Algoritmos aproximados para o Problema do Maior Conjunto Controlado Generalizado

Ivairton Monteiro Santos*, Carlos Alberto Martinhon, Luiz Satoru Ochi

Universidade Federal Fluminense, Departamento de Ciência da Computação
Rua Passo da Pátria, 156 - Bloco E - 3 andar - Boa Viagem
24210-240, Niterói, RJ, Brasil
isantos@ic.uff.br, {mart, satoru}@dcc.ic.uff.br

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, o PMCC, entretanto, só será resolvido em tempo polinomial se $P = NP$. Neste trabalho apresentamos a noção de *f-controle* e introduzimos ainda o Problema do Maior Conjunto Controlado Generalizado (PMCCG), que associa pesos 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 0,5-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 $v \in V$ in *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 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 0.5-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.

1 Introdução

Dado 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 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 [12].

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, 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 $E_2 \backslash E_1$ (arestas optativas). Assim, decidir sobre esse problema, é equivalente a encontrar um grafo sanduíche com arestas E, tal que $E_1 \subseteq E \subseteq E_2$ [11]. Neste trabalho abordamos um tipo especial de problema em grafo sanduíche, o Problema do Maior Conjunto Controlado Generalizado discutido adiante.

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 $|N_G[i] \cap M| \geq |N_G[i]|/2$, onde $N_G[i] = \{i\} \cup \{j \in V | (i, j) \in E\}$ é vizinhança fechada de i.

O conjunto $M \subseteq V$ define um *monopólio* em G se todo vértice $i \in V$ é controlado por M. O Problema de Verificação de Monopólio - PVM

é um problema de decisão que retorna "SIM" caso seja encontrando um grafo sanduíche (se existir), tal que todos os vértices de G estejam controlados por M. Makino *et. al.*[15] demonstram que o grafo sanduíche para o PVM pode ser obtido em tempo polinomial por meio da solução de um Problema de Fluxo Máximo definido convenientemente.

Quando o conjunto $M \subseteq V$ não define um monopólio, ou seja, o problema de decisão PVM retorna "NÃO", tem-se então o 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. O PMCC é NP-difícil, mesmo que G_1 seja um grafo vazio ou G_2 seja um grafo completo [15].

Makino *et. al.*[15] introduzem a definição de vértice *f-controlado*. Dessa maneira, um vértice $i \in V$ será f-controlado por M em um grafo sanduíche 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 representa, de certa forma, o "grau de controle" necessário para que o vértice $i \in V$ seja f-controlado por $M \subseteq V$. Chamaremos de *folga mínima* a constante $f_i \in \mathbb{Z}$ associada ao vértice $i \in V$.

Neste trabalho apresentamos uma extensão do PMCC denominada PROBLEMA DO MAIOR CONJUNTO CONTROLADO GENERALIZADO - PMCCG, que incorpora o conceito de f-controle introduzido por Makino *et. al.*[15] e considera pesos associados a cada vértice $i \in V$. No PMCCG, o objetivo será maximizar o somatório dos pesos dos vértices f-controlados por M.

Na Seção 2, iremos apresentar as regras de redução, discutidas em [15,16], eliminando informações redundantes nos dados de entrada do PMCC. Como veremos adiante, estas regras serão aplicadas igualmente ao PMCCG, bastando substituir a definição de vértice controlado por f-controlado. O PMCCG será tratado na Seção 3. Apresentamos um algoritmo determinístico 0,5-aproximado (Seção 3.1), um modelo matemático para o problema (Seção 3.2) e um algoritmo para obtenção de soluções viáveis a partir da solução da relaxação linear (Seção 3.3). Na Seção 4 introduzimos uma busca local baseada em Busca Tabu com Reconexão por Caminhos (Tabu Search with Path Relinking e na Seção 5 exibimos alguns dos resultados computacionais obtidos e a comparação com o exato. Finalmente, na Seção 6, apresentamos as conclusões e considerações finais.

2 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 arestas pertencentes a $E_2 \backslash E_1$ (arestas optativas), de tal forma que o conjunto de soluções ótimas continue o mesmo.

Descreveremos agora as regras de redução apresentadas em [15,16] para o PMCC. 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 [15]: adicionar a E_1 as arestas de $D(M,M)$ (*Regra de redução 1*) e remover as arestas $D(U,U)$ de E_2 (*Regra de redução 2*), onde $U = V \backslash M$. O novo conjunto E deverá satisfazer a:

$$E \supseteq E_1 \cup D(M,M) \tag{1}$$

$$E \subseteq E_1 \cup D(M,M) \cup D(U,M) \tag{2}$$

Visando a introdução das regras de redução descritas em [16], 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 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 $E_2 \backslash E_1$ 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.

A partição de V descrita acima pode ser realizada, facilmente, após a adição e remoção de todas as arestas optativas de G. Assim, após a adição das arestas de $E_2 \backslash E_1$, identificamos diretamente aqueles vértices em M_{SC} que continuam controlados e os vértices de 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 1) apresentadas em [16]:

- Adicionar a E_1 as arestas $D(M_{SC} \cup M_{NC}, U_R)$ *(Regra de redução 3)*;
- Remover de E_2 as arestas $D(M_R, U_{SC} \cup U_{NC})$ *(Regra de redução 4)*;
- Aleatoriamente adicionar ou remover as aresta $D(M_{SC} \cup M_{NC}, U_{SC} \cup U_{NC})$ *(Regra de redução 5)*.

Na Regra de redução 1, a adição das arestas optativas irá somente colaborar para o controle de novos vértices em M. O oposto ocorre na Regra 2, onde as arestas optativas que prejudicariam o controle de novos 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 as chances

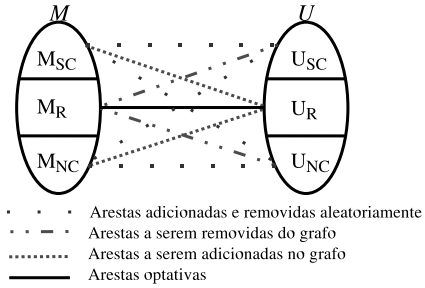

Figura 1. Relação entre os sub-conjuntos de vértices do grafo. Aplicação das regras de redução 3,4 e 5.

de controle de 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 M_R e U_R.

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 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(a) tem-se um grafo resultante após a aplicação da primeira e segunda regra de redução. A Figura 2(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 sanduíche G. Na Figura 2(c) tem-se a aplicação da quarta regra de redução. Agora o vértice 6 irá pertencer a U_{SC} por conseqüência da aplicação da terceira regra de redução. A aplicação em seqüência da terceira e quarta regras gera o grafo ilustrado na Figura 2(d).

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.

3 Problema do Maior Conjunto Controlado Generalizado - PMCCG

Imagine a seguinte situação, 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

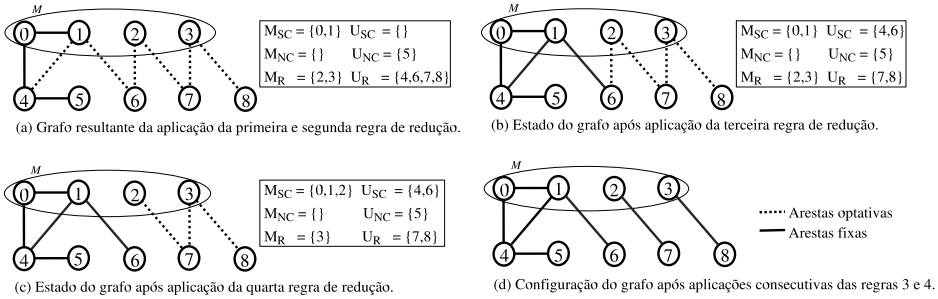

Figura 2. Exemplo de aplicação das regras de redução 3 e 4 consecutivamente

do dia seguinte. Para a surpresa, ou desapontamento dos cidadãos que votavam "SIM", o resultado encontrado mostrava uma vantagem de 2:1 para votos "NÃO" em sua vizinhança.

Como exemplo concreto, considere um grafo completo e bipartido com dois votos para "NÃO" de um lado e $n-2$ votos "SIM" no outro, veja na Figura 3.

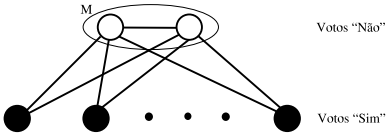

Figura 3. Dois vértices "NÃO" dão a impressão de ser a maioria na pesquisa local de todos os outros vértices. Na figura os vértices brancos representam a coalisão (conjunto M) e os vértices pretos os vértices controlados pela coalisão.

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 foram realizadas. Uma série de problemas práticos importantes envolvendo coalizões em grafos, maiorias locais e monopólios se enquadram nesse contexto. Maiores detalhes sobre esse assunto podem ser encontrados em [2,17].

A situação apresentada acima caracteriza, de certa forma, o PMCC apresentado em Makino *et. al.*[15], cujo interesse é promover um "controle" de uma maioria de votantes dado um sub-conjunto M 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. 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?

Imagine ainda que cada representante eleitor considere um número mínimo de votos (diferença entre os membros de dentro e fora da coalizão) para que sua escolha seja efetivada, teríamos então uma "folga mínima" ou "grau de convencimento" para esse eleitor. Por último, imagine também que existem eleitores que tenham maior "influência" ou maior "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 importantes. Essa situação corresponde à idéia dos *pesos* associados aos vértices do grafo, caracterizando agora o PMCCG.

O PMCCG agrega todas as extensões propostas para PMCC. Dessa maneira para cada vértice $i \in V$ teremos um *peso* $p_i \in \mathbb{Z}$ associado, que corresponde à "importância" ou "peso do voto", e $f_i \in \mathbb{Z}$, o valor da folga mínima ou "grau de convencimento" necessário ao vértice i para que ele seja f-controlado ou concorde com um determinado projeto.

Em seguida, apresentamos um algoritmo polinomial com razão de aproximação igual a 0,5, o modelo matemático para o PMCCG e um algoritmo para encontrar soluções viáveis iniciais por meio de uma relaxação linear do problema.

3.1 Algoritmo 0,5-aproximado para o PMCCG

O PMCC pode ser resolvido por um algoritmo determinístico, de tempo polinomial e razão de aproximação igual a 0,5 (vide Makino *et. al.*[15]). Veremos nessa seção que esse algoritmo pode ser estendido facilmente ao PMCCG, garantindo-se a mesma razão de aproximação.

O Algoritmo 1 pode ser empregado após a aplicação das regras de redução 1 e 2 descritas anteriormente. Além disso, suponha que W_1, 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 0,5-aproximado para PMCCG

1: $W_1 \leftarrow$ Somatório dos pesos dos vértices f-controlados por M, em $G = (V, E)$ para $E = E_2 \backslash \bar{D}$ onde $\bar{D} = \bigcup_{i \in M,\ p_i \geq 0} \bar{D}(\{i\}, U)$;
2: $W_2 \leftarrow$ Somatório dos pesos dos vértices f-controlados por M, em $G = (V, E)$ para $E = E_2 \backslash \tilde{D}$ onde $\tilde{D} = \bigcup_{j \in U,\ p_j < 0} \tilde{D}(M, \{j\})$;
3: $z_{H1} \leftarrow max\{W_1, W_2\}$;

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.*

Prova: É fácil ver que: $z_{max} \leq W_1 + W_2 \leq 2 \times max\{W_1, W_2\}$. Como $z_{H1} = max\{W_1, W_2\}$ então $z_{H1} \geq \frac{1}{2} z_{max}$ c.q.d. ■

3.2 Modelo matemático para o PMCCG

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 $p_i, f_i \in \mathbb{Z}$ 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:

$$z_{max} = maximizar \sum_{i \in V} p_i z_i \tag{3}$$

sujeito a:

$$\frac{\sum_{j \in M} a_{ij} x_{ij} - \sum_{j \in U} a_{ij} x_{ij} - f_i}{K} + 1 \geq z_i, \forall \ i \in V \tag{4}$$

$$x_{ij} = 1, \forall \ (i, j) \in E_1 \tag{5}$$

$$x_{ii} = 1, \forall \ i \in V \tag{6}$$

$$x_{ij} \in \{0, 1\}, \forall \ (i, j) \in E_2 \backslash E_1 \tag{7}$$

$$z_i \in \{0, 1\}, \forall \ i \in V \tag{8}$$

Na formulação apresentada temos a função objetivo (3) que calcula o somatório dos pesos dos vértices f-controlados por M. A desigualdade (4) 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 (5) e (6) 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 (7) e (8) (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. 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 (9) a seguir:

$$b_i = \left| \sum_{j \in M} a_{ij} x_{ij} - \sum_{j \in U} a_{ij} x_{ij} - f_i \right| \text{ para } i \in M_R \cup U_R \tag{9}$$

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 (4) pelas restrições (10), (11) e (12) descritas a seguir:

$$\frac{\sum_{j \in M} a_{ij} x_{ij} - \sum_{j \in U} a_{ij} x_{ij} - f_i}{b_i} + 1 \geq z_i, \forall\ i \in M_R \cup U_R \tag{10}$$

$$z_i = 1, \forall i \in M_{SC} \cup U_{SC} \tag{11}$$

$$z_i = 0, \forall i \in M_{NC} \cup U_{NC} \tag{12}$$

Seja $\bar{P}$ a relaxação linear associada a esta nova formulação. A desigualdade (10) tem a mesma função da restrição (4), 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 $b_i \leq K, \forall\ i \in V$. A igualdade (11) indica os vértices sempre f-controlados no grafo e a igualdade (12) 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 [19], essa relaxação pode ser resolvida em tempo polinomial através dos métodos de pontos interiores. Observe finalmente que, se $f_i = 0$ e $p_i = 1, \forall\ i \in V$ temos o PMCC conforme descrito em [15].

3.3 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 (que chamaremos Algoritmo 2) 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 $\bar{z}_i < 1$.

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 escolhermos 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 anexo.

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* $\tilde{z}_i \in [0,1], \forall\ i \in V$.

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

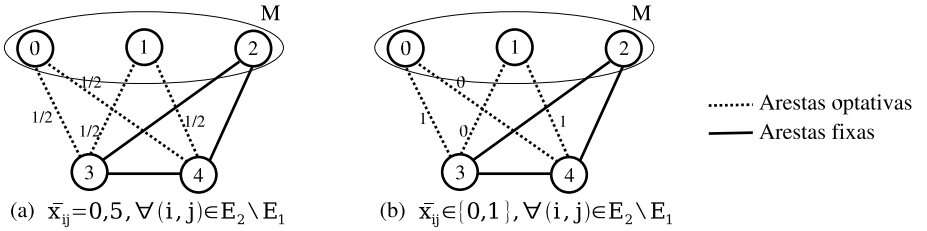

Figura 4. Exemplo de duas soluções ótimas, com coordenadas fracionárias e inteiras respectivamente.

Um novo procedimento, que chamaremos de Algoritmo 3, consiste em simplesmente selecionar a melhor solução obtida nos Algoritmos 1 e 2. Como demonstrado em [16], o Algoritmo 3 possui uma razão de performance superior 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 [16].

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

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

4 Busca Tabu com Reconexão por Caminhos aplicada ao PMCCG

O método de Busca Tabu (*Tabu Search*) - BT, proposto independentemente por [6] e [13] 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. Soluções ou movimentos presentes na *lista tabu* são proibidos temporariamente. Após um determinado número de iterações, a lista é atualizada de forma a permitir que novos movimentos ou soluções assumam o *status* de tabu. A lista tabu consiste de um conjunto de *soluções* ou *movimentos proibidos*, determinados por informações históricas das iterações precedentes no procedimento de busca local.

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 (fuga de ótimos locais). Podemos incluir ainda uma etapa de "pós-otimização" da solução onde um procedimento de Reconexão por Caminhos é ralizado 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*). 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 melhor que a solução global encontrada.

No caso do PMCCG, o procedimento de construção da solução inicial (linha 2) utilizado, consiste do Algoritmo 3 apresentado na Seção 3.3. 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 3.1. 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 são pesquisados e realizados na linha 7, caso promovam uma melhora na solução global, ou seja, caso o critério de aspiração seja satisfeito. 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. 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.

Essencialmente, na Reconexão por Caminhos, construímos um conjunto de soluções elite amarzenando as melhores soluções obtidas após cada busca local

Algoritmo 4 Pseudo código da BT com Reconexão por Caminhos aplicada a um problema de maximização

```
1: Contadores de iterações i, j ← 0
2: S, S* ← Solução inicial(), c(S'') ← −∞, Elite ← ∅
3: enquanto i ≤ i_max faça
4:    T ← ∅
5:    enquanto j ≤ j_max faça
6:       S' ← BuscaLocal(N(S)\T)
7:       S'' ← BuscaLocal(N(S) ∩ T) satisfazendo o critério de aspiração
8:       se c(S'') > c(S') então
9:          S' ← S''
10:      fim se
11:      se c(S') > c(S*) então
12:         S* ← S'
13:         j ← 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 ← S' e j ← j + 1
19:      Elite ← Atualiza(Elite,S)
20:   fim enquanto
21:   Diversificação(S)
22:   i ← i + 1
23: fim enquanto
24: S* ← Reconexão por Caminhos(Elite)
25: retorna S*
```

(linha 19). Em seguida, na etapa de pós-otimização, estas soluções são combinadas convenientemente visando a determinação de novas soluções de maior custo (linha 24).

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

4.1 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 do grafo sem que outros vértices sejam *descontrolados* (ou não-f-controlados). Representaremos esta estrutura de vizinhança por $N_0(S)$.

O procedimento de busca local proposto para o PMCCG é ilustrado sucintamente no Algoritmo 5. Dado um grafo sanduíche $G = (V, E)$ (solução corrente S), teremos um conjunto de arestas optativas E associadas e um conjunto $M_S \subseteq M_R$ (respectivamente $U_S \subseteq U_R$) dos vértices f-controlados por M em G. Considere também $L_S = M_S \cup U_S$, $L_D = (M_R \cup U_R \backslash L_S)$ e $\bar{L}_D = \{v \in L_D | p_v \geq 0\}$.

Nesta estrutura de vizinhança esperamos que vértices $v \in \bar{L}_D$ sejam f-controlados desde que nenhum outro vértice temporariamente f-controlado seja descontrolado. Note que, neste caso, uma nova solução de melhor custo, pertencente a $N_0(S)$ poderá ser gerada já que todos os pesos associados aos vértices de $\bar{L}_D$ são positivos. Ainda, 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, dado um grafo sanduíche G, teremos que $i \in V$ é f-controlado por M se, e somente se, $f_G(i) \geq 0$, caso contrário i será descontrolado (ou não f-controlado). Note ainda que $f_G(i) < 0, \forall\ i \in \bar{L}_D$.

Na busca local tentamos controlar vértices $v \in \bar{L}_D$ (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 $H_v = \{w \in V | (v, w) \in E_2 \backslash E_1$ e $f_G(w) \neq 0\}$. O vértice v poderá ser f-controlado desde que $|f_G(v)|$ seja menor ou igual a $|H_v|$ (linha 5). Basta então, adicionar/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 $M_R \backslash M_S$, será necessário remover $|f_G(v)|$ arestas optativas incidentes a v. Entretanto, caso $v \in U_R \backslash U_S$, 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 pelo BT

```
 1: Dado um grafo sanduíche G = (V, E) como entrada
 2: enquanto L̄_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 de 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:    L̄_D ← L̄_D\{v}
20: fim enquanto
21: retorna G
```

A lista tabu T é construida da seguinte forma: sempre que um grafo sanduíche G estiver associado a um máximo local, descontrolamos arbitrariamente um vértice $v \in L_S$ (conjunto dos vértices f-controlados em 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 $v \in M_S$, inserimos todas as arestas optativas incidentes a v. Caso contrário, se $v \in U_S$, eliminamos todas as arestas optativas incidentes a v. Após a inserção ou remoção de arestas optativas, computamos o novo custo associado.

Outras estruturas de vizinhanças bastante naturais (aqui representadas por $N_i(S)$ para $i \geq 1$) podem ser adotadas para o PMCCG. Poderemos efetuar uma busca local, descontrolando-se i vértices f-controlados (todos de M_R ou U_R respectivamente) 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 $N_0(S)$, além de exigirem um maior tempo computacional à medida que i aumenta. Em [18] discutimos a utilização de várias estruturas de vizinhança (através do método de vizinhança variável - VNS) aplicado ao PMCC.

Após a aplicação de uma busca local intensiva, executamos uma diversificação da solução corrente S, buscando agora 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.

4.2 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 drástica na solução corrente, resultando assim em uma nova semente (ou solução inicial). No PMCCG, dado um grafo sanduíche G, essa "perturbação drástica" pode ser realizada descontrolando-se aleatoriamente um subconjunto $K \subseteq L_S$ (onde $|K| \leq k$), de vértices f-controlados por M (sendo k parâmetro de entrada). 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 $v \in M_S$, inserimos todas as arestas optativas incidentes a v. Caso contrário, se $v \in U_S$, eliminamos todas as arestas optativas incidentes a v.

Esse procedimento não possui mecanismos de memória (lista tabu) associada, ele simplesmente gera uma nova solução inicial "diversificada", ou seja, uma solução com atributos bastante 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.

4.3 Reconexão por Caminhos

A técnica de Reconexão por Caminhos - RC (*Path Relinking*) introduzida inicialmente por Glover&Laguna [7] 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 devam possuir características ou atributos importantes que possam ser utilizados na determinação de novas soluções. Dessa maneira na geração dos caminhos (sequência de soluções intermediarias) entre soluções elites espera-se encontrar soluções melhores [8,9].

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 intermediarias, 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 [9].

As escolhas das soluções de 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 [9].

No PMCCG, consideramos como atributos, a informação associada aos vértices f-controlados ou não por M. Assim, dado um grafo sanduíche $G = (V, E)$ construimos 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 5 ilustra a aplicação da RC.

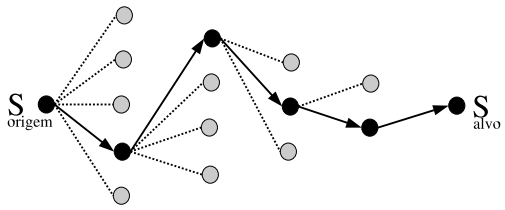

Figura 5. Exemplo da aplicação da reconexão por caminhos.

Neste trabalho, o conjunto elite é gerado durante a BT e consiste das p melhores soluções encontradas pela BT onde p é parâmetro de entrada. Optamos por utilizar 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 intermediaria 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 melhor solução 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* $S^{\star}$

5 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, respectivamente. Todos os algoritmos foram implementados usando a linguagem C 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 80%, a probabilidade de uma aresta ser fixa ou optativa, sendo que dentre este total de arestas selecionadas, 70% delas são optativas. Cada vértice teve seu peso e folga mínima definidos aleatoriamente dentro de um intervalo pré determinado. Para as instâncias com 50 vértices, foi definido o intervalo entre 1-10 para os pesos e 0-5 para as folgas mínimas. Nas instâncias com 75 vértices, intervalo entre 1-15 para os pesos e 0-7 para as folgas. Instâncias com 100 vértices, intervalo definido entre 1-20 para os pesos e 0-10 para as folgas. 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 no intervalo 0-10, seu nome deve ser: G100-20-10-01.

A Tabela 1 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 *MYK* e *Relaxação* e correspondem à solução obtida pelos Algoritmos 1 e 2, respectivamente, sendo que a solução inicial considerada, será aquela com melhor valor obtido entre as duas (Algoritmo 3), representada em negrito. Note que, para a maioria dos casos testados, a solução obtida pela relaxação linear é melhor que a solução baseada no algoritmo de Makino *et. al.*[15] para o PMCC.

O algoritmo da BT foi aplicado 10 vezes para cada instância. Os parâmetros do algoritmo foram definidos empiricamente. Dessa maneira, o valor das constantes i_{max} e j_{max} foram definidos em 5 e 50, respectivamente. Estes parâmetros foram selecionados empiricamente, entretanto, na maioria dos casos observados a melhor solução obtida pela BT sempre partia da solução inicial gerada pelo Algoritmo 3. 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 20% 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 tempo médio, em segundos, gasto pelas execuções da BT.

Tabela 1. Tabela de resultados da BT para instâncias com 50, 75 e 100 vértices, conhecendo o valor da solução ótima.

Instância	**Reg. de**	**Solução Inicial**		**Busca Tabu**		**Ótimo**	**Tempo(s)**
	Redução	MYK	Relaxação	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**	⋆ 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**	⋆ 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**	⋆ 602 (2)	597,94	602	1,15

A Tabela 2 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 de cada vértice. Esses valores obdecem à seguinte distribuição: instâncias com 300 vértices, pesos entre 1-30 e folgas entre 0-20; instâncias com 500 vértices, pesos entre 1-50 e folgas entre 0-30 e instâncias com 1000 vértices, pesos entre 1-100 e folgas entre 0-50.

A tabela apresenta a coluna *Aprox.*, em substituição aos valores ótimos associados de cada instância. Como a determinação do valor ótimo de "grandes" instâncias, se torna impraticável através de métodos exatos, calculamos a razão de aproximação da solução obtida pela BT, baseando-se no valor da relaxação linear (limite superior) correspondente. Dessa maneira, podemos ter uma melhor noção da qualidade da solução heurística obtida. A coluna *Aprox.*, apresenta o valor médio das aproximações encontradas nas execuções da BT.

A Tabela 2 confirma o melhor desempenho do Algoritmo 2 em relação ao Algoritmo 1. Além disso os valores das soluções heurísticas para as instâncias consideradas se encontram com aproximação superior a 0,94 do ótimo.

A Tabela 3 apresenta resultados para instâncias do PMCCG, utilizando a BT com RC. A tabela insere a coluna *Rec. por Caminhos* que descreve a solução obtida pela RC. O cálculo da aproximação (coluna *Aprox.*) é baseado na melhor solução retornada pelo algoritmo, ou seja, as soluções obtidas pela RC. A coluna *Tempo* descreve o tempo (em segundos) gasto pela BT com RC. As instâncias geradas obedecem às mesmas probabilidades conforme descrito anteriormente. Para as instâncias com 1000 vértices, a distribuição de pesos associados aos

Tabela 2. Tabela de resultados da BT para instâncias com 300, 500 e 1000 vértices.

Instância	Reg. de	Solução Inicial		Tabu		Aprox.	Tempo(s)
	Redução	MYK	Relaxação	Melhor valor	Média		
G300-30-20-01	58,94%	2136	**2845**	2847 (1)	2845,20	0,9904	5,09
G300-30-20-02	56,71%	3200	**4422**	4422 (10)	4422	0,9966	5,23
G300-30-20-03	61,59%	2371	**2949**	2957 (1)	2950,90	0,9796	5,84
G300-30-20-04	61,69%	3212	**3949**	3949 (10)	3949	0,9465	2,37
G300-30-20-05	60,00%	3158	**3807**	3845 (1)	3832,35	0,9462	3,05
G500-50-30-01	60,84%	8960	**10723**	10788 (1)	10744,35	0,9487	1,57
G500-50-30-02	59,76%	8858	**11247**	11247 (10)	11247	0,9822	3,58
G500-50-30-03	58,75%	9353	**12119**	12119 (10)	12119	0,9842	4,15
G500-50-30-04	62,55%	9061	**10614**	10652 (1)	10617,40	0,9460	2,21
G500-50-30-05	57,80%	8950	**12166**	12179 (1)	12170,90	0,9879	5,01
G1000-100-50-01	61,48%	36997	**44697**	44957 (1)	44741,25	0,9696	3,23
G1000-100-50-02	60,96%	36476	**45251**	45414 (1)	45381,10	0,9701	2,98
G1000-100-50-03	60,40%	36261	**45155**	45374 (1)	45322,30	0,9773	2,97
G1000-100-50-04	59,35%	37278	**47895**	47947 (2)	47910	0,9887	3,59
G1000-100-50-05	61,69%	38039	**44628**	44818 (1)	44776,40	0,9615	2,25

vértices variou entre 1-100 e a folga mínima entre 0-10. Instâncias com 2000 vértices, os pesos ficaram entre 1-200 e folgas entre 0-20. Em virtude da qualidade da solução dada pela BT, com uma aproximação elevada, a RC apresentou melhoras para um número reduzido de instâncias, a Tabela 3 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 inviavel, em virtude do tempo computacional elevado, dessa maneira consideramos como solução inicial a solução dada pelo Algoritmo 1. Sem o valor da relaxação linear, modificamos o cálculo da aproximação, que utiliza como limite superior o valor $|V| - |M_{NC} \cup U_{NC}|$, ou seja, o somatório dos pesos dos vértice nunca controlados é subtraido do total de pesos dos vértices do grafo.

Por fim, a Tabela 4 apresenta resultados obtidos para instâncias de tamanho 50, 75 e 100 vértices para o PMCC (caso particular do PMCCG). Os parâmetros utilizados na geração das instâncias permaneceram inalterados conforme descrito anteriormente. Nota-se através dos resultados obtidos que a estratégia proposta se comporta muito bem para o PMCC, chegando a obter a solução ótima na maioria das instâncias testadas.

6 Conclusões

Neste trabalho, apresentamos uma generalização do Problema do Maior Conjunto Controlado (introduzido por Makino *et. al.*[15]). Apresentamos um algoritmo 0,5-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 então em um procedimento de busca local e

Tabela 3. Tabela de resultados da BT com Reconxão por Caminhos.

Instância	**Reg. de**	**Solução Inicial**		**Busca**	**Rec. por**	**Aprox.**	**Tempo(s)**
	Redução	MYK	Relaxação	**Tabu**	**Caminhos**		
G1000-100-10-1	59,87%	35765	**48231**	48316	48399	0,9975	9,99
G1000-100-10-2	60,05%	36961	**49942**	50148	50231	0,9961	9,21
G1000-100-10-3	62,04%	38858	**48218**	48801	48804	0,9902	12,38
G1000-100-10-4	60,12%	38229	**50984**	51072	51093	0,9959	10,24
G1000-100-10-5	60,85%	37131	**48319**	48669	48705	0,9933	9,94
G1000-100-10-6	59,92%	37107	**49285**	49388	49419	0,9965	9,58
G1000-100-10-7	59,22%	35908	**48884**	49431	49442	0,9950	8,00
G2000-200-20-01	59,61%	147040	–	189180	189337	0,9287	2981
G2000-200-20-02	60,94%	145401	–	181085	181180	0,9088	54,21
G2000-200-20-03	59.03%	139436	–	184205	184360	0,9418	50,30
G2000-200-20-04	59.14%	141374	–	186677	186794	0,9395	2049
G2000-200-20-05	59.30%	144686	–	187512	187710	0,9382	2920
G2000-200-20-06	61.14%	146182	–	180753	180947	0,9039	4452
G2000-200-20-07	60.15%	149287	–	186041	186153	0,9179	3675

Tabela 4. Tabela de resultados da BT para instâncias com 50, 75 e 100 vértices para o PMCC, conhecendo o valor da solução ótima.

Instância	**Reg. de**	**Solução Inicial**		**Busca Tabu**		**Ótimo**	**Tempo(s)**
	Redução	MYK	Relaxação	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

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 resultados muito próximos do ótimo conhecido. Para as instâncias maiores (sem ótimo conhecido), os resultados obtidos pela Busca Tabu estiveram a uma distância inferior ou igual a 6% do valor ótimo.

ANEXO

Prova do Teorema 3

Prova: Note inicialmente de (10) que uma solução relaxada $(\bar{x}, \bar{z})$ de $\bar{P}$ deverá satisfazer à seguinte igualdade:

$$\frac{\sum_{i \in M} a_{ik}\bar{x}_{ik} - \sum_{j \in U} a_{jk}\bar{x}_{jk} - f_k}{b_k} + 1 = \bar{z}_k, \forall\ k \in M_R \cup U_R \tag{13}$$

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 $c' : E_2' \to \mathbb{R}^+$. A função c', que define as capacidades dos arcos de E_2', será construída a partir de $\bar{z}$ e da igualdade (13) da seguinte forma:

$$\begin{aligned} c'_{si} &= |E_1(i, M_R)| - |E_1(i, U_R)| - (\bar{z}_i - 1)b_i - f_i, \forall\ i \in M_R \\ c'_{ij} &= 1, \forall\ (i, j) \in E_2 \backslash E_1 \\ c'_{jt} &= (\bar{z}_j - 1)b_j - |E_1(M_R, j)| + |E_1(U_R, j)| + f_j, \forall\ j \in U_R \end{aligned} \tag{14}$$

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:

$$\sum_{j \in U_R} a_{ij}\hat{x}_{ij} = \hat{x}_{si} = c'_{si}, \forall\ i \in M_R \tag{15}$$

$$\sum_{i \in M_R} a_{ij}\hat{x}_{ij} = \hat{x}_{jt} = c'_{jt}, \forall\ j \in U_R \tag{16}$$

É fácil ver de (14), (15) e (16) 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 6.

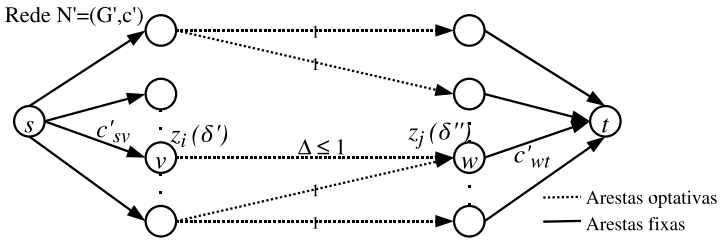

Figura 6. Enviando Δ unidades de fluxo em $p = (s, v, w, t)$.

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 (13) que:

$$\bar{z}_v = z_v(\delta' + \Delta) = \frac{|E_1(v, M_R)| - (|E_1(v, U_R)| + (\Delta + \delta')) - f_v}{b_v} + 1, \quad (17)$$

para $v \in M_R$

$$\bar{z}_w = z_w(\delta'' + \Delta) = \frac{(|E_1(w, M_R)| + (\Delta + \delta'')) - |E_1(w, U_R)| - f_w}{b_w} + 1, \quad (18)$$

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 (17) e (18), 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 (17) e (18) 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 (17) e (18) 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. ■

Referências

1. Ahuja, R. N., Magnanti, T. L., Orlin, J. B. *Network Flows: theory, algorithms and applications.* Prentice-Hall, 1993.
2. Bermond, J.C., Bond, J., Peleg, D., Perennes, S. *The power of small coalitions in graphs.* Discrete Appl. Math. - Elsevier Science Publishers B. V., vol. 127, pp. 399–414, 2003.
3. Carrano, A. V. *Establishing the order of human chromosome-specific DNA fragments.* In A. D. Woodhead and B. J. Barnhart, editors, Biotechnology and the Human Genome, pp. 37–50, Plenum Press, 1988
4. Gendreau, M., Soriano, P., Salvail, L. *Solving the maximum clique problem using a tabu search approach.* Annals of Operations Research N.41, pp 385–403, 1993.
5. Glover, F. *Future paths for integer programming and artificial intelligence.* Computers e Operations Research, volume 13, pp. 533–549, 1986.
6. Glover, F. *Tabu Search - Part I.* ORSA Journal on Computing, volume 1, N. 3, pp. 190–206, 1989.
7. Glover, F.,Laguna, M. *Modern Heuristic Techniques for Combinatoial Problems - Tabu Search.* Blackwell Scientific Publications, Oxford, pp. 70–150, 1993.
8. Glover, F. *Tabu search and adaptive memory programming - advances, applications and challenges.* in R.S. Barr, R.V. Helgason and J.L. Kennington, eds, 'Interfaces in Computer Science and Operations Research', Kluwer, pp. 1–75, 1997.
9. Glover, F., Laguna, M., Martí, R. *Fundamentals of scatter search and path relinking.* Control and Cybernetics, volume 29, n. 3, pp. 653–684, 2000.
10. Golumbic, M. C.,Shamir, R.. *Complexity and algorithms for reasoning about time: A graph-theoretic approach.* ACM, volume 40, pp. 1108–1133, 1993.
11. Golumbic, M. C., Kaplan, H., Shamir, R. *On the complexity of DNA physical mapping.* Advances in Applied Mathematics, volume 15, pp. 251–261, 1994.
12. Golumbic, M. C., Kaplan, H., Shamir, R. *Graph Sandwich Problems.* J. Algorithms, volume 19, N. 3, pp. 449–473, 1995.

13. Hansen, P. *The steepest ascent mildest descent heuristic for combinatorial programming.* Congress on Numerical Methods in Combinatorial Optimization, Capri-Italy, 1986.
14. Hansen, P., Mladenovic, N. *Variable Neighborhood search: Principles and applications.* European Journal of Operational Research, 130, pp. 449–467, 2001.
15. Makino, K., Yamashita, M., Kameda, T. *Max-and min-neighborhood monopolies.* Algorithmica, N. 34, pp. 240-260, 2002.
16. Martinhon, C. A., Protti, F. *An Improved Derandomized Approximation Algorithm for the Max-Controlled Set Problem.* WEA 2004, LNCS 3059, pp. 341–355, 2004.
17. Peleg, D. *Local majorities, coalitions and monopolies in graphs: a review.* Theoretical Computer Science, Vol.282, n.2, pp.231–257, 2002.
18. Santos, I. M., Martinhon, C. A., Ochi, L. S. *Uma metaheurística VNS aplicada ao Problema do Maior Conjunto Controlado.* Encontro Regional de Matemática Aplicada e Computacional-ERMAC 2004 SBMAC, Rio de Janeiro, v.1, pp. 30-30, 2004.
19. Wright, Stephen J. *Primal-Dual Interior Point Algorithms.* SIAM Publications, Philadelphia, 1997.