Universidade Federal Fluminense

IVAIRTON MONTEIRO SANTOS

Algoritmos Aproximados para o Problema do Maior Conjunto Controlado Generalizado

NITERÓI

2005

Universidade Federal Fluminense

IVAIRTON MONTEIRO SANTOS

Algoritmos Aproximados para o Problema do Maior Conjunto Controlado Generalizado

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização Combinatória e Inteligência Artificial.

Orientador:
Carlos Alberto de J. Martinhon

Co-orientador:
Luiz Satoru Ochi

NITERÓI

2005

Algoritmos Aproximados para o Problema do Maior Conjunto Controlado Generalizado

Ivairton Monteiro Santos

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre.

Aprovada por:

Prof. D.Sc. Carlos Alberto de Jesus Martinhon / IC-UFF
(Presidente)

Prof. D.Sc. Luiz Satoru Ochi / IC-UFF

Prof^a. D.Sc. Simone de Lima Martins / IC-UFF

Prof. D.Sc. Fabio Protti / UFRJ

Prof. D.Sc. Luerbio Faria / UERJ

Niterói, 22 de Julho de 2005.

À minha saudosa mãe, Ivani.

Agradecimentos

Agradeço em primeiro lugar ao nosso Senhor Deus que nos deu o maior de todos os dons, uma força divina que nos impulsiona a enfrentar qualquer desafio, que é a vida.

Obrigado ao professor Carlos Martinhon que soube como ninguém desempenhar o papel de orientador, tendo paciência, acreditando no meu trabalho e sabendo me direcionar a solucionar problemas que em algumas vezes nem eu mesmo sabia ser capaz. Ao professor Luiz Satoru, pelas explicações preciosas na área de metaheurísticas e pelos incentivos, sempre me empurrando para a frente. Aos dois muito obrigado pela amizade.

Agradeço às duas pessoas mais importantes na minha vida, ao meu pai José Airton e a minha princesa Fernanda Dalla. Ao meu pai por sempre me apoiar incondicionalmente em qualquer coisa que eu faça, mesmo que possa não ser do seu agrado. Sempre com palavras motivadoras, amigas e de aconchego, será sempre meu "porto seguro". Meu muito obrigado à pessoa mais linda que já conheci, dona de virtudes admiráveis, com um coração que desobedece as leis da física por ser tão grande e caber em um corpo pequenino. Sempre ofereceu seu carinho, amizade e amor sem esperar nada em troca, companheira em toda e qualquer situação, obrigado Fernanda, amo você.

Agradeço aos amigos que me acompanharam nessa caminha do mestrado, "Gaúcho" (Luis Rigo) grande amigo, presente em todas as horas, dono de dicas valiosas e idéias mirabolantes, como ir à praia numa véspera de prova. Obrigado à Luciana Pessoa, companheira desde o início da caminhada, sempre oferecendo o inigualável biscoito goiabinha, fonte de glicose para passarmos pelas disciplinas e desenvolver a dissertação. Obrigado à Adria, pelas dicas e idéias de valor incalculável na área de programação linear, como poderia ter passado sem elas? Ao meu amigo "Chups" (Stênio Sã), sempre com suas críticas muito bem vindas e idéias preciosas. Obrigado à "Renatinha" (Renatha Cappua), com soluções para os problemas mais cabeludos em Latex. Meu muito obrigado a todos aqueles que em algum momento me ajudaram seja com os códigos, idéias, e-mails, críticas e amizade, sei que sempre estiveram na torcida pelo meu sucesso, obrigado Geiza, Haroldo, "Super Jackes" (Jacques), Daniela, Viviane Thomé, Viviane Trindade, Luciana Brugiolo.

Resumo

Dado um grafo $G = (V, E)$ e um conjunto de vértices $M \subseteq V$, sendo $|V| = n$, dizemos que o vértice $v \in V$ é *controlado* por M se a maioria dos seus vértices adjacentes (incluindo ele mesmo) pertencem ao conjunto M. O conjunto M define um *monopólio* em G se M controla todos os vértices de V. Dado um conjunto $M \subseteq V$ e dois grafos $G_1 = (V, E_1)$ e $G_2 = (V, E_2)$, onde $E_1 \subseteq E_2$, temos o Problema de Verificação de Monopólio - PVM, que consiste em encontrar um grafo sanduíche $G = (V, E)$ (i.e., um grafo onde $E_1 \subseteq E \subseteq E_2$), tal que M defina um monopólio em G. Caso a resposta do PVM seja "não", temos então o Problema do Maior Conjunto Controlado - PMCC, cujo objetivo é encontrar um grafo sanduíche $G = (V, E)$, tal que o número de vértices de G controlados por M seja maximizado. O PVM pode ser resolvido em tempo polinomial, entretanto, o PMCC é NP-difícil. Neste trabalho apresentamos a noção de *f-controle* e introduzimos ainda o Problema do Maior Conjunto Controlado Generalizado (PMCCG), que associa pesos positivos e folgas mínimas a cada vértice de V. Nesse caso, o objetivo será maximizar o somatório dos pesos dos vértices *f-controlados* por M. Apresentamos um algoritmo $\frac{1}{2}$-aproximado para o PMCCG e um procedimento para a geração de soluções viáveis baseado na solução de uma relaxação linear para o problema. Essas soluções são utilizadas posteriormente em um método de busca local (Busca Tabu com Reconexão por Caminhos) visando a determinação de soluções de melhor qualidade para o PMCCG. Finalmente, apresentamos alguns resultados computacionais e comparamos os resultados obtidos com o valor ótimo encontrado para pequenas instâncias do problema.

Palavras chaves: Grafo sanduíche, Algoritmos Aproximados, Heurísticas Construtivas, Busca Tabu, Reconexão por Caminhos

Abstract

Given a graph $G = (V, E)$ and a set of vertices $M \subseteq V$, with $|V| = n$, we say that $v \in V$ is *controlled* by M, if the majority of v neighbors (including itself) belongs to M. The set M defines a *monopoly* in G if M controls all vertices of V. In the *Monopoly Verification Problem* - MVP, we are given a set $M \subseteq V$ and two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, with $E_1 \subseteq E_2$. The objective is to find a sandwich graph $G = (V, E)$ (i.e., a graph where $E_1 \subseteq E \subseteq E_2$), such that M defines a monopoly in G. However, if the answer to the MVP is "NO", we have the *Max-Controlled Set Problem* - MCSP, whose objective is to find a sandwich graph $G = (V, E)$, such that the total number of controlled vertices is maximized. The MVP can be solved in polynomial time, the MCSP, however is NP-hard. In this work we describe the notion of *f-controlled* vertices and introduce the *Generalized Max-Controlled Set Problem* - GMCSP, where positive weights and gaps are associated to all vertices of V. In this case, the objective is to maximize the summation of the weights of all vertices f-controlled by M. We present a $\frac{1}{2}$-approximation algorithm for the GMCSP and a new procedure for finding feasible solutions based on the solutions of a linear programming relaxation. These solutions are then used in a local search procedure (Tabu Search with Path Relinking) looking for solutions of better quality. Finally, we present some computational results and we compare these results with the optimum solutions values obtained for small instances of the problem.

Key words: Sandwich Graph Problems, Approximation Algorithms, Construction Heuristics, Tabu Search, Path Relinking.

Glossário

PVM	:	Problema de Verificação de Monopólio
PMCC	:	Problema do Maior Conjunto Controlado
PMCCG	:	Problema do Maior Conjunto Controlado Generalizado
BT	:	Busca Tabu
RC	:	Reconexão por Caminhos
VNS	:	*Variable Neighborhood Search*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Introduzimos neste trabalho o Problema do Maior Conjunto Controlado Generalizado - PMCCG, antes entretanto, apresentamos alguns conceitos básicos importantes, e descrevemos os problemas de Verificação de Monopólio e do Maior Conjunto Controlado, como descrito em Makino *et. al.* [19].

Dados dois grafos $G_1 = (V, E_1)$ e $G_2 = (V, E_2)$ tal que $E_1 \subseteq E_2$, dizemos que $G = (V, E)$, onde $E_1 \subseteq E \subseteq E_2$, é um *grafo sanduíche* com propriedade Π se, e somente se, $G = (V, E)$ satisfaz Π. Um problema de decisão envolvendo grafo sanduíche consiste em decidir se há algum grafo G para o par G_1, G_2 que satisfaça uma propriedade Π. Problemas utilizando grafos sanduíche podem ser vistos em diferentes áreas de pesquisa, como em mapeamento físico de DNA, raciocínio temporal, sincronização de processos paralelos, árvores evolutivas, sistemas esparsos de equações lineares, entre outros [13].

Neste trabalho as arestas pertencentes ao conjunto $E_2 \backslash E_1$ serão denominadas de *arestas optativas*, podendo assumir os estados de "ativada" ou "desativada". Uma aresta será considerada "desativada" quando ela não for selecionada no grafo sanduíche, e "ativada" em caso contrário.

A Figura 1.1 ilustra um grafo sanduíche G de G_1 e G_2 respectivamente. Observe na figura a presença de algumas arestas optativas ((1,2), (1,3), (1,5) e (3,4)) no grafo sanduíche G.

No mapeamento físico de DNA [3], por exemplo, as informações de interseções e não-interseções de pares de segmentos, são obtidas a partir da cadeia de DNA de maneira experimental. O problema consiste em como arranjar os vários segmentos, de modo que seus pares de interseções combinem com os dados experimentais. Na representação utilizando grafos os vértices correspondem aos segmentos, e dois vértices são conectados por uma aresta se os segmentos correspondentes possuem interseções, dessa maneira,

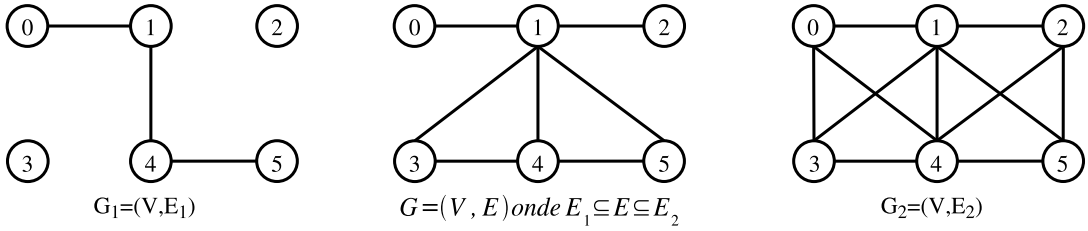


Figura 1.1: Exemplo de grafo sanduíche.

define-se o conjunto de arestas E_2. Na prática as informações de interseções são conhecidas parcialmente, por causa de experimentos incompletos ou resultados não conclusivos. A ambigüidade nas informações de interseções introduz o conjunto de arestas $E_2 \backslash E_1$. Assim, decidir sobre esse problema, é equivalente a encontrar um grafo sanduíche com arestas E, tal que $E_1 \subseteq E \subseteq E_2$ [12].

Dado um grafo não direcionado $G = (V, E)$ e um conjunto de vértices $M \subseteq V$, um vértice $i \in V$ é *controlado* por M se e somente se, $|N_G[i] \cap M| \geq |N_G[i]|/2$, onde $N_G[i] = \{i\} \cup \{j \in V | (i, j) \in E\}$ é *vizinhança fechada* de i, ou seja, um vértice é controlado por M se a maioria dos seus vértices vizinhos (incluindo ele mesmo) pertencem ao conjunto M. O conjunto $M \subseteq V$ define um *monopólio* em G se e somente se, todo vértice $i \in V$ é controlado por M.

De acordo com a notação apresentada por Makino *et. al.* [19], se $Cont(G, M)$ define o conjunto de vértices controlados por M em G, então M será um monopólio em G se, e somente se, $Cont(G, M) = V$. A Figura 1.2 ilustra um grafo com monopólio, todos os vértices do grafo são controlados por M.

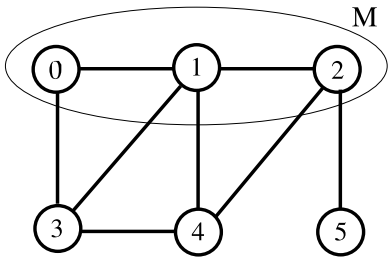


Figura 1.2: Grafo onde o conjunto de vértices pertencentes a M estabelece um monopólio em G.

A noção de monopólio foi introduzida por [18] para descrever o Problema de Maioria Local em um ambiente distribuído. Uma série de problemas práticos importantes envolvendo esses conceitos motivam esse estudo, por exemplo, sincronização de processos paralelos, árvores evolutivas, problemas de acordo em sistemas de agentes, superação de

falhas em computação distribuída, algoritmos com tolerância a falhas, tratamento de redundância ou replicação de dados em banco de dados distribuídos, aplicações em redes sociais, entre outros [2, 4, 13, 18, 22].

Como exemplo, vamos considerar um algoritmo em um sistema de agentes, para definir um padrão industrial entre muitos padrões propostos [4]. Suponha que cada agente conheça os padrões apoiados pelos demais agentes locais. Um acordo possível seria efetuar uma pesquisa entre todos os agentes e adotar o padrão candidato apoiado pela maioria. O sistema de agentes, entretanto, pode possuir características que não permitam que o acordo pela pesquisa global seja praticado. Para superar esse problema, algoritmos heurísticos têm sido sugeridos, baseados em informações parciais (locais) sobre a distribuição de opiniões de apoio dos agentes. Simplificando, considere que existam somente dois padrões propostos, A e B, e que o padrão candidato apoiado pela maioria de agentes seja o padrão selecionado. Se cada agente sabe a opinião do seu agente vizinho, uma heurística natural é fazer com que cada agente concorde com a opinião da maioria dos seus vizinhos, contando obviamente com a sua própria opinião. Isso é chamado de Sistema Determinístico de Votação da Maioria Local [18]. Peleg [2, 18, 22] entre outros, recentemente investigaram esse tipo de sistema e determinaram quantos agentes, que apoiam a proposta A, são necessários e suficientes para estabelecer um acordo com resultado em A. Em seu modelo, o sistema é representado por um grafo não direcionado $G = (V, E)$, onde V e E representam o conjunto de agentes e suas relações de vizinhança, respectivamente. Makino *et. al.* [19] descrevem que o conjunto de todos os agentes irá decidir exclusivamente pela proposta A se e somente se, existir um monopólio $M \subseteq V$ em G. Em sistemas determinísticos de votação da maioria local, assegurar apoio pelos membros do monopólio M, implica em assegurar um acordo unânime, conseqüentemente, os monopólios possuem um papel importante em tais sistemas.

Podemos descrever também outras aplicações de maioria local em diferentes áreas, como discutido em [22]: superação de falhas é um problema central em computação distribuída, a noção de controle da maioria pode ser empregada também nesta situação. A idéia consiste em eliminar danos causados por vértices falhos (seja por processamento incorreto ou processadores que se tornam inativos), ou restringir a influência deles. Isso pode ser feito mantendo cópias replicadas dos dados essenciais e executando uma eleição entre os processadores sempre que ocorrerem falhas, adotando os valores armazenados na maioria dos processadores com dados confiáveis, ou seja, aqueles que possuem dados consistentes frente aos que apresentam erros.

Redundância ou replicação de dados é uma técnica utilizada na área de algoritmos de gerenciamento de base de dados distribuídos e visa evitar falhas no armazenamento dos dados. O conjunto de regras para solução de inconsistências são freqüentemente baseados em um modelo pré definido, entretanto é possível estabelecer um modelo mais adequado à base (global) seguindo um padrão estabelecido pela maioria das bases (locais), priorizando a versão dos dados apoiados pela maioria das cópias disponíveis [22].

Sistemas de votação da maioria local tem aplicação também no contexto de alocação de recursos, como meio de assegurar *exclusão mútua*. Por exemplo, um processador eleito em um determinado grupo de processadores permite a um usuário entrar em uma seção crítica, nenhum outro usuário terá permissão de entrar em seção crítica, mesmo que solicite permissão a qualquer outro processador, sendo (a seção) negada até que o primeiro usuário tenha concluído sua seção [22].

Considere agora um conjunto $M \subseteq V$ e dois grafos $G_1 = (V, E_1)$, $G_2 = (V, E_2)$ onde $E_1 \subseteq E_2$. O Problema de Verificação de Monopólio - PVM é um problema de decisão que retorna "SIM" caso seja encontrando um grafo sanduíche G, tal que todos os vértices de G estejam controlados por M. Makino *et. al.* [19] demonstram que o grafo sanduíche para o PVM pode ser obtido em tempo polinomial por meio da solução de um Problema de Fluxo Máximo [1] definido convenientemente.

Quando não existe um grafo sanduíche tal que o conjunto $M \subseteq V$ define um monopólio, a solução do PVM é "NÃO". Neste caso, nos interessamos pelo Problema do Maior Conjunto Controlado - PMCC, cujo objetivo é encontrar um grafo sanduíche $G = (V, E)$, tal que o conjunto de vértices controlados por M seja maximizado. Assim, desejamos encontrar G, tal que $Cont(G, M)$ tenha a maior cardinalidade possível. O PMCC pertence à classe dos problemas NP-difícil, mesmo que G_1 seja um grafo vazio ou G_2 seja um grafo completo [19].

Os grafos ilustrados na Figura 1.3 não possuem monopólio definido por M, porque a resposta do PVM é "NÃO". No exemplo da Figura 1.3(a), $Cont(G, M)$ é igual a $\{1, 2, 5\}$ e possui cardinalidade 3. Os demais vértices estão não controlados, ou seja, a maioria dos seus vizinhos estão fora de M. Porém a solução dada por $Cont(G, M)$, não corresponde ao maior conjunto controlado. Para encontrar um conjunto de vértices controlados por M de maior cardinalidade, devemos adicionar e/ou remover arestas optativas buscando gerar uma nova configuração que promova o controle do maior número de vértices possível, maximizando $|Cont(G, M)|$. Na Figura 1.3(b) adicionamos a aresta optativa (1,3) e conseguimos uma nova solução onde $|Cont(G, M)|$ igual a 4.

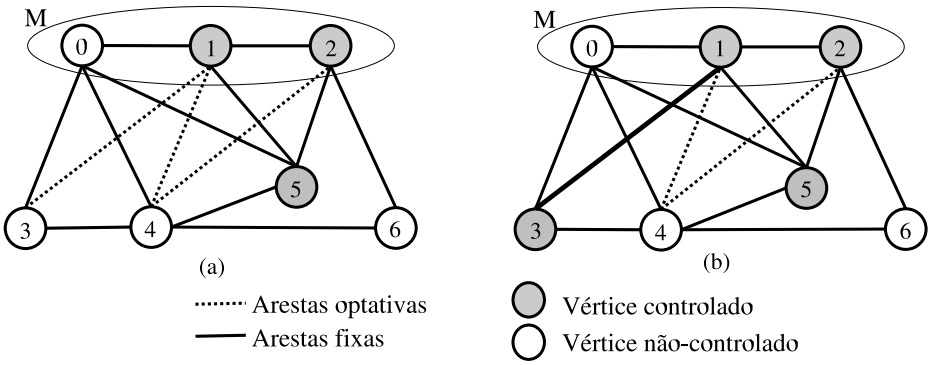


Figura 1.3: Instância do PMCC.

Considere agora a seguinte situação: imagine que cada vértice no grafo G representa um cidadão. No dia de amanhã, os cidadãos de G irão votar entre "SIM/NÃO" para uma questão polêmica. Curiosamente, cada cidadão decidiu por fazer uma pesquisa entre seus vizinhos, incluindo seu próprio voto, para ter um censo particular sobre a eleição do dia seguinte. Para a surpresa, ou desapontamento dos cidadãos que votavam "SIM", o resultado encontrado para cada vértice eleitor de "SIM" mostrava uma vantagem de 2:1 para votos "NÃO" em sua vizinhança.

Como exemplo concreto, considere um grafo com dois votos para "NÃO" de um lado e $n-2$ votos "SIM" no outro, veja na Figura 1.4.

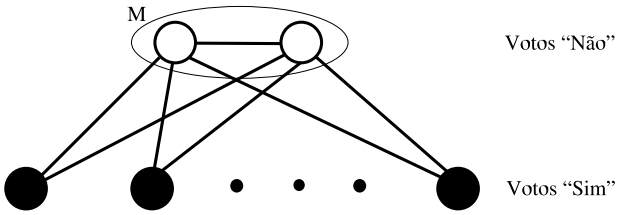


Figura 1.4: Dois vértices "NÃO" dão a impressão de ser a maioria na pesquisa local de todos os outros vértices. Na figura os vértices brancos representam uma coalizão (conjunto M) e os vértices pretos os vértices controlados pela coalizão.

Os eleitores de "SIM" podem se considerar derrotados e os de "NÃO" vitoriosos? Ou poderiam todas as pesquisas estarem erradas? Qual seria o número mínimo de votos "NÃO" que se teria na eleição, dado os resultados das pesquisas?

Se cada eleitor tiver seu voto influenciado pela sua pesquisa particular, na eleição hipotética tem-se uma vitória dos votos "NÃO", mesmo sendo a minoria no momento em que as pesquisas particulares foram realizadas.

A situação apresentada acima caracteriza de certa forma o PMCC, cujo interesse se-

ria promover um "controle" de uma maioria de votantes, dado um sub-conjunto $M \subseteq V$ de vértices, representando por exemplo uma "coalizão" ou "partido político". Poderíamos imaginar, por exemplo, que um projeto importante precisa ser votado em uma câmara legislativa e os vértices controlados por M são aqueles que concordam ou votam favoravelmente a um projeto proposto por M. É de interesse do partido (coalizão M), promover o controle da maioria dos representantes da câmara, ou seja, que o maior número possível de pessoas, dentro ou fora do partido, apóie um determinado projeto. Entretanto, como estabelecer parcerias e/ou contatos para que o partido consiga a maioria das opiniões a favor do seu projeto na câmara?

Pode-se perceber pelo exemplo dado pela Figura 1.4 a importância das conexões. A influência de um vértice é determinada pelo seu grau. O nível de influência de um conjunto de vértices (coalizão) é dado pela organização de suas arestas. O problema de encontrar essa organização de arestas que consiga controlar o maior número de vértices possível pode ser modelado convenientemente por um problema em grafos sanduíche.

Neste trabalho, introduzimos uma extensão do PMCC denominada Problema do Maior Conjunto Controlado Generalizado - PMCCG, que incorpora o conceito de f-controle e considera *pesos* $p_i \in \mathbb{Z}^+$ associados a cada vértice $i \in V$. No PMCCG, o objetivo será maximizar o somatório dos pesos dos vértices f-controlados por M. Para tentar resolver esse problema iremos aplicar técnicas heurísticas e metaheurísticas e tentar estabelecer razões de aproximação para os algoritmos propostos, buscando uma melhor avaliação da qualidade nas soluções encontradas. No Capítulo 2, introduzimos o PMCCG e conceitos básicos, além de apresentarmos as *regras de redução* na Seção 2.1, discutidas em [19, 20], capazes de eliminar informações redundantes nos dados de entrada do PMCC. Como veremos, estas regras serão aplicadas igualmente ao PMCCG, bastando substituir a definição de vértice controlado por f-controlado. Na Seção 2.2 é apresentado um algoritmo determinístico $\frac{1}{2}$-aproximado, na Seção 2.3 um modelo matemático para o problema e na Seção 2.4 um algoritmo para obtenção de soluções viáveis a partir da solução obtida pela relaxação linear. O Capítulo 3 aborda a aplicação da metaheurística Busca Tabu com Reconexão por Caminhos, visando incrementar ainda mais as soluções obtidas pelos algoritmos construtivos. Assim, na Seção 3.1 introduzimos uma busca local intensiva para o PMCCG, além de definirmos um procedimento de diversificação das soluções na Seção 3.2 e a técnica de Reconexão por Caminhos como procedimento de pós-otimização das soluções na Seção 3.3. No Capítulo 4 analisamos e apresentamos alguns resultados computacionais obtidos e a comparação com o exato para pequenas instâncias. Finalmente, no Capítulo 5, apresentamos as conclusões e sugestões de trabalhos futuros.

Capítulo 2

Problema do Maior Conjunto Controlado Generalizado - PMCCG

Consideramos agora a definição de vértice *f-controlado* introduzida por Makino *et. al.* [19]. Dado um grafo $G = (V, E)$ e um subconjunto $M \subseteq V$, um vértice $i \in V$ é f-controlado por $M \subseteq V$ em um grafo G se, e somente se, $|N_G[i] \cap M| - |N_G[i] \cap U| \geq f_i$ onde $f_i \in \mathbb{Z}$ e $U = V \backslash M$. A constante f_i (que chamaremos *folga mínima*) representa, de certa forma, o "grau de controle" ou "grau de convencimento" necessário para que o vértice $i \in V$ seja f-controlado por $M \subseteq V$. Observe por exemplo que, se $f_i = -\infty$ então o vértice i é sempre f-controlado por M. Caso contrário, se $f_i = +\infty$ então i será nunca f-controlado.

A Figura 2.1 ilustra um grafo com folgas mínimas (valores representados entre parênteses) associadas aos vértices. Considerando as folgas mínimas associadas, temos o f-controle de três vértices $\{0, 2, 4\}$, onde $|N_G[i] \cap M| - |N_G[i] \cap U| \geq f_i$. Entretanto, perceba que no exemplo apresentado, se $f_i = 0 \ \forall \ i \in V$, teríamos um monopólio, pois para todo vértice $i \in V$, $|N_G[i] \cap M| \geq |N_G[i]|/2$.

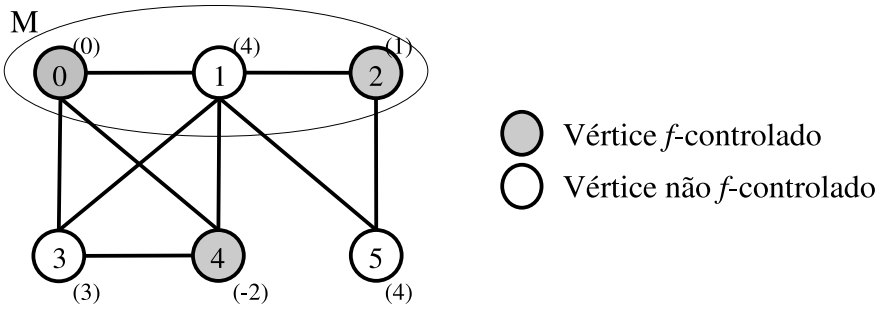


Figura 2.1: Grafo com folgas mínimas associadas aos vértices.

Dizemos que $G = (V, E)$ define um f-monopólio se e somente se, todos os vértices de

V são f-controlados por $M \subseteq V$.

Neste trabalho introduzimos uma extensão do PMCC denominada Problema do Maior Conjunto Controlado Generalizado - PMCCG, que incorpora o conceito de f-controle e considera também *pesos* $p_i \in \mathbb{Z}^+$ associados a cada vértice $i \in V$. A constante $p_i \in \mathbb{Z}^+$ pode representar de certa forma o "grau de importância" de um vértice $i \in V$. No PMCCG, o objetivo será maximizar o somatório dos pesos dos vértices f-controlados por M.

A Figura 2.2 ilustra um exemplo de um grafo com folgas mínimas (entre parênteses) e pesos (representados entre colchetes) associados aos vértices. Na Figura 2.2(a) verifica-se o f-controle de três vértices $\{0, 1, 4\}$ com um custo (somatório dos pesos dos vértices f-controlados) igual a 10. Neste caso, as arestas optativas não foram selecionadas. Já na Figura 2.2(b), com a adição de duas arestas optativas, ocorre uma redução na quantidade de vértices f-controlados. Entretanto, com o f-controle dos vértices $\{4, 5\}$, tem-se um maior custo associado, de valor 12.

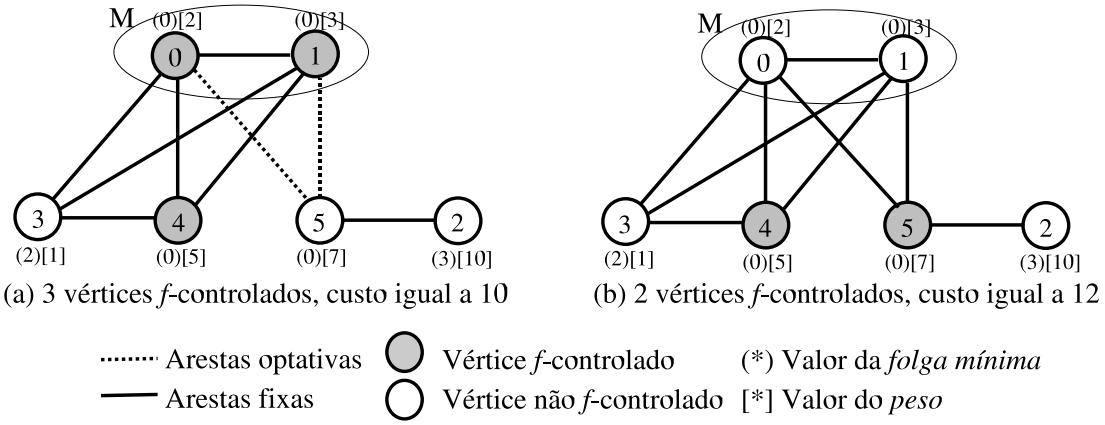


Figura 2.2: Exemplos de grafos com *pesos* e *folgas mínimas* associados aos vértices. O objetivo é maximizar o somatório dos pesos dos vértices f-controlados por M.

Considere novamente a situação apresentada no Capítulo 1, onde temos representados dois partidos políticos, M e $V \backslash M$, votando um projeto proposto por M. É de interesse do partido que o maior número possível de eleitores, dentro ou fora do partido, apóie o projeto. Imagine ainda que cada eleitor considera um número mínimo de votos (diferença entre os membros de dentro e fora de M) para que sua escolha seja efetivada, teríamos então uma "folga mínima" ou "grau de convencimento" para esse eleitor. Por último, imagine que existam eleitores que tenham maior "influência" ou "peso político" (seja um indivíduo com cargo importante ou de uma classe econômica privilegiada, por exemplo) e que tivéssemos o interesse de maximizar o "controle" desses eleitores mais importantes.

Essa situação corresponde à idéia dos pesos e folgas mínimas associados aos vértices do grafo, caracterizando o PMCCG. Observe que, se toda folga mínima for considerada com valor igual a 0 e todo peso associado a um vértice assumir valor igual a 1, temos o PMCC conforme descrito no capítulo anterior.

A seguir iremos apresentar as regras de redução para o problema, discutidas em [19, 20] que eliminam informações redundantes presentes nos dados de entrada do PMCC. Como veremos, estas regras serão aplicadas igualmente ao PMCCG, bastando substituir a definição de vértice controlado por f-controlado.

2.1 Regras de redução

As regras de redução colaboram reduzindo ou eliminando informações redundantes nos dados de entrada, sem interferir no resultado (conjunto de soluções ótimas) do problema original. Essas regras se aproveitam de características presentes na estrutura dos grafos G_1 e G_2, adicionando ou removendo permanentemente (fixando) arestas pertencentes a $E_2 \backslash E_1$ (arestas optativas), de tal forma que o conjunto de soluções ótimas continue o mesmo.

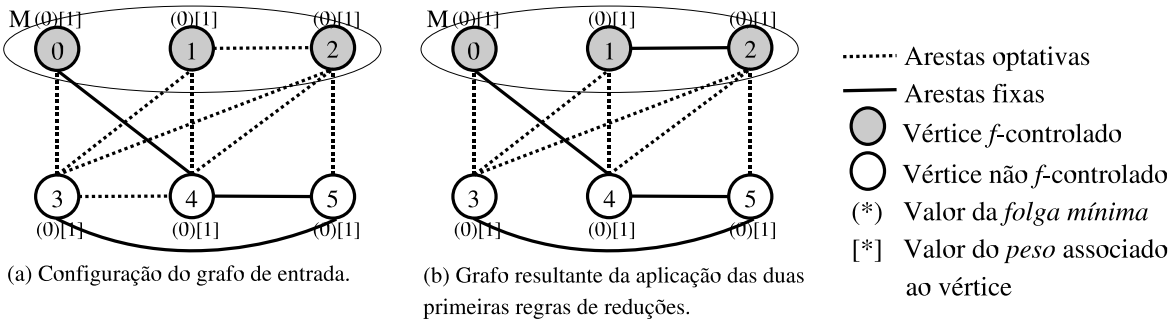
Descreveremos a seguir as regras de redução apresentadas em [19, 20] para o PMCC. As regras de redução serão aplicadas ao PMCCG da mesma maneira que são empregadas no PMCC, bastando substituir a definição de vértice controlado por M, para vértice f-controlado por M. Primeiramente, considere a seguinte notação auxiliar: dados dois grafos G_1 e G_2 como definidos anteriormente e dois conjuntos $A, B \subseteq V$, representamos por $D(A, B) = \{(i, j) \in E_2 \backslash E_1 \mid i \in A,\ j \in B\}$ o conjunto de arestas optativas com uma extremidade em A e outra em B. Temos então as seguintes regras de redução descritas em [19]: adicionar a E_1 as arestas de $D(M, M)$ (*Regra de redução 1*) e remover de E_2 as arestas $D(U, U)$ (*Regra de redução 2*). O novo conjunto E de um grafo sanduíche $G = (V, E)$ deverá satisfazer a:

$$E \supseteq E_1 \cup D(M, M) \tag{2.1}$$

$$E \subseteq E_1 \cup D(M, M) \cup D(U, M) \tag{2.2}$$

A Figura 2.3(a) ilustra um grafo de entrada $G = (V, E)$ qualquer antes da aplicação de qualquer regras de redução. A Figura 2.3(b) demonstra a aplicação das duas primei-

ras regras, as arestas optativas com extremidades em vértices pertencentes ao conjunto M são inseridas no grafo G_1 (aresta (1,2)) conforme (2.1), e as arestas optativas com extremidades em vértices de U são removidas de G_2 (aresta (3,4)), conforme (2.2).



(a) Configuração do grafo de entrada.

(b) Grafo resultante da aplicação das duas primeiras regras de reduções.

Figura 2.3: Aplicação das duas primeiras regras de redução, adicionando e removendo arestas optativas.

Note que as regras 1 e 2 podem ser aplicadas mesmo se tivéssemos instâncias com pesos negativos. Neste caso, se $(i,j) \in D(M,M)$ (analogamente, $(i,j) \in D(U,U)$) com $p_i < 0$ e $p_j < 0$ basta eliminar (i,j) do (respectivamente, adicionar (i,j) ao) conjunto de optativas. A dificuldade surge para aquelas situações onde $p_i \times p_j < 0$. Observe que, dessa forma, se torna difícil decidir sobre a inserção ou remoção da aresta optativa (i,j).

Visando a introdução das regras de redução descritas em [20], considere inicialmente a seguinte partição de V:

- M_{SC} e U_{SC}, vértices *sempre controlados*, consistem dos vértices pertencentes a M e U, respectivamente, que estão controlados em qualquer que seja o grafo sanduíche $G = (V, E)$. Em outras palavras, um vértice $i \in (M_{SC} \cup U_{SC})$ será sempre controlado por M, independentemente de quais arestas optativas $E_2 \backslash E_1$ sejam adicionadas ou removidas em G;

- M_{NC} e U_{NC}, vértices *nunca controlados*, consistem dos vértices pertencentes a M e U, respectivamente, que são sempre não controlados qualquer que seja o grafo sanduíche $G = (V, E)$. Ou seja, um vértice $i \in (M_{NC} \cup U_{NC})$ nunca será controlado por M, independentemente de quais arestas optativas em $E_2 \backslash E_1$ sejam adicionadas ou removidas em G;

- M_R e U_R, definidos por $M_R = M \backslash (M_{SC} \cup M_{NC})$ e $U_R = U \backslash (U_{SC} \cup U_{NC})$, são vértices que poderão estar controlados ou não de acordo com a adição ou remoção de arestas optativas.

Para melhor caracterizar os conjuntos acima, considere a seguinte notação auxiliar capaz de verificar se um vértice qualquer pode ser controlado. Considere uma variável binária x_{ij} associada a cada par $i, j \in V$, onde $x_{ij} \in \{0, 1\}$ e $x_{ij} = x_{ji}, \forall\ i, j \in V$. Seja a constante binária $a_{ij} = 1$ se e somente se, $i = j$ ou $(i, j) \in E_2$, e $a_{ij} = 0$ caso contrário.

Assim:

$$B_i = \sum_{j \in M} a_{ij} x_{ij} - \sum_{j \in U} a_{ij} x_{ij}, \text{ para } i = 1, ..., |V| \tag{2.3}$$

Na equação (2.3), $x_{ii} = 1$ para todo $i \in V$, $x_{ij} = 1$ para todo $(i, j) \in E_1$ e $x_{ij} = 0$ para todo $(i, j) \notin E_2$. As variáveis binárias restantes estarão associadas às arestas optativas $E_2 \backslash E_1$. Observe que, $B_i \geq 0$ para alguma atribuição $0 - 1$ às variáveis $x_{ij} \in \{0, 1\}$, se e somente se, o vértice i pode ser controlado por M em algum grafo sanduíche $G = (V, E)$. Os subconjuntos $M_{SC}, U_{SC}, M_{NC}, U_{NC}$ podem ser caracterizados da seguinte forma:

- $i \in M_{SC} \cup U_{SC}$ se e somente se, $B_i \geq 0$, qualquer que seja a atribuição 0-1 às variáveis x_{ij} (arestas optativas de G);
- $i \in M_{NC} \cup U_{NC}$ se e somente se, $B_i < 0$, qualquer que seja a atribuição 0-1 às variáveis x_{ij} (arestas optativas de G);

Dessa maneira, a partição de V descrita acima pode ser realizada facilmente após a adição e remoção de todas as arestas optativas de G. Ou seja, após a adição de todas as arestas de $E_2 \backslash E_1$, identificamos diretamente aqueles vértices em M_{SC} (que continuam controlados) e em U_{NC} (que continuam não controlados). Da mesma forma, após a remoção de todas as arestas optativas, identificamos diretamente os vértices de U_{SC} e M_{NC} que continuam controlados e não controlados, respectivamente.

Considere então as regras de redução (esquematizadas na Figura 2.4) apresentadas em [20]:

- Adicionar arestas a E_1, fazendo $x_{ij} = 1,\ \forall\ (i, j) \in D(M_{SC} \cup M_{NC}, U_R)$ *(Regra de redução 3)*;
- Remover arestas de E_2, fazendo $x_{ij} = 0,\ \forall\ (i, j) \in D(M_R, U_{SC} \cup U_{NC})$ *(Regra de redução 4)*;
- Aleatoriamente adicionar ou remover arestas atribuindo $0 - 1$ a $x_{ij},\ \forall\ (i, j) \in D(M_{SC} \cup M_{NC}, U_{SC} \cup U_{NC})$ *(Regra de redução 5)*.

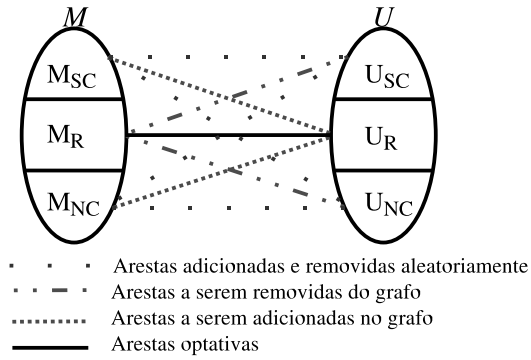


Figura 2.4: Relação entre os sub-conjuntos de vértices do grafo e esquema de aplicação das regras de redução 3,4 e 5.

Considere o grafo da Figura 2.5(a). A Figura 2.5(b) demonstra a aplicação da terceira regra de redução, onde as arestas $D(M_{SC} \cup M_{NC}, U_R)$ são adicionadas ao grafo (arestas (1,3) e (1,4)). Note agora que o vértice 4 será sempre f-controlado. A Figura 2.5(c) demonstra a aplicação da quarta regra de redução, onde as arestas $D(U_{SC} \cup U_{NC}, M_R)$ são removidas do grafo (arestas (5,2) e (4,2)).

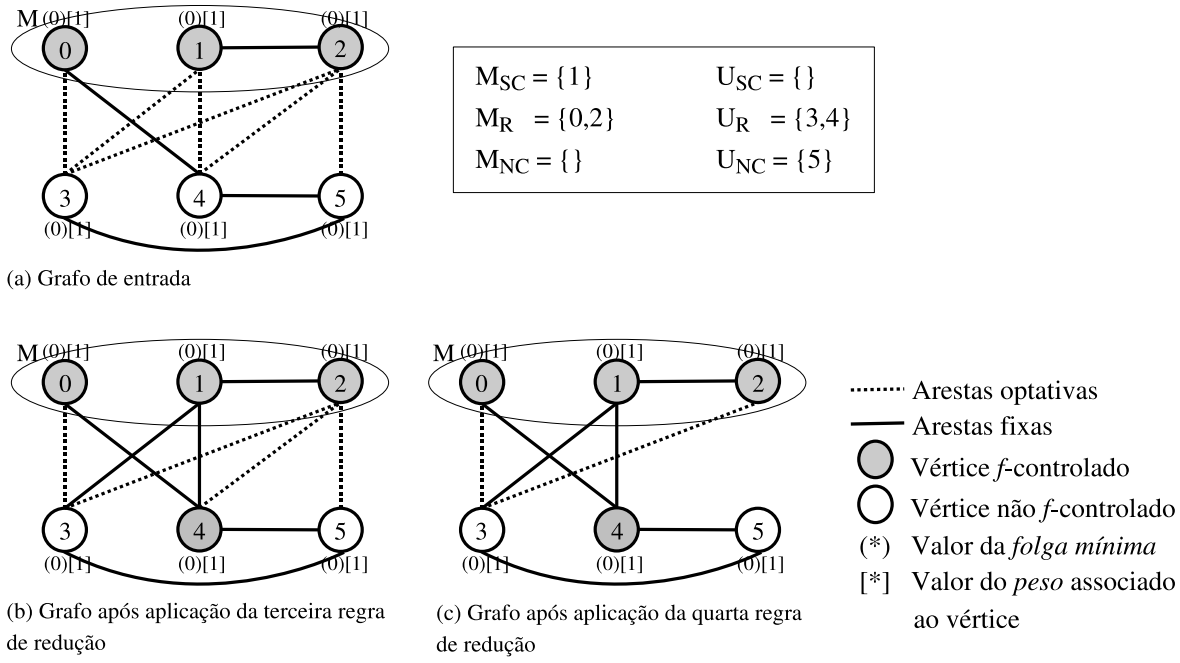
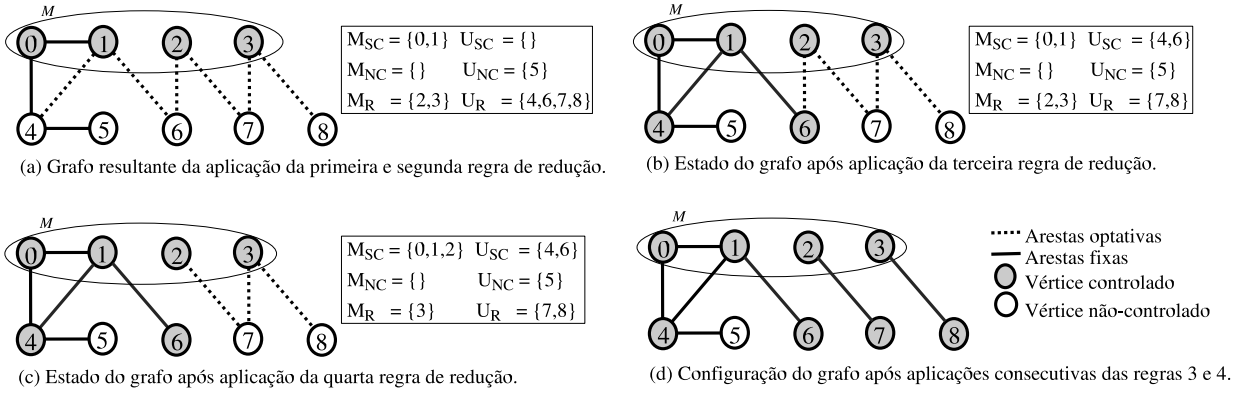


Figura 2.5: Aplicação da terceira e quarta regra de redução.

Na regra de redução 1, a adição das arestas optativas irá somente colaborar para o controle de vértices em M. Na regra 2, as arestas optativas que poderiam prejudicar o controle de vértices em U são eliminadas. Na regra 3, são adicionadas as arestas optativas incidentes àqueles vértices de M, sempre ou nunca controlados, facilitando o controle dos

outros vértices adjacentes em U_R. Na regra 4, se um vértice pertence a $U_{SC} \cup U_{NC}$, removemos todas as suas arestas optativas incidentes visando aumentar a chance de controle dos vértices adjacentes em M_R. Por fim, a regra 5 visa somente reduzir o espaço de soluções, já que a adição ou remoção de arestas entre vértices sempre ou nunca controlados não influencia na solução do problema. É fácil ver que, após a aplicação das regras de redução descritas acima, tem-se apenas arestas optativas com extremidades em vértices pertencentes a M_R e U_R respectivamente.

A ordem em que a terceira e quarta regras de redução são aplicadas pode interferir na remoção ou adição de uma aresta optativa. É possível que a aplicação das regras ocorra de maneira que uma regra colabore para a outra e vice-versa. Esse fato leva a aplicação consecutiva das regras 3 e 4 até que não seja mais possível adicionar ou remover arestas optativas. A Figura 2.6 ilustra um exemplo de aplicações sucessivas das regras 3 e 4 para o caso particular onde $f_i = 0$ e $p_i = 1, \forall\ i \in V$ (PMCC). Na Figura 2.6(a) tem-se um grafo resultante da aplicação da primeira e segunda regra de redução. A Figura 2.6(b) demonstra a aplicação da terceira regra de redução. Note que o vértice 1 pertence a M_{SC}, logo todas as suas arestas optativas incidentes serão adicionadas ao grafo G. Na Figura 2.6(c) tem-se a aplicação da quarta regra de redução. O vértice 6 pertence a U_{SC} por conseqüência da aplicação da terceira regra de redução. A aplicação em seqüência das regras de redução 3 e 4 gera o grafo ilustrado na Figura 2.6(d).



(a) Grafo resultante da aplicação da primeira e segunda regra de redução.

(b) Estado do grafo após aplicação da terceira regra de redução.

(c) Estado do grafo após aplicação da quarta regra de redução.

(d) Configuração do grafo após aplicações consecutivas das regras 3 e 4.

Figura 2.6: Aplicação das regras de redução 3 e 4 consecutivamente.

Após a aplicação das regras de redução descritas acima, identificamos alguns vértices que jamais poderão ser controlados, ou seja, todos os vértices $i \in (M_{NC} \cup U_{NC})$. Essa propriedade nos permite determinar um limite superior para o número de vértices controlados em um grafo, através da equação (2.4).

$$LS = \sum_{i \in V} p_i - \sum_{j \in (M_{NC} \cup U_{NC})} p_j \tag{2.4}$$

Em seguida, apresentamos um algoritmo polinomial com razão de aproximação igual a $\frac{1}{2}$, um modelo matemático para o PMCCG e um algoritmo para encontrar soluções viáveis iniciais por meio de uma relaxação linear do problema.

2.2 Algoritmo $\frac{1}{2}$-aproximado para o PMCCG

O PMCC pode ser resolvido por um algoritmo determinístico, de tempo polinomial e razão de aproximação $\frac{1}{2}$ (vide Makino *et. al.* [19]). Esse algoritmo pode ser estendido facilmente ao PMCCG, garantindo-se a mesma razão de aproximação.

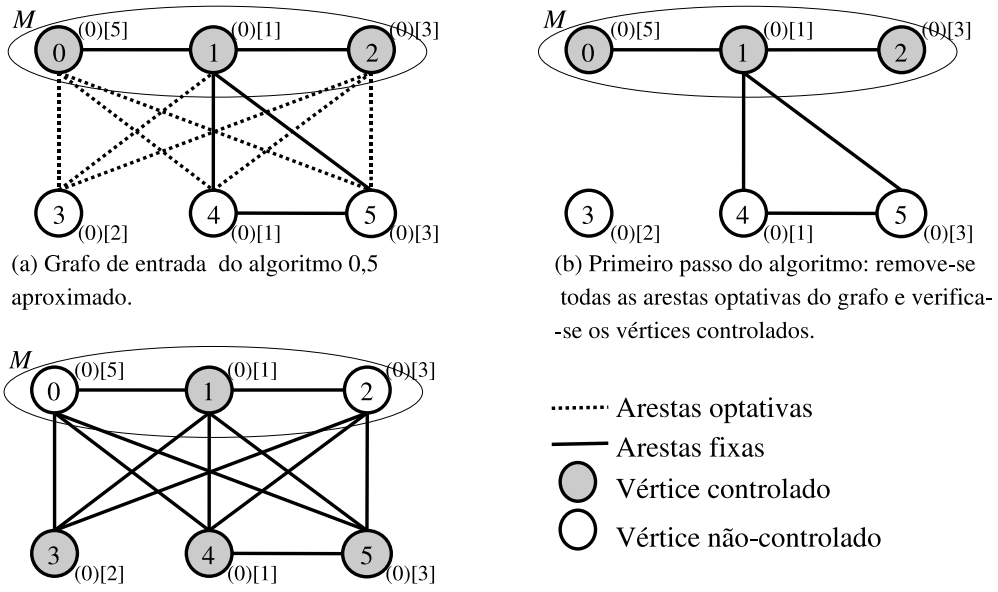
Originalmente o Algoritmo 1, descrito a seguir, faz uso das duas primeiras regras de redução. Neste trabalho o algoritmo é empregado após a aplicação das cinco regras de redução descritas anteriormente e considera pesos positivos associados aos vértices do grafo. Além disso, suponha que W_1 e W_2 representam o somatório dos pesos dos vértices f-controlados por M em $G = (V, E)$ para $E = E_1$, e $E = E_2$, respectivamente. A variável z_{H1} representa a melhor solução obtida em ambos os casos.

Algoritmo 1 Algoritmo $\frac{1}{2}$-aproximado para o PMCCG

1: $W_1 \leftarrow$ Somatório dos pesos dos vértices f-controlados por M, em $G = (V, E)$ para $E = E_1$;
2: $W_2 \leftarrow$ Somatório dos pesos dos vértices f-controlados por M, em $G = (V, E)$ para $E = E_2$;
3: $z_{H1} \leftarrow max\{W_1, W_2\}$;

A Figura 2.7 ilustra uma aplicação do Algoritmo 1. Considere como grafo de entrada do algoritmo a Figura 2.7(a), após a aplicação das regras de redução 1-5 descritas anteriormente. Na Figura 2.7(b) tem-se o primeiro passo do algoritmo, onde todas as arestas optativas são removidas. Note neste caso que $W_1 = 9$. A Figura 2.7(c) demonstra o segundo passo do algoritmo, após a adição de todas as arestas optativas. Neste caso, $W_2 = 7$. A solução dada pelo Algoritmo 1 para o exemplo, corresponde ao maior valor entre W_1 e W_2, ou seja, $z_{H1} = 9$ (dado por W_1).

Teorema 1 *Seja z_{max} o valor da solução ótima do PMCCG. O valor de z_{H1} fornecido pelo Algoritmo 1 satisfaz $z_{H1} \geq \frac{1}{2} z_{max}$, qualquer que seja a instância do problema.*



(a) Grafo de entrada do algoritmo 0,5 aproximado.

(b) Primeiro passo do algoritmo: remove-se todas as arestas optativas do grafo e verifica-se os vértices controlados.

(c) Segundo passo do algoritmo: adiciona-se todas as arestas optativas do grafo e verifica-se os vértices controlados.

Figura 2.7: Exemplo da aplicação do Algoritmo 1.

Prova: Seja $W_1 = W_1^M + W_1^U$ (analogamente $W_2 = W_2^M + W_2^U$), onde W_1^M e W_1^U (analogamente W_2^M e W_2^U) representam os somatórios dos pesos dos vértices f-controlados (por M) em M e U, respectivamente. Note que W_1^M e W_2^U representam a maior soma de pesos possível em M e U, respectivamente. Logo, como todos os pesos envolvidos são positivos temos: $z_{max} \leq W_1^M + W_2^U \leq W_1 + W_2 \leq 2 \times max\{W_1, W_2\}$. Como $z_{H1} = max\{W_1, W_2\}$ então $z_{H1} \geq \frac{1}{2} z_{max}$ c.q.d. ■

2.3 Modelo matemático para o PMCCG

Inicialmente Martinhon&Protti [20] definiram uma formulação de programação inteira para o PMCC. A fim de introduzir uma formulação de programação linear inteira para o PMCCG, definimos inicialmente as variáveis binárias z_i, que determinam quais vértices i de V serão f-controlados ou não por M. As variáveis binárias x_{ij} são usadas para decidir quais arestas pertencentes a $E_2 \backslash E_1$ serão consideradas ou não no grafo sanduíche. As constantes $p_i \in \mathbb{Z}^+$ e $f_i \in \mathbb{Z}$ correspondem, respectivamente, ao peso e folga mínima do vértice $i \in V$. As constantes binárias $a_{ij} \in \{0, 1\}$ são associadas às arestas $(i, j) \in E_2$, sendo $a_{ij} = 1$, se e somente se, $i = j$ ou $(i, j) \in E_2$. Assumimos ainda que $a_{ij} = a_{ji}$, $\forall\, i, j \in V$.

Considere $K = |V| + max\{|f_i| \mid i \in V\}$, apresentamos então o seguinte modelo de

programação linear inteira para o PMCCG:

$$z_{max} = maximizar \sum_{i \in V} p_i z_i \tag{2.5}$$

sujeito a:

$$\frac{\sum_{j \in M} a_{ij}x_{ij} - \sum_{j \in U} a_{ij}x_{ij} - f_i}{K} + 1 \geq z_i, \forall\ i \in V \tag{2.6}$$

$$x_{ij} = 1, \forall\ (i,j) \in E_1 \tag{2.7}$$

$$x_{ii} = 1, \forall\ i \in V \tag{2.8}$$

$$x_{ij} \in \{0,1\}, \forall\ (i,j) \in E_2 \backslash E_1 \tag{2.9}$$

$$z_i \in \{0,1\}, \forall\ i \in V \tag{2.10}$$

Na formulação apresentada temos a função objetivo (2.5) que calcula o somatório dos pesos dos vértices f-controlados por M. A desigualdade (2.6) garante o f-controle do vértice i por M, sempre que o primeiro membro da inequação for maior ou igual a 1. Caso contrário, o vértice i não será f-controlado por M e z_i será fixado em 0. A divisão por K é realizada para manter a diferença entre os dois somatórios e f_i sempre maior que -1, garantindo sempre $z_i \geq 0$. As igualdades (2.7) e (2.8) definem o conjunto de arestas fixas em G_1. A relaxação linear do PMCCG, representada simplesmente por $\tilde{P}$, é obtida substituindo-se as restrições (2.9) e (2.10) (associadas às variáveis de decisão) por $x_{ij} \in [0,1]$ e $z_i \in [0,1]$, respectivamente.

Uma formulação mais forte para o PMCCG pode ser obtida após a aplicação das regras de redução descritas na Seção 2.1. Dessa maneira, considere os conjuntos M_R, M_{SC}, M_{NC}, U_R, U_{SC}, U_{NC} como definidos anteriormente, obtidos substituindo-se a definição de vértice controlado por f-controlado. Considere ainda uma constante b_i, que possui o valor correspondente à pior folga que um vértice $i \in M_R \cup U_R$ pode assumir. Assim, b_i será calculado com o auxílio da equação (2.11) a seguir:

$$b_i = \left| \sum_{j \in M} a_{ij}x_{ij} - \sum_{j \in U} a_{ij}x_{ij} - f_i \right| \text{ para } i \in M_R \cup U_R \tag{2.11}$$

A determinação de b_i é realizada então da seguinte forma, se $i \in M_R$ fazemos $x_{ij} = 1,\ \forall\ (i,j) \in E_2 \backslash E_1$. Analogamente, se $i \in U_R$ fazemos $x_{ij} = 0,\ \ \forall\ (i,j) \in E_2 \backslash E_1$. Obviamente, em ambos os casos teremos $x_{ij} = 1,\ \forall\ (i,j) \in E_1$.

Desta forma, uma formulação mais forte para o PMCCG pode ser obtida substituindo-

se a restrição (2.6) pelas restrições (2.12), (2.13) e (2.14) descritas a seguir:

$$\frac{\sum_{j\in M} a_{ij}x_{ij} - \sum_{j\in U} a_{ij}x_{ij} - f_i}{b_i} + 1 \geq z_i, \forall\ i \in M_R \cup U_R \tag{2.12}$$

$$z_i = 1, \forall i \in M_{SC} \cup U_{SC} \tag{2.13}$$

$$z_i = 0, \forall i \in M_{NC} \cup U_{NC} \tag{2.14}$$

Considere $\bar{P}$ a relaxação linear associada a esta nova formulação. A desigualdade (2.12) tem a mesma função da restrição (2.6), garantir o f-controle do vértice i por M, sempre que o primeiro membro da inequação for maior ou igual a 1. Entretanto, a nova restrição garante uma relaxação linear de melhor qualidade para o PMCCG, visto que $b_i \leq K, \forall\ i \in M_R \cup U_R$. A igualdade (2.13) indica os vértices sempre f-controlados no grafo e a igualdade (2.14) os vértices nunca f-controlados. Formalmente, temos o seguinte resultado:

Teorema 2 *Sejam $\tilde{z}_{max}$ e $\bar{z}_{max}$ os valores ótimos de $\tilde{P}$ e $\bar{P}$ respectivamente. Então $\tilde{z}_{max} \geq \bar{z}_{max}$.*

Representaremos por $\bar{x}_{ij} \in [0,1],\ \forall\ (i,j) \in E_2$, e $\bar{z}_i \in [0,1],\ \forall\ i \in V$, ou simplesmente $(\bar{x}, \bar{z})$, uma solução ótima de $\bar{P}$. Como discutido em [24], essa relaxação pode ser resolvida em tempo polinomial através dos métodos de pontos interiores.

2.4 Solução para o PMCCG baseada na relaxação linear

Mostraremos nesta seção como obter uma solução viável para o PMCCG a partir do problema relaxado $\bar{P}$ como descrito anteriormente. Assim, um procedimento bastante natural para o problema pode ser obtido da seguinte forma: dada uma solução $(\bar{x}, \bar{z})$ de $\bar{P}$ com coordenadas $\bar{x}_{ij} \in [0,1], \forall\ (i,j) \in E_2$ e $\bar{z}_i \in [0,1], \forall\ i \in V$, defina como f-controlado, todos os vértices $i \in V$ onde $\bar{z}_i = 1$, e como não f-controlado os demais vértices de V, onde $\bar{z}_i < 1$.

Na construção do Algoritmo 2, considere L_G e L_D os conjuntos dos vértices f-controlados e não f-controlados por M respectivamente.

Como veremos adiante, se $\bar{P}$ possui uma única solução ótima, os valores $\bar{x}_{ij} \in [0,1]$ obtidos após a solução de $\bar{P}$ serão sempre inteiros, significando portanto, que ao esco-

Algoritmo 2 Algoritmo baseado na relaxação linear

1: $L_G \leftarrow \oslash, L_D \leftarrow \oslash$
2: **para todo** $i \in V$ **faça**
3: **se** $\bar{z}_i = 1$ **então**
4: $L_G \leftarrow L_G \cup \{i\}$
5: **se não**
6: $L_D \leftarrow L_D \cup \{i\}$
7: **fim se**
8: **fim para**
9: *retorna* G

lhermos os vértices f-controlados ou não por M através de $\bar{z}$, teremos automaticamente um mecanismo para decidir quais arestas optativas serão ou não selecionadas ao final do procedimento. Formalmente, temos a seguinte proposição demonstrada em seguida.

Teorema 3 *Considere uma solução ótima* $(\bar{x}, \bar{z})$ *de* $\bar{P}$ *com coordenadas* $\bar{x}_{ij} \in [0,1], \forall (i,j) \in E_2$, $\bar{z}_i \in [0,1], \forall i \in V$, *e valor ótimo* $\bar{z}_{max} = \sum_{i \in V} p_i \bar{z}_i$. *Se* $\bar{x}_{ij} \in (0,1)$ *para alguma aresta* $(i,j) \in E_2 \backslash E_1$, *existirá então, uma nova solução relaxada* $(\tilde{x}, \tilde{z})$ *de* $\bar{P}$ *onde* $\bar{z}_{max} = \sum_{i \in V} p_i \tilde{z}_i$ *sendo* $\tilde{x}_{ij} \in \{0,1\}, \forall\ (i,j) \in E_2$ *e* $\tilde{z}_i \in [0,1], \forall\ i \in V$.

Prova: Note inicialmente de (2.12) que uma solução relaxada $(\bar{x}, \bar{z})$ de $\bar{P}$ deverá satisfazer à seguinte igualdade:

$$\frac{\sum_{i \in M} a_{ik}\bar{x}_{ik} - \sum_{j \in U} a_{jk}\bar{x}_{jk} - f_k}{b_k} + 1 = \bar{z}_k, \forall\ k \in M_R \cup U_R \tag{2.15}$$

Sem perda de generalidade, considere $V = M_R \cup U_R$. Considere ainda V' e E_2', dois conjuntos de vértices e arestas definidos da seguinte forma:

$$V' = V \cup \{s,t\}$$
$$E_2' = E_2 \backslash E_1 \cup \{(s,i), \forall\ i \in M_R\} \cup \{(j,t), \forall\ j \in U_R\}$$

Construímos agora uma rede auxiliar $N' = (G', c')$ onde $G' = (V', E_2')$ e $c' : E_2' \rightarrow \mathbb{R}^+$. A função c', que define as capacidades dos arcos de E_2', será construída a partir de $\bar{z}$ e da igualdade (2.15) da seguinte forma:

$$\begin{aligned} c'_{si} &= |E_1(i, M_R)| - |E_1(i, U_R)| - (\bar{z}_i - 1)b_i - f_i, \forall\ i \in M_R \\ c'_{ij} &= 1, \forall\ (i,j) \in E_2 \backslash E_1 \\ c'_{jt} &= (\bar{z}_j - 1)b_j - |E_1(M_R, j)| + |E_1(U_R, j)| + f_j, \forall\ j \in U_R \end{aligned} \tag{2.16}$$

Onde $E_1(M_R, j)$ e $E_1(M_R, i)$ (respectivamente $E_1(U_R, i)$ e $E_1(U_R, j)$) representam o conjunto das arestas fixas de G_1 com extremidades em M_R (em U_R) e V, respectivamente.

Utilizaremos inicialmente o algoritmo dos Caminhos Aumentantes Sucessivos (*Augmenting Path Algorithm*) para resolver o problema de fluxo máximo de s a t em N' (para maiores detalhes vide [1]). Assim, da lei de conservação de fluxo, obtemos uma nova solução $(\hat{x}, \bar{z})$ de $\bar{P}$ onde:

$$\sum_{j \in U_R} a_{ij}\hat{x}_{ij} = \hat{x}_{si} = c'_{si}, \forall\ i \in M_R \tag{2.17}$$

$$\sum_{i \in M_R} a_{ij}\hat{x}_{ij} = \hat{x}_{jt} = c'_{jt}, \forall\ j \in U_R \tag{2.18}$$

É fácil ver de (2.16), (2.17) e (2.18) que ao resolvermos o problema de fluxo máximo de s a t pelo Algoritmo de Caminhos Aumentantes, estaremos garantindo uma nova solução $(\hat{x}, \bar{z})$ onde $\bar{z}_{max} = \sum_{i \in V} p_i \bar{z}_i$. Note ainda que não podemos assegurar que as variáveis $\hat{x}_{ij}$ (associadas às arestas optativas) sejam binárias já que as capacidades $c'_{si}, c'_{jt}, \forall\ i \in M_R$ e $j \in U_R$ não representam necessariamente valores inteiros.

Considere então de nossa hipótese, que exista uma variável fracionária $\hat{x}_{vw} \in (0, 1)$, associada a alguma aresta $(v, w) \in E_2 \backslash E_1$. Mostraremos que uma nova solução relaxada $(\tilde{x}, \tilde{z})$ de $\bar{P}$ com componentes $\tilde{x}_{ij} \in \{0, 1\}, \forall\ (i, j) \in E_2 \backslash E_1$, poderá ser obtida resolvendo-se um problema de fluxo máximo de s a t, em uma nova rede $N'' = (G', c'')$ onde $c'' : E'_2 \rightarrow \mathbb{Z}^+$ é definida convenientemente, e $\bar{z}_{max} = \sum_{i \in V} p_i \tilde{z}_i$.

Suponha então que $\Delta \in [0, 1]$ unidades de fluxo devam ser enviadas no caminho $p = (s, v, w, t)$ (onde $v \in M_R$ e $w \in U_R$). Considere ainda, que $\delta' = \lfloor c'_{sv} \rfloor$ e $\delta'' = \lfloor c'_{wt} \rfloor$ unidades de fluxo já tenham sido enviadas por outros caminhos aumentantes passando por v e w, respectivamente, vide Figura 2.8.

Assim, seja $\Delta = min\{r_{sv}, r_{wt}, 1\}$, onde $r_{sv} = c'_{sv} - \delta'$ e $r_{wt} = c'_{wt} - \delta''$, as capacidades residuais na rede residual N'_R (associada a N'). Logo, ao enviarmos $\Delta \in [0, 1]$ unidades de fluxo no caminho p em N' teremos $\hat{x}_{sv} = \delta' + \Delta$, $\hat{x}_{wt} = \delta'' + \Delta$ e $\hat{x}_{vw} = \Delta$, respectivamente. Obviamente, se $\Delta = 0$ ou $\Delta = 1$ já teremos $\hat{x}_{vw} \in \{0, 1\}$. Considere então $\Delta \in (0, 1)$. Concluímos agora de (2.15) que:

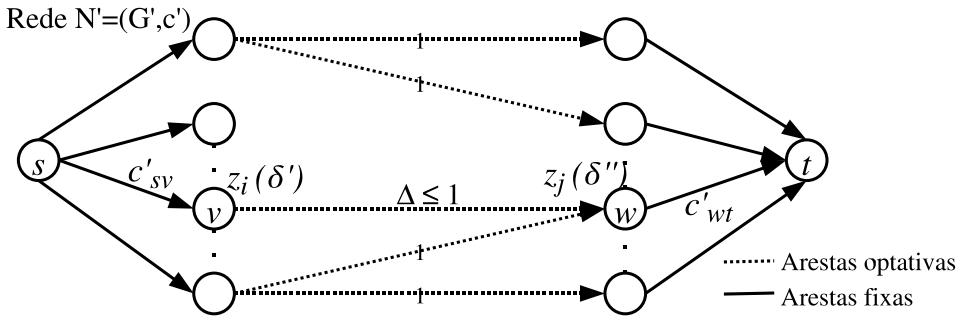


Figura 2.8: Enviando Δ unidades de fluxo em $p = (s, v, w, t)$.

$$\bar{z}_v = z_v(\delta' + \Delta) = \frac{|E_1(v, M_R)| - (|E_1(v, U_R)| + (\Delta + \delta')) - f_v}{b_v} + 1, \quad (2.19)$$

para $v \in M_R$

$$\bar{z}_w = z_w(\delta'' + \Delta) = \frac{(|E_1(w, M_R)| + (\Delta + \delta'')) - |E_1(w, U_R)| - f_w}{b_w} + 1, \quad (2.20)$$

para $w \in U_R$

Vejamos agora como determinar as capacidades c''_{sv}, c''_{vw} e c''_{wt} no caminho p na nova rede N''.

Essas novas capacidades deverão ser definidas de maneira que, ao resolvermos o problema do fluxo máximo de s a t em N'', os arcos c''_{si} e c''_{jt} sejam inteiros e saturados, além disso devemos ter, $p_v z_v(\delta' + \Delta) + p_w z_w(\delta'' + \Delta) = p_v z_v(\delta' + \Delta') + p_w z_w(\delta'' + \Delta')$ onde $\Delta' \in \{0, 1\}$, ou seja, $p_v \bar{z}_v + p_w \bar{z}_w = p_v \tilde{z}_v + p_w \tilde{z}_w$. Assim, repetindo-se o processo para todo arco $(v, w) \in E_2 \backslash E_1$ onde $\hat{x}_{vw}$ é fracionário, teremos uma nova rede $N'' = (G', c'')$ onde c'' é vetor de coordenadas inteiras e $\bar{z}_{max} = \sum_{i \in V} p_i \bar{z}_i = \sum_{i \in V} p_i \tilde{z}_i$.

Note ainda de (2.19) e (2.20), que se aumentarmos o fluxo no arco (v, w) em N'' de Δ para $\Delta' = 1$ teremos sempre $z_v(\delta' + 1) \leq z_v(\delta' + \Delta), \forall\ v \in M_R$ e $z_w(\delta'' + 1) \geq z_w(\delta'' + \Delta), \forall\ w \in U_R$. Analogamente, ao diminuirmos o fluxo de Δ para $\Delta' = 0$ teremos sempre $z_v(\delta') \geq z_v(\delta' + \Delta), \forall\ v \in M_R$ e $z_w(\delta'') \leq z_w(\delta'' + \Delta), \forall\ w \in U_R$. Observe agora que, se $p_v > p_w$ (respectivamente $p_w > p_v$), ao substituir Δ por $\Delta' = 1$ (Δ por $\Delta' = 0$) teremos um novo fluxo ótimo em N'' com custo maior que $\bar{z}_{max}$, o que é absurdo pois $\bar{z}_{max}$ é valor ótimo de $\bar{P}$. Concluímos então que, se $p_v \neq p_w$ teremos então $\Delta \in \{0, 1\}$.

Considere agora $p_v = p_w$. Neste caso, temos as seguintes possibilidades, obtidas após resolvermos o problema de fluxo máximo de s a t em N':

i. $\bar{z}_v = z_v(\delta' + \Delta) = 0$ e $\bar{z}_w = z_w(\delta'' + \Delta) \leq 1$. Nesse caso, fazemos $c''_{sv} = \delta' + 1, c''_{vw} = 1$ e $c''_{wt} = \delta'' + 1$. É fácil ver que, ao substituir Δ por $\Delta' = 1$ no caminho p de N'', obtemos uma nova solução viável de $\bar{P}$, onde $\tilde{z}_v = z_v(\delta' + 1) = 0$ e $\tilde{z}_w = z_w(\delta'' + 1) \geq z_w(\delta'' + \Delta)$. Entretanto, se $z_w(\delta'' + 1) > z_w(\delta'' + \Delta)$, teríamos uma nova solução relaxada de $\bar{P}$ onde $\bar{z}_v = \tilde{z}_v$ e $\bar{z}_w < \tilde{z}_w$ e portanto $\sum_{i \in V} p_i \tilde{z}_i > \bar{z}_{max}$, o que nos levaria a um absurdo. Por outro lado, se $z_w(\delta'' + 1) = z_w(\delta'' + \Delta)$, temos uma nova solução de $\bar{P}$ onde $\tilde{x}_{vw} = \Delta' = 1$, $\bar{z}_v = \tilde{z}_v$ e $\bar{z}_w = \tilde{z}_w$.

ii. $\bar{z} = z_v(\delta' + \Delta) \leq 1$ e $\bar{z}_w = z_w(\delta'' + \Delta) = 0$. Nesse caso, fazemos $c''_{sv} = \delta', c''_{vw} = 0$ e $c''_{wt} = \delta''$. Novamente, ao substituir Δ por $\Delta' = 0$ no caminho p em N'', obtemos uma nova solução, onde $\tilde{z}_v = z(\delta') \geq z(\delta' + \Delta)$ e $z_w(\delta'') = z_w(\delta'' + \Delta)$. Se $z_v(\delta') = z_v(\delta' + \Delta)$, temos uma nova solução de $\bar{P}$ onde $\tilde{x}_{vw} = \Delta' = 0$, $\bar{z}_v = \tilde{z}_v$ e $\bar{z}_w = \tilde{z}_w$. Por outro lado, se $z(\delta') > z(\delta' + \Delta)$ teríamos uma nova solução de $\bar{P}$ onde $\bar{z}_v < \tilde{z}_v$ e $\bar{z}_w = \tilde{z}_w$ e portanto $\sum_{i=V} p_i \tilde{z}_i > \bar{z}_{max}$, o que nos levaria a um absurdo.

iii. $\bar{z}_v = z_v(\delta' + \Delta) \in (0, 1)$ e $\bar{z}_w = z_w(\delta'' + \Delta) \in (0, 1)$. Mostraremos primeiramente que isto só será possível se $b_w = b_v$. Como δ' e δ'', representam quantidades inteiras de fluxo já enviadas de s a t passando por v e w, respectivamente, ao fazer $\Delta' = 0$ ou $\Delta' = 1$ em N'' teremos de (2.19) e (2.20) que $z_v(\delta' + \Delta') \in [0, 1]$ e $z_w(\delta'' + \Delta') \in [0, 1]$.

Suponha agora, sem perda de generalidade, que $b_w > b_v$. De (2.19) e (2.20) concluímos que $\bar{z}_v - \tilde{z}_v + \bar{z}_w - \tilde{z}_w = z_v(\delta' + \Delta) - z_v(\delta' + \Delta') + z_w(\delta'' + \Delta) - z_w(\delta'' + \Delta') = -\frac{(\delta' + \Delta)}{b_v} + \frac{(\delta' + \Delta')}{b_v} + \frac{(\delta'' + \Delta)}{b_w} - \frac{(\delta'' + \Delta')}{b_w} = (b_w - b_v)(\Delta' - \Delta)$.

Como $\Delta \in (0, 1)$, e $b_w > b_v$, ao fazer $\Delta' = 0$ obtemos uma nova solução onde: $\bar{z}_v + \bar{z}_w < \tilde{z}_v + \tilde{z}_w$, o que é absurdo pois teríamos uma nova solução melhor do que aquela obtida em $\bar{P}$, ou seja, teríamos uma solução onde $\sum_{i \in V} p_i \tilde{z}_i > \bar{z}_{max}$. Analogamente, se $b_v > b_w$, chegamos à mesma contradição fazendo $\Delta' = 1$. Devemos ter portanto, $b_w = b_v$. Nesse caso, podemos fazer $\Delta' = 0$ ou 1 indistintamente. Portanto fazemos $c''_{sv} = c''_{wt} = \delta' + \Delta'$ e $c''_{vw} = \Delta'$ em N''.

iv. $\bar{z}_v = z_v(\delta' + \Delta) = 1$ e $\bar{z}_w = z_w(\delta'' + \Delta) \leq 1$. Neste caso fazemos $c''_{sv} = \delta'$, $c''_{sw} = 0$ e $c''_{wt} = \delta''$. Como δ' representa uma quantidade inteira de fluxo já enviada de s a t passando por v (algoritmo dos caminhos aumentantes), ao fazer $\Delta' = 1$ em N'' o vértice v continua controlado pois $f_v \in \mathbb{Z}^+$. Fazemos então $c''_{sv} = \delta' + 1$, $c''_{vw} = 1$ e $c''_{wt} = \delta'' + 1$. Analogamente, se $\bar{z}_v = z_v(\delta' + \Delta) \leq 1$ e $\bar{z}_w = z_w(\delta'' + \Delta) = 1$ fazemos $\Delta' = 0$.

Repetimos os processo até que todas as arestas $(i,j) \in E_2 \backslash E_1$ com $\hat{x}_{ij} \in (0,1)$, tenham sido pesquisadas.

Portanto ao resolvermos o problema de fluxo máximo de s a t em N'' podemos garantir a existência de uma nova solução com coordenadas inteiras $\tilde{x}_{ij} \in \{0,1\}, \forall\ (i,j) \in E_2 \backslash E_1$. Dessa forma, obtemos uma nova solução (para a rede N'') onde $\sum_{i \in V} p_i \bar{z}_i = \sum_{i \in V} p_i \tilde{z}_i = \bar{z}_{max}$. c.q.d. ■

Note no exemplo da Figura 2.9(a), que para $f_i = 0$ e $p_i = 1, \forall i \in V$, uma solução ótima de $\bar{P}$, pode ser obtida fazendo-se $\bar{x}_{ij} = 0,5, \forall\ (i,j) \in E_2 \backslash E_1$. Na Figura 2.9(b) entretanto, observamos uma outra solução ótima onde $\tilde{x}_{ij} \in \{0,1\}, \forall (i,j) \in E_2 \backslash E_1$. Em ambos os casos, 4 vértices são controlados (solução ótima).

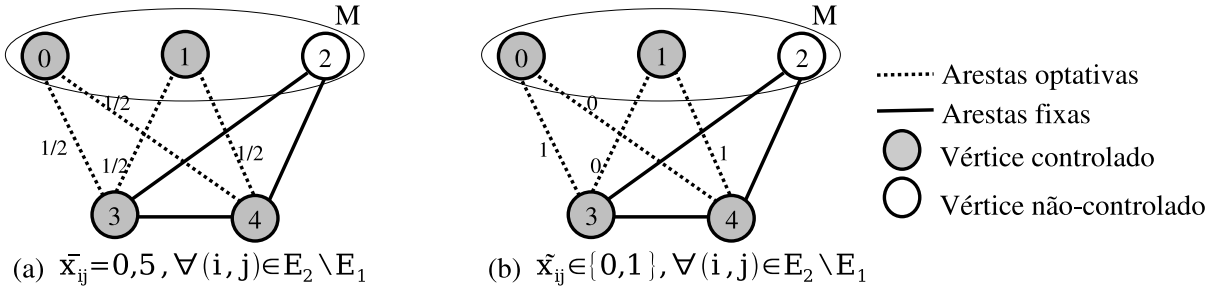


Figura 2.9: Exemplo de duas soluções ótimas, com coordenadas fracionárias (a) e coordenadas inteiras (b).

Um novo procedimento, ilustrado pelo Algoritmo 3, consiste em simplesmente selecionar a melhor solução (com o melhor custo $c(G)$ associado) obtida entre os Algoritmos 1 e 2. Como demonstrado em [20], o Algoritmo 3 possui uma razão de performance maior que $\frac{1}{2}$ para o PMCC (situação particular do PMCCG onde $f_i = 0$ e $p_i = 1$, $\forall\ i \in V$). Aquelas instâncias onde $n \leq 4$, podem ser resolvidas facilmente, de maneira exata e em tempo polinomial. Para maiores detalhes sobre esse procedimento e sua análise de aproximação vide [20].

Algoritmo 3 Combinação dos Algoritmos 1 e 2

1: $G' \leftarrow$ *Algoritmo* 1
2: $G'' \leftarrow$ *Algoritmo* 2
3: **se** $c(G') > c(G'')$ **então**
4: *retorna* G'
5: **se não**
6: *retorna* G''
7: **fim se**

Teorema 4 *O Algoritmo 3 garante, em tempo polinomial, uma razão de performance para o PMCC igual a* $\frac{1}{2} + \frac{1+\sqrt{n}}{2(n-1)}$, $\forall\ n > 4$.

A solução obtida através da relaxação linear (Algoritmo 2), não garante obviamente uma solução ótima para o PMCCG. Para exemplificar essa situação, considere os grafos da Figura 2.10 onde $p_i = 1$ e $f_i = 0$, $\forall\ i \in V$. O grafo da Figura 2.10(a) corresponde ao grafo de entrada. Na Figura 2.10(b) temos o grafo obtido através do Algoritmo 2, com os respectivos valores de $\bar{z}_i$ e $\bar{x}_{ij}$. Note a presença das arestas (0,6), (1,5) e (1,7) e a ausência das arestas (0,7) e (1,6) em G. Dessa maneira, teremos apenas 3 vértices f-controlados por M (associados a $\bar{z}_i = 1$). Entretanto, a Figura 2.10(c) demonstra que uma outra solução pode ser obtida com 4 vértices f-controlados.

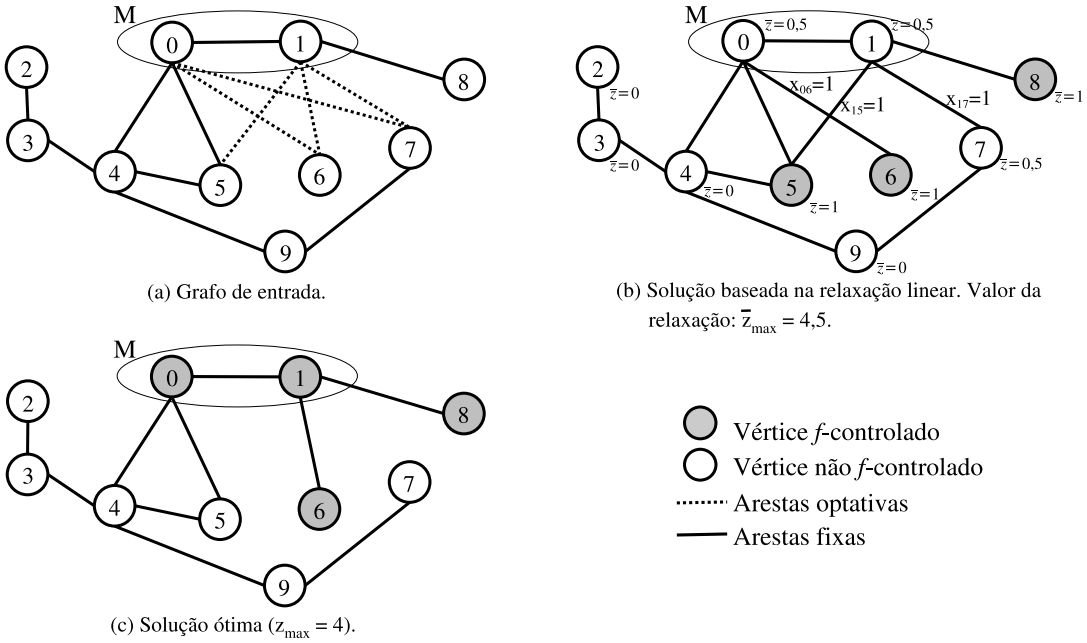


Figura 2.10: Exemplo de grafo em que o Algoritmo 2 retorna uma solução com custo estritamente menor que a solução ótima.

Visando incrementar ainda mais as soluções obtidas pelo Algoritmo 3, descrevemos a seguir um procedimento de busca local para o PMCCG baseado no procedimento de Busca Tabu.

Capítulo 3

Busca Tabu com Reconexão por Caminhos aplicada ao PMCCG

O método de Busca Tabu (*Tabu Search*) - BT, proposto independentemente por [7] e [15] em 1986, é um procedimento iterativo para a solução de problemas de otimização combinatória e que aceita movimentos de piora da solução corrente para tentar escapar de ótimos locais evitando, dessa forma, retornar a regiões previamente pesquisadas.

Dentre as metaheurísticas existentes na literatura, a BT tem se mostrado competitiva na resolução de diferentes problemas de otimização combinatória. Na BT, em cada iteração uma busca local é executada vasculhando-se uma vizinhança $N(S)$ da solução corrente S. A *lista tabu* consiste de um conjunto de *soluções* ou *movimentos proibidos*, determinados por informações históricas das iterações precedentes no procedimento de busca local. Dessa maneira, os elementos presentes na lista tabu são proibidos temporariamente. Após um determinado número de iterações, a lista é atualizada de forma a remover elementos da lista e permitir que novos elementos assumam o estado de tabu.

A BT para um problema de maximização, apresentada no Algoritmo 4, contém as etapas normalmente encontradas nos modelos tradicionais da literatura. Dessa forma, o método inclui construção da solução inicial, fase de busca local intensiva e diversificação das soluções (fuga de ótimos locais). Podemos incluir ainda uma etapa de "pós-otimização" da solução, onde um procedimento de Reconexão por Caminhos - RC, é realizado a partir de um conjunto elite ou *pool* de soluções.

Considere S (com custo $c(S)$ associado) a solução corrente no algoritmo e $S^{\star}$ a melhor solução encontrada (*solução global*) em uma etapa de intensificação da busca local. A lista tabu será representada por T e irá conter os atributos de cada movimento, dessa maneira, uma série de movimentos proibidos será definida implicitamente. O critério de aspiração

permite que sejam exploradas regiões ainda não visitadas no espaço de soluções, garantindo que movimentos presentes na lista Tabu sejam executados, desde que promovam uma solução com custo melhor que a solução global encontrada.

Algoritmo 4 Pseudo código da BT com RC aplicada a um problema de maximização

1: Contadores de iterações $i, j \leftarrow 0$
2: $S, S^\star \leftarrow$ *Solução inicial()*, $c(S'') \leftarrow -\infty$, *Elite* $\leftarrow \oslash$
3: **enquanto** $i \leq i_{max}$ **faça**
4: $T \leftarrow \oslash$
5: **enquanto** $j \leq j_{max}$ **faça**
6: $S' \leftarrow$ *BuscaLocal*$(N(S) \backslash T)$
7: $S'' \leftarrow$ *BuscaLocal*$(N(S) \cap T)$ satisfazendo o critério de aspiração
8: **se** $c(S'') > c(S')$ **então**
9: $S' \leftarrow S''$
10: **fim se**
11: **se** $c(S') > c(S^\star)$ **então**
12: $S^\star \leftarrow S'$
13: $j \leftarrow 0$
14: **fim se**
15: **se** $c(S') < c(S)$ **então**
16: Insira o movimento (S', S) na lista tabu T
17: **fim se**
18: $S \leftarrow S'$ e $j \leftarrow j + 1$
19: *Elite* $\leftarrow$ *Atualiza(Elite,S)*
20: **fim enquanto**
21: $S^\star \leftarrow$ *Diversificação(S)*
22: $i \leftarrow i + 1$
23: **fim enquanto**
24: $S^\star \leftarrow$ *Reconexão por Caminhos(Elite)*
25: *retorna* $S^\star$

No caso do PMCCG, o procedimento de construção da solução inicial (linha 2) utilizado, consiste do Algoritmo 3 apresentado na Seção 2.4. Naquelas instâncias onde o número de restrições é muito grande para serem resolvidas por Programação Linear, geramos a solução inicial utilizando o Algoritmo 1 como descrito na Seção 2.2. Os contadores i e j, denotam o número de iterações do algoritmo e da busca local respectivamente, sendo que a busca local é interrompida sempre que j_{max} iterações ocorram sem melhoria da solução global. Na linha 6, a busca local em $N(S)$ é executada desprezando-se os movimentos proibidos, presentes na lista tabu. Esses movimentos serão pesquisados e realizados na linha 7. Caso o critério de aspiração seja satisfeito, eles serão considerados nas linhas 8 e 9. A solução global será atualizada nas linhas 11-13, caso as buscas locais encontrem uma solução com custo associado maior que o custo da solução global. Para escapar de máximos locais, a busca local permite também movimentos de piora da solução corrente.

Nesse caso deve-se garantir que uma busca conseguinte não retorne novamente a uma região já pesquisada. Essa garantia é dada pela inserção do movimento de "volta" na lista tabu, representada por um ou mais atributos do movimento que levaria a uma solução já encontrada (linhas 15 e 16). A lista tabu se baseia normalmente no conceito de fila. Assim, em cada atualização um movimento proibido é incluído no final da lista (linha 16), ficando presente na lista tabu por um número de iterações equivalente à capacidade dessa lista. O conjunto *elite* é formado, iterativamente, pelas k melhores soluções encontradas pela busca local (linha 19), sendo k um parâmetro de entrada. O conjunto elite é necessário para a execução da Reconexão por Caminhos - RC, discutida mais adiante. A função de diversificação, executada na linha 21, efetua uma perturbação na solução corrente, com o objetivo de fugir de ótimos locais, indo para regiões ainda não pesquisadas pela busca local.

As etapas de busca local, diversificação da solução corrente e reconexão por caminhos são descritas detalhadamente a seguir.

3.1 Fase de busca local intensiva no PMCCG

Uma das etapas mais importante da BT, se refere à etapa de busca local intensiva onde a vizinhança de uma solução corrente S é pesquisada. A qualidade dessa busca local pode ser medida em função do tamanho dessa vizinhança e do número de vezes em que ela é efetuada numa dada região. A idéia, a princípio, é investigar uma mesma região enquanto ela se mostrar promissora.

A busca local (linhas 6 e 7 do Algoritmo 4) analisa a partir de uma solução inicial (semente) soluções vizinhas, utilizando um critério normalmente guloso, ou seja, priorizando o controle dos vértices com maior peso associado. Na busca local aqui realizada, buscamos o f-controle de novos vértices sem que outros vértices sejam *descontrolados* (se tornem não f-controlados). Representaremos esta estrutura de vizinhança por $N_0(G)$.

O procedimento de busca local proposto para o PMCCG é ilustrado no Algoritmo 5. Dado um grafo sanduíche $G = (V, E)$ (solução corrente S), teremos um conjunto de arestas optativas E associadas e um conjunto $M_G \subseteq M_R$ (respectivamente $U_G \subseteq U_R$) dos vértices f-controlados por M em G. Considere também $L_G = M_G \cup U_G$ e $L_D = (M_R \cup U_R \backslash L_G)$.

Nessa estrutura de vizinhança ($N_0(G)$), esperamos que vértices $v \in L_D$ sejam f-controlados desde que nenhum outro vértice $w \in L_G$ seja descontrolado. Note que neste caso, uma nova solução de melhor custo, pertencente a $N_0(G)$ poderá ser gerada já que

todos os pesos associados aos vértices de L_D são positivos. Dado um grafo sanduíche G, chamaremos de *folga corrente* de um vértice $i \in V$ o valor $f_G(i) = |N_G[i] \cap M| - |N_G[i] \cap U| - f_i$. Assim, teremos que $i \in V$ é f-controlado por M se, e somente se, $f_G(i) \geq 0$, caso contrário i será não f-controlado. Note ainda que $f_G(i) < 0, \forall\, i \in L_D$.

Na busca local tentamos controlar vértices $v \in L_D$ (linhas 2,3 e 4), desde que o vértice selecionado não esteja presente nas soluções definidas implicitamente pela lista tabu (representada por T). Representaremos os vizinhos de v que podem auxiliar em seu f-controle por $H_v = \{w \in V | (v,w) \in E_2 \backslash E \text{ e } f_G(w) \neq 0\}$. Um vértice v poderá ser f-controlado desde que $|f_G(v)| \leq |H_v|$ (linha 5). Basta então, adicionar ou remover arestas optativas em número suficiente (dado por $|f_G(v)|$) para se estabelecer o f-controle de v. A escolha de quais vértices vizinhos em H_v serão utilizados, é realizada de maneira aleatória (linha 7). Note que, se o vértice v a ser f-controlado pertence a $M_R \backslash M_G$, será necessário remover $|f_G(v)|$ arestas optativas incidentes a v. Entretanto, caso $v \in U_R \backslash U_G$, será necessário adicionar $|f_G(v)|$ arestas optativas para que v seja f-controlado (linhas de 8-11). Por fim, ocorre a atualização das folgas correntes dos vértices v e seus vizinhos $w \in H_v$ (linhas 13 e 14) e a atualização da lista de vértices descontrolados (linha 19).

Algoritmo 5 Procedimento de busca-local utilizado pela BT

```
1: Dado um grafo sanduíche G = (V, E) como entrada
2: enquanto L_D ≠ ∅ faça
3:    Seleciona aleatoriamente v ∈ L_D
4:    se v ∉ T então
5:       se |H_v| ≥ |f_G(v)| então
6:          enquanto f_G(v) < 0 faça
7:             w ← Escolhe vértice w ∈ H_v aleatoriamente
8:             se v ∈ M_R então
9:                E ← E\{(v, w)}
10:            se não
11:               E ← E ∪ {(v, w)}
12:            fim se
13:            f_G(v) ← f_G(v) + 1
14:            f_G(w) ← f_G(w) − 1
15:            H_v ← H_v\{w}
16:         fim enquanto
17:      fim se
18:   fim se
19:   L_D ← L_D\{v}
20: fim enquanto
21: retorna G
```

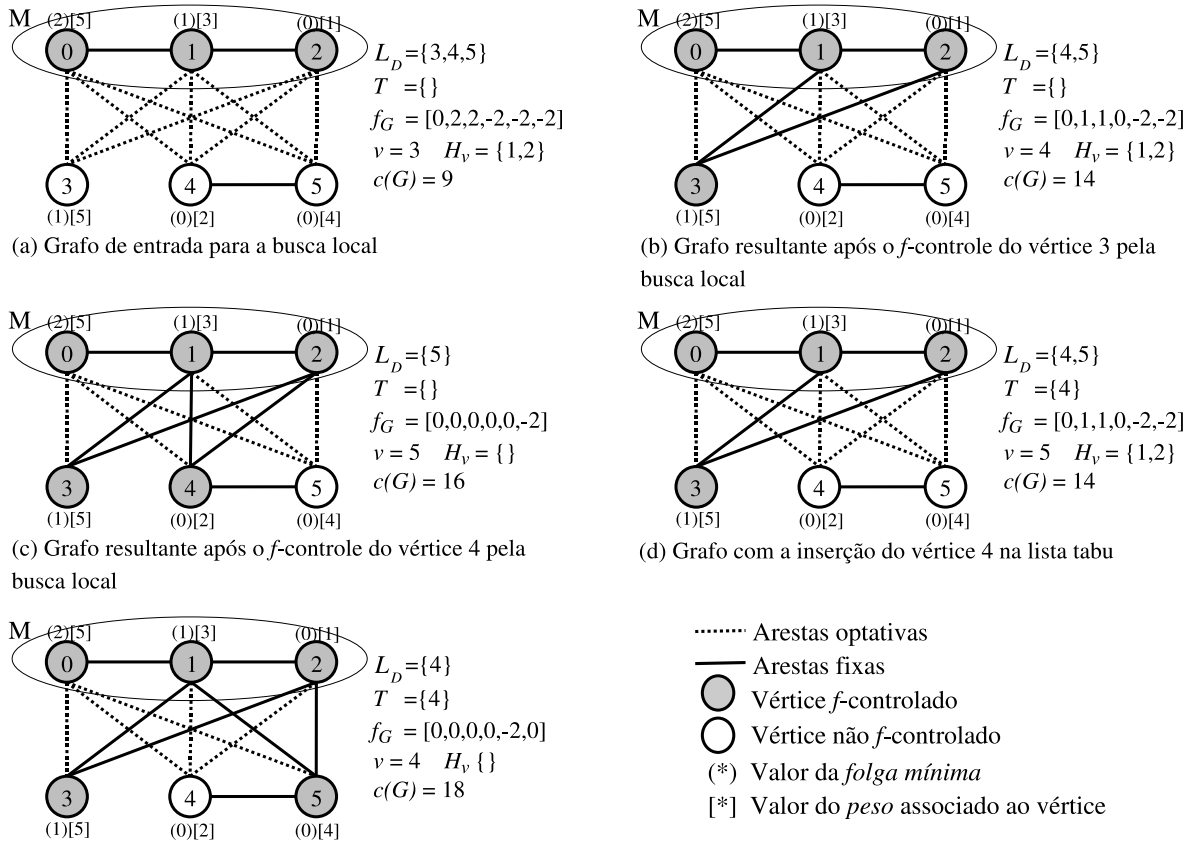
A lista tabu T é construída da seguinte forma: sempre que um grafo sanduíche G estiver associado a um máximo local, descontrolamos arbitrariamente um vértice $v \in L_G$,

inserindo-o em seguida no final da lista tabu. Os novos vértices adicionados deverão permanecer em T por $|T|$ iterações. O descontrole de v é realizado da seguinte forma: se $v \in M_G$, inserimos todas as arestas optativas incidentes a v. Caso contrário, se $v \in U_G$, removemos todas as arestas optativas incidentes a v. Após a inserção ou remoção de arestas optativas, computamos o novo custo associado obtido.

Como exemplo da aplicação da busca local, considere como entrada o grafo ilustrado na Figura 3.1(a). Inicialmente não há movimentos presentes na lista tabu, então qualquer vértice pertencente a L_D é candidato a ser f-controlado. Na Figura 3.1(b), o vértice 3 foi escolhido aleatoriamente e se tornou f-controlado pela adição das arestas optativas (1,3) e (2,3). Observe que a aresta optativa $(0, 3)$ não foi utilizada, pois o vértice vizinho 0 não pode auxiliar no f-controle do vértice escolhido. Na seqüência (Figura 3.1(c)), o vértice 4, também escolhido aleatoriamente, se torna f-controlado pela adição das arestas optativas (1,4) e (2,4). Nesse momento, a busca local (com estrutura de vizinhança $N_0(G)$) chegaria ao fim, porque o vértice 5 não pode se tornar f-controlado sem que ocorra o descontrole de um vértice vizinho. Chegamos portanto a um máximo local. Porém imagine que na seqüência o vértice 4 (selecionado aleatoriamente) seja descontrolado e o movimento que leva ao controle do vértice 4 venha a ser inserido na lista tabu, conforme a Figura 3.1(d). Nesse cenário, a busca local continua conforme ilustra a 3.1(e), controlando o vértice 5, gerando inclusive, uma solução de custo melhor que as encontradas anteriormente.

Outras estruturas de vizinhanças bastante naturais (aqui representadas por $N_i(G)$ para $i \geq 1$) podem ser adotadas para o PMCCG. Poderemos efetuar uma busca local, descontrolando-se i vértices f-controlados de L_G, desde que um ganho na função objetivo seja observado. Entretanto, constatamos em testes realizados empiricamente que essas estruturas de vizinhança foram pouco eficientes quando comparadas à estrutura $N_0(G)$, além de exigirem um maior tempo computacional à medida que i aumenta. Esse fato, frustrou uma tentativa inicial de utilizar o método de Busca por Vizinhança Variável (*Variable Neighborhood Search*-VNS [15, 16]) por apresentar baixa eficiência nas estruturas de vizinhança diferentes de $N_0(G)$, isso serviu também como motivação na adoção da BT para o PMCCG, como uma alternativa mais adequada às características do problema. Em [23] discutimos a utilização de várias estruturas de vizinhança (através do método VNS) aplicado ao PMCC.

A busca local que utiliza o critério de aspiração visa gerar soluções melhores que o ótimo global, permitindo a utilização de movimentos presentes na lista tabu. Para o PMCCG, o critério de aspiração consiste em tentar f-controlar vértices presentes na



(a) Grafo de entrada para a busca local

(b) Grafo resultante após o f-controle do vértice 3 pela busca local

(c) Grafo resultante após o f-controle do vértice 4 pela busca local

(d) Grafo com a inserção do vértice 4 na lista tabu

(e) Grafo resultante após o f-controle do vértice 5 pela busca local.

Figura 3.1: Exemplo da aplicação da busca local.

lista tabu (no máximo igual a $|T|$), desde que nenhum outro vértice f-controlado seja descontrolado. Este critério é empregado sempre que o novo custo associado for maior que a solução global. Caso o custo da solução obtida for inferior ao custo da solução global, retornamos à configuração anterior.

Após a aplicação de uma busca local intensiva, executamos uma diversificação da solução corrente S, buscando uma fuga de ótimos locais "já pesquisados". O "grau" de diversificação está relacionado ao peso e número de vértices envolvidos nessa perturbação. A seção seguinte trata desse processo de diversificação.

3.2 Diversificação da solução

Uma busca local deve ser interrompida em uma dada região (após j_{max} iterações) se ela não for capaz de produzir soluções melhores que a solução global $S^{\star}$.

Essa mudança de região pode ser viabilizada através de uma perturbação mais drás-

tica na solução corrente, resultando assim em uma nova semente (solução inicial). No PMCCG, dado um grafo sanduíche G, essa "perturbação drástica" pode ser realizada descontrolando-se simultaneamente e aleatoriamente um subconjunto $K \subseteq L_G$ (onde $|K| \leq k$, sendo k um parâmetro de entrada), de vértices f-controlados por M. Esta operação, obviamente, afeta os demais vértices do grafo, alterando as folgas correntes dos vértices vizinhos associados. Assim, se $v \in K$ temos então, duas possibilidades: para $v \in M_G$, inserimos todas as arestas optativas incidentes a v. Ou caso contrário, se $v \in U_G$, removemos todas as arestas optativas incidentes a v. Note ainda que, se dois vértices v, w pertencem a $K \cap M_R$ e $K \cap U_R$ respectivamente, pode ocorrer que v e w não sejam descontrolados simultaneamente, a Figura 3.2 ilustra essa situação. Caso o vértice v venha a ser descontrolado, o vértice w passará a ser f-controlado. Ainda, caso o vértice w venha a ser descontrolado, o vértice v voltaria a ser f-controlado. A ordem em que os vértices $|K|$ vértices são descontrolados é definida aleatoriamente.

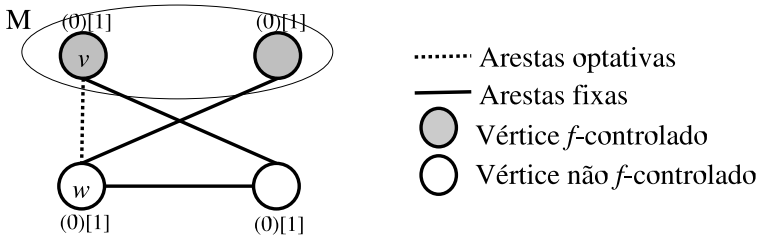


Figura 3.2: Exemplo de um subgrafo onde dois vértices não podem ser descontrolados simultaneamente.

O procedimento de diversificação não possui mecanismos de memória (lista tabu) associada, ele simplesmente gera uma nova solução inicial "diversificada", ou seja, uma solução com atributos distintos das soluções anteriores. Com isso, o objetivo é analisar uma região "distante" da anterior à procura de ótimos locais de melhor qualidade. A nova semente será usada como solução inicial para a busca local descrita anteriormente.

3.3 Reconexão por Caminhos

A técnica de Reconexão por Caminhos - RC (*Path Relinking*) introduzida inicialmente por Glover&Laguna [9] em conjunto com o método de Busca Tabu, é uma maneira de explorar trajetórias entre soluções elites. A idéia fundamental deste método é que boas soluções para um problema devem possuir características semelhantes. Dessa maneira a geração dos caminhos (seqüência de soluções intermediárias) entre soluções elites espera encontrar soluções melhores [8, 11].

A RC pode ser considerada como um método evolutivo, onde as soluções são geradas através da combinação dos elementos de outras soluções. Ao contrário de outros procedimentos (como algoritmos genéticos) que consideram movimentos aleatórios na geração de soluções, a RC utiliza regras sistemáticas e determinísticas combinando soluções. Para gerar um caminho, é necessário a escolha de uma *solução origem* (representando o ponto de partida) e uma *solução alvo* (representando o ponto de chegada) selecionadas em um conjunto *elite* de soluções. Durante a geração do caminho os atributos da solução alvo são gradativamente incluídos nas soluções intermediárias, enquanto que os atributos da solução origem vão sendo desprezados. Atributos idênticos devem permanecer inalterados durante todo o processo.

Uma implementação de RC deve atentar para algumas características importantes: construção do conjunto elite, escolha das soluções origem e alvo e regras para a movimentação entre as soluções.

A qualidade e o nível de diversidade das soluções presentes no conjunto elite tem forte influência na qualidade das soluções geradas pela RC [11].

As escolhas das soluções origem e alvo também são importantes para garantir a qualidade e performance das novas soluções geradas e pode ocorrer de diversas maneiras: pode-se, por exemplo, considerar a melhor e pior solução presente na elite, a melhor e a segunda melhor solução, a melhor e a solução mais diversificada (segundo uma dada métrica) em relação à melhor solução, ou podem ser escolhidas randomicamente [11].

No PMCCG, consideramos como atributos, a informação associada aos vértices f-controlados ou não por M. Assim, dado um grafo sanduíche $G = (V, E)$ construímos um vetor binário S associado, indicando quais vértices são ou não f-controlados por $M \subseteq V$. Dessa forma, teremos $S[i] = 1$, se e somente se, o vértice $i \in V$ é f-controlado por M, e $S[i] = 0$, caso contrário. O primeiro passo consiste em identificar os vértices com atributos diferentes entre duas soluções selecionadas do conjunto elite (representadas respectivamente por S_{origem} e S_{alvo}). Em seguida adiciona-se gradativamente na solução origem (um vértice por vez) o atributo correspondente da solução alvo. Ao final da primeira iteração escolhe-se a melhor solução encontrada e fixa-se o atributo correspondente a essa melhora. O procedimento ocorre sucessivamente percorrendo-se o caminho de S_{origem} a S_{alvo}. A Figura 3.3 ilustra conceitualmente a aplicação da RC. Os vértices mais escuros correspondem às melhores soluções intermediárias encontradas. Assim, podemos ver como um caminho é percorrido e a geração das soluções intermediárias (possivelmente encontrando soluções com custo melhor que a solução global).

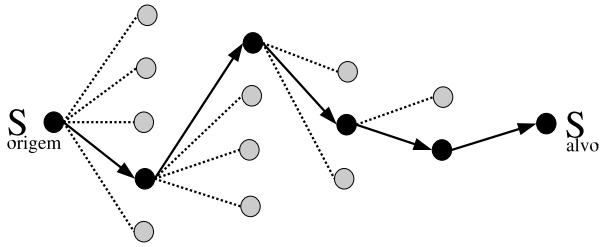


Figura 3.3: Exemplo da aplicação da reconexão por caminhos.

Neste trabalho, o conjunto elite é gerado pela busca local durante a BT e consiste das p melhores soluções encontradas pela BT, onde p é um parâmetro de entrada. Optamos por utilizar na RC as soluções com maior diversidade, ou seja, a solução origem corresponde à melhor solução encontrada pela BT e a solução alvo será aquela presente na elite com maior diversidade em relação à solução origem.

O Algoritmo 6 demonstra a aplicação da RC para o PMCCG. Considere dois grafos sanduíche, G_{origem} e G_{alvo} associados a S_{origem} e S_{alvo} respectivamente. No algoritmo, S_{origem} representa a melhor solução encontrada pela BT e S_{alvo} representa a solução mais distinta de S_{origem}, presente no conjunto elite (linha 1). A variável R representa o conjunto dos vértices com atributos distintos entre as soluções S_{origem} e S_{alvo} (linha 2). Considere que na linha 6, a coordenada $S_{alvo}[i]$ retorna o atributo associado ao vértice i da solução alvo, ou seja, o vértice i na solução S' irá assumir os atributos correspondentes do vértice i da solução alvo. Na linha 10, a coordenada $S_{origem}[i]$ restaura os atributos associados ao vértice i da solução origem. Dessa maneira, ao modificar o atributo de um vértice i de 1 para 0 (f-controlado para não f-controlado) simplesmente adiciona-se (respectivamente remove-se) todas as arestas incidentes ao vértice $i \in M_R$ (respectivamente $i \in U_R$). Entretanto, para a mudança de 0 para 1 (não f-controlado para f-controlado) é necessário utilizar o mesmo conjunto de arestas incidentes a i de acordo com o grafo G_{alvo} (solução alvo). Após verificar a aplicação (individual) dos atributos presentes na solução alvo, escolhe-se (fixa) aquela solução intermediária que apresentou o melhor custo (linha 12). As atualizações das melhores soluções intermediárias ocorrem nas linhas 7 e 8 e nas linhas 13 e 14 ocorre a atualização da solução global do algoritmo, caso a RC consiga melhora-la.

Algoritmo 6 Reconexão por Caminhos para o PMCCG

1: $S_{origem} \leftarrow S^{\star}$, $S_{alvo} \leftarrow$ solução mais distante em relação a $S\star$
2: $R \leftarrow$ conjunto dos vértices com atributos distintos entre S_{origem} e S_{alvo}
3: **enquanto** $R \neq \oslash$ **faça**
4: $S' \leftarrow S_{origem}$, $S'' \leftarrow \oslash$
5: **para** $i \in R$ **faça**
6: $S'[i] \leftarrow S_{alvo}[i]$
7: **se** $c(S') > c(S'')$ **então**
8: $S'' \leftarrow S'$
9: **fim se**
10: $S'[i] \leftarrow S_{origem}[i]$
11: **fim para**
12: $S_{origem} \leftarrow S''$
13: **se** $c(S_{origem}) > c(S^{\star})$ **então**
14: $S^{\star} \leftarrow S_{origem}$
15: **fim se**
16: $R \leftarrow R \backslash \{i\}$
17: **fim enquanto**
18: *retorna* $S^{\star}$

Capítulo 4

Resultados computacionais

Para efeito de avaliação de nossas heurísticas para o PMCCG, utilizamos o pacote *open source* GNU/GLPK versão 4.8, para geração de soluções exatas em instâncias contendo 50, 75 e 100 vértices. Todos os algoritmos foram implementados usando a linguagem C, seguindo o padrão C ANSI, com compilador gcc versão 3.3.2 em ambiente Linux (distribuição Mandrake 9.1 e Fedora Core 2). Os testes foram executados em máquinas similares com processadores Pentium IV, 2.60Ghz com 512 Mb de memória RAM em ambiente compartilhado.

Todas as instâncias testadas foram geradas aleatoriamente obedecendo-se aos seguintes parâmetros (definidos empiricamente): fixamos em 27% a probabilidade de um vértice pertencer ao conjunto M. Em 70% a probabilidade de uma aresta ser fixa ou optativa, sendo que dentre este total de arestas selecionadas, 80% delas são optativas. Cada vértice teve seu peso e folga mínima definidos aleatoriamente dentro de um intervalo pré determinado. As instâncias geradas, tem nomenclatura baseada nesses parâmetros, de acordo com o seguinte modelo: "G Quantidade de vértices-Peso máximo-Folga máxima-Número identificador da instância". Como exemplo, considere a primeira instância gerada com 100 vértices, com intervalo de pesos entre 1-20 e folga mínima com intervalo 0-10, seu nome será dado por: G100-20-10-01.

No procedimento de geração dessas instâncias esperamos que o PVM retorne "NÃO" como solução. Dessa maneira, é importante que sempre exista pelo menos um vértice nunca controlado. Isto pode ser implementado adicionando-se, por exemplo, estruturas (subgrafos) onde é impossível o f-controle de dois vértices simultaneamente, conforme ilustrado no capítulo anterior na Figura 3.2. Outra possibilidade é simplesmente adicionarmos um vértice nunca f-controlado fazendo simplesmente $f_i = +\infty$. Assim, geramos instâncias para o PMCCG onde M nunca define um f-monopólio.

A Tabela 4.1 ilustra execuções distintas do Algoritmo 2 utilizando as relaxações $\bar{P}$ e $\tilde{P}$ separadamente (conforme descrito nas Seções 2.3 e 2.4). Note que uma formulação mais forte para o PMCCG interfere não somente no valor do limite superior (Teorema 2) mas afeta também no valor da solução heurística. Observe que os melhores valores obtidos pelo Algoritmo 2 utilizaram as soluções determinadas pela relaxação linear mais forte. De agora em diante, no Algoritmo 2, utilizaremos apenas as soluções obtidas a partir de $\bar{P}$.

Tabela 4.1: Tabela de comparação das soluções obtidas pelo Algoritmo 2 utilizando as relaxações $\bar{P}$ e $\tilde{P}$.

Instância	**Relaxação $\tilde{P}$**	**Algoritmo 2 com $\tilde{P}$**	**Relaxação $\bar{P}$**	**Algoritmo 2 com $\bar{P}$**	**Solução ótima**
G50-10-5-01	96,43	81	90,58	85	87
G50-10-5-02	201,08	173	190,89	176	185
G50-10-5-03	220,23	195	212,38	198	207
G50-10-5-04	268,93	267	268,60	268	268
G50-10-5-05	197,96	130	180,23	151	161
G100-20-10-01	403,81	319	389,20	363	379
G100-20-10-02	524,06	325	464,04	338	401
G100-20-10-03	1.007,79	865	963,94	940	–
G100-20-10-04	501,36	321	467,58	420	439
G100-20-10-05	975,31	844	930,94	894	–

A Tabela 4.2 apresenta os resultados obtidos para algumas instâncias geradas aleatoriamente, com o conhecimento das respectivas soluções ótimas. As regras de redução foram aplicadas em todas as instâncias testadas. A porcentagem de redução do número de arestas optativas, é apresentada na coluna *Reg. de Redução*. Pelos resultados obtidos é possível notar a importância da aplicação das regras de redução, chegando a reduzir o número de arestas optativas, em alguns casos, em mais de 90%. As soluções iniciais geradas são apresentadas nas colunas *Alg.1* e *Alg.2* e correspondem à solução obtida pelos Algoritmos 1 e 2, respectivamente, sendo que a solução inicial considerada pela BT, será aquela com melhor custo obtido entre as duas (Algoritmo 3), representada em negrito. Note que, para a maioria dos casos testados, a solução obtida através da relaxação linear (Algoritmo 2) é melhor que a solução baseada no Algoritmo 1.

O algoritmo da BT foi aplicado 10 vezes para cada instância, com os parâmetros i_{max} e j_{max} definidos empiricamente em 5 e 50, respectivamente. A capacidade da lista tabu corresponde a 5% do total de vértices em $M_R \cup U_R$ e o procedimento de diversificação descontrola no máximo 5% do total dos vértices f-controlados por M em uma solução corrente G. As soluções da BT são apresentadas na tabela da seguinte maneira: a coluna *Melhor valor* apresenta o melhor valor obtido entre as execuções da BT (o valor entre

parênteses, descreve o número de vezes que esta solução foi obtida entre as 10 execuções e o símbolo ($\star$) indica aquelas instâncias onde o valor ótimo foi obtido). A coluna *Média* descreve a média entre as soluções obtidas em todas as execuções. A solução ótima da instância, obtida com o auxílio do pacote GNU/GLPK, é exibida na coluna *Ótimo*. Finalmente, a coluna *Tempo* traz o pior tempo obtido, em segundos, gasto pela execução da BT.

Tabela 4.2: Tabela de resultados da BT para instâncias com 50, 75 e 100 vértices, conhecendo o custo da solução ótima.

Instância	**Reg. de Redução**	**Solução Inicial**		**Busca Tabu**		**Ótimo**	**Tempo(s)**
		Alg.1	Alg.2	Melhor valor	Média		
G50-10-5-01	69,22%	151	**176**	184 (2)	181,05	185	12,28
G50-10-5-02	65,25%	165	**198**	201 (2)	198,04	207	5,29
G50-10-5-03	59,75%	172	**268**	$\star$ 268 (10)	268	268	6,52
G50-10-5-04	65,38%	136	**151**	160 (2)	155,65	161	0,32
G50-10-5-05	59,57%	151	**218**	218 (10)	218	219	0,50
G75-15-7-01	81,04%	374	**391**	415 (1)	401,20	416	1,31
G75-15-7-02	51,40%	372	**565**	$\star$ 565 (10)	565	565	18,40
G75-15-7-03	57,47%	370	**509**	515 (1)	509,55	521	0,83
G75-15-7-04	79,75%	291	**297**	318 (2)	304,05	320	13,09
G75-15-7-05	62,38%	398	**535**	545 (3)	539,15	548	0,96
G100-20-10-01	95,73%	329	**363**	374 (3)	364,45	379	35,21
G100-20-10-02	91,02%	**360**	338	395 (1)	385,18	401	8,94
G100-20-10-03	94,10%	**354**	340	384 (3)	377,78	388	24,68
G100-20-10-04	88,12%	362	**420**	435 (2)	422,50	439	2,97
G100-20-10-05	81,26%	514	**578**	$\star$ 602 (2)	597,94	602	1,15

A Tabela 4.3 apresenta resultados de instâncias maiores, com 300, 500 e 1000 vértices, os parâmetros utilizados na geração das instâncias permaneceram inalterados sofrendo modificações somente os intervalos dos pesos e folgas mínimas associados a cada vértice. A BT também foi executada sob a mesma configuração de parâmetros, conforme descrito anteriormente.

Esta tabela, diferentemente da Tabela 4.2, apresenta a coluna *Aprox.*, em substituição aos valores ótimos obtidos pelo método exato, impraticável em instâncias "grandes". Dessa maneira, calculamos a aproximação da solução obtida pela BT, baseando-se no valor da relaxação linear (limite superior) correspondente à instância testada. Assim, podemos ter uma melhor noção da qualidade da solução heurística obtida. A coluna *Aprox.* apresenta o valor médio das aproximações obtidas pelas execuções da BT.

Os resultados obtidos confirmam o melhor desempenho do Algoritmo 2 em relação ao Algoritmo 1. Além disso, os custos obtidos pela BT, para as instâncias consideradas, se

encontram com aproximação superior a 0,94 do valor ótimo.

Tabela 4.3: Tabela de resultados da BT para instâncias com 300, 500 e 1000 vértices.

Instância	Reg. de Redução	Solução Inicial		Busca Tabu		Aprox.	Tempo(s)
		Alg.1	Alg.2	Melhor valor	Média		
G300-30-20-01	58,94%	2.136	**2.845**	2.847 (1)	2.845,20	0,9904	0,51
G300-30-20-02	56,71%	3.200	**4.422**	4.422 (10)	4.422,00	0,9966	0,52
G300-30-20-03	61,59%	2.371	**2.949**	2.957 (1)	2.950,90	0,9796	0,58
G300-30-20-04	61,69%	3.212	**3.949**	3.949 (10)	3.949,00	0,9465	0,24
G300-30-20-05	60,00%	3.158	**3.807**	3.845 (1)	3.832,35	0,9462	0,31
G500-50-30-01	60,84%	8.960	**10.723**	10.788 (1)	10.744,35	0,9487	1,57
G500-50-30-02	59,76%	8.858	**11.247**	11.247 (10)	11.247,00	0,9822	3,58
G500-50-30-03	58,75%	9.353	**12.119**	12.119 (10)	12.119,00	0,9842	4,15
G500-50-30-04	62,55%	9.061	**10.614**	10.652 (1)	10.617,40	0,9460	2,21
G500-50-30-05	57,80%	8.950	**12.166**	12.179 (1)	12.170,90	0,9879	5,01
G1000-100-50-01	61,48%	36.997	**44.697**	44.957 (1)	44.741,25	0,9696	3,23
G1000-100-50-02	60,96%	36.476	**45.251**	45.414 (1)	45.381,10	0,9701	2,98
G1000-100-50-03	60,40%	36.261	**45.155**	45.374 (1)	45.322,30	0,9773	2,97
G1000-100-50-04	59,35%	37.278	**47.895**	47.947 (2)	47.910,00	0,9887	3,59
G1000-100-50-05	61,69%	38.039	**44.628**	44.818 (1)	44.776,40	0,9615	2,25

Na Tabela 4.4 obtemos resultados para instâncias do PMCCG contendo 300, 500, 1000 e 2000 vértices, utilizando a BT com RC. A tabela possui uma nova coluna *Rec. por Caminhos* que apresenta a melhor solução obtida pela RC. O cálculo da aproximação (coluna *Aprox.*) é baseado na melhor solução retornada pelo algoritmo (BT com RC), dividido por um limite superior para o valor ótimo. Na coluna *Tempo* verificamos o tempo (em segundos) gasto pela BT com RC. As instâncias geradas obedecem às mesmas probabilidades conforme descrito anteriormente e a BT também foi executada conforme os parâmetros já definidos.

Em virtude das soluções obtidas pela BT apresentarem normalmente uma aproximação elevada, a RC não apresentou melhoras para todas as instâncias consideradas. Podemos perceber isso sobretudo nas instâncias contendo 300 e 500 vértices. Mesmo nas instâncias maiores, essa melhora ocorre em um número reduzido de instâncias. A Tabela 4.4 apresenta algumas instâncias onde a RC incrementou as soluções obtidas pela BT.

Para as instâncias com número de vértices superior a 1000, o cálculo da relaxação linear se tornou inviável em virtude do tempo computacional elevado. Para essas instâncias consideramos como solução inicial a solução dada pelo Algoritmo 1.

Utilizando a razão de aproximação do Algoritmo 1 como referência, pode-se determinar, em alguns casos, uma nova razão de aproximação da solução ótima obtida após a execução de um método de busca local. De fato, as soluções obtidas pelo Algoritmo

1 estão a pelo menos 50% da solução ótima. Em alguns casos, se $z_{H1} > \frac{LS}{2}$ para um grafo analisado, significa que a nova razão de aproximação RA será superior a $\frac{1}{2}$. Como $z_{max} \leq LS$, sempre que $z_{H1} > \frac{LS}{2}$ pode-se calcular uma nova aproximação através de uma regra de três simples, caso contrário a aproximação irá permanecer igual a $\frac{1}{2}$. Como exemplo, considere a situação onde um grafo possui $LS = 95$ e um algoritmo de busca local retorna $z = 70$. Após a aplicação de uma regra de três uma nova aproximação será dada pela equação 4.1

$$\begin{aligned} RA &= \frac{2 \times (70 \times 0,5)}{95} \\ RA &= 0,73 \end{aligned} \tag{4.1}$$

Sem o valor da relaxação linear, modificamos o cálculo da aproximação das soluções, passando a considerar como limite superior o custo de maior número possível de vértices a serem f-controlados dado por $LS = \sum_{i \in V} p_i - \sum_{j \in M_{NC} \cup U_{NC}} p_j$. Como LS representa um limite superior mais fraco em relação à solução obtida através da relaxação linear e a solução gerada pelo Algoritmo 1 é geralmente pior que a solução gerada pelo Algoritmo 3, temos uma conseqüente queda nos valores obtidos para a aproximação das soluções.

A Tabela 4.5 apresenta alguns resultados obtidos para instâncias de tamanho 50, 75 e 100 vértices para o PMCC (caso particular do PMCCG que considera $p_i = 1$ e $f_i = 0 \; \forall \; i \in V$). Nota-se através dos resultados obtidos que os algoritmos utilizados para o PMCCG se comportam muito bem também para o PMCC, chegando ao valor ótimo para várias das instâncias testadas.

Podemos avaliar a importância da geração de uma boa solução inicial (semente) através da Tabela 4.6, que compara as soluções obtidas pela BT com RC considerando soluções iniciais fornecidas pelos algoritmos 1 e 2 (executados com parâmetros idênticos: tamanho da lista tabu, porcentagem de diversificação, número de iterações). Na Tabela, a coluna *Alg.1* corresponde ao custo associado à solução inicial gerada pelo Algoritmo 1, e a coluna *Alg.2* ao custo obtido pela solução gerada pelo Algoritmo 2, respectivamente, temos a solução obtida pela BT com PR nas colunas *TB/RC-Alg.1* e *TB/RC-Alg.2*.

A seguir apresentamos alguns histogramas para ilustrar o comportamento da BT. Para isso consideramos três instâncias contendo 300, 500 e 1000 vértices, geradas conforme às probabilidades já especificadas. Os pesos e folgas mínimas variaram de acordo com o tamanho da instância testada. Os pesos associados aos vértices assumiram valor de no

Tabela 4.4: Tabela de resultados da BT com Reconexão por Caminhos para instâncias com 300, 500, 1000 e 2000 vértices.

Instância	Reg. de Redução	Solução Inicial		Busca Tabu	Rec. por Caminhos	Aprox.	Tempo(s)
		Alg.1	Alg.2				
G300-30-3-1	62.50%	4.140	**4.418**	4.422	4.422	0,9850	0,53
G300-30-3-2	59.87%	3.463	**4.625**	4.625	4.625	0,9943	0,52
G300-30-3-3	61.01%	3.729	**4.746**	4.749	4.749	0,9931	0,58
G500-50-5-1	58.89%	8.667	**12.269**	12.269	12.269	0,9990	1,86
G500-50-5-2	61.52%	9.873	**12.699**	12.705	12.705	0,9948	2,00
G500-50-5-3	60.77%	9.123	**12.133**	12.134	12.134	0,9928	2,34
G1000-100-10-1	59,87%	35.765	**48.231**	48.316	48.399	0,9975	9,99
G1000-100-10-2	60,05%	36.961	**49.942**	50.148	50.231	0,9961	9,21
G1000-100-10-3	62,04%	38.858	**48.218**	48.801	48.804	0,9902	12,38
G1000-100-10-4	60,12%	38.229	**50.984**	51.072	51.093	0,9959	10,24
G1000-100-10-5	60,85%	37.131	**48.319**	48.669	48.705	0,9933	9,94
G1000-100-10-6	59,92%	37.107	**49.285**	49.388	49.419	0,9965	9,58
G1000-100-10-7	59,22%	35.908	**48.884**	49.431	49.442	0,9950	8,00
G2000-200-20-1	59,61%	147.040	–	189.180	189.337	0,9287	2.981
G2000-200-20-2	60,94%	145.401	–	181.085	181.180	0,9088	54,21
G2000-200-20-3	59.03%	139.436	–	184.205	184.360	0,9418	50,30
G2000-200-20-4	59.14%	141.374	–	186.677	186.794	0,9395	2.049
G2000-200-20-5	59.30%	144.686	–	187.512	187.710	0,9382	2.920
G2000-200-20-6	61.14%	146.182	–	180.753	180.947	0,9039	4.452
G2000-200-20-7	60.15%	149.287	–	186.041	186.153	0,9179	3.675

máximo 10% e a folga mínima valor de no máximo de 1%.

Para efeito de comparação, variamos e combinamos os parâmetros correspondentes ao tamanho da lista tabu e ao número de vértices a serem descontrolados pelo procedimento de diversificação, buscando verificar de que maneira esses parâmetros afetam o comportamento da BT. Para o parâmetro correspondente ao tamanho da lista tabu, assumimos os valores correspondentes a 1% ou 5% em relação à quantidade de vértices f-controlados na solução inicial (semente). O número de vértices a serem descontrolados no procedimento de diversificação foi definido em 5% ou 20%, também em relação ao número de vértices f-controlados dado pela solução inicial.

Os títulos de cada gráfico nos informam sobre os parâmetros utilizados na execução. A quantidade de vértices presentes na instância testada é expressa logo após a palavra "Grafo". O valor associado à letra "D", refere-se ao número de vezes (i_{max}) que o procedimento de diversificação foi empregado durante a execução do algoritmo. O valor relacionado à letra "I" descreve o valor do critério de parada, ou seja, o número máximo de interações sem melhora (j_{max}) considerado pela busca local. O tamanho da lista tabu é descrito em porcentagem junto à letra "T" e finalmente, associado à letra "d" temos, em

Tabela 4.5: Tabela de resultados da BT para instâncias do PMCC com 50, 75 e 100 vértices, conhecendo o custo da solução ótima.

Instância	**Reg. de Redução**	**Solução Inicial**		**Busca Tabu**		**Ótimo**	**Tempo(s)**
		Alg.1	Alg.2	Melhor valor	Média		
G50-1-0-01	63,28%	32	**39**	$\star$ 41 $_{(1)}$	40,10	41	0,01
G50-1-0-02	60,44%	32	**46**	$\star$ 46 $_{(10)}$	46	46	0,01
G50-1-0-03	66,66%	34	**45**	$\star$ 45 $_{(10)}$	45	45	0,01
G50-1-0-04	59,30%	34	**42**	$\star$ 46 $_{(4)}$	44,85	46	0,01
G50-1-0-05	60,58%	32	**48**	$\star$ 48 $_{(10)}$	48	48	0,01
G75-1-0-01	60,82%	55	**68**	$\star$ 68 $_{(10)}$	68	68	0,01
G75-1-0-02	71,11%	50	**53**	53 $_{(10)}$	53	54	0,02
G75-1-0-03	64,63%	48	**62**	$\star$ 62 $_{(10)}$	62	62	0,02
G75-1-0-04	70,87%	44	**50**	$\star$ 52 $_{(3)}$	50,60	52	0,02
G75-1-0-05	70,51%	51	**55**	55 $_{(10)}$	55	57	0,02
G100-1-0-01	59,24%	70	**90**	90 $_{(10)}$	90	91	0,02
G100-1-0-02	75,58%	57	**61**	61 $_{(10)}$	61	62	0,03
G100-1-0-03	63,35%	68	**88**	$\star$ 88 $_{(10)}$	88	88	0,02
G100-1-0-04	66,92%	72	**81**	81 $_{(10)}$	81	82	0,03
G100-1-0-05	60,64%	70	**92**	92 $_{(10)}$	92	94	0,02

porcentagem, a quantidade de vértices descontrolados pelo procedimento de diversificação.

A análise é feita para uma execução do algoritmo BT com RC, para cada combinação de parâmetros e cada instância. Nos gráficos apresentados, o eixo horizontal (das abscissas) informa a iteração do algoritmo e o eixo vertical (ordenadas) o custo obtido pela solução dada pela busca local. Podemos observar que geralmente as melhores soluções obtidas são encontradas a partir das sementes geradas pelo Algoritmo 3. Nas execuções com diversificação acentuada (em 20%) pode-se perceber no gráfico uma forte queda do custo das soluções. Dessa maneira, uma diversificação muito forte tende a atrapalhar o procedimento de busca, assim, uma diversificação de 5% dos vértices f-controlados, se mostrou mais adequada para o problema. Na avaliação do tamanho da lista lista tabu, verificamos pelos resultados obtidos que uma lista com tamanho em 1% é mais apropriada ao problema. Apesar do critério de aspiração, uma lista tabu muito grande pode impedir que um ou mais vértices com alto custo associado ("importancia") sejam f-controlados simultaneamente, dessa maneira verificamos uma queda do custo médio das soluções obtidas, além de uma tendência em evitar que a busca local se recupere de ótimos locais ruins.

Tabela 4.6: Tabela de comparação entre soluções obtidas pela BT com RC considerando soluções iniciais geradas pelos algoritmos 1 e 2.

Instância	Alg.1	TB/RC Alg.1	Tempo(s)	Alg.2	TB/RC Alg.2	Tempo(s)
G300-30-3-1	3.175	4.081	1.15	4.290	4.291	0.47
G300-30-3-2	3.509	4.201	2.05	4.484	4.488	0.36
G300-30-3-3	3.468	4.410	1.05	4.698	4.698	0.31
G500-50-5-1	9.061	11.372	1.60	12.200	12.200	1.66
G500-50-5-2	8.439	11.433	1.50	11.890	11.890	1.22
G500-50-5-3	9.161	11.900	10.93	12.522	12.522	1.39
G1000-100-10-1	36.346	43.752	5.53	47.105	47.121	5.42
G1000-100-10-2	38.338	45.038	117.68	48.115	48.129	5.11
G1000-100-10-3	36.957	43.857	131.16	47.054	47.123	7.32
G1200-120-12-1	54.236	66.794	12.57	70.659	70.725	13.47
G1200-120-12-2	52.779	66.475	297.44	69.242	69.806	19.78
G1200-120-12-3	51.673	66.145	393.92	69.410	69.776	12.65

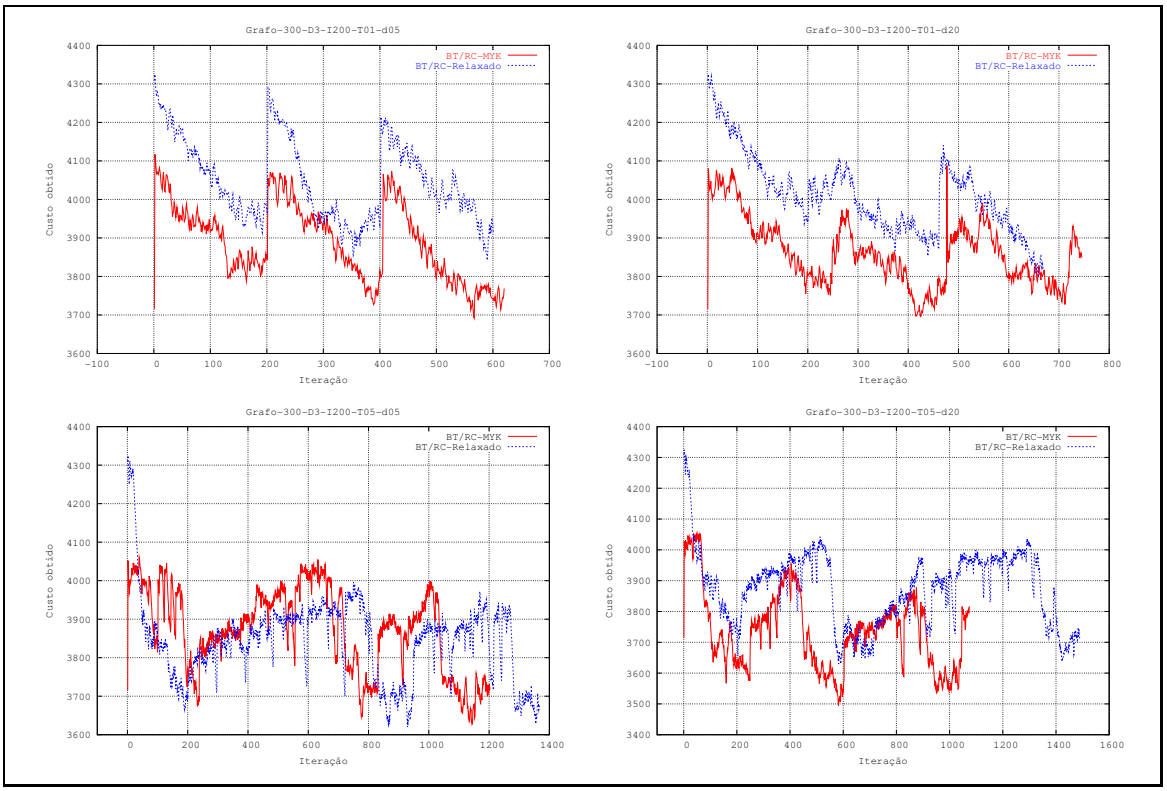


Figura 4.1: Histograma da execução da BT para uma instância contendo 300 vértices.

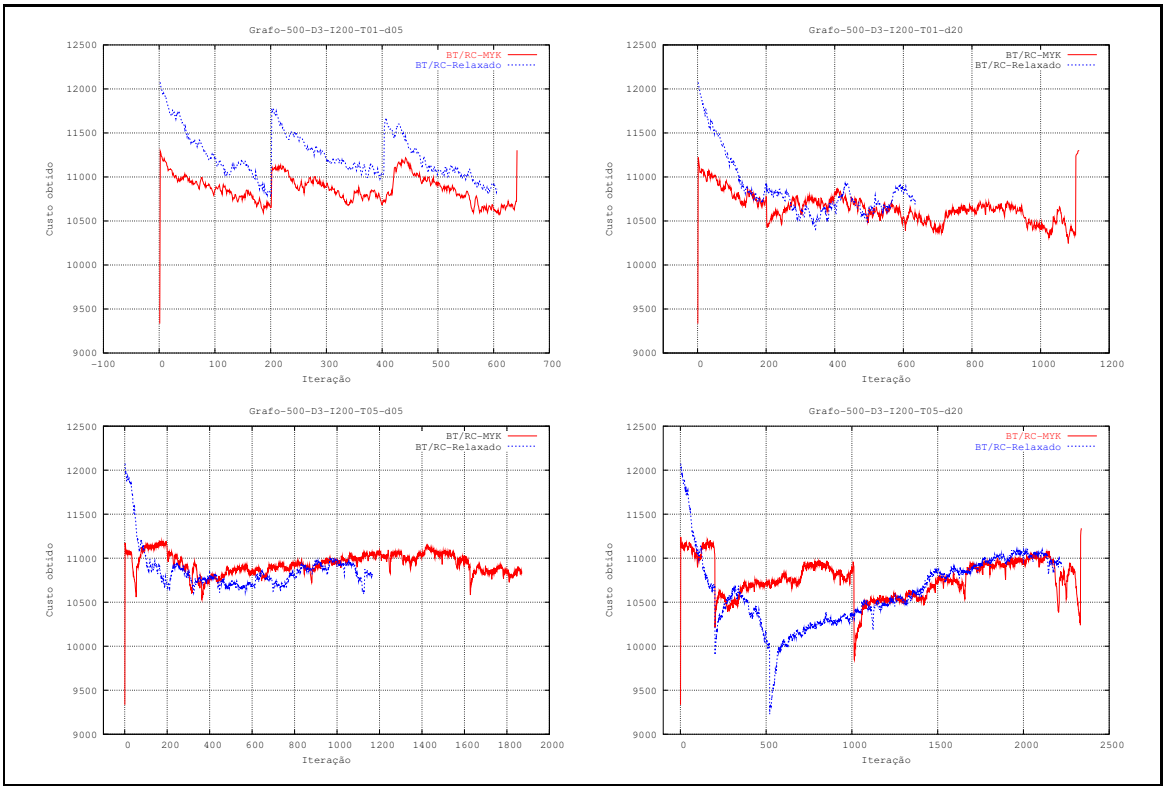


Figura 4.2: Histograma da execução da BT para uma instância contendo 500 vértices.

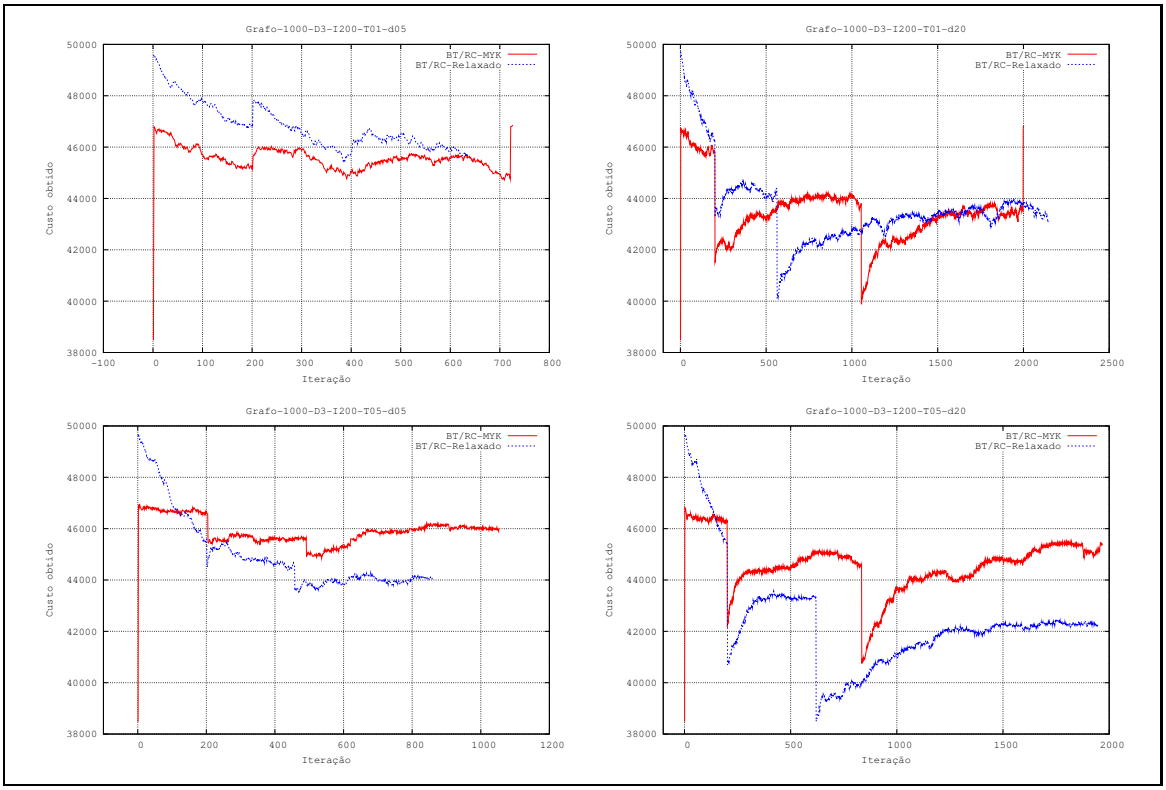


Figura 4.3: Histograma da execução da BT para uma instância contendo 1000 vértices.

Capítulo 5

Conclusões e propostas de trabalhos futuros

Neste trabalho, apresentamos uma generalização do Problema do Maior Conjunto Controlado (introduzido por Makino *et. al.* [19]) denominada Problema do Maior Conjunto Controlado Generalizado - PMCCG. Apresentamos um algoritmo $\frac{1}{2}$-aproximado para o PMCCG e um procedimento para a geração de soluções viáveis baseado na solução de uma relaxação linear para o problema. Essas soluções foram utilizadas em um procedimento de busca local e pós-otimização (Busca Tabu com Reconexão por Caminhos), visando a determinação de soluções de melhor qualidade. Finalmente, apresentamos alguns resultados computacionais e comparamos os resultados obtidos com o valor ótimo encontrado para pequenas instâncias do problema, obtendo em média, soluções com o custo muito próximo da solução ótima. Para as instâncias maiores (sem custo ótimo conhecido), os resultados obtidos pela Busca Tabu estiveram a uma distância inferior ou igual a 6% da solução ótima.

Como trabalhos futuros, acreditamos que seja possível desenvolver uma BT com parâmetros ajustados dinamicamente às características de cada instância. Por exemplo, aumentar ou diminuir a porcentagem de vértices descontrolados pela diversificação, de acordo com o conjunto de soluções geradas previamente, tamanho da lista tabu, critério de parada, etc. Algumas outras estratégias podem ser adotadas no procedimento de busca local, como por exemplo, garantir que vértices freqüentemente f-controlados em boas soluções estejam presentes em qualquer solução, ou ainda que no procedimento de diversificação, não ocorra o descontrole de vértices importantes (com pesos associados elevados).

Outra possibilidade é generalizarmos ainda mais o problema considerando também vértices com pesos negativos. Este caso é mais complicado já que as arestas optativas entre dois vértices de M ou de U respectivamente, não podem ser eliminadas diretamente pelas

regras de redução, interferindo portanto em todas as heurísticas de construção e de busca local vistas anteriormente. Note por exemplo que, se $(i, j) \in D(M, M)$ (respectivamente $(i, j) \in D(U, U)$) e $p_i \times p_j < 0$, não temos um mecanismo seguro (nas regras de redução) para decidir sobre a fixação ou não da aresta optativa (i, j).

Outra possibilidade interessante de pesquisa é a determinação da razão de aproximação obtida pelo algoritmo de construção baseado na solução da relaxação linear (Algoritmo 2). Os resultados computacionais realizados indicaram uma nítida superioridade desta heurística em relação ao algoritmo $\frac{1}{2}$-aproximado (Algoritmo 1) baseado nas idéias apresentadas em Makino *et. al.* [19].

Finalmente, conforme observado nos resultados da Tabela 4.1 e buscando ainda um maior incremento das soluções obtidas pelo Algoritmo 2, podemos estudar o impacto da inclusão de novas desigualdades válidas para o PMCCG.

Referências

[1] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. *Network Flows: theory, algorithms and applications*, 1 ed. Prentice-Hall, 1993.

[2] Bermond, J. C., Bond, J., Peleg, D., and Perennes, S. The power of small coalitions in graphs. *Discrete Appl. Math. 127*, 3 (2003), 399–414.

[3] Carrano, A. V. Establishing the order of human chromosome-specific dna. In *Biotechnology and the Human Genome*, A. D. Woodhed and B. J. Barnhart, Eds. Plenum Press, 1988, p. 37–50.

[4] Fitoussi, D., and Tennenholtz, M. Minimal social laws. In *Proc. AAAI'98* (1998), p. 26–31.

[5] Gendreau, M., Soriano, P., and Salvail, L. Solving the maximum clique problem using a tabu tearch approach. *Annals of Operations Research*, 41 (1993), 385–403.

[6] Glover, F. Future paths for integer programming and artificial intelligence. *Computers e Operations Research 13* (1986), 533–549.

[7] Glover, F. Tabu search - part i. *ORSA Journal on Computing 1*, 3 (1989), 190–206.

[8] Glover, F. *Tabu search and adaptive memory programming - advances, applications and challenges.* Interfaces in Computer Science and Operations Research. Kluwer, 1997.

[9] Glover, F., and Laguna, M. *Modern Heuristic Techniques for Combinatoial Problems - Tabu Search.* Blackwell Scientific Publications, Oxford, 1993.

[10] Glover, F., and Laguna, M. *Tabu Search.* Kluwer Academic Publishers, 1998.

[11] Glover, F., Laguna, M., and Martí, R. Fundamentals of scatter search and path relinking. *Control and Cybernetics 29*, 3 (2000), 653–684.

[12] Golumbic, M. C., Kaplan, H., and Shamir, R. On the complexity of dna physical mapping. *Advances in Applied Mathematics 15* (1994), 251–261.

[13] Golumbic, M. C., Kaplan, H., and Shamir, R. Graph sandwich problems. *J. Algorithms 19*, 3 (1995), 449–473.

[14] Golumbic, M. C., and Shamir, R. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *ACM 40* (1993), 1108–1133.

[15] Hansen, P. The steppest ascent mildest descent heuristic for combinatorial programming. In *Congress on Numerical Methods in Combinatorial Optimization* (Capri-Italy, 1986).

[16] Hansen, P., and Mladenovic, N. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130 (2001), 449–467.

[17] Hassin, Y., and Peleg, D. Distributed probabilistic polling and applications to proportionate agreement. *Inf. Comput. 171*, 2 (2001), 248–268.

[18] Linial, N., Peleg, D., Rabinovich, Y., and Saks, M. Sphere packing and local majorities in graphs. *Proc. 2nd Israel Symposium on Theoretical Computer Science, IEEE Computer Soc. Press* (1993), 141–149.

[19] Makino, K., Yamashita, M., and Kameda, T. Max-and min-neighborhood monopolies. *Algorithmica*, 34 (2002), 240–260.

[20] Martinhon, C. A., and Protti, F. An improved derandomized approximation algorithm for the max-controlled set problem. In *WEA LNCC* (2004), vol. 3059, p. 341–355.

[21] Peleg, D. Size bounds for dynamic monopolies. *Discrete Appl. Math. 86*, 2-3 (1998), 263–273.

[22] Peleg, D. Local majorities, coalitions and monopolies in graphs: a review. *Theor. Comput. Sci. 282*, 2 (2002), 231–257.

[23] Santos, I. M., Martinhon, C. A., and Ochi, L. S. A vns metaheuristic for the max-controlled set problem. In *Encontro Regional de Matemática Aplicada e Computacional-ERMAC* (Rio de Janeiro, 2004), vol. 1, SBMAC, p. 30.

[24] Wright, S. J. *Primal-Dual Interior Point Algorithms.* SIAM Publications, Philadelphia, 1997.