos grafos analisados praticamente não influencia no resultado da função CDF. Desta forma, pode-se afirmar que, para grafos que possuem grau médio dos nodos igual a 4. por exemplo, existe 90% de chance da diferença do $\#C(v)$ médio exato e do $\#C(v)$ médio heurístico ser inferior a $4 * 0.237$, ou seja 0.948. A tabela 5.4 apresenta, para alguns valores de grau médio dos nodos, qual é o valor correspondente a 90% das medições da diferença média entre os $\#C(v)$'s calculado da maneira *grau medio dos nodos* $* 0.237$. Analisando as informações da tabela, por exemplo, observa-se que, para os grafos analisados que possuem grau médio dos nodos igual a 5, existe 90% de chance da diferença média entre os $\#C(v)$'s ser inferior a 1.185.

Grau médio dos nodos	**Diferença entre os $\#C(v)$'s**
3	0.711
4	0.948
5	1.185
6	1.422
7	1.659
8	1.896

Tabela 5.4: Valores correspondentes a 90% das medições da diferença média entre os $\#C(v)$'s.

Conclui-se que em 90% dos casos, a diferença média entre os $\#C(v)$'s é inferior a 23,7% do valor do grau médio dos nodos. Ou seja, a heurística cumpre sua função em retornar valores de $\#C(v)$ próximos aos valores retornados pelo algoritmo exato. Além da avaliação dos valores de $\#C(v)$ retornados pela heurística, será apresentada uma avaliação da cobertura de falhas obtida pelos melhores desvios retornados pela heurística.

5.3.2 Avaliação dos Resultados da Cobertura de Falhas

5.3.2.1 Cobertura de Falhas Obtida por Desvios Exatos e por Desvios Heurísticos

A tabela 5.6 apresenta os resultados da cobertura de falhas obtida pelos 3 melhores desvios retornados pelo algoritmo exato. Em praticamente todos os tamanhos de grafos e todo grau médio dos nodos, o melhor desvio exato não obtém a melhor cobertura de falhas, se comparada com a cobertura de falhas obtida pelo segundo melhor desvio exato e com a cobertura de falhas obtida pelo terceiro melhor desvio exato. Esta situação não é observada somente nos grafos que possuem 10 nodos e grau médio dos nodos pertencente ao intervalo $[5, 8]$. O mesmo comportamento é observado na tabela 5.8, que apresenta os resultados da cobertura de falhas obtida pelos 3 melhores desvios heurísticos. A figura 5.2 apresenta graficamente os resultados da cobertura de falhas obtida pelos 3 melhores desvios exatos em todos os grafos analisados. Novamente observa-se que a cobertura de falhas obtida pelo melhor desvio exato é inferior à cobertura de falhas obtida pelo segundo e pelo terceiro melhores desvios exatos.

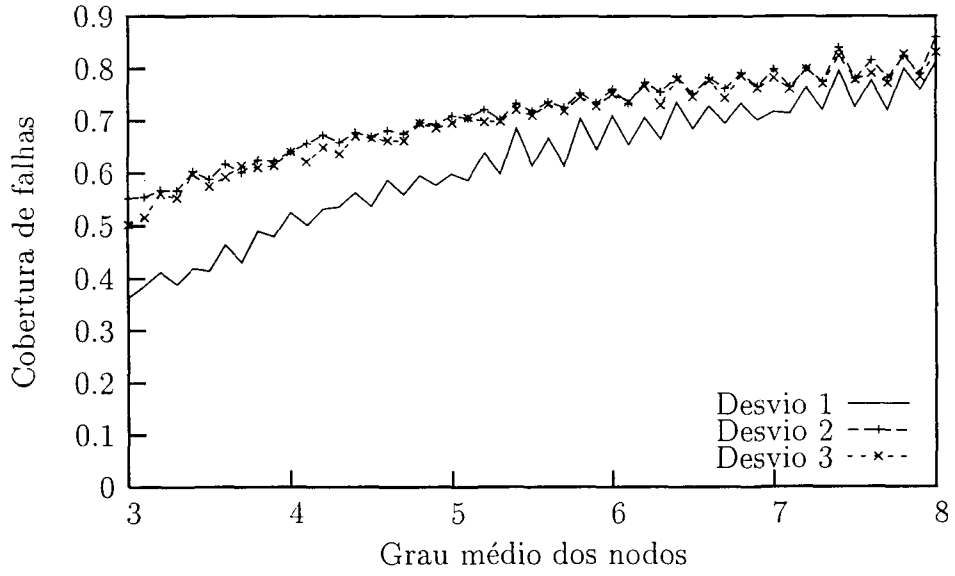


Figura 5.2: Cobertura de falhas obtida pelos 3 melhores desvios exatos.

Caso o nodo escolhido como desvio pertença ao caminho que o nodo origem utiliza para alcançar o nodo destino, existe a possibilidade da rota alternativa criada através do

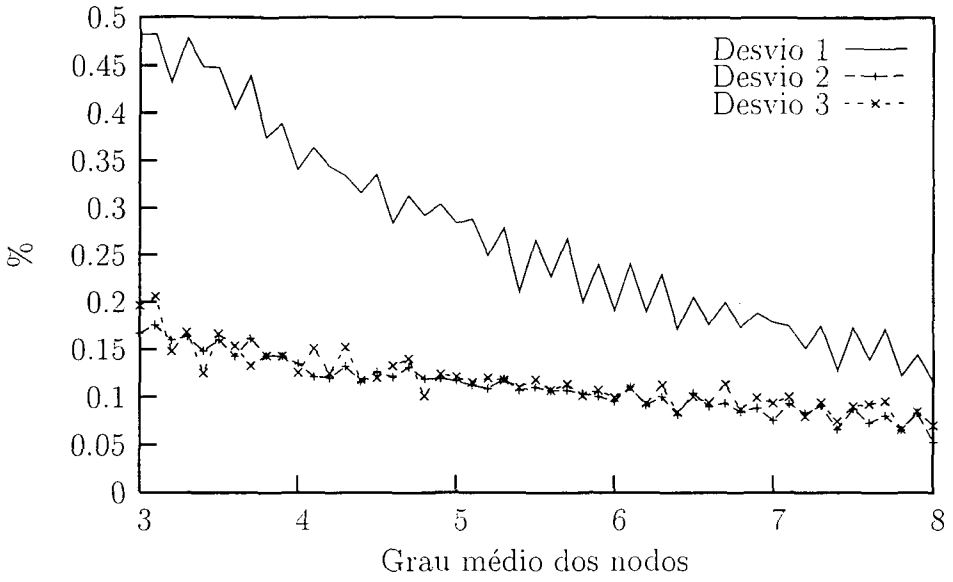


Figura 5.3: Percentagem de vezes em que os melhores desvios exatos pertencem ao caminho utilizado pelos nodos comunicantes.

desvio ser idêntica à rota falha, não sendo possível recuperar a falha da rede. A cobertura de falhas obtida pelo melhor desvio exato é pior que a cobertura de falhas obtida pelo segundo melhor desvio exato, como pode ser observado nas tabelas 5.6, 5.8 e na figura 5.2. Tal resultado deve-se ao fato do percentual de vezes em que o melhor desvio exato pertence ao caminho utilizado pelos nodos comunicantes ser maior que o percentual de vezes em que o segundo melhor desvio exato pertence ao caminho utilizado pelos nodos comunicantes, como pode ser observado no gráfico da figura 5.3. Na figura 5.3, observa-se também que a diferença entre o percentual de vezes em que o melhor desvio exato e o percentual de vezes em que o segundo melhor desvio exato pertencem ao caminho utilizado pelos nodos comunicantes diminui com o aumento do grau médio dos nodos do grafo. Tal resultado explica o fato da diferença entre a cobertura de falhas alcançada pelo segundo melhor desvio exato e a cobertura de falhas alcançada pelo melhor desvio exato diminuir com o aumento do grau médio dos nodos, como pode ser observado nas tabelas 5.6, 5.8 e na figura 5.2.

Conclui-se que a estratégia de seleção de desvios baseando-se apenas em critérios de conectividade não obteve o resultado esperado. Foi mostrado que o melhor desvio, ou seja, o nodo que apresenta a maior conectividade, obteve uma cobertura de falhas

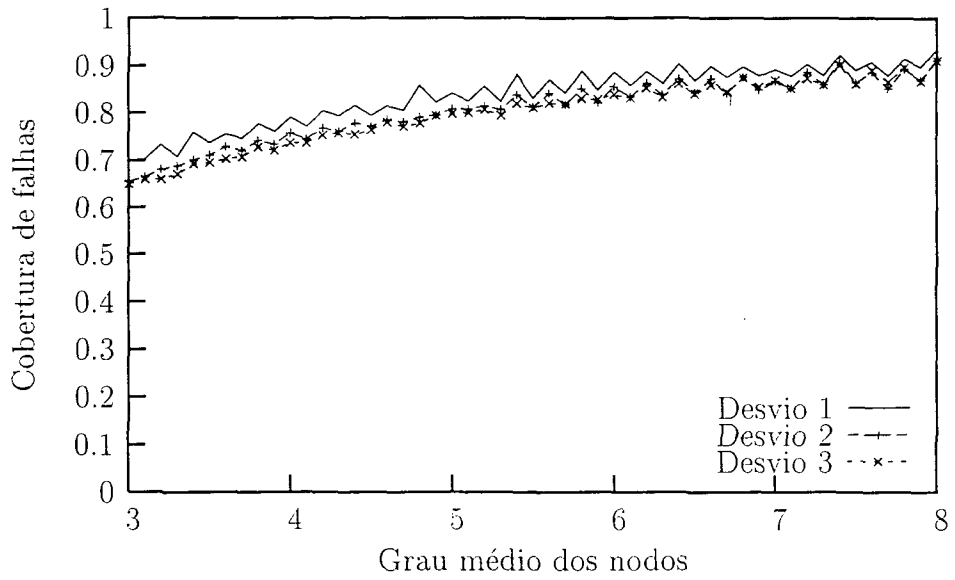


Figura 5.4: Cobertura de falhas obtida pelos 3 melhores desvios exatos e não pertencentes ao caminho.

inferior à cobertura de falhas alcançada pelo segundo melhor desvio, que possui uma menor conectividade.

5.3.2.2 Cobertura de Falhas Obtida por Desvios Exatos não Pertencentes ao Caminho

Na tabela 5.7, que apresenta os resultados da cobertura de falhas obtida pelos 3 melhores desvios exatos não pertencentes ao caminho, observa-se que, para todo tamanho de grafo e todo grau médio dos nodos, o melhor desvio exato não pertencente ao caminho sempre obtém uma melhor cobertura de falhas, se comparada com a cobertura de falhas obtida pelo segundo e pelo terceiro melhor desvio exato não pertencente ao caminho. A figura 5.4 apresenta graficamente os resultados da cobertura de falhas obtida pelos 3 melhores desvios exatos não pertencentes ao caminho. Novamente pode-se observar que a cobertura de falhas obtida pelo melhor desvio é superior à cobertura de falhas obtida pelo segundo melhor desvio, que é superior à cobertura de falhas obtida pelo terceiro melhor desvio.

Comparando os resultados da cobertura de falhas obtida pelos melhores desvios exatos não pertencentes ao caminho (tabelas 5.7, 5.9 e figura 5.4), com os resultados da cobertura de falhas obtida pelos melhores desvios exatos (tabelas 5.6, 5.8 e figura 5.2),

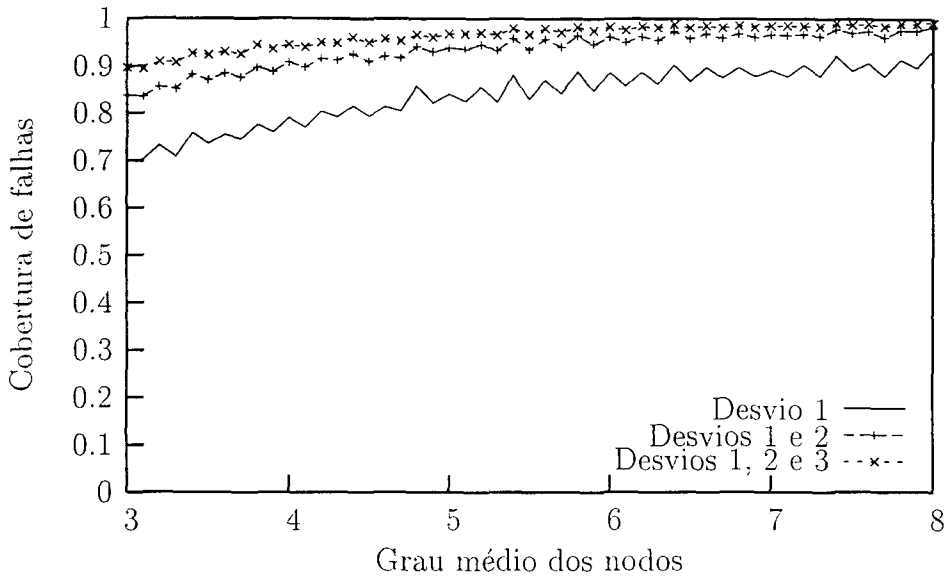


Figura 5.5: Cobertura de falhas acumulada obtida pelos 3 melhores desvios exatos não pertencentes ao caminho.

constata-se que a cobertura de falhas obtida pela utilização dos melhores desvios exatos não pertencentes ao caminho é superior à cobertura de falhas obtida pela utilização dos melhores desvios exatos. Nos experimentos realizados, a percentagem de vezes em que as rotas criadas através de desvios pertencentes ao caminho foram idênticas às rotas falhas foi de 98.7%. Isto explica o fato da seleção de desvios, sem verificar se os mesmos pertencem ou não ao caminho que o nodo origem utiliza para alcançar o nodo destino, levar a resultados piores de cobertura de falhas.

O gráfico apresentado na figura 5.4, mostra que a cobertura de falhas obtida pelo melhor desvio exato não pertencente ao caminho em grafos que possuem grau médio dos nodos igual a 3 é de 70%. Ou seja, mesmo para grafos com número reduzido de arestas, a cobertura de falhas alcançou um valor alto. Observa-se que, conforme o esperado, a cobertura de falhas aumenta de acordo com o aumento no grau médio dos nodos, chegando a 90% em grafos que possuem grau médio dos nodos igual a 8. A figura 5.5, apresenta a cobertura de falhas acumulada obtida pelo 3 melhores desvios exatos não percentences ao caminho. Observa-se que a cobertura de falhas em grafos que possuem grau médio dos nodos igual a 3 salta para 81%, quando são utilizados o melhor e o segundo melhor desvio exato não pertencente ao caminho, e para 87%, quando são utilizados o melhor,

o segundo melhor e o terceiro melhor desvio exato não pertencente ao caminho. Além disso, quando são utilizados os três melhores desvios, a cobertura de falhas atinge valores superiores a 95%, nos casos em que o grau médio dos nodos é superior a 4.4, e atinge valores superiores a 98%, nos casos em que o grau médio dos nodos é superior a 6.8.

Conclui-se que a estratégia de seleção de desvios baseada nos critérios de conectividade propostos e que não pertencem ao caminho utilizado pelos nodos comunicantes obteve os resultados esperados de cobertura de falhas. Foi mostrado que o melhor desvio exato não pertencente ao caminho, ou seja, o nodo que apresenta a maior conectividade e que não faz parte do caminho utilizado pelos nodos comunicantes, obteve uma cobertura de falhas superior à cobertura de falhas alcançada pelo segundo melhor desvio exato não pertencente ao caminho, que possui uma conectividade menor. Foi demonstrado também que a cobertura de falhas obtida pelo melhor desvio exato não pertencente ao caminho atinge valores altos, mesmo para grafos em que o grau médio dos nodos é baixo. Além disso, foi mostrado que a cobertura de falhas pode ser incrementada utilizando mais de um desvio simultaneamente. Em alguns grafos, a cobertura de falhas acumulada foi superior a 98%.

5.3.2.3 Cobertura de Falhas Obtida por Desvios Heurísticos não Pertencentes ao Caminho

A tabela 5.9 apresenta os resultados da cobertura de falhas obtida pelos 3 melhores desvios heurísticos não pertencentes ao caminho. Observa-se que, para todo tamanho de grafo e todo grau médio dos nodos, o melhor desvio heurístico não pertencente ao caminho sempre obtém uma melhor cobertura de falhas, se comparada com a cobertura de falhas obtida pelo segundo e pelo terceiro melhor desvio heurístico não pertencente ao caminho.

A figura 5.6 apresenta o gráfico da cobertura de falhas obtida pelos 3 melhores desvios heurísticos não pertencentes ao caminho. Já a figura 5.7 apresenta o gráfico da cobertura de falhas acumulada obtida pelos 3 melhores desvios heurísticos não pertencentes ao caminho. Observa-se que a cobertura de falhas obtida pelos desvios heurísticos não pertencentes ao caminho é similar à cobertura de falhas obtida pelos desvios exatos não

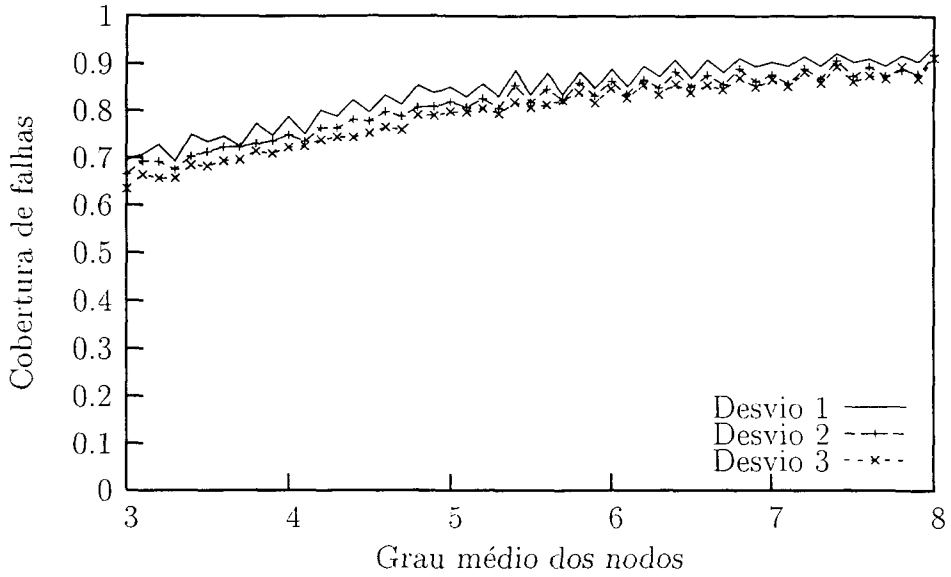


Figura 5.6: Cobertura de falhas obtida pelos três melhores desvios heurísticos não pertencentes ao caminho.

pertencentes ao caminho, apresentados nas figuras 5.4 e 5.6. A similaridade também pode ser observada na figura 5.8, que apresenta o gráfico da cobertura de falhas obtida pelo melhor desvio exato não pertencente ao caminho e pelo melhor desvio heurístico não pertencente ao caminho. Observa-se que a cobertura de falhas obtida pelo melhor desvio heurístico é praticamente idêntica à cobertura de falhas obtida pelo melhor desvio exato. Aliás, quando os resultados gerais da cobertura de falhas apresentados na tabela 5.7 são comparados com os resultados gerais da cobertura de falhas apresentados na tabela 5.9, observa-se que os desvios heurísticos obtiveram cobertura de falhas ligeiramente superior à cobertura de falhas obtida pelos desvios exatos.

Conclui-se que não existem impeditivos em utilizar a heurística para a seleção dos melhores desvios, visto que os desvios selecionados pela heurística obtiveram cobertura de falhas ligeiramente superior à cobertura de falhas obtida pelos desvios retornados pelo algoritmo exato.

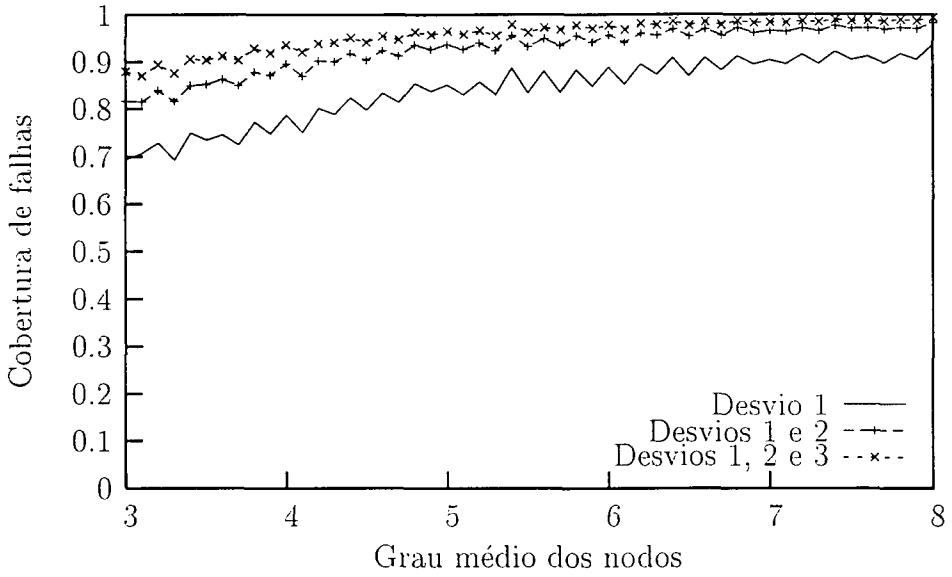


Figura 5.7: Cobertura de falhas acumulada obtida pelos 3 melhores desvios heurísticos não pertencentes ao caminho.

5.3.2.4 Comparação da Cobertura de Falhas em Diversos Tamanhos de Grafos

A figura 5.9 apresenta a cobertura de falhas obtida pelo melhor desvio exato não pertencente ao caminho em diversos tamanhos de grafos. Observa-se que, a cobertura de falhas obtida em grafos que possuem 10 nodos é significativamente superior à cobertura de falhas obtida em grafos que possuem 50 e 100 nodos. Porém, a cobertura de falhas obtida em grafos que possuem 50 e 100 nodos são praticamente idênticas. Analisando os resultados gerais de cobertura de falhas apresentados na tabela 5.7, observa-se que existe uma tendência de diminuição na cobertura de falhas de acordo com o aumento no número de nodos do grafo.

5.3.2.5 Comparação da Estratégia de Seleção de Desvios Baseada nos Critérios de Conectividade Propostos com uma Estratégia de Seleção Aleatória de Desvios

A cobertura de falhas obtida através da seleção aleatória de desvios também foi realizada com o intuito de comparar a estratégia de seleção de desvios baseada nos critérios de

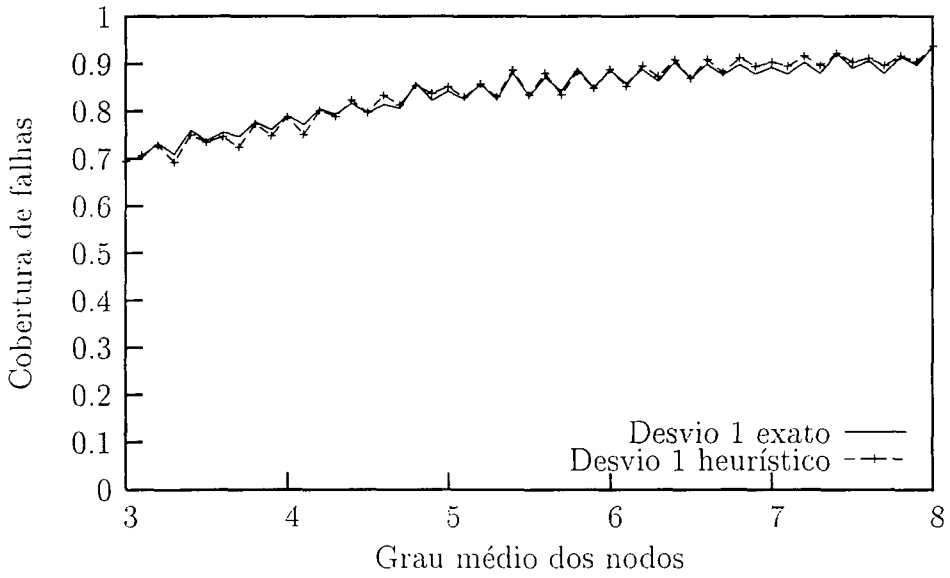


Figura 5.8: Cobertura de falhas obtida pelo melhor desvio exato e pelo melhor desvio heurístico não pertencentes ao caminho.

conectividade propostos com outra estratégia de seleção de desvios. A figura 5.10 apresenta o gráfico da cobertura de falhas obtida pelo melhor desvio exato não pertencente ao caminho, a cobertura de falhas obtida pela seleção aleatória de desvios e a cobertura de falhas obtida pela seleção aleatória de desvios não pertencentes ao caminho. Observa-se que a cobertura de falhas obtida por desvios selecionados através dos critérios de conectividade propostos é sempre maior que a cobertura de falhas obtida pela seleção aleatória de desvios e pela seleção aleatória de desvios não pertencentes ao caminho. Em grafos com grau médio dos nodos igual a 3, por exemplo, a cobertura de falhas obtida por desvios selecionados aleatoriamente foi de 50% e a cobertura de falhas obtida por desvios selecionados a partir dos critérios de conectividade propostos foi de 70%, ou seja, uma melhoria de 20%.

A tabela 5.5 apresenta as médias para a cobertura de falhas obtida pelos melhores desvios exatos não pertencentes ao caminho, por desvios selecionados aleatoriamente e por desvios não pertencentes ao caminho também selecionados aleatoriamente. São apresentadas as médias para a cobertura de falhas acumulada quando 1, 2, 3 e 4 desvios são utilizados. Observa-se que, quando somente um desvio é utilizado, a diferença média da cobertura de falhas obtida pelo melhor desvio exato não pertencente ao caminho e pela

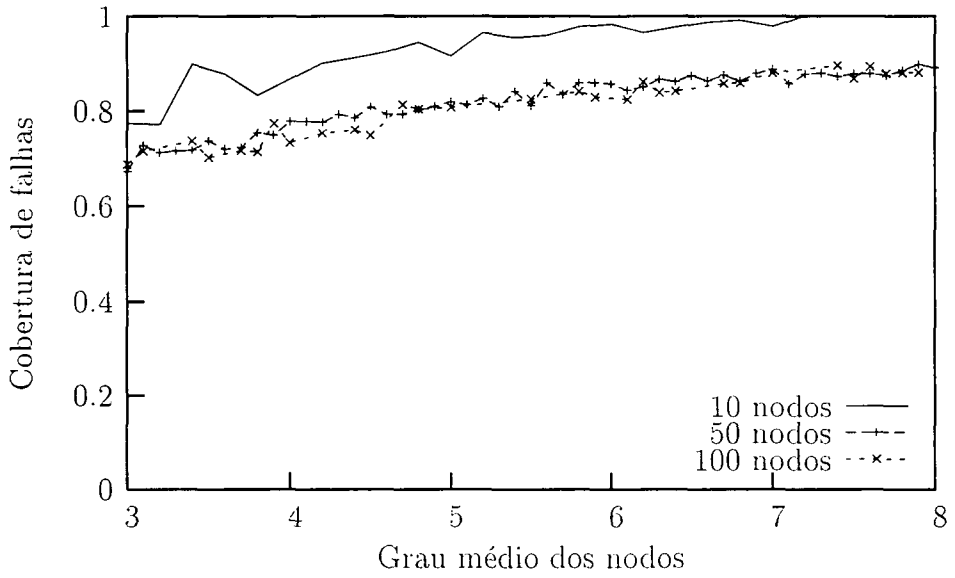


Figura 5.9: Cobertura de falhas obtida pelo melhor desvio não pertencente ao caminho em diversos tamanhos de grafos.

seleção aleatória de desvios é de 15.7%. Já a diferença média da cobertura de falhas obtida pelo melhor desvio exato não pertencente ao caminho e pela seleção aleatória de desvios não pertencentes ao caminho é de 6.7%. Chama a atenção o fato da diferença média da cobertura de falhas obtida pelo melhor desvio exato não pertencente ao caminho e pela seleção aleatória de desvios ser de apenas 15.7%, e da diferença média da cobertura de falhas obtida pelo melhor desvio exato não pertencente ao caminho e pela seleção aleatória de desvios não pertencentes ao caminho ser de apenas 6.7%. De qualquer forma, o cálculo da cobertura de falhas mostra claramente que o uso de desvios selecionados a partir dos critérios de conectividade propostos aumenta a confiabilidade do roteamento em todos os casos.

Além disso, observa-se que a diferença média da cobertura de falhas obtida pelo melhor desvio exato não pertencente ao caminho e pela seleção aleatória de desvios, bem como a diferença média da cobertura de falhas obtida pelo melhor desvio exato não pertencente ao caminho e pela seleção aleatória de desvios não pertencentes ao caminho diminui com aumento do grau médio dos nodos do grafo. Tal fato é observado quando 1, 2 e 3 desvios são utilizados. Ou seja, a estratégia de seleção de desvios baseada nos critérios de conectividade propostos faz diferença em grafos esparsos, que são mais próximos das

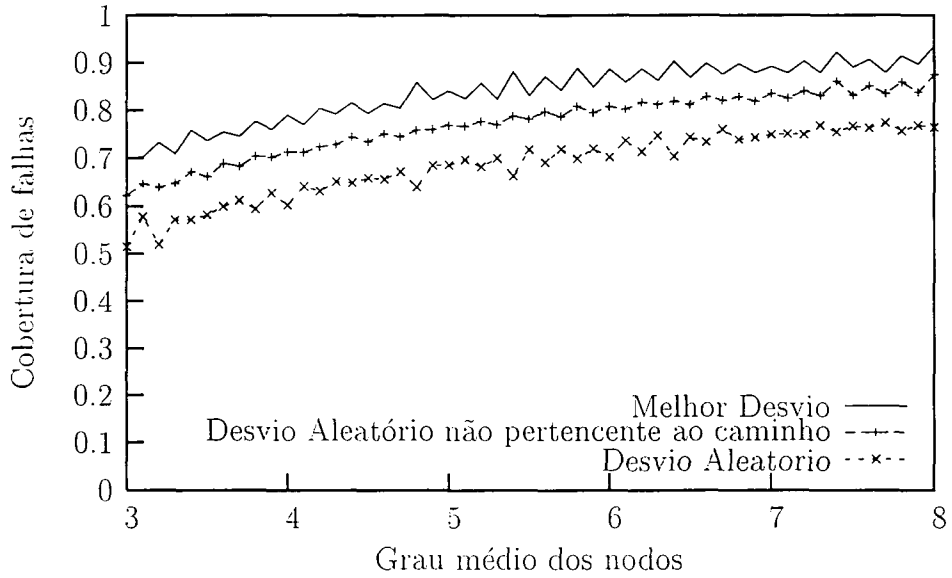


Figura 5.10: Cobertura de falhas obtida pelo melhor desvio exato não pertencente ao caminho e por desvios escolhidos aleatoriamente.

topologias utilizadas na prática.

Observa-se também que a diferença média da cobertura de falhas obtida pelo melhor desvio exato não pertencente ao caminho e pela seleção aleatória de desvios, bem como a diferença média da cobertura de falhas obtida pelo melhor desvio exato não pertencente ao caminho e pela seleção aleatória de desvios não pertencentes ao caminho diminui quando uma maior quantidade de desvios são utilizados. Quando 3 desvios são utilizados simultaneamente, a cobertura de falhas obtida pela seleção aleatória de desvios não pertencentes ao caminho possui valores ligeiramente inferiores à cobertura de falhas obtida pelos 3 melhores desvios exatos não pertencentes ao caminho. Já para desvios selecionados aleatoriamente, são necessários 4 desvios para que a cobertura de falhas seja semelhante à cobertura de falhas obtida pelos 3 melhores desvios exatos não pertencentes ao caminho.

Uma possível explicação para o fato da cobertura de falhas acumulada obtida pelos melhores desvios exatos não pertencentes ao caminho não ser tão maior que a cobertura de falhas acumulada obtida por desvios aleatórios não pertencentes ao caminho é intuitiva. Caso o primeiro desvio aleatório não pertencente ao caminho não consiga desviar da aresta falha, um segundo desvio é utilizado. Tal desvio também é escolhido de maneira aleatória, possuindo grande probabilidade de pertencer a um outro componente do grafo.

Num. Desvios	Desvio Utilizado	Cobertura de falhas por Intervalo de Grau Médio dos Nodos e Geral					
		[3, 4)	[4, 5)	[5, 6)	[6, 7)	[7, 8]	Geral
1	Exato não pertencente ao caminho	0.740	0.810	0.854	0.883	0.900	0.838
	Aleatório não pertencente ao caminho	0.668	0.738	0.785	0.818	0.844	0.771
	Aleatório	0.576	0.648	0.695	0.730	0.756	0.681
2	Exato não pertencente ao caminho	0.870	0.920	0.947	0.963	0.971	0.934
	Aleatório não pertencente ao caminho	0.845	0.896	0.926	0.945	0.957	0.914
	Aleatório	0.784	0.845	0.884	0.907	0.925	0.869
3	Exato não pertencente ao caminho	0.921	0.955	0.973	0.983	0.987	0.964
	Aleatório não pertencente ao caminho	0.912	0.947	0.967	0.979	0.985	0.958
	Aleatório	0.874	0.920	0.947	0.961	0.972	0.935
4	Aleatório	0.918	0.954	0.973	0.982	0.988	0.963

Tabela 5.5: Médias para a cobertura de falhas acumulada obtida pelos melhores desvios exatos não pertencentes ao caminho, por desvios selecionados aleatoriamente e por desvios não pertencentes ao caminho selecionados aleatoriamente.

Em todos os experimentos realizados, somente uma aresta do grafo foi marcada como falha em cada etapa. Desta forma, é grande a probabilidade do segundo desvio aleatório não pertencente ao caminho desviar da aresta falha, já que o mesmo possui grande probabilidade de estar situado em um outro componente do grafo. Em contrapartida, o algoritmo de seleção de desvios retorna uma lista de nodos ordenada por grau de conectividade. Desta forma, existe a tendência dos três melhores desvios exatos não pertencentes ao caminho pertencerem ao mesmo componente do grafo, possuindo probabilidade semelhante de desviar de uma aresta falha. Com isso, quando mais nodos são utilizados, a cobertura de falhas acumulada obtida por desvios aleatórios não pertencentes ao caminho tende a aproximar-se da cobertura de falhas acumulada obtida pelos melhores desvios exatos não pertencentes ao caminho.

Porém, caso existam mais arestas falhas no grafo, o segundo desvio aleatório não pertencente ao caminho possui maior probabilidade de não conseguir desviar de uma outra aresta falha se comparado ao segundo melhor desvio exato não pertencente ao caminho, pois a conectividade do nodo não foi levado em consideração na sua seleção. Desta forma, acredita-se que na existência de mais arestas falhas no grafo, a diferença da cobertura de

falhas obtida por desvios exatos não pertencentes ao caminho e da cobertura de falhas obtida por desvios aleatórios não pertencentes ao caminho tende a aumentar.

Num. Nodos	Desvio(s) Utilizado(s)	Cobertura de falhas por Intervalo de Grau Médio dos Nodos e Geral					
		$[3,4)$	$[4,5)$	$[5,6)$	$[6,7)$	$[7,8]$	Geral
10	1	0.453	0.648	0.807	0.870	0.956	0.747
	2	0.597	0.699	0.776	0.845	0.946	0.773
	3	0.605	0.690	0.762	0.856	0.927	0.768
	1 e 2	0.814	0.928	0.975	0.986	0.998	0.940
	1, 2 e 3	0.918	0.974	0.994	0.997	0.998	0.976
20	1	0.401	0.526	0.605	0.690	0.754	0.595
	2	0.576	0.660	0.715	0.764	0.787	0.700
	3	0.576	0.662	0.707	0.742	0.774	0.692
	1 e 2	0.762	0.845	0.892	0.937	0.948	0.877
	1, 2 e 3	0.884	0.934	0.959	0.977	0.983	0.947
30	1	0.405	0.517	0.600	0.673	0.714	0.582
	2	0.580	0.660	0.704	0.752	0.776	0.694
	3	0.570	0.663	0.697	0.741	0.774	0.689
	1 e 2	0.742	0.837	0.878	0.922	0.934	0.863
	1, 2 e 3	0.861	0.924	0.950	0.972	0.979	0.937
40	1	0.401	0.538	0.611	0.665	0.717	0.587
	2	0.567	0.664	0.719	0.750	0.776	0.695
	3	0.558	0.629	0.705	0.746	0.772	0.682
	1 e 2	0.699	0.828	0.892	0.913	0.934	0.853
	1, 2 e 3	0.826	0.914	0.954	0.969	0.978	0.928
50	1	0.421	0.549	0.612	0.670	0.715	0.593
	2	0.602	0.670	0.716	0.749	0.780	0.704
	3	0.589	0.660	0.706	0.738	0.773	0.693
	1 e 2	0.741	0.829	0.881	0.912	0.933	0.859
	1, 2 e 3	0.858	0.916	0.948	0.965	0.976	0.933
60	1	0.440	0.543	0.621	0.682	0.725	0.602
	2	0.603	0.678	0.724	0.753	0.776	0.707
	3	0.574	0.654	0.722	0.739	0.766	0.691
	1 e 2	0.743	0.829	0.877	0.913	0.930	0.858
	1, 2 e 3	0.849	0.910	0.946	0.965	0.973	0.929
70	1	0.460	0.563	0.619	0.678	0.725	0.609
	2	0.608	0.683	0.721	0.760	0.783	0.711
	3	0.552	0.657	0.720	0.751	0.772	0.690
	1 e 2	0.753	0.835	0.878	0.913	0.935	0.863
	1, 2 e 3	0.849	0.918	0.947	0.965	0.976	0.931
100	1	0.527	0.584	0.667	0.678	0.718	0.635
	2	0.642	0.687	0.748	0.753	0.794	0.725
	3	0.609	0.669	0.741	0.762	0.790	0.714
	1 e 2	0.788	0.827	0.894	0.893	0.940	0.868
	1, 2 e 3	0.874	0.907	0.956	0.956	0.977	0.934

Tabela 5.6: Cobertura de falhas obtida pelos 3 melhores desvios exatos.

Num. Nodos	Desvio(s) Utilizado(s)	Cobertura de falhas por Intervalo de Grau Médio dos Nodos e Geral					
		$[3,4)$	$[4,5)$	$[5,6)$	$[6,7)$	$[7,8]$	Geral
10	1	0.821	0.910	0.959	0.980	0.998	0.934
	2	0.723	0.802	0.883	0.936	0.996	0.868
	3	0.691	0.774	0.839	0.908	0.979	0.838
	1 e 2	0.917	0.968	0.993	0.995	0.998	0.974
	1, 2 e 3	0.960	0.984	0.997	0.999	0.999	0.988
20	1	0.770	0.821	0.855	0.887	0.908	0.848
	2	0.725	0.782	0.823	0.842	0.854	0.805
	3	0.692	0.770	0.821	0.837	0.865	0.797
	1 e 2	0.896	0.927	0.952	0.969	0.972	0.943
	1, 2 e 3	0.939	0.958	0.979	0.987	0.991	0.971
30	1	0.743	0.805	0.836	0.873	0.882	0.828
	2	0.705	0.775	0.814	0.843	0.862	0.800
	3	0.696	0.764	0.810	0.847	0.861	0.796
	1 e 2	0.872	0.919	0.940	0.961	0.969	0.932
	1, 2 e 3	0.922	0.953	0.970	0.983	0.987	0.963
40	1	0.702	0.788	0.843	0.864	0.881	0.816
	2	0.691	0.771	0.817	0.845	0.862	0.797
	3	0.687	0.761	0.813	0.840	0.862	0.793
	1 e 2	0.842	0.909	0.944	0.958	0.968	0.924
	1, 2 e 3	0.905	0.948	0.970	0.980	0.986	0.958
50	1	0.724	0.791	0.834	0.861	0.879	0.818
	2	0.697	0.769	0.812	0.840	0.858	0.795
	3	0.683	0.764	0.804	0.837	0.858	0.789
	1 e 2	0.860	0.910	0.936	0.954	0.964	0.925
	1, 2 e 3	0.911	0.950	0.965	0.978	0.983	0.957
60	1	0.712	0.780	0.829	0.859	0.876	0.811
	2	0.694	0.757	0.813	0.838	0.858	0.792
	3	0.684	0.755	0.806	0.842	0.861	0.790
	1 e 2	0.852	0.902	0.934	0.952	0.960	0.920
	1, 2 e 3	0.907	0.944	0.966	0.977	0.981	0.955
70	1	0.713	0.784	0.827	0.863	0.881	0.813
	2	0.689	0.770	0.805	0.844	0.861	0.794
	3	0.689	0.760	0.801	0.840	0.859	0.790
	1 e 2	0.852	0.908	0.934	0.953	0.964	0.922
	1, 2 e 3	0.904	0.948	0.966	0.977	0.982	0.956
100	1	0.729	0.766	0.832	0.844	0.884	0.811
	2	0.706	0.758	0.817	0.841	0.861	0.797
	3	0.708	0.756	0.811	0.835	0.855	0.793
	1 e 2	0.863	0.894	0.938	0.942	0.967	0.921
	1, 2 e 3	0.914	0.941	0.970	0.973	0.984	0.956

Tabela 5.7: Cobertura de falhas obtida pelos 3 melhores desvios exatos não pertencentes ao caminho.

Num. Nodos	Desvio(s) Utilizado(s)	Cobertura de falhas por Intervalo de Grau Médio dos Nodos e Geral					
		$[3,4)$	$[4,5)$	$[5,6)$	$[6,7)$	$[7,8]$	Geral
10	1	0.427	0.654	0.810	0.923	0.977	0.758
	2	0.599	0.704	0.782	0.857	0.945	0.777
	3	0.607	0.687	0.780	0.838	0.938	0.770
	1 e 2	0.809	0.938	0.979	0.995	0.999	0.944
	1, 2 e 3	0.914	0.974	0.994	0.998	1.000	0.976
20	1	0.346	0.502	0.575	0.667	0.755	0.569
	2	0.509	0.645	0.703	0.758	0.795	0.682
	3	0.576	0.661	0.715	0.752	0.781	0.697
	1 e 2	0.675	0.826	0.886	0.930	0.959	0.855
	1, 2 e 3	0.844	0.923	0.959	0.972	0.983	0.936
30	1	0.344	0.466	0.539	0.650	0.659	0.531
	2	0.484	0.637	0.675	0.742	0.777	0.663
	3	0.542	0.665	0.711	0.744	0.783	0.689
	1 e 2	0.623	0.795	0.850	0.910	0.938	0.823
	1, 2 e 3	0.791	0.907	0.941	0.965	0.979	0.917
40	1	0.352	0.471	0.544	0.616	0.642	0.525
	2	0.479	0.628	0.687	0.739	0.767	0.660
	3	0.535	0.658	0.714	0.752	0.783	0.689
	1 e 2	0.603	0.779	0.847	0.899	0.923	0.810
	1, 2 e 3	0.765	0.893	0.935	0.964	0.975	0.906
50	1	0.354	0.468	0.533	0.617	0.657	0.526
	2	0.462	0.618	0.661	0.729	0.758	0.646
	3	0.533	0.668	0.696	0.745	0.772	0.683
	1 e 2	0.579	0.767	0.822	0.885	0.916	0.794
	1, 2 e 3	0.749	0.894	0.924	0.954	0.971	0.898
60	1	0.392	0.451	0.551	0.610	0.644	0.529
	2	0.487	0.612	0.671	0.730	0.755	0.651
	3	0.538	0.673	0.699	0.752	0.774	0.687
	1 e 2	0.604	0.747	0.821	0.881	0.907	0.792
	1, 2 e 3	0.750	0.881	0.925	0.954	0.965	0.895
70	1	0.384	0.463	0.573	0.615	0.649	0.537
	2	0.468	0.608	0.683	0.731	0.755	0.649
	3	0.523	0.666	0.704	0.761	0.776	0.686
	1 e 2	0.584	0.743	0.826	0.882	0.905	0.788
	1, 2 e 3	0.733	0.876	0.920	0.954	0.967	0.890
100	1	0.434	0.470	0.574	0.580	0.678	0.547
	2	0.535	0.615	0.689	0.716	0.764	0.663
	3	0.575	0.673	0.728	0.754	0.779	0.702
	1 e 2	0.649	0.740	0.826	0.856	0.906	0.795
	1, 2 e 3	0.782	0.871	0.928	0.943	0.964	0.898

Tabela 5.8: Cobertura de falhas obtida pelos 3 melhores desvios heurísticos.

Num. Nodos	Desvio(s) Utilizado(s)	Cobertura de falhas por Intervalo de Grau Médio dos Nodos e Geral					
		$[3,4)$	$[4,5)$	$[5,6)$	$[6,7)$	$[7,8]$	Geral
10	1	0.834	0.917	0.972	0.991	0.994	0.942
	2	0.720	0.810	0.902	0.954	0.983	0.874
	3	0.674	0.772	0.831	0.910	0.967	0.831
	1 e 2	0.915	0.970	0.991	0.996	0.999	0.974
	1, 2 e 3	0.958	0.985	0.997	0.999	1.000	0.988
20	1	0.771	0.816	0.864	0.899	0.926	0.855
	2	0.725	0.782	0.826	0.846	0.867	0.809
	3	0.683	0.771	0.815	0.830	0.857	0.791
	1 e 2	0.884	0.919	0.953	0.965	0.976	0.939
	1, 2 e 3	0.931	0.956	0.980	0.983	0.991	0.968
30	1	0.729	0.804	0.841	0.873	0.910	0.831
	2	0.712	0.780	0.819	0.845	0.869	0.805
	3	0.684	0.755	0.810	0.844	0.854	0.790
	1 e 2	0.850	0.910	0.939	0.955	0.973	0.925
	1, 2 e 3	0.902	0.944	0.968	0.980	0.989	0.956
40	1	0.705	0.787	0.840	0.871	0.894	0.819
	2	0.695	0.771	0.821	0.850	0.868	0.801
	3	0.674	0.747	0.809	0.839	0.862	0.786
	1 e 2	0.827	0.898	0.932	0.957	0.968	0.916
	1, 2 e 3	0.883	0.938	0.962	0.979	0.985	0.949
50	1	0.704	0.793	0.832	0.861	0.883	0.815
	2	0.703	0.776	0.819	0.846	0.866	0.802
	3	0.682	0.747	0.808	0.837	0.865	0.788
	1 e 2	0.822	0.901	0.928	0.949	0.964	0.913
	1, 2 e 3	0.881	0.940	0.959	0.973	0.983	0.947
60	1	0.695	0.787	0.830	0.865	0.883	0.812
	2	0.692	0.769	0.825	0.851	0.870	0.801
	3	0.679	0.745	0.815	0.838	0.866	0.789
	1 e 2	0.816	0.893	0.929	0.950	0.959	0.909
	1, 2 e 3	0.875	0.934	0.961	0.974	0.980	0.945
70	1	0.693	0.787	0.818	0.868	0.886	0.811
	2	0.694	0.772	0.808	0.854	0.872	0.800
	3	0.682	0.749	0.799	0.842	0.864	0.787
	1 e 2	0.813	0.891	0.918	0.950	0.962	0.907
	1, 2 e 3	0.874	0.933	0.955	0.973	0.981	0.943
100	1	0.719	0.776	0.832	0.859	0.876	0.812
	2	0.710	0.760	0.825	0.854	0.867	0.803
	3	0.703	0.736	0.812	0.836	0.862	0.790
	1 e 2	0.836	0.882	0.929	0.945	0.957	0.910
	1, 2 e 3	0.893	0.926	0.961	0.969	0.978	0.945

Tabela 5.9: Cobertura de falhas obtida pelos 3 melhores desvios heurísticos não pertencentes ao caminho.

Capítulo 6

Conclusão

Este trabalho apresentou uma abordagem para roteamento tolerante a falhas baseado em desvios de alta conectividade. O objetivo principal é permitir que os nodos da rede continuem a se comunicar, mesmo durante o período de tempo em que as tabelas de rotas dos roteadores ainda não foram atualizadas para refletir uma falha ocorrida na rede. Para este propósito, rotas alternativas que possuem grande probabilidade de desviar de uma falha ocorrida na rede são utilizadas. Rotas alternativas são criadas através de nodos chamados desvios, que pertencem a componentes de alta conectividade. O número de caminhos distintos que atravessam um componente de maior conectividade é maior, se comparado a um componente de menor conectividade. Desta forma, é maior a probabilidade da rota alternativa criada através de um desvio pertencente a um componente de maior conectividade ser funcional, mesmo na presença de falhas na rede. Os critérios de conectividade $\#C(v)$ e $MCC(v)$, que indicam a conectividade de cada nodo da rede, são utilizados na seleção do melhor desvio para um par de nodos. Os critérios de conectividade $\#C(v)$ e $MCC(v)$ podem ser calculados utilizando uma árvore de Gomory-Hu ou árvore de corte. Além disso, o critério de conectividade $\#C(v)$ pode ser estimado utilizando uma heurística que possui ordem de complexidade linear, evitando assim a ordem de complexidade necessária para a construção da árvore de corte. Os algoritmos de cálculo dos critérios de conectividade $\#C(v)$ e $MCC(v)$ utilizando a árvore de corte e a heurística proposta, e o algoritmo de seleção do melhor desvio foram implementados na linguagem Java. Um al-

goritmo que realiza o cálculo da cobertura de falhas quando o primeiro, segundo e terceiro melhores desvios são utilizados também foi implementado, assim como um algoritmo que realiza uma comparação do $\#C(v)$ médio exato e do $\#C(v)$médio heurístico. Resultados experimentais foram obtidos utilizando grafos aleatórios de diferentes tamanhos e quantidade de arestas, gerados pelo método Waxman, que se propõe a capturar a característica de localidade existente nas redes reais. Os resultados experimentais demonstram que a heurística obtém valores de $\#C(v)$ próximo aos valores obtidos pelo algoritmo exato. A cobertura de falhas obtida pelo melhor desvio não pertencente ao caminho pode chegar a 90%. Quando os três melhores desvios não pertencentes ao caminho são utilizados, a cobertura de falhas pode ser superior a 98%. A cobertura de falhas obtida pelos melhores desvios heurísticos não pertencentes ao caminho apresentou valores similares. Estes resultados visam comprovar que rotas alternativas criadas através de desvios selecionados a partir dos critérios de conectividade propostos possuem boa probabilidade de desviar de uma falha ocorrida na rede, mesmo sem a informação da localização da falha. Na comparação com uma estratégia de seleção aleatória de desvios, a cobertura de falhas obtida pela estratégia de seleção de desvios baseada nos critérios de conectividade propostos foi sempre superior. Em tal comparação, a estratégia de seleção de desvios baseada nos critérios de conectividade propostos fez maior diferença em grafos esparsos, que são mais próximos das topologias utilizadas na prática.

Trabalhos futuros incluem o desenvolvimento de um algoritmo que selecione a rota alternativa de maior conectividade, considerando todos os nodos da rota. A especificação de um protocolo de roteamento que seleciona a melhor rota para um determinado destino baseando-se nos critérios de conectividade propostos, e sua implementação em uma rede real são também metas para desenvolvimento futuro.

Referências Bibliográficas

[1] Y. Rekhter. Application of the Border Gateway Protocol in the Internet, RFC-1772, 1995.

[2] Y. Rekhter. A Border Gateway Protocol 4 (BGP-4), RFC-1771, 1995.

[3] D. E. Comer. *Internetworking with TCP/IP*, volume 1. Prentice-Hall Inc., fourth edition, 2000.

[4] C. Hedrick. Routing Information Protocol, RFC-1058, 1988.

[5] E. W. Dijkstra. A Note on Two Problems in Connection with Graphs. In *Numerische Mathematic*, páginas 269–271, 1959.

[6] J. Moy. OSPF Version 2, RFC-1583, 1994.

[7] Christian Huitema. *Routing in the Internet.* Prentice Hall PTR, second edition, 1999.

[8] Craig Labovitz, Abha Ahuja, Abhijit Bose, e Farnam Jahanian. Delayed Internet Routing Convergence. In *SIGCOMM*, páginas 175–187, 2000.

[9] Jaime Cohen e E.P. Duarte Jr. Fault-Tolerant Routing of TCP/IP PDU's on General Tolopogy Backbones. In *Third International Workshop on Design of Reliable Communication Networks*, 2001.

[10] R. E. Gomory e T. C. Hu. Multi-Terminal Network Flows. *SIAM Journal on Applied Mathematics*, páginas 9:551–556, 1961.

[11] Ellen W. Zegura, Kenneth L. Calvert, e Michael J. Donahoo. A Quantitative Comparison of Graph-Based Models for Internet Topology. *IEEE/ACM Transactions on Networking*, 5(6):770–783, 1997.

[12] C. Jin, Q. Chen, e S. Jamin. Inet: Internet Topology Generator, Technical Report CSE-TR443-00, Department of EECS, University of Michigan, 2000.

[13] J. Winick e S. Jamin. Inet-3.0: Internet Topology Generator, Technical Report CSE-TR456-02, Department of EECS, University of Michigan, 2002.

[14] Michalis Faloutsos, Petros Faloutsos, e Christos Faloutsos. On Power-law Relationships of the Internet Topology. In *SIGCOMM*, páginas 251–262, 1999.

[15] The Source for Java Technology. http://java.sun.com, acessado em 01/07/2003.

[16] D. Walend. JDigraph. http://jdigraph.sourceforge.net, acessado em 20/04/2003.

[17] Sam Halabi e Danny McPherson. *Interdomain Routing Architecture*. Cisco Press, second edition, 2001.

[18] Y. Rekhter. A Border Gateway Protocol 4 (BGP-4), draft-ietf-idr-bgp4-17.txt, 2002.

[19] D. L. Mills. Exterior Gateway Protocol Formal Specification, RFC-904, 1984.

[20] Craig Labovitz, Abha Ahuja, Roger Wattenhofer, e Venkatachary Srinivasan. The Impact of Internet Policy and Topology on Delayed Routing Convergence. In *INFOCOM*, páginas 537–546, 2001.

[21] Timothy Griffin e Gordon T. Wilfong. An Analysis of BGP Convergence Properties. In *SIGCOMM*, páginas 277–288, 1999.

[22] Kannan Varadhan, Ramesh Govindan, e Deborah Estrin. Persistent Route Oscillations in Inter-Domain Routing. *Computer Networks (Amsterdam, Netherlands: 1999)*, 32(1):1–16, 2000.

[23] E.P. Duarte Jr., G. Mansfield, S. Noguchi, e M. Miyazaki. Fault-Tolerant Network Management. In *ISACC'94*, 1994. Monterrey, Mexico.

[24] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, e J. Zahorjan. Detour: Informed Internet Routing and Transport. *IEEE Micro*, 1999.

[25] T. H. Cormen, C. E. Leiserson, e R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill, second edition, 1990.

[26] F. Harary. *Graph Theory*. Perseus Books Publishing, 1969.

[27] J. Evans e E. Minieka. *Optimization Algorithms for Networks and Graphs*. Dekker, Monticello, second edition, 1992.

[28] D. Gusfield. Very Simple Method for All Pairs Network Flow Analisys. *SIAM Journal on Computing*, páginas 19(1):143–155, 1990.

[29] L. R. Ford e D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[30] Chandra Chekuri, Andrew V. Goldberg, David R. Karger, Matthew S. Levine, e Clifford Stein. Experimental Study of Minimum Cut Algorithms. In *Symposium on Discrete Algorithms*, páginas 324–333, 1997.

[31] Jonatan Schroeder, Nortan P. Molina Jr., e Pedro R. Torres Jr. Busca de Desvios em Redes Baseada nos Critérios de Conectividade #C(v) e MCC(v), 2003.

[32] S. Khuller e B. Raghavachari. Improved Approximation Algorithms for Uniform Connectivity Problems. *Journal of Algorithms*, 21, 1996.

[33] P. O. B. Netto. *Grafos, Teorias, Modelos, Algoritmos*. Editora Edgard Blücher Ltda, second edition, 1996.

[34] Mapa do Backbone RNP2. http://www.rnp.br/backbone/bkb-mapa.html, acessado em 20/07/2002.

[35] Bernard M. Waxman. Routing of Multipoint Connections. In *IEEE Journal of Selected Areas in Communications/6(9)*, páginas 1617–1622, 1988.

[36] Reinhard Diestel. *Graph Theory.* Springer-Verlog, second edition, 2000.

[37] Georgia Tech Internet Topology Models (GT-ITM). http://www.cc.gatech.edu/projects/gtitm/, acessado em 03/03/2003.

[38] Inet Topology Generator. http://topology.eecs.umich.edu/inet/, acessado em 23/11/2002.

[39] R. Santini. Implementação Java da Estratégia de Seleção de Desvios. http://www.inf.ufpr.br/~elias, acessado em 01/08/2003.

[40] Dale Dougherty e Arnold Robbins. *Sed & Awk.* O'Reilly & Associates, second edition, 1997.

Anexo 1

Script rand.sh utilizado na geração de números reais aleatórios

```
#!/bin/sh

# Script rand.sh para a geracao de numeros reais randomicos.
# Parametros:
# 1: Quantidade de numeros que serao gerados
# 2: Semente
# 3: Limite inferior
# 4: Limite superior

awk '
BEGIN {
   srand('$2');
   for (i=0;i<'$1';i++) {
      if ('$4'!=0)
         printf ('$3'+(rand()*('$4'-'$3')))" "
      else
         printf rand()" "
   }
   print
} '
```

Script rand_seed.sh utilizado na geração de números inteiros aleatórios

```
#!/bin/sh

# Script rand_seed.sh para a geracao de numeros inteiros randomicos.
# Parametros:
# 1: Quantidade de numeros que serao gerados
# 2: Semente
# 3: Limite inferior
# 4: Limite superior

awk '
BEGIN {
   srand('$2');
   for (i=0;i<'$1';i++) {
```

```
      if ('$4'!=0)
         printf ("%d ",'$3'+(rand()*('$4'-'$3')))
      else
         printf rand()" "
   }
   print
} '
```

Script gera_grafo.sh utilizado na geração de grafos aleatórios utilizando o método Waxman

```
#!/bin/sh

# Programa gerador de grafos usando o metodo waxman
# Parametros:
# 1: num de nodos do grafo
# 2: Numero de grafos de cada intervalo que serao gerados
# 3: arquivo com valores de alfa
# 4: arquivo com valores de beta
# 5: arquivo com valores da semente
# 6: diretorio onde serao colocados os arquivos contendo os grafos
#    gerados

# Obtem Parametros
nodos=$1
numGrafos=$2
alfaFile=$3
betaFile=$4
seedFile=$5
diretorio=$6

# OUTRAS VARIAVEIS UTILIZADAS
MAX_TIMEOUT=5

# Joga valores de alfa e beta em arrays
read -a alfaArray < $alfaFile
read -a betaArray < $betaFile
read -a seedArray < $seedFile

alfaLimit=${#alfaArray[*]}
betaLimit=${#betaArray[*]}
seedLimit=${#seedArray[*]}

i=0
grafosgrau3=0
grafosgrau4=0
grafosgrau5=0
```

```
grafosgrau6=0
grafosgrau7=0
while (( (($grafosgrau3 < $numGrafos) || ($grafosgrau4 < $numGrafos) ||
  ($grafosgrau5 < $numGrafos) || ($grafosgrau6 < $numGrafos) ||
  ($grafosgrau7 < $numGrafos)) && ($i < $alfaLimit) && ($i < $betaLimit)
  && ($i < $seedLimit) ))
do
  a=${alfaArray[$i]}
  b=${betaArray[$i]}
  s=${seedArray[$i]}

  # Gera arquivo utilizado pelo itm e roda o itm
  echo "geo 1 $s" > grafo-$nodos-1-$s-$a-$b.itm
  echo $nodos" "$nodos" 1 "$a" "$b >> grafo-$nodos-1-$s-$a-$b.itm
  ./itm grafo-$nodos-1-$s-$a-$b.itm &

  # Mata o processo se o mesmo demorar 10 segundos para executar
  timeout=0;
  pid=`ps -ef | grep grafo-$nodos-1-$s-$a-$b.itm | grep -v grep |
       awk '{print $2}'`
  while (( `ps -ef | grep grafo-$nodos-1-$s-$a-$b.itm | grep -v grep |
    wc -l` && $timeout < $MAX_TIMEOUT ))
  do
     sleep 1;
     if (( timeout == $(($MAX_TIMEOUT - 1)) ))
     then
        kill -9 $pid
        echo "Timeout no itm - alfa beta: "$a" "$b
     fi
     timeout=$(($timeout+1))
  done

  if (( $timeout != $MAX_TIMEOUT ))
  then
      ./sgb2alt grafo-$nodos-1-$s-$a-$b.itm-0.gb \
         grafo-$nodos-1-$s-$a-$b.itm-0.alt

      # Gera arquivo definitivo que eh entendido pela aplicacao
      echo "# geo 1 $s" > grafo-$nodos-1-$s-$a-$b.txt
      echo "# "$nodos" "$nodos" 1 "$a" "$b >> \
         grafo-$nodos-1-$s-$a-$b.txt
      ./alt2graph.sh grafo-$nodos-1-$s-$a-$b.itm-0.alt >> \
         grafo-$nodos-1-$s-$a-$b.txt

      # Verifica o grau medio dos nodos estah entre valores
      # preh-estabelecidos
      edges=`wc -l grafo-$nodos-1-$s-$a-$b.txt | awk '{print $1}'`
      edges=$(($edges-2))
```

```
      if (( $(( ($edges>=((3*$nodos)/2)) && ($edges<((4*$nodos)/2)) \
         && ($grafosgrau3 < $numGrafos) )) ))
      then
         # Move arquivo para o diretorio e incrementa o contador.
         mv grafo-$nodos-1-$s-$a-$b.txt $diretorio
         echo $nodos" "$edges" 1 "$s" "$a" "$b
         grafosgrau3=$(($grafosgrau3+1))
      elif (( $(( ($edges>=((4*$nodos)/2)) && ($edges<((5*$nodos)/2)) \
        && ($grafosgrau4 < $numGrafos) )) ))
      then
         # Move arquivo para o diretorio e incrementa o contador.
         mv grafo-$nodos-1-$s-$a-$b.txt $diretorio
         echo $nodos" "$edges" 1 "$s" "$a" "$b
         grafosgrau4=$(($grafosgrau4+1))
      elif (( $(( ($edges>=((5*$nodos)/2)) && ($edges<((6*$nodos)/2)) \
         && ($grafosgrau5 < $numGrafos) )) ))
      then
         # Move arquivo para o diretorio e incrementa o contador.
         mv grafo-$nodos-1-$s-$a-$b.txt $diretorio
         echo $nodos" "$edges" 1 "$s" "$a" "$b
         grafosgrau5=$(($grafosgrau5+1))
      elif (( $(( ($edges>=((6*$nodos)/2)) && ($edges<((7*$nodos)/2)) \
         && ($grafosgrau6 < $numGrafos) )) ))
      then
         # Move arquivo para o diretorio e incrementa o contador.
         mv grafo-$nodos-1-$s-$a-$b.txt $diretorio
         echo $nodos" "$edges" 1 "$s" "$a" "$b
         grafosgrau6=$(($grafosgrau6+1))
      elif (( $(( ($edges>=((7*$nodos)/2)) && ($edges<=((8*$nodos)/2)) \
         && ($grafosgrau7 < $numGrafos) )) ))
      then
         # Move arquivo para o diretorio e incrementa o contador.
         mv grafo-$nodos-1-$s-$a-$b.txt $diretorio
         echo $nodos" "$edges" 1 "$s" "$a" "$b
         grafosgrau7=$(($grafosgrau7+1))
      fi
      echo "grau3:"$grafosgrau3" grau4:"$grafosgrau4" \
      grau5:"$grafosgrau5" grau6:"$grafosgrau6" grau7:"$grafosgrau7
   fi
   rm -f grafo-$nodos-1-$s-$a-$b.txt grafo-$nodos-1-$s-$a-$b.itm \
   grafo-$nodos-1-$s-$a-$b.itm-0.gb grafo-$nodos-1-$s-$a-$b.itm-0.alt
   i=$(($i+1))
done
echo "Tentativa de geracao: "$i
```

Script alt2graph.sh utilizado na conversão do formato dos arquivos de grafos

```
#!/bin/bash
# uso: alt2graph.sh <arquivo>
COMECA=`grep -w EDGES $1 -n | awk -F: '{ print $1 }'`
TOTAL=`wc -l $1 | awk '{ print $1 }'`
LINHAS=$(($TOTAL-$COMECA))
tail -n $LINHAS $1 | awk '{ print $1" " $2}'
```