UNIVERSIDADE FEDERAL FLUMINENSE

GILBERTO FARIAS DE SOUSA FILHO

Algoritmos Heurísticos e Exatos para o Problema de Edição de Biclusters

NITERÓI

2016

UNIVERSIDADE FEDERAL FLUMINENSE

GILBERTO FARIAS DE SOUSA FILHO

Algoritmos Heurísticos e Exatos para o Problema de Edição de Biclusters

Proposta de Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Algoritmos e Otimização.

Orientador:
Fabio Protti

Coorientador:
Luiz Satoru Ochi

NITERÓI

2016

GILBERTO FARIAS DE SOUSA FILHO

Algoritmos Heurísticos e Exatos para o Problema de Edição de *Biclusters*

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Algoritmos e Otimização.

Aprovada em Agosto de 2016.

BANCA EXAMINADORA

Prof. Fabio Protti - Orientador, IC/UFF

Prof. Luiz Satoru Ochi - Coorientador, IC/UFF

Prof. Eduardo Uchoa Barboza, DEP/UFF

Prof. Lucídio Formiga dos Anjos Cabral, CI/UFPB

Prof. Yuri Abitibol de Menezes Frota, IC/UFF

Profa. Simone de Lima Martins, IC/UFF

Prof. Luidi Simonetti, COPPE/UFRJ

Niterói

2016

Resumo

O Problema de Edição de *Biclusters* (PEB), é NP-difícil e consiste na edição de um número mínimo de arestas de um bigrafo de entrada $G = (V_1, V_2, E)$ com o objetivo de transformá-lo em uma união disjunta de subgrafos bipartidos completos. Aplicações do PEB incluem mineração de dados, análise de dados de expressões genéticas, criação de células de manufaturas de produtos industriais, entre outros. Neste trabalho, propomos técnicas exatas e heurísticas para o PEB. Para isso, apresentamos a criação de novas instâncias teste, além da definição de uma nova regra de redução da dimensão destas instâncias. Propomos modelos de programação linear inteira, um estudo poliedral do PEB para aplicar técnicas de *Branch-and-cut*. Reformulação do MPI, onde este modelo passa a ter crescimento exponencial do número de variáveis em relação à dimensão do problema, levando assim ao uso da técnica de *Branch-and-price*, e para tanto, propomos algoritmos de Geração de Colunas (GC) para resolver a relaxação linear deste modelo, uma formulação inteira para o subproblema de *pricing*, bem como uma heurística. Ainda propomos, neste trabalho, heurísticas construtivas, movimentos de vizinhança, estruturas auxiliares, com o objetivo de acelerar a avaliação destes movimentos, tendo como objetivo aplicar estas heurísticas aos *frameworks* clássicos GRASP, ILS e VNS na resolução do PEB. Resultados computacionais mostram que nossas propostas são mais eficazes e eficientes que os procedimentos da literatura, encontrando soluções ótimas anteriormente não provadas para algumas das instâncias testadas.

Palavras-chave: *bicluster editing problem*, *biclusterização*, *politopo de particionamento em bicliques*.

Sumário

Capítulo 1

Introdução

O conceito de agrupar dados em *clusters* aparece em inúmeros contextos e disciplinas. Este tema tem sido estudado extensivamente e vários algoritmos exatos, aproximativos e heurísticos foram propostos, onde o objetivo é encontrar uma partição no conjunto de dados de entrada cujos elementos de um mesmo *cluster* sejam similares, enquanto elementos de *clusters* distintos tenham pouca similaridade. Esta similaridade é geralmente modelada como um grafo: cada vértice representa uma entidade do dado, e dois vértices são conectados por uma aresta se as entidades que eles representam possuem alguma similaridade em um contexto específico. Se os dados são perfeitamente agrupados, então o resultado será um grafo *clusterizado*, que é um grafo no qual cada componente conexa é uma clique. Neste contexto, um simples modelo de *clusterização* é definido como o problema de edição de *clusters* (em inglês *Cluster Editing Problem* (CEP) [Bansal *et al.*, 2004, Shamir *et al.*, 2004]). Este problema tem como objetivo encontrar um conjunto mínimo de arestas a serem editadas a fim de transformar o grafo de entrada $G = (V, E)$ em um grafo *clusterizado*. Editar uma aresta (i, j) consiste em adicioná-la (se $(i, j) \notin E$) ou deletá-la (se $(i, j) \in E$).

Em algumas situações, o modelo padrão de *clusterização* não é satisfatório. Um importante exemplo, apresentado por [Guo *et al.*, 2008], é a *clusterização* de dados de expressão genética, onde sob certas condições o nível da expressão de um número de genes é medido. Neste caso, *clusterizar* apenas genes ou condições não fornece significado suficiente. Pretende-se encontrar um subconjunto de genes e um subconjunto de condições que juntos se comportem de uma maneira consistente. Tal situação configura o particionamento de bigrafos, do inglês *biclustering* [Madeira & Oliveira, 2004, Tanay *et al.*, 2006].

O conceito de *biclusterização* foi introduzido na década de 1970 [Kluger *et al.*, 2003], e abordado no contexto da biologia computacional por [Cheng & Church, 2000]. O Pro-

blema de Edição de *Biclusters* (PEB) é uma simples representação de *biclusterização*, em inglês *Bicluster Editing Problem* (BEP). Neste caso, a solução é composta por bicliques, que são subgrafos bipartidos completos, disjuntos em vértices.

1.1 Motivação

Podemos identificar algumas aplicações para o PEB em diferentes áreas do conhecimento, entre elas:

Mineração de Dados: informações são tipicamente armazenados em base de dados, onde cada registro possui um conjunto de atributos. Nas aplicações com centenas de milhares de registros, uma alternativa para descobrir novas informações e conhecimentos ocorre através de *biclusterização*. Imagine livros vendidos em uma livraria. O objetivo é encontrar padrões ocultos de compras. A descoberta de compras similares pode ajudar a livraria na recomendação de novos produtos a seus clientes, pela identificação de padrões em suas compras, criando perfis por associações a serem usados em futuras promoções. Toda esta informação não seria perceptível, uma vez que muitos dos padrões de compra não são aparentes. Ao emparelhar produtos comprados e clientes, um algoritmo de *biclusterização* pode nos dar uma boa visão das relações ocultas entre os dados.

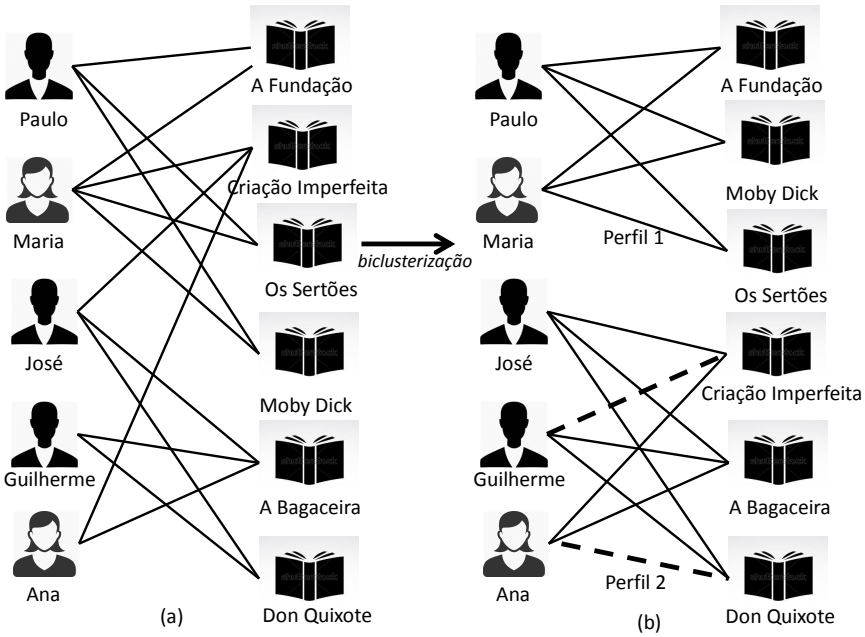


Figura 1.1: (a) Relação de compra entre clientes e produtos de uma livraria. (b) Perfis de clientes criados a partir da *biclusterização* dos dados.

A Figura 1.1(a) ilustra os relacionamentos de clientes de uma livraria fictícia e

seus livros comprados no passado. Já a Figura 1.1(b) apresenta a formação de dois perfis de clientes a partir de um processo de *biclusterização* dos dados, agrupando clientes que tenham similaridade nos livros comprados. As ligações representadas por linhas tracejadas são relações fictícias, como exemplo temos a cliente Ana que não comprou o livro Don Quixote, sendo agrupada no perfil de quem já comprou este livro, o sistema assim poderá usar esta relação para recomendar, para Ana, a compra do livro.

Projeto de redes *multicast*: uma sessão *multicast* pode ser definida como um subconjunto de clientes que requerem o mesmo conteúdo. Cada cliente pode requisitar várias sessões *multicast*. A principal limitação é que as redes de telecomunicações não suportam controlar muitas sessões *multicast* simultaneamente. Uma possível solução é agrupar as sessões em um número limitado. Um exemplo de uso de sessões *multicast* é o serviço de TV sobre o protocolo IP (*Internet Protocol*), em que um subconjunto de clientes requisitam os mesmos canais ou o mesmo subconjunto de canais durante um intervalo de tempo. Neste cenário, os *biclusters* seriam formados por subconjuntos de clientes e seus canais requisitados, criando assim uma única sessão *multicast* para atende-los.

Células de Manufatura: a gestão da produção é um fator muito importante para o sucesso de uma indústria. Através dela busca-se organizar o ambiente de produção de maneira a economizar custos e tempo de produção, sem perda de qualidade. Para as indústrias que possuem uma grande variedade e volume médio de produção, não há necessidade que as máquinas sejam dedicadas à produção de apenas um produto. Uma abordagem muito usada é a formação de grupos de máquinas com funcionalidades idênticas (grupos de tornos, fresadeiras, etc). Assim, uma parte (peça) de um determinado produto que necessite de operações de manufatura de mais de um tipo de máquina, precisará percorrer todos os grupos que contenham os tipos de máquinas necessários para a sua completa manufatura.

Neste tipo de sistema de produção, máquinas de diferentes funcionalidades são agrupadas em uma célula, a qual é dedicada à produção de uma família de produtos (partes que possuem alto nível de similaridade entre si no que dizem respeito às máquinas necessárias para sua manufatura). O sistema de produção passa a ser formado por células de máquinas e famílias de partes, também chamados de células de manufatura.

A Figura 1.2(a) apresenta uma matriz indicando as máquinas em que cada peça precisa ser manufaturada para finalizar sua confecção. Já a Figura 1.2(b) apresenta a criação de duas células de manufaturas, indicando quais conjuntos de máquinas e famílias de peças devem se agrupar. As relações que ficaram fora das células formadas, indicam que

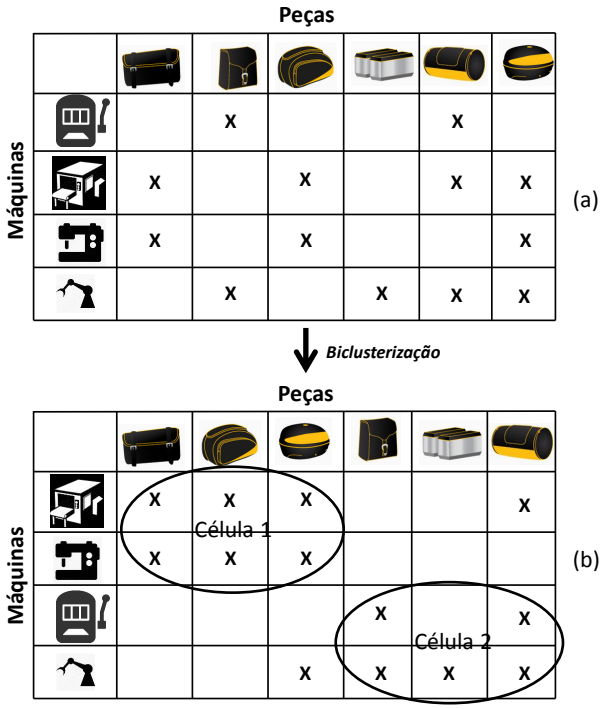


Figura 1.2: (a) Relação de peças a serem manufaturas pelas máquinas de uma indústria. (b) Células de Manufatura geradas pela *biclusterização*.

a peça relacionada deverá em algum momento sair de sua atual célula e se deslocar para a célula onde a máquina, a que ela também se relaciona, se encontra. Estes "erros"devem ser minimizados na construção das células. O trabalho de [Pinheiro *et al.*, 2016] foi o primeiro a associar o problema de Células de Manufatura a um problema de *biclusterização*.

1.2 Objetivos

Os objetivos deste trabalho são:

- Revisão dos problemas de *clusterização*, mas especificamente problemas de *biclusterização*, descrevendo suas aplicações práticas e os métodos propostos na literatura.
- Realizar o estudo poliedral do *Biclique Partitioning Polytope*, que é o politopo do Problema de Edição de *Biclusters*
- Desenvolver técnicas exatas para o Problema de Edição de *Biclusters*.
- Desenvolver *frameworks* heurísticos clássicos para a resolução de problemas de grande porte para o Problema de Edição de *Biclusters*.

1.3 Esboço da Tese

A continuação deste trabalho está organizado da seguinte forma.

- Capítulo 2 define formalmente o Problema de Edição de *Biclusters*, descreve trabalhos relativos da literatura, apresenta dois novos Modelos de Programação Linear Inteira, a geração das instâncias testes e uma nova regra de redução da dimensão de instância para o problema.
- Capítulo 3 realiza um estudo poliedral do PEB (*Biclique Partitioning Polytope*), descrevendo novas facetas para o seu poliedro e propondo seu uso para a técnica de *Branch-and-cut*.
- Capítulo 4 apresenta uma nova formulação para o PEB a ser utilizada na técnica de *Branch-and-price*, propondo também a formulação do subproblema de *pricing*, algoritmos de Geração de Colunas e técnicas de *branching*.
- Capítulo 5 apresenta heurísticas a serem adaptadas às meta-heurísticas clássicas VNS, ILS, GRASP.
- Capítulo 6 descreve os resultados computacionais que validam as propostas implementadas neste trabalho.

Capítulo 2

Problema de Edição de Biclusters

Dado um grafo $G = (V, E)$, editar uma aresta (i, j) consiste em adicioná-la no grafo (se $(i, j) \notin E$) ou removê-la (se $(i, j) \in E$). O PEB consiste em editar o número mínimo de arestas afim de transformar um grafo bipartido de entrada $G = (V_1 \cup V_2, E)$ em uma união disjunta de subgrafos bipartidos completos, em outras palavras, em uma solução final, cada componente conexa é uma biclique (grafo bipartido completo). As bicliques que formam uma solução são chamadas de *biclusters*.

Consideramos apenas grafos bipartidos não direcionados. Seja P_4 um caminho com 4 vértices, onde $ijkl$ é um P_4 em que os vértices i e l tem grau 1, e j e k tem grau 2. A vizinhança de vértices(aberta) do vértice j é representada por $N(j)$, e a vizinhança fechada $N(j) \cup \{j\}$ é representado por $N[j]$. Nós estendemos esta notação para um conjunto de vértices S, $N(S) = \bigcup_{j \in S} N(j)$. Para um vértice j, $N^2[j] = N(N(j))$ representa o conjunto de vértices que estão a uma distância exatamente 2 de j, onde a distância entre dois vértices é definida pelo número de arestas do caminho de comprimento mínimo que os liga. Note que $j \in N^2[j]$. Para um grafo $G = (V, E)$, nós definimos $\overline{E} = \{(i, j) \mid (i, j) \notin E\}$.

Figura 2.1(a) apresenta uma instância G para o PEB, onde as partições V_1 e V_2 contém 4 e 3 vértices.

Note que uma solução viável do PEB satisfaz as seguintes condições:

- Cada par de vértices i e j da mesma parte contidas na mesma componente B tem a mesma vizinhança ($N(i) = N(j)$).
- Cada vértice j em uma componente conexa B satisfaz $N(j) \cup N^2[j] = B$.

Figura 2.1(b) apresenta uma solução ótima, formada por dois *biclusters*: *bicluster*

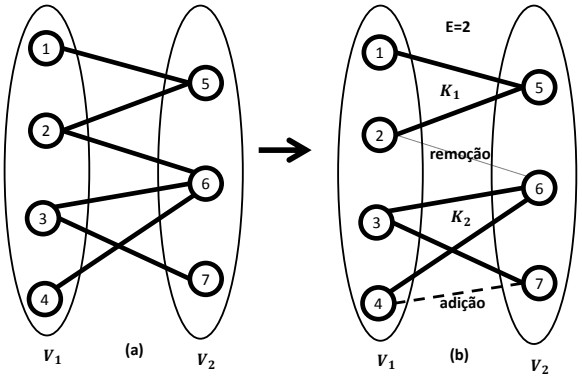


Figura 2.1: (a) Pequena instância para o PEB. (b) Uma solução ótima para a instância exemplo.

K_1 com vértices $\{1,2\} \in V_1$ e $\{5\} \in V_2$, e *bicluster* K_2 com vértices $\{3,4\} \in V_1$ e $\{6,7\} \in V_2$. Como podemos ver, a aresta $(2,6)$ foi removida e a aresta $(4,7)$ adicionada. Por isso, duas edições foram realizadas, e este é o valor da solução ótima.

2.1 Modelos de Programação Linear Inteira

Em [Amit, 2004], Amit apresenta a formulação $(\mathcal{F}_{P_4})$ para o PEB, descrita a seguir. Seja $G = (V_1 \cup V_2, E)$ um grafo bipartido, com as partes (conjuntos estáveis) V_1 e V_2. Se $(i,j) \in E$, atribuímos $w(i,j) = +1$, caso contrário nós atribuímos $w(i,j) = -1$. Se $w(i,j) = +1$, nós dizemos que (i,j) é uma *aresta positiva*, caso contrário (i,j) é uma *aresta negativa*. Utilizamos uma variável binária x_{ij} para cada par (i,j), com $i \in V_1$ e $j \in V_2$, tal que $x_{ij} = 1$ se e somente se i e j terminarem no mesmo *bicluster*. O significado do valor original da variável x_{ij} foi invertido, em [Amit, 2004] dois vértices i e j terminam no mesmo *bicluster* se $x_{ij} = 0$. Esta mudança tem o objetivo de tornar o modelo mais didático. O modelo de programação linear inteira com a inversão do valor da variável x_{ij} é apresentado a seguir.

$$(\mathcal{F}_{P_4}) \min \sum_{+(i,j)} (1 - x_{ij}) + \sum_{-(i,j)} x_{ij} \tag{2.1}$$

$$\text{s.t. } x_{ij} + x_{jk} + x_{kl} - x_{il} \leq 2, \qquad \forall i,k \in V_1 \text{ e } \forall j,l \in V_2 \tag{2.2}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in V_1 \text{ e } \forall j \in V_2. \tag{2.3}$$

onde $+(i,j) = \{(i,j) \mid w(i,j) = +1\}$, e $-(i,j) = \{(i,j) \mid w(i,j) = -1\}$.

A função objetivo (2.1) mede o número de *erros* no grafo de entrada, isto é, o número mínimo de arestas que devem ser editadas com o objetivo de obter uma solução. Erros incluem *erros positivos*, isto é, arestas positivas (i, j) que devem ser removidas (e para os quais $x_{ij} = 0$ na solução final), bem como *erros negativos*, isto é, arestas negativas (i, j) que devem ser adicionadas (e para os quais $x_{ij} = 1$ na solução final). Note que as inequações (2.2) garantem que se os pares (i, j), (j, k) e (k, l) finalizarem no mesmo *bicluster*, então o par (i, l) deve estar também no mesmo *bicluster*. Em outras palavras, as inequações (2.2) evitam a existência de um P_4 $ijkl$ como um subgrafo induzido de G. A configuração de arestas proibidas entre os vértices $ijkl$ está ilustrada na Figura 2.2.

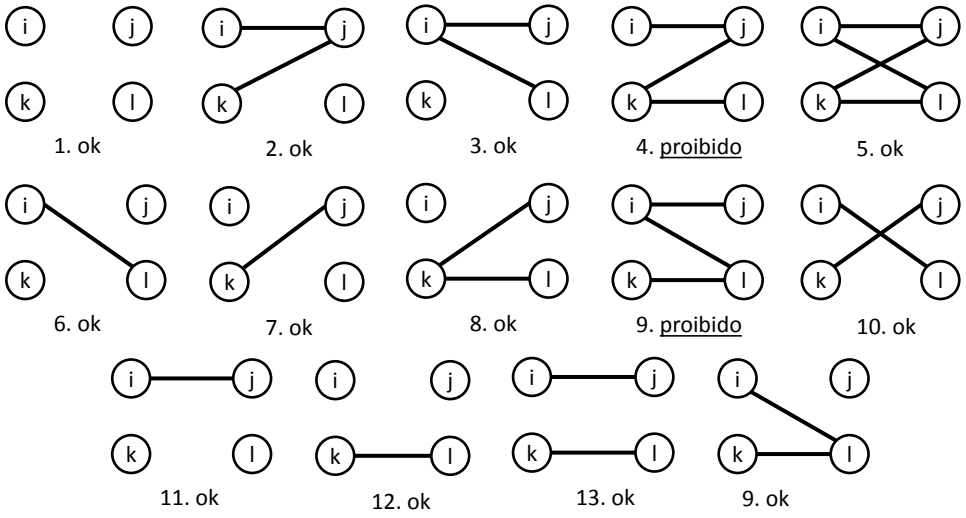


Figura 2.2: Configurações das arestas entre *ijkl* permitidas e proibidas pelas inequações de $(\mathcal{F}_{P_4})$.

Para que o modelo anterior trate grafos ponderados, propomos a substituição da função objetivo (2.1) pela seguinte função objetivo, onde os valores para $w(i, j)$ não são restritos para o conjunto $\{-1, +1\}$:

$$\text{Minimize} \sum_{+(i,j)} w(i,j)(1 - x_{ij}) + \sum_{-(i,j)} |w(i,j)| \, x_{ij}. \tag{2.4}$$

Assim, $+(i, j) = \{(i, j) \mid w(i, j) > 0\}$, e $-(i, j) = \{(i, j) \mid w(i, j) < 0\}$. Elementos do subconjunto $+(i, j)$ são arestas positivas e elementos do subconjunto $-(i, j)$ são arestas negativas. O valor de uma solução viável $G' = (V, E')$, denotado por $valor(G')$, é dado por:

$$valor(G') = \sum_{+(i,j) \cap \bar{E}'} w(i,j) + \sum_{-(i,j) \cap E'} |w(i,j)| \, . \tag{2.5}$$

Na equação anterior, a primeira soma refere-se às arestas positivas que foram removidas, e a segunda às arestas negativas que foram adicionadas.

2.1.1 Formulação $\mathcal{F}_{K_1}$

Propomos neste trabalho dois novos modelos de programação linear inteira ($\mathcal{F}_{K_1}$) e ($\mathcal{F}_{K_2}$) que adotam a abordagem de indexação dos *biclusters* formados na solução final. Estes modelos continuam adotando como entrada os conjuntos $G = (V_1, V_2, E)$, $+(i,j)$ e $-(i,j)$ e o custo das arestas w_{ij}. Seja K o número máximo de *biclusters* que possam existir na solução final, como podemos ter uma solução contendo apenas um vértice em cada *bicluster* (*singletons*) temos que $K = |V_1| + |V_2|$. A variável binária de decisão x_{ik} indica, se $x_{ik} = 1$, que o vértice i está no *bicluster* de índice k, caso contrário ele não pertence ao *bicluster* k. Sendo assim temos a seguinte formulação para ($\mathcal{F}_{K_1}$):

$$(\mathcal{F}_{K_1}) \min \sum_{+(i,j)} w_{ij}(1 - \sum_{k \in K} x_{ik}x_{jk}) + \sum_{-(i,j)} |w_{ij}| \, (\sum_{k \in K} x_{ik}x_{jk}) \tag{2.6}$$

$$\text{s.t.} \sum_{k \in K} x_{ik} = 1, \qquad \forall i \in V_1 \cup V_2 \tag{2.7}$$

$$x_{ik} \in \{0,1\}, \qquad \forall i \in V_1 \cup V_2 \text{ e } \forall k \in K. \tag{2.8}$$

A função objetivo (2.6) computa o peso das arestas negativas que ligam vértices contidos no mesmo *bicluster* e das arestas positivas que ligam vértices que não estão no mesmo *bicluster* da solução final. As restrições (2.7) garantem que cada vértice esteja em um e somente um *bicluster* de índice k na solução final.

A linearização de $\mathcal{F}_{K_1}$ é realizada pela definição de um novo conjunto de variáveis binárias y_{ijk}. Para cada par de vértices $i \in V_1$, $j \in V_2$, a variável y_{ijk} assume o valor 1 se a aresta $\{i,j\}$ estiver no *bicluster* k, ou seja, ambos os vértice i e j estão no *bicluster* k. Caso contrário $y_{ijk} = 0$. A linearização de $\mathcal{F}_{K_1}$ pode ser expressa como se segue:

$$(\mathcal{F}_{K_1}) \min \sum_{+(i,j)} w_{ij}(1 - \sum_{k \in K} y_{ijk}) + \sum_{-(i,j)} |w_{ij}| \, (\sum_{k \in K} y_{ijk}) \tag{2.9}$$

$$\text{s.t. } (2.7), (2.8)$$

$$x_{ik} + x_{jk} \leq y_{ijk} + 1, \qquad \forall i \in V_1,\ \forall j \in V_2 \text{ e } \forall k \in K \tag{2.10}$$

$$y_{ijk} \leq x_{ik}, \qquad \forall i \in V_1,\ \forall j \in V_2 \text{ e } \forall k \in K \tag{2.11}$$

$$y_{ijk} \leq x_{jk}, \qquad \forall i \in V_1,\ \forall j \in V_2 \text{ e } \forall k \in K \tag{2.12}$$

$$y_{ijk} \in \{0,1\}, \qquad \forall i \in V_1,\ \forall j \in V_2 \text{ e } \forall k \in K. \tag{2.13}$$

2.1.2 Formulação $\mathcal{F}_{K_2}$

Para a formulação de $(\mathcal{F}_{K_2})$ adotamos todos os parâmetros de entrada de $(\mathcal{F}_{K_1})$, e a variável binária de decisão x_{ik}. Inserimos a variável binária de decisão y_{ij}. Se $y_{ij} = 1$ e a aresta $(i,j) \in +(i,j)$ (aresta positiva) então indica que (i,j) não foi removida na solução final, se a aresta $(i,j) \in -(i,j)$ (aresta negativa) então (i,j) foi adicionada na solução final. Assim, temos a seguinte formulação para $(\mathcal{F}_{K_2})$:

$$(\mathcal{F}_{K_2}) \min \left(\sum_{+(i,j)} w_{ij} - \sum_{+(i,j)} w_{ij} y_{ij} \right) + \sum_{-(i,j)} |w_{ij}|\, y_{ij} \tag{2.14}$$

$$\text{s.t. } x_{ik} + x_{jk} \leq y_{ij} + 1, \qquad \forall (i,j) \in -(i,j) \text{ e } \forall k \in K \tag{2.15}$$

$$y_{ij} \leq 2x_{ik} - x_{jk} + 1, \qquad \forall (i,j) \in +(i,j) \text{ e } \forall k \in K \tag{2.16}$$

$$y_{ij} \leq 2x_{jk} - x_{ik} + 1, \qquad \forall (i,j) \in +(i,j) \text{ e } \forall k \in K \tag{2.17}$$

$$\sum_{k \in K} x_{ik} = 1, \qquad \forall i \in V_1 \cup V_2 \tag{2.18}$$

$$x_{ik} \in \{0,1\}, \qquad \forall i \in V_1 \cup V_2 \text{ e } \forall k \in K \tag{2.19}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E. \tag{2.20}$$

A função objetivo (2.14), como a função objetivo (2.6), computa a soma do peso das arestas positivas removidas e das arestas negativas adicionadas. As restrições (2.15) marcam a aresta $(i,j) \in -(i,j)$ como adicionada ($y_{ij} = 1$) caso os vértices i e j estejam no mesmo *bicluster* de índice k. Já as restrições (2.16) e (2.17) indicam que a aresta $(i,j) \in +(i,j)$ foi removida ($y_{ij} = 0$) caso os vértices i e j estejam em *biclusters* distintos. Por fim, as restrições (2.18) garantem que cada vértice esteja em um e somente um *bicluster* de índice k na solução final.

2.1.3 Remoção de soluções simétricas

Os modelos $\mathcal{F}_{K_1}$ e $\mathcal{F}_{K_2}$ sofrem de simetria como ilustrado na Tabela 2.1, isto é, a troca do conjunto de vértices entre quaisquer dois *biclusters* produz soluções equivalentes alternativas. Esta característica aumenta o tempo computacional devido ao aumento no espaço de busca das soluções.

Tabela 2.1: Duas soluções equivalentes para o PEB.

Solução 1		**Solução 2**	
Bicluster	Vértices	Bicluster	Vértices
1	2, 4, 5	1	1, 3, 6
2	-	2	2, 4, 5
3	1, 3, 6	3	-

Inequações válidas para eliminar simetrias foram propostas na literatura (veja, por exemplo, [Köhler *et al.*, 2013]). Para eliminar simetria para o PEB propomos dois conjuntos de inequações.

O primeiro conjunto substitui as inequações (2.7) e (2.18) das formulações $\mathcal{F}_{K_1}$ e $\mathcal{F}_{K_2}$ respectivamente, pelas inequações (2.21). Estes cortes forçam cada vértice a ser atribuído para um *bicluster* cujo índice seja menor ou igual ao índice do vértice, ou seja, vértice 1 para o *bicluster* 1, vértice 2 para o *bicluster* 1 ou 2, vértice 3 para o *bicluster* 1, 2 ou 3 e etc.

$$\sum_{k=1}^{v} x_{vk} = 1, \qquad \forall v \in V_1 \cup V_2 \tag{2.21}$$

O segundo conjunto de inequações, que elimina simetria, força que o vértice de menor índice em um *bicluster* tenha o índice igual do *bicluster*. Como conseqüência se o vértice k não estiver no *bicluster* k, então o *bicluster* k fica vazio. O conjunto de inequações (2.22) formula esta ideia e deve ser adicionada às formulações $\mathcal{F}_{K_1}$ e $\mathcal{F}_{K_2}$.

$$x_{vk} \leq x_{kk}, \qquad \forall v \in V_1 \cup V_2, k \in \{1..v\} \tag{2.22}$$

2.2 Geração e Redução de Instâncias

Esta seção discute como gerar instâncias de grafos bipartidos aleatórios para o PEB com vários níveis de dificuldade. Além de descrever as regras de redução da literatura e uma nova regra proposta neste trabalho.

2.2.1 Algoritmo de Geração

Estudos relativos ao PEB, em geral, apresentavam provas formais de garantia da performance de suas soluções, em vez de analisar resultados empíricos. Portanto há a necessidade de criar uma bateria de instâncias testes para realizar nosso experimentos computacionais.

Seguimos o modelo de grafos aleatórios propostos em [Gilbert, 1959]. Denota-se por $G(n, m, p)$ um grafo bipartido aleatório com n vértices na partição V_1, m vértices na partição V_2, tal que cada aresta entre partições possa existir independentemente com probabilidade p. Em [Bastos *et al.*, 2014] é proposto um procedimento simples para geração de grafos aleatórios gerais (não necessariamente bipartido). Adaptamos o mesmo para a geração de grafos bipartidos. Primeiro, inicializa-se os subconjuntos V_1 e V_2 com n e m elementos, respectivamente; o conjunto de arestas é iniciado vazio. Depois, para cada par de vértices $i \in V_1$, $j \in V_2$, decide-se com probabilidade p se (i, j) é adicionado à E. Seja G o grafo bipartido gerado neste esquema. Note que a *densidade* de G, é definida como $d = |E|/(n * m)$.

Para determinar a dificuldade de uma instância, [Bastos *et al.*, 2014] utilizou um critério simples baseado no número de arestas editadas necessárias para atingir a otimalidade, denotado por $opt(G)$. Para duas instâncias G_1 e G_2 com o mesmo número de vértices $n * m$, dizemos que G_1 é *mais difícil* que G_2 se $opt(G_1) > opt(G_2)$.

O impacto da densidade na dificuldade das instâncias é apresentado na Figura 2.3. O valores $opt(G(n, m, p))$ foram calculados para vários pares $(n*m, p)$. A figura apresenta os resultados para alguns valores de $n * m$. Note que, em geral, as instâncias mais difíceis são geradas para $p \in \{0.5, 0.6, 0.7\}$. Outra característica interessante de $G(n, m, p)$ é que sua densidade tem distribuição homogênea, no sentido que para cada subconjunto E' de arestas de $G(n, m, p)$, a densidade esperada do sugrafo induzido por E' é p também. Tal comportamento é outro fator contribuinte para a dificuldade das instâncias geradas, uma vez que, intuitivamente, uma distribuição irregular na densidade das arestas facilitaria a identificação de subgrafos densos candidatos diretos a se torna *biclusters* em uma solução

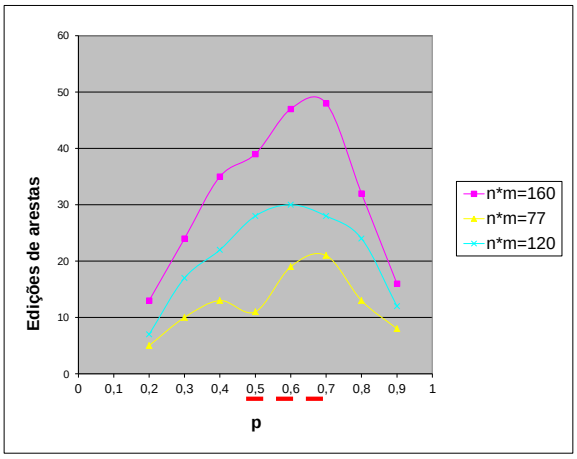


Figura 2.3: Valores $opt(G(n, m, p))$ para $n * m \in \{77, 120, 160\}$.

final.

2.2.2 Regras de Redução

No trabalho [Guo *et al.*, 2008] é proposto e apresentado duas regras de redução; a segunda trabalha com o conceito de conjuntos críticos independentes.

Definição 2.2.1. *Um conjunto S de vértices é chamado um conjunto crítico independente se todos vértices em S tiverem a mesma vizinhança aberta de vértices e S seja maximal sobre esta propriedade.*

Observe que cada conjunto crítico independente é um conjunto independente. A conexão entre conjuntos independentes críticos e o PEB é dado pelo seguinte lema.

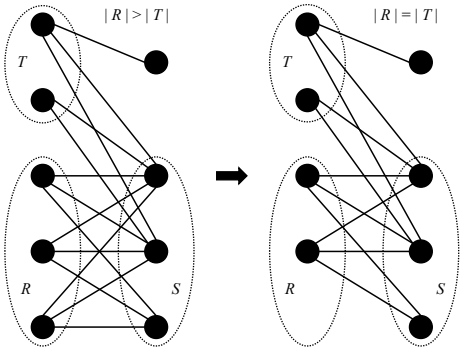


Figura 2.4: Aplicado à Regra 2.

Lema 2.2.1. *[Guo* et al.*, 2008] Para qualquer conjunto independente crítico S, existe uma solução ótima do PEB em que qualquer dois vértices i e j de S terminam no mesmo bicluster.*

As duas regras de redução propostas em [Guo *et al.*, 2008] são apresentadas a seguir:

Regra 1. *Remova todas as componentes conexas que são bicliques do grafo.*

Regra 2. *Considere um conjunto crítico independente R. Seja $S = N(R)$ e $T = N(S) \backslash R$. Se $|R| > |T|$ então remova vértices arbitrários de R até $|R| = |T|$.*

Como ilustrado na Figura 2.4, um vértice de R foi removido sem influenciar o valor ótimo da instância. A corretude da Regra 2 é provada em [Guo *et al.*, 2008].

Agora propomos uma nova regra de redução. Considere a versão ponderada do PEB, onde arestas que atravessam as partições são ponderadas.

Nova Regra. *Considere um conjunto crítico independente R, e seja $S = N(R)$. Agrupe todos os vértices de R em um único vértice i; arestas paralelas também devem ser agrupadas em uma única aresta e, do qual o peso é a soma dos pesos destas arestas agrupadas.*

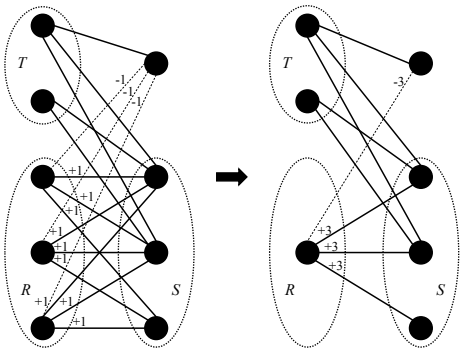


Figura 2.5: Aplicando a Nova Regra de Redução.

Como ilustrado na Figura 2.5, os três vértices de R são agrupados em um único vértice. As arestas agrupadas tem peso $+3$ ou -3.

Lema 2.2.2. *Nova Regra é correta.*

Demonstração. Seja G uma solução ótima para o PEB tal que o conjunto R esteja contido em um *bicluster* B. Também, seja G' uma solução do problema reduzido, gerado de G pelo colapso dos vértices de R em um conjunto R' contido no *bicluster* B'.

Visto que o Lema 2.2.1 prova que existe uma solução ótima do problema original contendo todo o conjunto R no mesmo *bicluster*, simplesmente demonstramos que a solução G' tem o mesmo custo de G, assim provando que G' também será ótimo para o problema reduzido.

O custo da solução G pode ser definido como

$$valor(G) = valor(\bar{R}) + valor(R),$$

onde $valor(\bar{R})$ é o custo de edição das arestas que não possuem um vértice terminal em R, e $valor(R)$ é o custo de edição das arestas que possuem um vértice terminal em R.

Como as arestas que não possuem terminais em R ou R' não são afetadas pela regra de redução, $valor(\bar{R}) = valor(\bar{R}')$.

Note que $valor(R)$ pode ser definida como:

$$valor(R) = \sum_{\substack{v \in N(R) \\ v \notin V(B)}} \sum_{i \in R} w(i,v) \quad - \sum_{\substack{v \notin N(R) \\ v \in V(B)}} \sum_{i \in R} w(i,v). \tag{2.23}$$

Agora, seja r o vértice único de R'. Então:

$$value(R') = \sum_{\substack{v \in N(R') \\ v \notin V(B')}} w(r,v) \quad - \sum_{\substack{v \notin N(R') \\ v \in V(B')}} w(r,v). \tag{2.24}$$

Visto que a regra de redução colapsa arestas que tem o mesmo vértice terminal $v \notin R$ pela adição de seus pesos, os pesos das arestas que possuem terminais em R' são dados por:

$$w(r,v) = \sum_{i \in R} w(i,v), \quad \forall v \in V(B) \text{ and } v \neq r. \tag{2.25}$$

Deste modo, a partir de (2.23), (2.24), e (2.25), temos $valor(R) = valor(R')$, e portanto $valor(G) = valor(G')$. □

Note que Nova Regra funciona com grafos não ponderados, produzindo grafos ponderados.

2.3 Trabalhos da literatura

Na literatura, muitos algoritmos de clusterização procuram particionar objetos em *clusters* maximizando a similaridade dentro do cluster, ou minimizando a similaridade entre clusters, baseado em alguma medida de similaridade. Por exemplo, muitos algoritmos foram propostos por clusterizar entradas baseados em métricas como distância Euclideana, similaridade coseno e coeficiente de Jaccard (vide [Subhashini & Kumar, 2010]). Heurísticas que seguem estes modelos são MSR [Cheng & Church, 2000], Plaid [Lazzeroni & Owen, 2000], OPSM [Ben-Dor *et al.*, 2002], ISA [Bergmann *et al.*, 2003], Spectral [Kluger *et al.*, 2003], xMOTIFs [Murali & Kasif, 2003] e BiMax [Prelić *et al.*, 2006]. Outros algoritmos mais recentes, tais como COALESCE [Huttenhower *et al.*, 2009], CPB [Bozdağ *et al.*, 2009], QUBIC [Li *et al.*, 2009], e FABIA [Hochreiter *et al.*, 2010], adotam este modelo de *biclusterização*.

Um simples modelo de *biclusterização* é edição de *bicluster* onde cada solução é avaliada pelas arestas editadas durante a construção dos *biclusters*. Como condição de consistência para um *bicluster*, é necessária a formação de um biclique, que é, um subgrafo bipartido completo. Além disso, não é permitido aos *biclusters* em uma solução final ter arestas sobrepostas entre eles.

Diversas aplicações em mineração de dados e tratamento de expressões genéticas são resolvidas com a *biclusterização*. Mas apesar de sua importância, existem poucos resultados para o PEB comparado ao CEP. Em [Amit, 2004], os autores provam que o *Problema de Edição de Biclusters* (PEB) é NP-Difícil e apresentam um algoritmo de aproximação com fator 11 baseado na relaxação de um modelo de programação linear. Usando técnicas de decomposição de grafos, o problema pode ser resolvido com complexidade $O(4^k + m)$ [Protti *et al.*, 2009], onde k é o número máximo de arestas editadas permitidas e m o número de arestas do grafo. Além disso, [Sun *et al.*, 2013] propõem um algoritmo exato de fixação de parâmetros para o PEB. Em [Guo *et al.*, 2008], duas regras de redução das instâncias e um algoritmo de aproximação fator 4 baseada em heurísticas aleatórias são apresentados para o PEB. Entretanto, [Ailon *et al.*, 2012] demonstram que o algoritmo descrito em [Guo *et al.*, 2008] tem uma taxa de aproximação ilimitado, além disso, os autores apresentaram dois algoritmos aproximativos de fator 4 para o problema. O primeiro método usa técnicas de LP-rounding e o segundo é uma heurística. [Pinheiro *et al.*, 2016] apresentam um algoritmo exato *branch-and-cut* adotando uma abordagem via programação dinâmica como algoritmo de separação de inequações violadas. Finalmente, nos últimos anos duas heurísticas foram propostas para o PEB: a

heurística de remoção de arestas (EDH), apresentado em [Sun *et al.*, 2013], e a heurística Bi-Force, apresentada em [Sun *et al.*, 2014].

Capítulo 3

Estudo do Politopo do PEB para uso na técnica de branch-and-cut

O MPI $\mathcal{F}_{P_4}$ para o Problema de Edição de *Biclusters*, descrito na Seção 6.3, é composto pelo politopo conhecido como Politopo de Particionamento em Bicliques (em inglês *Biclique Partitioning Polytope* - BPP). Muitos problemas nas literatura possuem este politopo como base para a modelagem de seus próprios politopos. Entre eles temos:

- *K-Biclique Vertex Partition Problem* [Liu, 2013]: seja $G = (V_1, V_2, E)$ um bigrafo de entrada; este problema consiste em encontrar k bicliques cujos vértices $v \in V_1 \cup V_2$ sejam cobertos unicamente por um deste bicliques;
- *Minimum Biclique Cover and Partition* [Amilhastre *et al.*, 1998]: seja $G = (V_1, V_2, E)$ um bigrafo de entrada; este problema consiste em encontrar um número mínimo de bicliques que cubram todas as arestas $(i, j) \in E$. Não há edições nas arestas, apenas seu agrupamento em bicliques distintos.

Logo, o estudo deste politopo será uma contribuição para o PEB e todos os problemas que tenham estas inequações como base para seus próprios politopos.

Neste capítulo descreveremos melhor o politopo BPP com o objetivo de utilizar as novas inequações propostas em uma técnica de *Branch-and-Cut* para o PEB.

3.1 Politopo de particionamento em bicliques

Seja $K_{n,m} = (V_1, V_2, E_{n,m})$, com $|V_1| = n$ e $|V_2| = m$, o bigrafo completo de ordem $n + m$. Para evitar trivialidades, assumimos que $n \geq 2$ e $m \geq 2$. Seja $P_{n,m}$ a envoltória convexa de vetores incidentes do particionamento em bicliques de $K_{n,m}$, isto é,

$$P_{n,m} = conv\{\chi^A \in \mathbb{R}^{nm} |\ A \text{ é um particionamento em bicliques de } K_{n,m}\}. \quad (3.1)$$

$P_{n,m}$ é chamado o politopo de particionamento em bicliques (de ordem $n+m$). A Figura 3.1 apresenta a representação de um particionamento em bicliques A e seu vetor incidente correspondente.

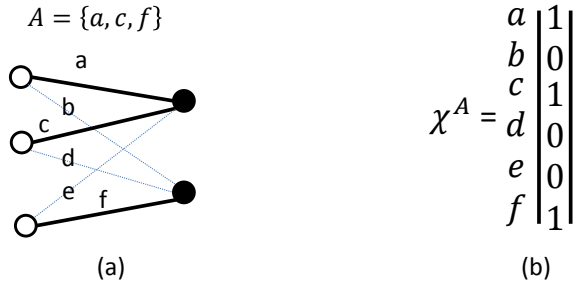


Figura 3.1: (a) Particionamento em Biclique A. (b) Vetor incidente χ^A

O Problema de Particionamento em Bicliques (em inglês, *Biclique Partitioning Problem* - BPP) pode ser visto como um Modelo de Programação Linear (MPL) da forma

$$\min w^T x$$
$$\text{sujeito a } x \in P_{n,m}$$

uma vez que cada solução básica do MPL é um vetor incidente do particionamento em bicliques. Entretanto, com o objetivo de aplicar técnicas de programação linear para resolver este problema, precisamos de uma descrição do $P_{n,m}$ por meio de um sistema de inequações lineares. Como o BPP é um problema NP-Difícil[Amit, 2004], é improvável que possamos encontrar uma descrição completa do $P_{n,m}$.

Iniciamos o BPP como um problema de programação linear inteira. Visto que $P_{n,m}$ está contido em um hipercubo unitário, as *inequações triviais*

$$0 \leq x_e \leq 1 \quad \forall e \in E_{n,m} \quad (3.2)$$

são claramente válidas. Além disso, se A é um particionamento em bicliques e se $a = uw$, $b = ux$ e $c = vx$ são três arestas em A com pontos finais u, x em comum então a aresta $d = vw$ também deve estar em A. Assim para todo quadrado $\{a, b, c, d\}$ de $K_{n,m}$, ilustrado na Figura 3.2, *a inequação quadrada*

$$x_a + x_b + x_c - x_d \leq 2 \tag{3.3}$$

é satisfeita por todo vetor em $P_{n,m}$, e portanto é válida para $P_{n,m}$.

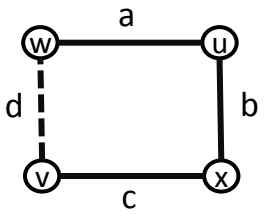


Figura 3.2: Quadrado $\{a, b, c, d\}$.

Note que cada quadrado $\{a, b, c, d\}$ induz de fato quatro inequações quadradas, a saber

$$\begin{aligned} x_a + x_b + x_c - x_d \leq 2 \quad & x_a + x_b - x_c + x_d \leq 2 \\ x_a - x_b + x_c + x_d \leq 2 \quad & -x_a + x_b + x_c + x_d \leq 2. \end{aligned}$$

Adiante iremos falar de inequações quadradas $x_a + x_b + x_c - x_d \leq 2$ induzidas por um quadrado $\{a, b, c, d\}$ e assumimos que ela representa todas as quatro possíveis inequações quadradas.

Considere agora o politopo

$$\begin{aligned} F_{n,m} = \{x \in \mathbb{R}^{E_{n,m}} | 0 \leq x_e \leq 1, & \qquad \forall e \in E_{n,m}, \\ x_a + x_b + x_c - x_d \leq 2 & \qquad \text{para todos os quadrados } \{a, b, c, d\} \text{ de } K_{n,m}\}. \end{aligned}$$

Das observações anteriores decorre que $P_{n,m} \subseteq F_{n,m}$, e é fácil ver que os pontos inteiros de $F_{n,m}$ são exatamente os vetores de incidência do particionamento em bicliques de K_{nm}. Então $P_{n,m} = conv\{x \in F_{n,m} | x \text{ é inteiro}\}$ e isto implica que

$$\begin{aligned} \min w^T x & \\ \text{s.t. } 0 \leq x_e \leq 1, & \qquad \forall e \in E_{nm}, \\ x_a + x_b + x_c - x_d \leq 2, & \qquad \text{para todos os quadrados } \{a, b, c, d\} \text{ de } K_{nm}, \\ x \text{ é inteiro}, & \end{aligned}$$

é uma formulação de programação inteira de BPP. Observe que $P_{n,m}$ contém o vetor nulo e todos os vetores unitários; então $P_{n,m}$ tem dimensão cheia, isto é,

$$dim\ P_{n,m} = |E_{nm}| = nm.$$

Isto implica que para toda faceta de $P_{n,m}$ existe uma única inequação que a define.

Teorema 3.1.1. *Para todo politopo de particionamento em bicliques $P_{n,m}$, $n \geq 2$ e $m \geq 2$, podemos afirmar:*

a) Toda restrição de não negatividade $x_e \geq 0$ define uma faceta de $P_{n,m}$;

b) Toda inequação de limite superior $x_e \leq 1$ define uma faceta de $P_{n,m}$;

c) Toda inequação quadrada $x_a + x_b + x_c - x_d \leq 2$ define uma faceta de $P_{n,m}$.

Demonstração. (a) Seja $e \in E_{n,m}$. Então $x_e = 0$ é satisfeita pelo vetor nulo e todos os vetores unitários $\chi^{\{f\}}$, $f \in E_{n,m}$, $f \neq e$. Estes $|E_{n,m}|$ vetores são pertencentes a $P_{n,m}$ e são afim independentes.

(b) Seja $e \in E_{n,m}$. Então $x_e = 1$ é satisfeito pelo vetor unitário $\chi^{\{e\}}$ e os vetores $\chi^{\{e,f\}}$, $\forall f \in E_{n,m}$, $f \neq e$. Estes $|E_{n,m}|$ vetores pertencentes a $P_{n,m}$ e são afim independentes.

(c) Seja $\{a, b, c, d\}$ um quadrado em $K_{n,m}$, ilustrado na Figura 3.2, e $a = uw$, $b = ux$, $c = vx$, $d = vw$. Seja também $a^T x \leq a_0$ a inequação (3.3), isto é, $a^T x = x_a + x_b + x_c - x_d \leq 2 = a_0$, e seja $b^T x \leq b_0$ uma faceta definida para $P_{n,m}$ tal que $F_a = \{x \in P_{n,m} | a^T x = a_0\} \subseteq F_b = \{x \in P_{n,m} | b^T x = b_0\}$. Claramente, $F_a \neq P_{n,m}$; assim se pudermos provar que $b = \alpha a$ para algum $\alpha \in \mathbb{R}$, uma vez que $F_a \neq \emptyset$ poderemos concluir que (3.3) define uma faceta de $P_{n,m}$.

Observando a Figura 3.2 é possível perceber que $\chi^{\{a,b\}}$, $\chi^{\{a,c\}}$, $\chi^{\{b,c\}}$, $\chi^{\{a,b,c,d\}} \in F_a \subseteq F_b$, e portanto, $0 = b_0 - b_0 = b^T\chi^{\{a,b\}} - b^T\chi^{\{a,c\}} = b_b - b_c$ e $0 = b_0 - b_0 = b^T\chi^{\{a,b\}} - b^T\chi^{\{b,c\}} = b_a - b_c$; então $b_a = b_b = b_c = \alpha$. Fazendo $0 = b_0 - b_0 = b^T\chi^{\{a,b,c,d\}} - b^T\chi^{\{b,c\}} = b_a + b_d$, então $b_a = -b_d$. Logo, $b_d = -\alpha$.

Por fim, iremos mostrar que os outros coeficientes da inequação são iguais a 0. Observando a Figura 3.2 é possível perceber que para qualquer $e \in E_{nm}$, onde $e \notin (\delta(x) \cup \delta(w))$, temos $\chi^{\{a,b,e\}} \in F_a \subseteq F_b$, e portanto, $0 = b_0 - b_0 = \chi^{\{a,b,e\}} - \chi^{\{a,b\}} = b_e$.

Agora, para qualquer $e \in E_{nm}$, onde $e \in (\delta(x) \cup \delta(w) - \{a, b, c, d\})$, temos $\chi^{\{b,c,e\}} \in F_a \subseteq F_b$, e portanto, $0 = b_0 - b_0 = b^T\chi^{\{b,c,e\}} - b^T\chi^{\{b,c\}} = b_e$. Logo, para qualquer $e \in (E_{n,m} - \{a, b, c, d\})$ vale que $b_e = 0$.

Estas observações implicam que a inequação (3.3) define uma faceta de $P_{n,m}$.

□

A seguinte observação define uma propriedade estrutural das inequações facetas de $P_{n,m}$. Esta prova foi apresentada em [Grötschel & Wakabayashi, 1990] para o politopo de de particionamento em cliques, sendo a lógica a mesma para o politopo aqui abordado.

Proposição 3.1.2. *Sejam $a^Tx \leq \alpha$ uma faceta não trivial para $P_{n,m}$, $V_{nm} = V_1 \cup V_2$ e seja $E_a = \{e \in E_{nm} | a_e \neq 0\}$. Então o subgrafo $H = K_{nm}[E_a]$ é conexo.*

Demonstração. Suponha H seja desconexo. Seja W_1 o conjunto de vértices de uma componente conexa não trivial de H, e seja $W_2 = V_{nm} \setminus W_1$. Para $i = 1, 2$ seja $F_i = E_{nm}(W_i)$ o conjunto de arestas em K_{nm} com ambos os vértices terminais em W_i, e seja a^i um vetor definido por $(a^i)_e = a_e$ se $e \in F_i$ e $(a^i)_e = 0$, caso contrário. Agora seja D uma partição em biclique de K_{nm} tal que $a^T\chi^D \leq \alpha$, e seja $D_1 = D \cap F_1$ e $D_2 = D \cap F_2$. Claramente D_1 e D_2 são partições de bicliques de K_{nm}. Sejam $\alpha_1 = a^T\chi^{D_1}$ e $\alpha_2 = a^T\chi^{D_2}$. Então $\alpha_1 + \alpha_2 = \alpha$, e além disto $a^{1T}x \leq \alpha_1$ e $a^{2T}x \leq \alpha_2$ são inequações válidas para $P_{n,m}$. Visto que $a^Tx \leq \alpha$ é a soma de duas inequações, temos uma contradição. □

3.2 Definindo facetas

Nesta seção definiremos novas inequações que descrevem o $P_{n,m}$.

3.2.1 Inequações Escada

Definição 3.2.1. *(Grafo Escada) O Grafo Escada $L_k = (V_1, V_2, E^+ \cup E^-)$ é um grafo bipartido de dimensão $k \times k$. Seu conjunto de arestas E é dividido em dois subconjuntos E^+ e E^- e seu grafo base é L_2, ilustrado na Figura 3.3, que é o grafo das inequações quadradas. A construção do grafo L_k é recursivamente definida como se segue:*

if $k = 2$ **then return** L_2
else
$V_1[L_k] = V_1[L_{k-1}] \cup \{k\}$

$V_2[L_k] = V_2[L_{k-1}] \cup \{k'\}$

if k é ímpar **then**

$E^+[L_k] = E^+[L_{k-1}] \cup \{(k, k'), (k-1, k')\}$

$E^-[L_k] = E^-[L_{k-1}] \cup \{(k, (k-1)'), (k, (k-2)')\}$

else

$E^+[L_k] = E^+[L_{k-1}] \cup \{(k, k'), (k, (k-1)')\}$

$E^-[L_k] = E^-[L_{k-1}] \cup \{(k-1, k'), (k-2, k')\}$

Uma possível topologia do grafo L_k é ilustrada na Figura 3.3.

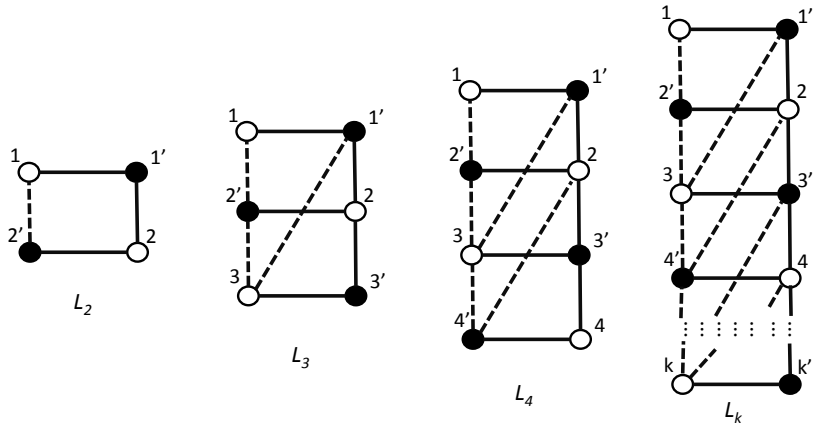


Figura 3.3: Topologia para o Grafo Escada.

Teorema 3.2.1. *Seja L_k o Grafo Escada com $k \geq 2$, a correspondente inequação*

$$x(E^+[L_k]) - x(E^-[L_k]) \leq k, \tag{3.4}$$

é válida para $P_{n,m}$ e define uma faceta.

Demonstração. Provamos a validade de (3.4) para o grafo escada pela indução em $k \geq 2$. A Figura 3.4 apresenta subgrafos específicos induzidos por vértices de L_k. Podemos observar que as arestas do subgrafo U são as únicas que aparecem apenas uma vez nos outros subgrafos.

Para $k = 2$ a validade é imediata. Pela hipótese de indução, adicionamos as inequações (3.4) dos subgrafos Escada L^1_{k-1}, L^2_{k-1} e L_2 de L_k

$$2(x(E^+[L_k] \setminus E^+[U]) - x(E^-[L_k] \setminus E^-[U])) + x(E^+[U]) - x(E^-[U]) \leq 2k$$

Adicionando $-x(E^-[U]) \leq 0$ e $x(E^+[U]) \leq |E^+[U]|$ (onde $|E^+[U]| = 1$) para a

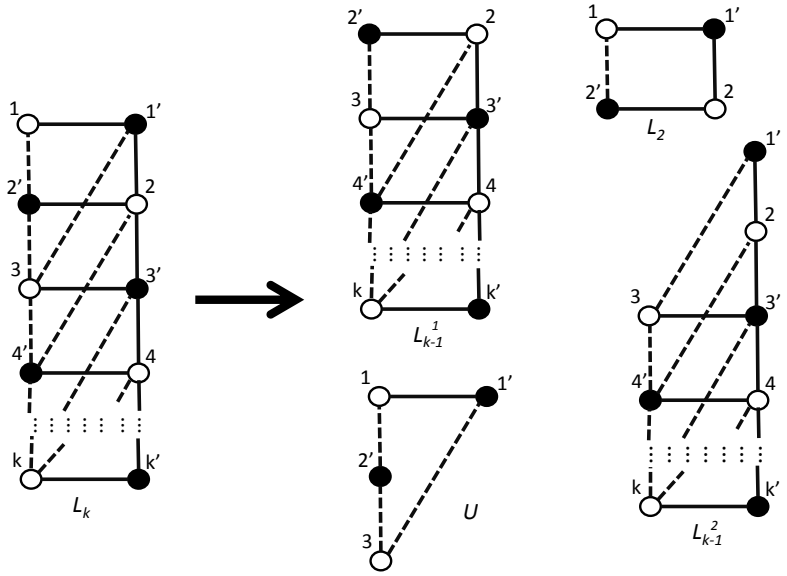


Figura 3.4: Subgrafo L^1_{k-1}, L^2_{k-1}, L_2 e U do Grafo Escada L_k.

inequação acima, temos a seguinte inequação válida para $P_{n,m}$

$$2(x(E^+[L_k]) - x(E^-[L_k])) \leq 2k + 1$$

e, portanto,

$$x(E^+[L_k]) - x(E^-[L_k]) \leq \lfloor k + 1/2 \rfloor = k$$

é válida para $P_{n,m}$.

Agora provaremos que (3.4) define uma faceta de $P_{n,m}$.

Seja $a^T x \leq a_0$ a inequação (3.4), isto é, $a^T x = x(E^+[L_k]) - x(E^-[L_k]) \leq k = a_0$, e seja $b^T x \leq b_0$ uma faceta definida para $P_{n,m}$ tal que $F_a = \{x \in P_{n,m} | a^T x = a_0\} \subseteq F_b = \{x \in P_{n,m} | b^T x = b_0\}$. Claramente, $F_a \neq P_{n,m}$; assim se pudermos provar que $b = \alpha a$, para algum $\alpha \in \mathbb{R}$ uma vez que $F_a \neq \emptyset$ podemos concluir que (3.4) define uma faceta de $P_{n,m}$. Para isto, iniciamos estabelecendo o seguinte:

Lema 3.2.2. *Existe $\alpha \in \mathbb{R}$ tal que $b_e = \alpha$ para todo $e \in E^+[L_k]$.*

Demonstração. Para provar o Lema 3.2.2 definimos o conjunto M (ilustrado na Figura 3.5) como

$$M = \{(i, i') \in E^+[L_k]\}$$

sendo M um *k-emparelhamento* do grafo L_k.

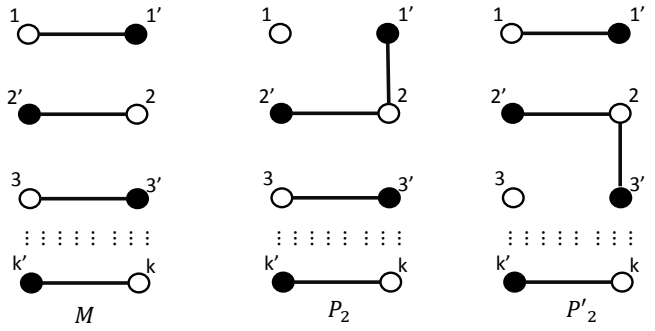


Figura 3.5: Subgrafos M, P_2 e P_2' do Grafo Escada L_k.

Para $i \in \{1, 3, .., 2 \cdot \lceil k/2 \rceil - 1\}$ e $j = i + 1$ considere os particionamentos em bicliques:

$$I_i = M \cup \{(i-1), i'\} - \{(i-1), (i-1)'\} \text{ e}$$

$$I_i' = M \cup \{(i+1), i'\} - \{(i+1), (i+1)'\}.$$

$$P_j = M \cup \{j, (j-1)'\} - \{(j-1), (j-1)'\},$$

$$P_j' = M \cup \{j, (j+1)'\} - \{(j+1), (j+1)'\}.$$

Observando os subgrafos M, P_2 e P_2' de L_k, ilustrados na Figura 3.5, é possível perceber que χ^M, χ^{P_j}, $\chi^{P_j'}$, χ^{I_i}, $\chi^{I_i'} \in F_a \subseteq F_b$, e portanto $0 = b_0 - b_0 = b^T\chi^M - b^T\chi^{P_j} = b_{j(j-1)'} - b_{(j-1)(j-1)'}$ e $0 = b_0 - b_0 = b^T\chi^M - b^T\chi^{I_i'} = b_{(i+1)i'} - b_{(i+1)(i+1)'}$. Logo $b_{j(j-1)'} = b_{(j-1)(j-1)'}$ e $b_{(i+1)i'} = b_{(i+1)(i+1)'}$. Como $j = i + 1$ temos que $b_{j(j-1)'} = b_{(i+1)i'}$. Aplicando este processo para todo $i \in \{1, 3, .., 2 \cdot \lceil k/2 \rceil - 1\}$ e $j = i + 1$ e sabendo que L_k é um grafo conexo, temos que para toda aresta $e \in E^+[L_k]$ vale $b_e = \alpha$.

□

Lema 3.2.3. *Seja K_{nm} a biclique de dimensão $n \times m$ relacionada ao $P_{n,m}$. Então para todo $e \in \{E[K_{nm}] \setminus (E^+[L_k] \cup E^-[L_k])\}$, $b_e = 0$.*

Demonstração. Para $i \in \{3, 5, .., 2 \cdot \lceil k/2 \rceil - 1\}$, $j \in \{i-2, .., 1\}$, $p \in \{4, 6, .., 2 \cdot \lfloor k/2 \rfloor\}$ e $l \in \{k-2, .., 2\}$ considere os particionamentos em bicliques:

$$Z_{ij} = M \cup \{\{j, i'\}, \{(j+1), j'\}\} - \{j, j'\} \text{ e } Z_{pl}' = M \cup \{\{p, l'\}, \{l, (l+1)'\}\} - \{l, l'\}.$$

Observando o subgrafo Z_{31} de L_k, ilustrado na Figura 3.6, é possível perceber que $\chi^{Z_{ij}}$, $\chi^{Z_{pl}'} \in F_a \subseteq F_b$, e portanto $0 = b_0 - b_0 = b^T\chi^{Z_{ij}} - b^T\chi^M = b_{ji'} + b_{(j+1)j'} - b_{jj'}$; como já sabemos que $b_{(j+1)j'} = b_{jj'}$, então $b_{ji'} = 0$. E ainda $0 = b_0 - b_0 = b^T\chi^{Z_{pl}'} - b^T\chi^M =$

$b_{pl'} + b_{l(l+1)'} - b_{ll'}$; como já sabemos que $b_{l(l+1)'} = b_{ll'}$, segue que $b_{pl'} = 0$. Aplicando este processo para todo $i \in \{3, 5, .., 2 \cdot \lceil k/2 \rceil - 1\}$, $j \in \{i-2, .., 1\}$, $p \in \{4, 6, .., 2 \cdot \lfloor k/2 \rfloor\}$ e $l \in \{p-2, .., 2\}$, temos que para as arestas $(i, j'), (p, l') \in \{E[K_{nm}] \setminus (E^+[L_k] \cup E^-[L_k])\}$ vale que $b_{ij'} = 0$ e $b_{pl'} = 0$.

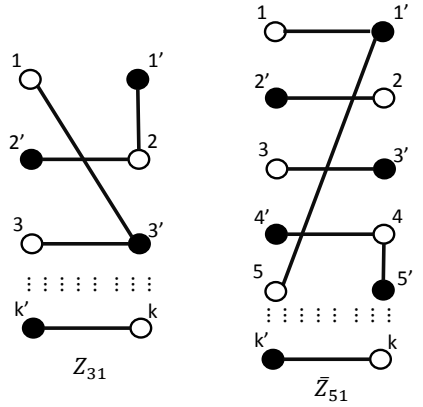


Figura 3.6: Subgrafos Z_{31} e $\bar{Z}_{51}$ do Grafo Escada L_k.

Para $i \in \{5, 7, .., 2 \cdot \lceil k/2 \rceil - 1\}$, $j \in \{i-4, .., 1\}$, $p \in \{6, 8, .., 2 \cdot \lfloor k/2 \rfloor\}$ e $l \in \{p-4, .., 2\}$ considere os particionamentos em bicliques:

$$\bar{Z}_{ij} = M \cup \{\{i, j'\}, \{(i-1), i'\}\} - \{i, i'\} \text{ e } \bar{Z'}_{pl} = M \cup \{\{l, p'\}, \{p, (p-1)'\}\} - \{p, p'\}.$$

Observando o subgrafo $\bar{Z}_{51}$ de L_k, ilustrado na Figura 3.6, é possível perceber que $\chi^{\bar{Z}_{ij}}$, $\chi^{\bar{Z'}_{pl}} \in F_a \subseteq F_b$, e portanto $0 = b_0 - b_0 = b^T\chi^{\bar{Z}_{ij}} - b^T\chi^M = b_{ij'} + b_{(i-1)i'} - b_{i,i'}$; como já sabemos que $b_{(i-1)i'} = b_{ii'}$, então $b_{ij'} = 0$. E ainda $0 = b_0 - b_0 = b^T\chi^{\bar{Z'}_{pl}} - b^T\chi^M = b_{lp'} + b_{p(p-1)'} - b_{p,p'}$; como $b_{p(p-1)'} = b_{pp'}$, segue que $b_{lp'} = 0$. Aplicando este processo para todo $i \in \{5, 7, .., 2 \cdot \lceil k/2 \rceil - 1\}$, $j \in \{i-4, .., 1\}$, $p \in \{6, 8, .., 2 \cdot \lfloor k/2 \rfloor\}$ e $l \in \{p-4, .., 2\}$, temos que para as arestas $(i, j'), (p, l') \in \{E[K_{nm}] \setminus (E^+[L_k] \cup E^-[L_k])\}$ vale que $b_{ij'} = 0$ e $b_{pl'} = 0$.

Por fim, para todo $(i, j) \in \{E[K_{nm}] \setminus (E^+[L_k] \cup E^-[L_k])\}$, $i \notin V_1[L_k]$ ou $j \notin V_2[L_k]$, temos que $\chi^{\{M,(i,j)\}} \in F_a \subseteq F_b$, e portanto, $0 = b_0 - b_0 = b^T\chi^{\{M,(i,j)\}} - b^T\chi^M = b_{ij}$. Logo, para toda aresta $(i, j) \in \{E[K_{nm}] \setminus (E^+[L_k] \cup E^-[L_k])\}$ temos $b_{ij} = 0$.

Assim, para todo $e \in \{E[K_{nm}] \setminus (E^+[L_k] \cup E^-[L_k])\}$ vale que $b_e = 0$.

□

Lema 3.2.4. *Existe $\alpha \in \mathbb{R}$ tal que $b_e = -\alpha$ para todo $e \in E^-[L_k]$.*

Demonstração. Para provar o Lema 3.2.4, primeiro abordaremos as arestas verticais $e \in E^-[L_k]$. Logo, para $i \in \{2, 4, .., 2 \cdot \lfloor k/2 \rfloor\}$ e $j \in \{1, 3, .., 2 \cdot \lceil k/2 \rceil - 1\}$ definimos os

particionamentos em bicliques:

$$\bar{A}_i = M \cup \{\{i,(i-1)'\},\{(i-1),i'\}\} \text{ e } \bar{A'}_j = M \cup \{\{(j-1),j'\},\{j,(j-1)'\}\}.$$

Observando o subgrafo $\bar{A}_3$ de L_k, ilustrado na Figura 3.7, é possível perceber que $\chi^{\bar{A}_i}$, $\chi^{\bar{A'}_j} \in F_a \subseteq F_b$, e portanto $0 = b_0 - b_0 = b^T\chi^{\bar{A}_i} - b^T\chi^M = b_{i(i-1)'} + b_{(i-1)i'}$ e $0 = b_0 - b_0 = b^T\chi^{\bar{A'}_i} - b^T\chi^M = b_{(j-1)j'} + b_{j(j-1)'}$. Logo $b_{i(i-1)'} = -b_{(i-1)i'}$ e $b_{(j-1)j'} = -b_{j(j-1)'}$. Aplicando este processo para todo $i \in \{2,4,..,2\cdot\lfloor k/2\rfloor\}$ e $j \in \{1,3,..,2\cdot\lceil k/2\rceil - 1\}$, temos que para toda aresta vertical $e \in E^+[L_k]$ vale que $b_e = -\alpha$.

Abordando as arestas transversais $e \in E^-[L_k]$, para $i \in \{2,4,..,2\cdot\lfloor (k-1)/2\rfloor\}$ e $j \in \{1,3,..,2\cdot\lceil (k-1)/2\rceil - 1\}$ definimos os particionamentos em bicliques:

$$\bar{C}_i = M \cup \{\{i,(i-1)'\},\{i,(i+1)'\},\{(i-1),(i+1)'\},\{(i+1),(i-1)'\}\} - \{i,i'\} \text{ e}$$
$$\bar{C'}_j = M \cup \{\{(j-1),j'\},\{(j+1),j'\},\{(j+1),(j-1)'\},\{(j-1),(j+1)'\}\} - \{j,j'\}.$$

Observando o subgrafo $\bar{C}_2$ de L_k, ilustrado na Figura 3.7, é possível perceber que $\chi^{\bar{C}_i}$, $\chi^{\bar{C'}_j} \in F_a \subseteq F_b$, e portanto $0 = b_0 - b_0 = b^T\chi^{\bar{C}_i} - b^T\chi^M = b_{i(i-1)'} + b_{i(i+1)'} + b_{(i-1)(i+1)'} + b_{(i+1)(i-1)'} - b_{i,i'}$. Como $b_{i(i+1)'} = b_{ii'}$ e $b_{(i-1)(i+1)'} = 0$, então $b_{i(i-1)'} = -b_{(i+1)(i-1)'}$. Fazendo $0 = b_0 - b_0 = b^T\chi^{\bar{C'}_j} - b^T\chi^M = b_{(j-1)j'} + b_{(j+1)j'} + b_{(j+1)(j-1)'} + b_{(j-1)(j+1)'} - b_{j,j'}$, e como já sabemos que $b_{(j+1)j'} = b_{jj'}$ e $b_{(j+1)(j-1)'} = 0$, segue que o $b_{(j-1)j'} = -b_{(j-1)(j+1)'}$. Aplicando este processo para todo $i \in \{2,4,..,2\cdot\lfloor (k-1)/2\rfloor\}$ e $j \in \{1,3,..,2\cdot\lceil (k-1)/2\rceil - 1\}$, temos que as arestas transversais $e \in E^-[L_k]$ possuem $b_e = -\alpha$.

Assim, para todo $e \in E^-[L_k]$ vale que $b_e = -\alpha$.

□

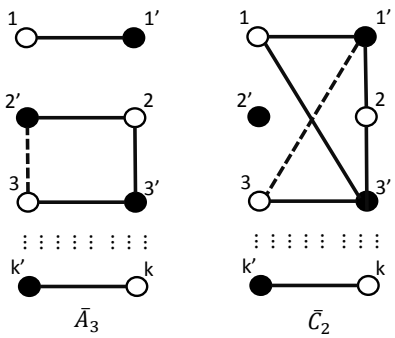


Figura 3.7: Subgrafos $\bar{A}_3$ e $\bar{C}_2$ de L_k.

Os três lemas implicam que a inequação (3.4) define uma faceta de $P_{n,m}$.

□

3.2.2 Inequações Fole

Definição 3.2.2. *(Grafo Fole) O Grafo Fole $B_k = (V_1, V_2, E^+ \cup E^-)$ é um grafo bipartido de dimensão $k \times k$. Seu conjunto de arestas E é dividido em dois subconjuntos E^+ e E^- e seu grafo base B_2, ilustrado na Figura 3.8, é o grafo da inequação quadrada. A construção do grafo B_k é definida recursivamente a seguir:*

if $k = 2$ **then return** B_2
else if $k \geq 3$ **then**
$V_1[B_k] = V_1[B_{k-1}] \cup \{k\}$
$V_2[B_k] = V_2[B_{k-1}] \cup \{k'\}$
$E^+[B_k] = E^+[B_{k-1}] \cup \{(k, k'), (1, k')\}$
$E^-[B_k] = E^-[B_{k-1}]$
for $j \in V_2[B_{k-1}]$ **do**
$E^-[B_k] = E^-[B_k] \cup \{(k, j)\}$

Uma possível topologia do grafo B_k é ilustrada na Figura 3.8.

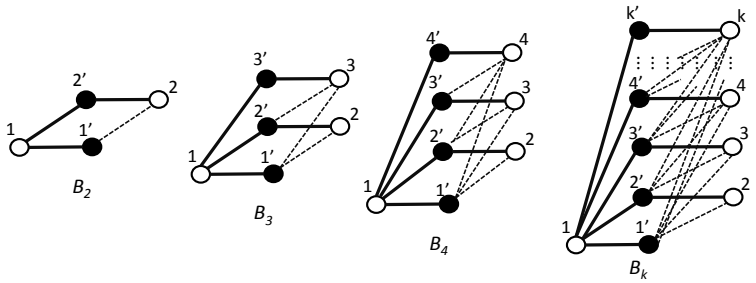


Figura 3.8: Topologia para o Grafo Fole.

Teorema 3.2.5. *Seja B_k o Grafo Fole com $k \geq 2$, a correspondente inequação*

$$x(E^+[B_k]) - x(E^-[B_k]) \leq k, \tag{3.5}$$

é válida para $P_{n,m}$ e define uma faceta.

Demonstração. Provamos a validade de (3.5) para o grafo fole pela indução em $k \geq 2$. A Figura 3.9 apresenta subgrafos específicos induzidos por vértices de B_k. Podemos ver que as arestas do subgrafo U são as únicas que aparecem apenas uma vez nos outros subgrafos.

Para $k = 2$ a validade é imediata. Pela hipótese de indução, adicionamos as inequações (3.5) dos subgrafos Fole B^1_{k-1}, B^2_{k-1} e B_2 de B_k

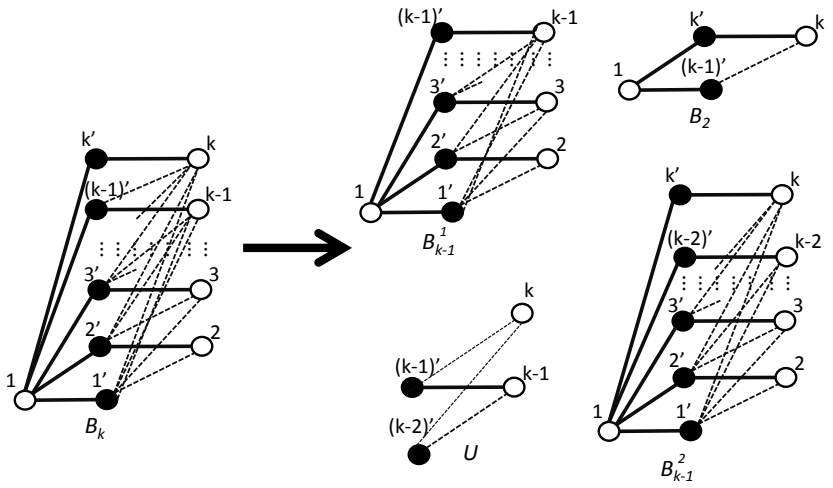


Figura 3.9: Subgrafos B^1_{k-1}, B^2_{k-1}, B_2 e U do Grafo Fole B_k.

$$2(x(E^+[B_k] \setminus E^+[U]) - x(E^-[B_k] \setminus E^-[U])) + x(E^+[U]) - x(E^-[U]) \leq 2k$$

Adicionando $-x(E^-[U]) \leq 0$ e $x(E^+[U]) \leq |E^+[U]|$ (onde $|E^+[U]| = 1$) para a inequação anterior, temos a seguinte inequação válida para $P_{n,m}$

$$2(x(E^+[B_k]) - x(E^-[B_k])) \leq 2k + 1$$

e, portanto,

$$x(E^+[B_k]) - x(E^-[B_k]) \leq \lfloor k + 1/2 \rfloor = k$$

é válida para $P_{n,m}$.

Agora provaremos que (3.5) define uma faceta de $P_{n,m}$.

Seja $a^T x \leq a_0$ a inequação (3.5), isto é, $a^T x = x(E^+[B_k]) - x(E^-[B_k]) \leq k = a_0$, e seja $b^T x \leq b_0$ uma inequação definindo faceta para $P_{n,m}$ tal que $F_a = \{x \in P_{n,m} | a^T x = a_0\} \subseteq F_b = \{x \in P_{n,m} | b^T x = b_0\}$. Claramente, $F_a \neq P_{n,m}$. Assim, se pudermos provar que $b = \alpha a$ para algum $\alpha \in \mathbb{R}$, uma vez que $F_a \neq \emptyset$. Podemos concluir que (3.5) define uma faceta de $P_{n,m}$. Para isto iniciamos estabelecendo o seguinte:

Lema 3.2.6. *Existe $\alpha \in \mathbb{R}$ tal que $b_e = \alpha$ para todo $e \in E^+[B_k]$.*

Demonstração. Para provar o Lema 3.2.6 definimos uma nova topologia para o conjunto M (ilustrado na Figura 3.10).

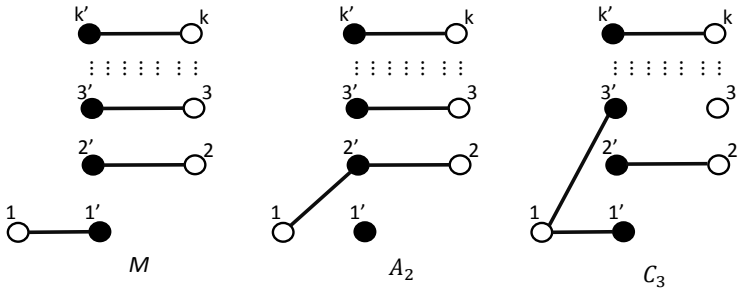


Figura 3.10: Subgrafos M, A_2 e C_3 do Grafo Fole B_k.

Para $i \in \{2, .., k\}$ considere os particionamentos em bicliques:

$$A_i = M \cup \{1, i'\} - \{1, 1'\} \text{ e } C_i = M \cup \{1, i'\} - \{i, i'\}.$$

Observando os subgrafos M, A_2 e C_3 de B_k, ilustrados na Figura 3.10, é possível perceber que χ^M, χ^{A_i}, $\chi^{C_i} \in F_a \subseteq F_b$, e portanto $0 = b_0 - b_0 = b^T\chi^M - b^T\chi^{A_i} = b_{11'} - b_{1i'}$ e $0 = b_0 - b_0 = b^T\chi^M - b^T\chi^{C_i} = b_{ii'} - b_{1i'}$. Logo $b_{11'} = b_{1i'}$ e $b_{ii'} = b_{1i'}$. Aplicando este processo para todo $i \in \{2, .., k\}$ e sabendo que B_k é um grafo conexo, temos que para toda aresta $(i, j) \in E^+[B_k]$ vale que $b_{ij} = \alpha$.

□

Lema 3.2.7. *Seja K_{nm} a biclique de dimensão $n \times m$ relacionada ao $P_{n,m}$. Então para todo $e \in \{E[K_{nm}] \setminus (E^+[B_k] \cup E^-[B_k])\}$ vale que $b_e = 0$.*

Demonstração. Para $i \in \{3.., k\}$ e $j \in \{i-1, ..2\}$ considere o particionamento em bicliques:

$$Z_{ij} = M \cup \{\{j, i'\}, \{1, j'\}\} - \{j, j'\}.$$

Observando o subgrafo Z_{32} de B_k, ilustrado na Figura 3.11, é possível perceber que $\chi^{Z_{ij}} \in F_a \subseteq F_b$, e portanto $0 = b_0 - b_0 = b^T\chi^{Z_{ij}} - b^T\chi^M = b_{ji'} + b_{1j'} - b_{jj'}$. Como $b_{1j'} = b_{jj'}$, então $b_{ji'} = 0$ para todo $i \in \{3.., k\}$ e $j \in \{i - 1, ..2\}$.

Por fim, para todo $(i, j) \in \{E[K_{nm}] \setminus (E^+[B_k] \cup E^-[B_k])\}$, $i \notin V_1[B_k]$ ou $j \notin V_2[B_k]$, temos que $\chi^{\{M,(i,j)\}} \in F_a \subseteq F_b$, e portanto $0 = b_0 - b_0 = b^T\chi^{\{M,(i,j)\}} - b^T\chi^M = b_{ij}$. Logo, para toda aresta $(i, j) \in \{E[K_{nm}] \setminus (E^+[B_k] \cup E^-[B_k])\}$ vale que $b_{ij} = 0$.

□

Lema 3.2.8. *Existe $\alpha \in \mathbb{R}$ tal que $b_e = -\alpha$ para todo $e \in E^-[B_k]$.*

Demonstração. Para provar o Lema 3.2.8, primeiro abordaremos as arestas $\{i, 1'\} \in E^-[B_k]$. Logo, para $i \in \{2, .., k\}$ definimos o particionamento em bicliques:

$$\bar{A}_i = M \cup \{\{1, i'\}, \{i, 1'\}\}.$$

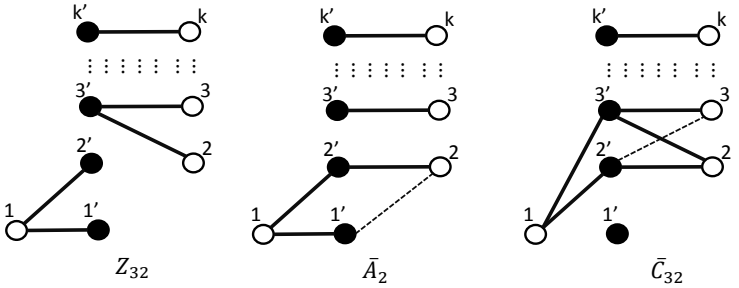


Figura 3.11: Subgrafos Z_{32}, $\bar{A}_2$ e $\bar{C}_{32}$ de B_k.

Observando o subgrafo $\bar{A}_2$ de B_k, ilustrado na Figura 3.11, é possível perceber que $\chi^{\bar{A}_i} \in F_a \subseteq F_b$, e portanto $0 = b_0 - b_0 = b^T\chi^{\bar{A}_i} - \chi^M = b_{i1'} + b_{1i'}$. Logo $b_{1i'} = -b_{i1'}$. Aplicando este processo para todo $i \in \{2, .., k\}$, temos que para toda aresta $(i, 1') \in E^-[B_k])$ vale que $b_{i1'} = -\alpha$. Abordando as outras arestas de $E^-[B_k]$, definimos para $i \in \{3.., k\}$ e $j \in \{i-1, .., 2\}$ o seguinte particionamento em bicliques:

$$\bar{C}_{ij} = M \cup \{\{i, j'\}, \{j, i'\}, \{1, i'\}, \{1, j'\}\} - \{1, 1'\}.$$

Observando o subgrafo $\bar{C}_{32}$ de B_k, ilustrado na Figura 3.11, é possível perceber que $\chi^{\bar{C}_{ij}} \in F_a \subseteq F_b$, e portanto $0 = b_0 - b_0 = b^T\chi^{\bar{C}_{ij}} - b^T\chi^M = b_{ij'} + b_{ji'} + b_{1i'} + b_{1j'} - b_{11'}$. Como $b_{1j'} = b_{11'}$ e $b_{j,i'} = 0$, então $b_{1,i'} = -b_{ij'}$. Aplicando este processo para $i \in \{3.., k\}$ e $j \in \{i-1, .., 2\}$, temos que todas as arestas $(i, j) \in E^-[B_k]$ vale que $b_{ij} = -\alpha$.

Assim, para todo $e \in E^-[B_k]$ vale que $b_e = -\alpha$.

□

Os três lemas implicam que a inequação (3.5) define uma faceta de $P_{n,m}$.

□

3.2.3 Inequações Grid

Definição 3.2.3. *(Grafo Grid) O Grafo Grid $G_k = (V_1, V_2, E^+ \cup E^-)$ é um grafo bipartido de dimensão $k \times k$. Seu conjunto de arestas E é dividido em dois subconjuntos E^+ e E^- e*

seu grafo base G_2, ilustrado na Figura 3.12, é o grafo da inequação quadrada. A construção do grafo G_k é definida recursivamente a seguir:

if $k = 2$ **then return** G_2
else
$V_1[G_k] = V_1[G_{k-1}] \cup \{k\}$
$V_2[G_k] = V_2[G_{k-1}] \cup \{k'\}$
$E^+[G_k] = E^+[G_{k-1}] \cup \{(k, k'), (k, (k-2)')\}$
$E^-[G_k] = E^-[G_{k-1}]$
if k é ímpar **then**
$E^-[G_k] = E^-[G_k] \cup \{(k-1, k'), (k-2, k')\}$
if $k > 3$ **then**
$E^-[G_k] = E^-[G_k] \cup \{(k-4, k')\}$
else
$E^+[G_k] = E^+[G_k] \cup \{(k, (k-1)')\}$
for $j \in (k-3)..(k-1)$ **do**
$E^-[G_k] = E^-[G_k] \cup \{(j, k')\}$
$E^-[G_k] = E^-[G_k] \cup \{(k-2, (k-1)'), (k, (k-3)')\}$
if $k > 4$ **then**
$E^-[G_k] = E^-[G_k] \cup \{(k, (k-4)'), (k-4, (k-1)')\}$

Uma possível topologia do grafo G_k está ilustrada na Figura 3.12.

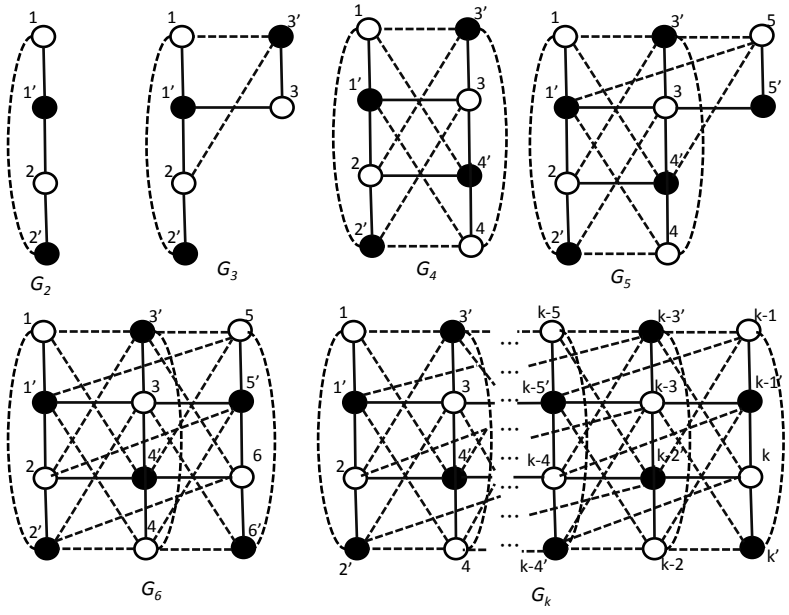


Figura 3.12: Topologia para o Grafo Grid.

Teorema 3.2.9. *Seja G_k o Grafo Grid com $k \geq 2$, a correspondente inequação*

$$x(E^+[G_k]) - x(E^-[G_k]) \leq k, \tag{3.6}$$

é válida para $P_{n,m}$ e define uma faceta.

Demonstração. Seja $N_k^+(v) = \{u \in G_k \mid (v,u) \in E^+[G_k]\}$ a vizinhança positiva do vértice v e $C_k = \{v \in G_k \mid \left|N_k^+(v)\right| > 1\}$, o conjunto de vértices centrais de G_k. Suponha $|C_k| = k$. Seja também $a_k^T x \leq k$ a inequação para o grafo Grid G_k, B um particionamento em bicliques qualquer, χ^B o vetor de incidência do particionamento B, sendo $\chi_{ij}^B = 1$ quando a aresta $(i,j) \in B$ e $\chi_{ij}^B = 0$ caso contrario. Por fim, $a_k^T \chi^B$ o valor assumido pela inequação de G_k induzido pelo particionamento B. Para facilitar a descrição desta prova sempre que dissermos que uma variável x_{ij} está em B, isto significa $(i,j) \in B$ ou que $\chi_{ij}^B = 1$.

Provamos a validade de (3.6) para o grafo Grid G_k e que para todo $v \in C_k$, quando $a_k^T \chi^B = k$ então $\sum_{u \in N_k^+(v)} \chi_{vu}^B \geq 1$, por indução em $k \geq 2$.

Para $k = 2$ a validade é imediata. Como G_2 possui três variáveis positivas cujas arestas associadas incidem sobre os dois vértices de C_2 e sendo necessário conter pelo menos duas destas arestas em B para que $a_2^T \chi^B = 2$, cada vértice $v \in C_2$ terá $\sum_{u \in N_2^+(v)} \chi_{vu}^B \geq 1$.

Para $k = 3$ podemos observar pela ilustração da Figura 3.13(a) a adição das variáveis positivas $x_{31'}$ e $x_{33'}$, sendo o vértice $1' \in C_2$ chamado de *vértice cola*. Também é possível perceber a adição de três variáveis negativas que ligam o vértice $3'$ para todos os vizinhos do *vértice cola* ($N_3^+(1')$). Sabemos que $a_3^T x \leq 3$. Se apenas uma das novas variáveis positivas estiver em B, como $a_2^T \chi^B \leq 2$, neste caso $a_3^T \chi^B \leq 3$. Caso as duas novas variáveis positivas estejam em B e $a_2^T \chi^B = 2$, então $a_3^T \chi^B = 2 + 2 > 3$; entretanto, $\sum_{u \in N_2^+(1')} \chi_{1'u}^B \geq 1$, criando com as arestas de G_3, em B, possíveis P_4s. Logo, em B existirá pelo menos uma das variáveis negativas $x_{i3'}$ onde $i \in N_2^+(1')$ para que B continue um particionamento em bicliques. Sendo assim, $a_3^T \chi^B \leq 3$.

É possível perceber que $C_3 = C_2 \cup \{3\}$ e que para $a_3^T \chi^B = 3$ então $\sum_{u \in N_3^+(3)} \chi_{3u}^B \geq 1$.

Para $k = 4$ podemos observar pela ilustração da Figura 3.13(b) a adição de três variáveis positivas $x_{24'}$, $x_{34'}$ e $x_{44'}$, sendo $2, 3 \in C_3$ os *vértices cola*. Também é possível perceber a adição de três variáveis negativas que ligam o vértice 4 para todos os vizinhos dos *vértices cola* ($N_4^+(2) \cup N_4^+(3)$). Sabemos que $a_4^T x \leq 4$. Se apenas uma das novas variáveis positivas estiver em B, como $a_3^T \chi^B \leq 3$ teremos $a_4^T \chi^B \leq 4$. Caso as três novas

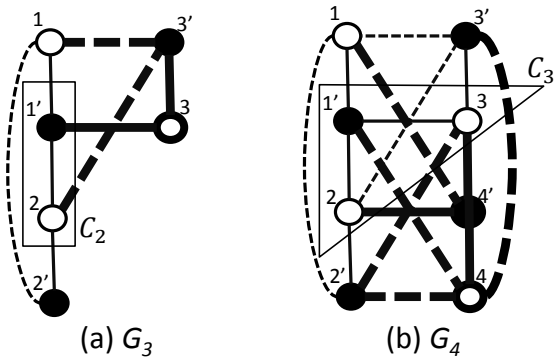


Figura 3.13: Formação dos grafos Grid G_3 e G_4.

variáveis positivas estejam em B e $a_3^T\chi^B = 3$, então, $a_4^T\chi^B = 3+3 > 4$; entretanto, $\sum_{u\in N_3^+(2)} \chi_{2u}^B \geq 1$ e $\sum_{u\in N_3^+(3)} \chi_{3u}^B \geq 1$, criando com as três arestas de G_4, em B, possíveis P_4s. Logo, em B existirão pelo menos duas variáveis negativas x_{4i}, onde $i \in N_3^+(2) \cup N_3^+(3)$, para que B continue um particionamento em bicliques. Sendo assim, $a_4^T\chi^B \leq 4$.

No caso de apenas duas novas variáveis positivas estarem em χ^B, o tratamento será análogo ao descrito anteriormente para G_3 ou G_4, dependendo apenas do número de *vértices cola* envolvidos.

É possível perceber que $C_4 = C_3 \cup \{4'\}$ e que, para $a_4^T\chi^B = 4$, temos $\sum_{u\in N_4^+(4')} \chi_{4'u}^B \geq 1$.

Agora suponha que G_{k-1} é viável e que para todo $v \in C_{k-1}$, quando $a_{k-1}^T\chi^B = k-1$ vale $\sum_{u\in N_{k-1}^+(v)} \chi_{vu}^B \geq 1$. Sendo assim, iremos provar que G_k também é, e para isto abordaremos os casos quando k for par ou ímpar.

Na expansão de G_k, quando k é ímpar, temos a adição de duas variáveis positivas $x_{kk-2'}$ e $x_{kk'}$, sendo $k-2' \in C_{k-1}$ o *vértice cola*. Também é possível perceber a adição de três variáveis negativas que ligam o vértice k' para todos vizinhos do *vértice cola* $(N_k^+(k-2'))$. Sabemos que $a_k^T x \leq k$. Se apenas uma das novas variáveis positivas estiver em B, como $a_{k-1}^T\chi^B \leq k-1$ teremos $a_k^T\chi^B \leq k$. Caso as duas novas variáveis positivas estejam em B e $a_{k-1}^T\chi^B = k-1$, vale $a_k^T\chi^B = k-1+2 > k$; entretanto, $\sum_{u\in N_{k-1}^+(k-2')} \chi_{vu}^B \geq 1$, criando com as duas arestas de G_k, em B, possíveis P_4s. Logo, em B existirá pelo menos uma das variáveis negativas $x_{ik'}$, onde $i \in N_{k-1}^+(k-2')$, para que B continue um particionamento em bicliques. Sendo assim, $a_k^T\chi^B \leq k$.

É possível perceber que $C_k = C_{k-1} \cup \{k\}$ e que, para $a_k^T\chi^B = k$, temos $\sum_{u\in N_k^+(k)} \chi_{ku}^B \geq 1$.

Na expansão de G_k, quando k é par, temos a adição de três variáveis positivas

$x_{k-2k'}$, $x_{k-1k'}$ e $x_{kk'}$, sendo $k-2$, $k-1 \in C_{k-1}$ os *vértices cola*. Também é possível perceber a adição de quatro variáveis negativas, três delas ligam o vértice k para os vizinhos dos *vértices cola* $(N^+_{k-1}(k-2) \cup N^+_{k-1}(k-1))$ e a variável $x_{k-4k-1'}$, onde $k-4 \in N^+_{k-1}(k-1)$. Cobrindo assim todos os vizinhos dos *vértices cola*. Sabemos que $a_k^T x \leq k$. Se apenas uma das novas variáveis positivas estiver em B, como $a_{k-1}^T \chi^B \leq k-1$ teremos $a_k^T \chi^B \leq k$. Caso as três novas variáveis positivas estejam em B e $a_{k-1}^T \chi^B = k-1$ vale $a_k^T \chi^B = k-1+3 > k$; entretanto, $\sum_{u \in N^+_{k-1}(k-2)} \chi^B_{k-2u} \geq 1$ e $\sum_{u \in N^+_{k-1}(k-1)} \chi^B_{k-1u} \geq 1$, criando com as três arestas, em B, possíveis P_4s. Logo, em B existirão pelo menos duas variáveis negativas x_{ki}, onde $i \in N^+_{k-1}(k-2) \cup N^+_{k-1}(k-1)$, para que B continue um particionamento em bicliques. Sendo assim, $a_k^T \chi^B \leq k$.

No caso de apenas duas novas variáveis positivas estarem em B, o tratamento será análogo ao descrito anteriormente para G_{k-1} ou G_k, dependendo apenas do número de *vértices cola* envolvidos.

É possível perceber que $C_k = C_{k-1} \cup \{k'\}$ e que, para $a_k^T \chi^B = k$, temos $\sum_{u \in N^+_k(k')} \chi^B_{k'u} \geq 1$.

Sendo assim, (3.6) é válida para $P_{n,m}$.

Agora provaremos que (3.6) define uma faceta de $P_{n,m}$.

Seja $a^T x \leq a_0$ a inequação (3.6), isto é, $a^T x = x(E^+[G_k]) - x(E^-[G_k]) \leq n = a_0$, e seja $b^T x \leq b_0$ uma faceta definida para $P_{n,m}$ tal que $F_a = \{x \in P_{n,m} | a^T x = a_0\} \subseteq F_b = \{x \in P_{n,m} | b^T x = b_0\}$. Claramente, $F_a \neq P_{n,m}$; assim se pudermos provar que $b = \alpha a$ para algum $\alpha \in \mathbb{R}$, uma vez que $F_a \neq \emptyset$ podemos concluir que (3.6) define uma faceta de $P_{n,m}$. Para isto, iniciamos estabelecendo o seguinte:

Lema 3.2.10. *Existe $\alpha \in \mathbb{R}$ tal que $b_e = \alpha$ para todo $e \in E^+[G_k]$.*

Demonstração. Para provar o Lema 3.2.10 definimos o conjunto M (ilustrado na Figura 3.14) como

$$M = \{(i, i') \in E^+[G_k]\}$$

sendo M um *k-emparelhamento* do grafo G_k. Além disso, considerando os particionamentos em bicliques:

$$P_i = M \cup \{(i+1), i'\} - \{i, i'\} \ ,$$

$$T_i = M \cup \{(i+2), i'\} - \{i, i'\} \text{ e } \bar{T}_i = M \cup \{(i-2), i'\} - \{i, i'\}.$$

Observando os subgrafos M, P_3 e $\bar{T}_3$ de G_k, ilustrados na Figura 3.14, é possível

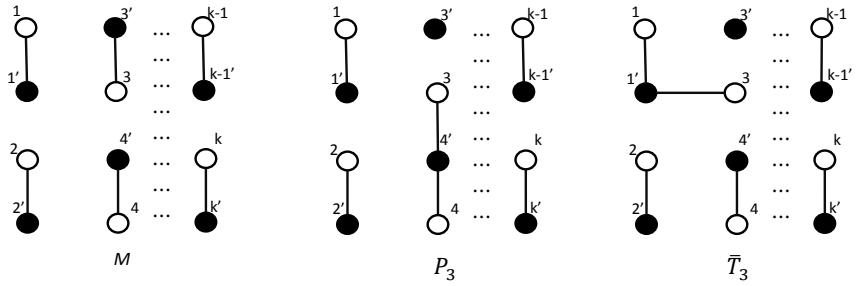


Figura 3.14: Subgrafos M, P_3 e $\bar{T}_3$ do Grafo Grid G_k.

perceber que $\chi^M, \chi^{P_i}, \chi^{T_i}, \chi^{\bar{T}_i} \in F_a \subseteq F_b$, e portanto $0 = b_0 - b_0 = \chi^M - \chi^{P_i} = b_{(i+1)i'} - b_{ii'}$, $0 = b_0 - b_0 = \chi^M - \chi^{T_i} = b_{(i+2)i'} - b_{ii'}$ e $0 = b_0 - b_0 = \chi^M - \chi^{\bar{T}_i} = b_{(i-2)i'} - b_{ii'}$. Logo $b_{(i+1)i'} = b_{ii'}$, $b_{(i+2)i'} = b_{ii'}$ e $b_{(i-2)i'} = b_{ii'}$. Aplicando este processo para todo $i \in \{1, 2, .., k\}$ e sabendo que G_k é um grafo conexo, temos que para toda aresta $e \in E^+[G_k]$ vale que $b_e = \alpha$.

□

Lema 3.2.11. *Seja K_{nm} a biclique de dimensão $n \times m$ relacionada ao $P_{n,m}$. Então para todo $e \in \{E[K_{nm}] \setminus (E^+[G_k] \cup E^-[G_k])\}$ vale que $b_e = 0$.*

Demonstração. Considere os particionamentos em bicliques:

$$W_{ij} = M \cup \{\{i, j'\}, \{(i+2), i'\}, \{(j-2), j'\}\} - \{\{(i+2)(i+2)'\}, \{j, j'\}\},$$

$$Y_{ij} = M \cup \{\{i, j'\}, \{(i+2), i'\}, \{j, (j-2)'\}\} - \{\{i, i'\}, \{j, j'\}\} \text{ e}$$

$$Z_{ij} = M \cup \{\{i, j'\}, \{j, i'\}\}.$$

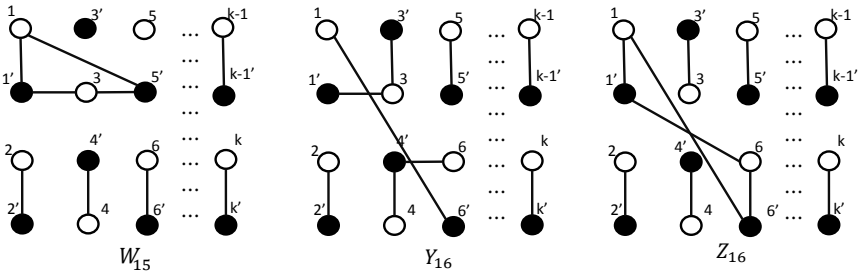


Figura 3.15: Subgrafos W_{15}, Y_{16} e Z_{16} do Grafo Grid G_k.

Observando o subgrafo W_{15} de G_k, ilustrado na Figura 3.15, é possível perceber que $W_{ij} \in F_a \subseteq F_b$, e portanto, $0 = b_0 - b_0 = b^T\chi^{W_{ij}} - b^T\chi^M = b_{ij'} + b_{(i+2)i'} + b_{(j-2)j'} - b_{jj'} -$

$b_{(j-2)j'}$. Como $b_{(i+2)i'} = b_{(j-2)j'} = b_{jj'} = b_{(j-2)j'}$, então $b_{ij'} = 0$ para todo $i \in \{1, 2.., k-4\}$ e $j = i + 4$.

Observando os subgrafos Y_{16} e Z_{16} de G_k, ilustrados na Figura 3.15, é possível perceber que Y_{ij}, $Y_{ij} - \{ij'\}$, $Z_{ij} \in F_a \subseteq F_b$, e portanto, $0 = b_0 - b_0 = b^T\chi^{Y_{ij}} - b^T\chi^{Y_{ij}-\{ij'\}} = b_{ij'}$, além disto, $0 = b_0 - b_0 = b^T\chi^{Z_{ij}} - b^T\chi^M = b_{ij'} = -b_{ji'}$. Como $b_{ij'} = 0$, segue que $b_{ij'} = b_{ji'} = 0$ para todo $i \in \{1, 2.., k-5\}$ e $j \in \{i + 5..k\}$.

Por fim, para todo $(i, j) \in \{E[K_{nm}] \setminus (E^+[G_k] \cup E^-[G_k])\}$, $i \notin V_1[G_k]$ ou $j \notin V_2[G_k]$, temos que $\chi^{\{M,(i,j)\}} \in F_a \subseteq F_b$, e portanto, $0 = b_0 - b_0 = b^T\chi^{\{M,(i,j)\}} - b^T\chi^M = b_{ij}$. Logo, para toda aresta $(i, j) \in \{E[K_{nm}] \setminus (E^+[G_k] \cup E^-[G_k])\}$ vale que $b_{ij} = 0$.

□

Lema 3.2.12. *Existe $\alpha \in \mathbb{R}$ tal que $b_e = -\alpha$ para todo $e \in E^-[G_k]$.*

Demonstração. Para provar o Lema 3.2.12, abordaremos inicialmente as arestas verticais e horizontais de $E^-[G_k]$, para isto, definimos os seguintes particionamentos em bicliques:

$$V_{ij} = M \cup \{\{i, j'\}, \{j, i'\}\} \text{ e}$$

$$H_{ij} = M \cup \{\{i, j'\}, \{j, i'\}\}.$$

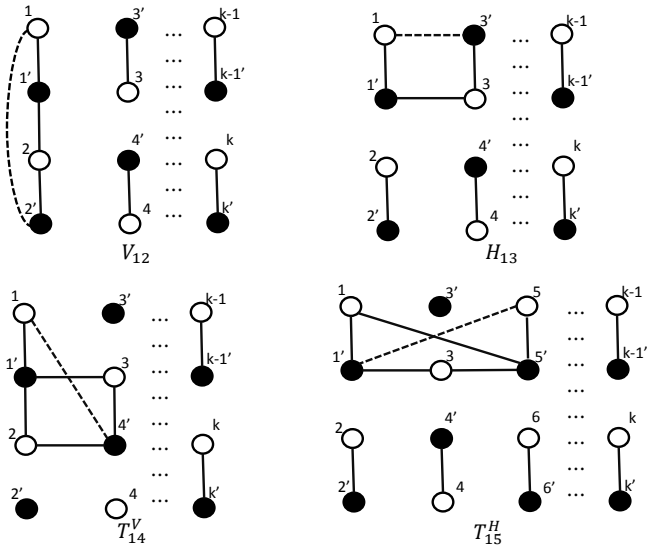


Figura 3.16: Subgrafos V_{12}, H_{13}, T_{14}^V e T_{15}^H de G_k.

Observando os subgrafos V_{12}, H_{13} de G_k, ilustrados na Figura 3.16, é possível perceber que V_{ij}, $H_{ij} \in F_a \subseteq F_b$, e portanto, $0 = b_0 - b_0 = b^T\chi^{V_{ij}} - b^T\chi^M = b_{ji'} + b_{ij'}$.

Logo, $b_{ij'} = -b_{ji'}$, e para todo $i \in \{1, 3, .., n-1\}$ e $j = i+1$, temos que para as arestas verticais de $E^-[G_k] b_{ij'} = -\alpha$. Como $0 = b_0 - b_0 = b^T\chi^{H_{ij}} - b^T\chi^M = b_{ij'} + b_{ji'}$ então $b_{ij'} = -b_{ji'}$. Logo, para todo $i \in \{1, 2, .., k-1\}$ e $j = i+2$, temos que para todas as arestas horizontais de $E^-[G_k]$ vale que $b_{ij'} = -\alpha$.

Abordando as arestas transversais de $E^-[G_k]$, definimos os seguintes particionamentos em bicliques:

$$T^V_{ij} = M \cup \{\{i, j'\}, \{(j-1), i'\}, \{(i+1), j'\}, \{(i+1), i'\}, \{(j-1), j'\}\}$$

$$-\{\{j, j'\}, \{(j-1), (j-1)'\}, \{(i+1), (i+1)'\}\} \text{ e}$$

$$\bar{T}^V_{ij} = M \cup \{\{j, i'\}, \{i, (j+1)'\}, \{(j+1), i'\}, \{j, (j+1)'\}\} - \{j, j'\} \text{ e}$$

$$T^H_{ij} = M \cup \{\{i, j'\}, \{j, i'\}, \{(i+2), i'\}, \{(j-2), j'\}\} - \{(i+2), (i+2)'\}.$$

Observando o subgrafo T^V_{14} de G_k, ilustrado na Figura 3.16, é possível perceber que T^V_{ij}, $\bar{T}^V_{ij} \in F_a \subseteq F_b$, e portanto, $0 = b_0 - b_0 = b^T\chi^{T^V_{ij}} - b^T\chi^M = b_{(j-1),i'} + b_{(i+1),i'} + b_{(i+1),j'} + b_{(j-1),j'} + b_{i,j'} - b_{j,j'} - b_{(i+1),(i+1)'} - b_{(j-1),(j-1)'}$. Como $b_{(i+1),i'} = b_{(i+1),j'} = b_{(j-1),j'} = b_{j,j'} = b_{(i+1),(i+1)'} = b_{(j-1),(j-1)'}$, e $b_{(j-1)i'} = \alpha$, então $b_{i,j'} = -\alpha$. Aplicando este processo para todo $i \in \{1, 3.., k-1\}$ e $j = i+3$, temos que para toadas as arestas transversais do primeiro grupo $(i, j) \in E^-[G_k]$ vale que $b_{ij} = -\alpha$. Aplicando o mesmo processo sobre os particionamentos em bicliques $\bar{T}^V_{ij}$, temos que $b_{ji'} = -\alpha$, e para todo $i \in \{2, 4.., k-1\}$ e $j = i+1$, temos que para todas as arestas transversais do segundo grupo $(i, j) \in E^-[G_k]$ vale que $b_{ij} = -\alpha$.

Finalmente, observando o subgrafo T^H_{15} de G_k, ilustrado na Figura 3.16, é possível perceber que $T^H_{ij} \in F_a \subseteq F_b$, e portanto, $0 = b_0 - b_0 = b^T\chi^{T^H_{ij}} - b^T\chi^M = b_{(i+2)i'} + b_{(i+2)i'} + b_{(j-2)j'} + b_{ij'} + b_{ji'} - b_{(i+2)(i+2)'}$. Como $b_{(i+2)i'} = b_{(i+2)(i+2)'}$, $b_{i,j'} = 0$ e $b_{(j-2)j'} = \alpha$, então $b_{j,i'} = -\alpha$. Aplicando este processo para todo $i \in \{1, 2.., k-4\}$ e $j = i+4$, temos que para todas as arestas transversais do terceiro grupo $(i, j) \in E^-[G_k]$ vale que $b_{ij} = -\alpha$.

Assim, para todo $e \in E^-[G_k]$ vale que $b_e = -\alpha$.

□

Os três lemas implicam que a inequação (3.6) define uma faceta de $P_{n,m}$.

□

3.3 Algoritmos de separação das Inequações Fole (B_k), Escada (L_k) e Grid (G_k)

Quando um MPI possui um conjunto de inequações de dimensão exponencial, sua resolução através de programação inteira clássica (*Branch-and-Bound*) se torna inviável. Uma possibilidade de contornar este problema é iniciar a resolução do modelo original restrito, ou seja, com menos inequações que o modelo completo, e adicionar apenas as inequações que forem identificadas como violadas. Como as inequações propostas neste capítulo para o MPI $\mathcal{F}_{P_4}$ têm crescimento exponencial, pretende-se criar algoritmos para identificação da violação destas inequações com o objetivo de aplicá-las em técnicas de *Branch-and-Cut*.

Nesta seção apresentamos um modelo geral para a separação das três novas famílias de inequações e uma heurística de separação para cada família embutida em um algoritmo de *Branch-and-Cut* definido. Ao final realizamos uma análise da eficiência das heurísticas de separação com o modelo exato.

3.3.1 MPI para separação das Inequações L_k, B_k e G_k para $\mathcal{F}_{P_4}$

Violar uma inequação L_k significa que ao resolver o modelo restrito, sem todas estas inequações e as inequações de integralidade das variáveis (*Branch-and-Bound*), encontramos em $\bar{x}_{ij}$, que representa o valor da solução do modelo restrito, uma inequação cujo valor de suas variáveis associadas seja

$$x(E^+[L_k]) - x(E^-[L_k]) > k.$$

Logo, a restrição relacionada foi violada e deve ser adicionada. Esta regra também é válida para as inequações da família B_k e G_K. Modelamos o problema de identificar a máxima violação das inequações L_k, B_k e G_k como descrito a seguir.

Seja $P_k = \{1, 2, .., 2k\}$ o conjunto de posições que os vértices nos grafos suporte L_k, B_k e G_k podem assumir. Seja também P_k^+ o conjunto de pares $< p_1, p_2 >$, onde $p_1, p_2 \in P_k$, que indicam o posicionamento do par de vértices associados às variáveis positivas das inequações, e analogamente, P_k^- o conjunto de pares $< p_1, p_2 >$ associados às variáveis negativas das inequações. Assim, podemos representar a configuração das variáveis de qualquer um dos grafos suporte pelos conjuntos P_k^+ e P_k^-, como na Figura 3.17(a), que apresenta a representação de um grafo L_4, e na Figura 3.17(b) que representa

um grafo G_4.

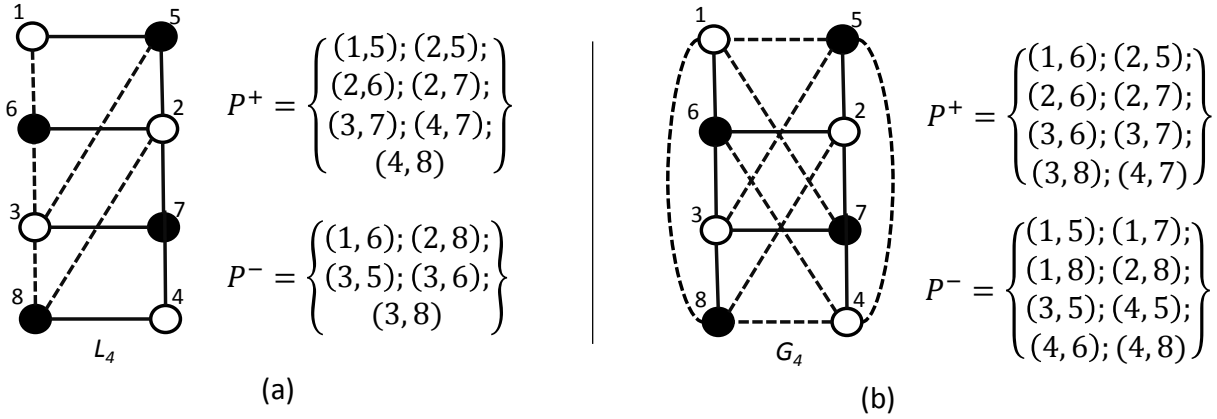


Figura 3.17: Representação do posicionamento dos vértices nos grafos suporte das inequações. (a)Representação de um grafo L_4 (b)Representação de um grafo G_4

Por fim, seja w_{ij} o valor das variáveis $\bar{x}_{ij}$ após a resolução do modelo restrito $\mathcal{F}_{P_4}$. E x_{vp} e $y_{ijp_1p_2}$, as variáveis de decisão, onde $x_{vp} = 1$ indica que o vértice $v \in V_1 \cup V_2$ está atribuído à posição $p \in P_k$ do grafo suporte, sendo $x_{vp} = 0$ em caso contrário. Já $y_{ijp_1p_2} = 1$ indica que o par de vértices $i, j \in V_1 \cup V_2$ foi atribuído ao par de posições $< p_1, p_2 > \in P_k^+ \cup P_k^-$, e $y_{ijp_1p_2} = 0$ em caso contrário. Logo, temos a seguinte formulação para identificar a máxima violação das inequações suporte.

$$(\mathcal{F}_{BLG}) \max \sum_{i \in V_1} \sum_{j \in V_2} \sum_{<p_1,p_2> \in P_k^+} w_{ij} y_{ijp_1p_2} - \sum_{i \in V_1} \sum_{j \in V_2} \sum_{<p_1,p_2> \in P_k^-} w_{ij} y_{ijp_1p_2} \tag{3.7}$$

sujeito a

$$y_{ijp_1p_2} \geq x_{ip_1} + x_{jp_2} - 1, \qquad \forall i \in V_1, \forall j \in V_2 \text{ e } \forall < p_1, p_2 > \in P_k^+ \cup P_k^- \tag{3.8}$$

$$y_{ijp_1p_2} \leq x_{ip_1}, \qquad \forall i \in V_1, \forall j \in V_2 \text{ e } \forall < p_1, p_2 > \in P_k^+ \cup P_k^- \tag{3.9}$$

$$y_{ijp_1p_2} \leq x_{jp_2}, \qquad \forall i \in V_1, \forall j \in V_2 \text{ e } \forall < p_1, p_2 > \in P_k^+ \cup P_k^- \tag{3.10}$$

$$\sum_{v \in V_1 \cup V_2} x_{vp} = 1, \qquad \forall p \in P_k \tag{3.11}$$

$$\sum_{p \in P} x_{vp} \leq 1, \qquad \forall v \in V_1 \cup V_2 \tag{3.12}$$

$$x_{ip}, y_{ijp_1p_2} \in \{0,1\} \qquad \forall i \in V_1, \forall j \in V_2, p \in P_k \text{ e } \forall < p_1, p_2 > \in P_k^+ \cup P_k^-. \tag{3.13}$$

A função objetivo (3.7) procura maximizar a violação da inequação mapeada em P_k^+ e P_k^-. As restrições (3.8-3.10) garante que o par de vértices $i \in V_1$ e $j \in V_2$ sejam alocados ao par de posições $< p_1, p_2 >$, ou seja, $y_{ijp_1p_2} = 1$ apenas se $x_{ip_1} = 1$ e $x_{ip_2} = 1$,

contabilizando assim o valor de w_{ij} na função objetivo. Já a restrição (3.11) garante que toda posição $p \in P_k$ é atribuído a um e somente um vértice $v \in V_1 \cup V_2$. Por fim, o conjunto de restrições (3.12) permite que cada vértice v esteja alocado a no máximo uma posição p.

3.3.2 *Branch-and-Cut* e heurística de separação das Inequações L_k, B_k e G_k para $\mathcal{F}_{P_4}$

Nesta seção apresentamos um algoritmo de *Branch-and-Cut* para $\mathcal{F}_{P_4}$, ilustrado na Figura 3.18, e propomos heurísticas de separação das novas inequações definidas. Como já foi dito existe um número exponencial de inequações para os grafos suporte L_k, B_k e G_k. Logo, não devemos gerá-los todos de uma só vez e sim seguindo uma estratégia de plano de cortes, adicionando as inequações apenas se forem violadas na solução fracionada corrente.

Iniciamos a otimização do modelo $\mathcal{F}_{P_4}$ com o conjunto vazio de restrições. Seja $\bar{x}$ o vetor correspondente a uma solução intermediária da relaxação linear. Criamos um processo iterativo onde inicialmente checamos se $\bar{x}$ viola qualquer inequação quadrada através da estratégia de programação dinâmica definida em [Pinheiro *et al.*, 2016]. Esta estratégia cria um conjunto Q de grafos suporte para as inequações quadradas violadas que foram identificadas. Sendo estes grafos base para a criação dos outros grafos suporte L_k, B_k e G_k que serviram para identificar a violação das inequações Escada, Fole e Grid, respectivamente. Se o conjunto Q não for vazio, criamos heuristicamente os conjuntos L, B e G com grafos suporte das inequações violadas, como descrito mais adiante no texto. Em seguida, adicionamos ao modelo todas as inequações correspondentes aos grafos suporte contidos em Q, L, B e G, resolvemos o modelo e iteramos. Caso o conjunto Q esteja vazio checamos se $\bar{x}$ é inteiro. Sendo verdade, $\bar{x}$ é uma solução ótima. Se $\bar{x}$ possui valores fracionados realizamos *branch* e voltamos a iterar no algoritmo.

Heurística de separação das inequações L_k, B_k e G_k

Como os grafos suporte das inequações L_k, B_k e G_k são recursivos, propomos heurísticas de separação destas inequações utilizando os grafos do nível $k-1$ para encontrar as inequações violadas no nível k. Como Q é o conjunto de inequações quadradas, que são inequações base ($k = 2$) das três novas famílias, logo, Q servirá para a criação das inequações do nível $k = 3$, e este novo conjunto de inequações servirá para a construção das inequações do próximo nível e assim por diante.

Cada família de inequações possui suas peculiaridades na construção do conjunto

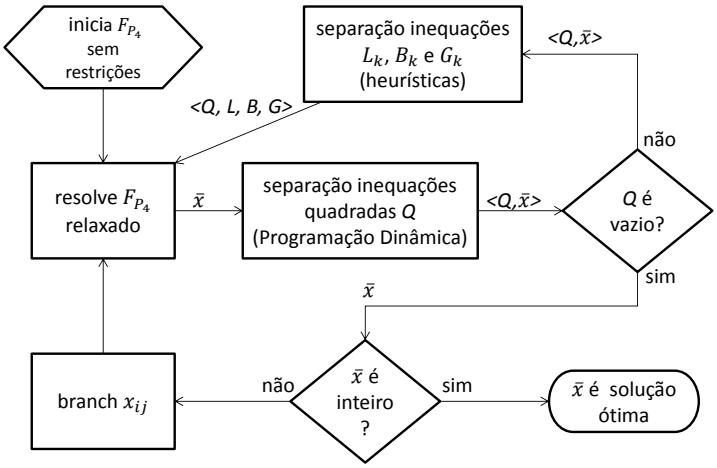


Figura 3.18: Fluxograma do algoritmo de *Branch-and-cut* para o modelo $\mathcal{F}_{P_4}$.

de grafos suporte do próximo nível; estas regras estão detalhadas na seção 3.2. Logo, a heurística de separação proposta é genérica para as três familias de inequações, mudando apenas a forma como avalia se a nova inequação está violada ou não. Por isto, descreveremos de forma geral esta heurística; para isto, iremos nos reportar de forma genérica a uma inequação qualquer das três famílias no nível de recorrência k como I_k. Considere ainda I^k o conjunto de todas as I_k inequações violadas, e por fim considere I o conjunto de todas as I_k violadas em todos os níveis k.

Como Q é o conjunto de inequações quadradas violadas que foram identificadas pelo processo de programação dinâmica ([Pinheiro *et al.*, 2016]), e sendo estas inequações base ($k = 2$) para as outras famílias, a heurística de separação recebe Q como dado de entrada e inicializa $I^2 = Q$. Para $k = 3$ até $k = min\{|V_1|, |V_2|\}$ fazemos uma inspeção em I^{k-1} , onde para cada $I_{k-1} \in I^{k-1}$ procuramos um par de arestas $\{i, j\} \notin I_{k-1}$ cujo grafo suporte $I_k = I_{k-1} \cup \{i, j\}$ tenha sua inequação associada violada. Lembrando que esta composição de um grafo suporte I_{k-1} em outro grafo suporte I_k é específico de cada família e detalhado na seção 3.2. Identificada a nova inequação violada I_k fazemos $I^k = I^k \cup I_k$, e uma vez finalizada a inspeção de I_{k-1}, fazemos $I = I \cup I^k$ e incrementamos o valor de k. Atingida a condição de parada no valor de k, retornamos I como saída da heurística.

Como dito anteriormente esta heurística acima definida é genérica para cada família de inequações. Logo, existem três implementações distintas com os mesmos comandos gerais, havendo apenas as devidas modificações na hora de compor os grafos suporte e avaliar a violação de cada família. Sendo assim, a implementação da heurística de separação para a família Fole, por exemplo, ao final irá retornar o conjunto B de inequações Fole violadas. Uma possível parametrização desta heurística está na definição do valor

final de k, podendo assumir qualquer valor entre 3 e $min\{|V_1|, |V_2|\}$, pois como os grafos suporte são de dimensão $k \times k$, o valor máximo assumido por k deve ser o tamanho da menor partição do bigrafo de entrada do problema.

3.3.3 Comparação entre os algoritmos de separação das Inequações Fole (B_k), Escada (L_k) e Grid (G_k)

Para avaliar a eficiência da heurística de separação das novas inequações, proposta na seção anterior, implementamos a técnica de Plano de Cortes para o modelo restrito $\mathcal{F}_{P_4}$ até a resolução do nó raiz, ou seja, sem a adição de inequações de integralidade das variáveis (*Branch-and-Bound*). A comparação da heurística se deu com o MPI $\mathcal{F}_{B_n}$ de separação das inequações. Esta implementação utilizou o resolvedor matemático CPLEX, habilitado para fornecer um "*pool* "das melhores soluções encontradas. Sempre que uma solução encontrada tanto na heurística quanto no MPI tiver valor acima de n, sua inequação associada estará violada, sendo assim, adicionada ao modelo restrito.

Tabela 3.1: Comparação entre o MPI $\mathcal{F}_{BLG}$ e a heurística de separação das inequações Fole (B_k), Escada (L_k) e Grid (G_k), instâncias do grupo BC_1.

Instância			**MPI**			**HEU**			
n	m	K	LB	cortes	*tempo* (s)	LB	$\Delta(\%)$	cortes	*tempo* (s)
5	7	3	7	81	18998	7	0	100	0,01
		4	7	70	31758	7	0	170	0,01
		5	7	121	62573	7	0	213	0,01
6	8	3	8	78	18837	8	0	169	0,02
		4	8	90	41662	8	0	213	0,02
		5	8	123	191050	8	0	394	0,02
		6	8	153	835107	8	0	331	0,02
7	11	3	11	90	41643	11	0	208	0,02
		4	11	118	164151	11	0	364	0,02
		5	11	161	895142	11	0	416	0,02
		6	11	189	3600601	11	0	445	0,02
		7	11	221	21093056	11	0	506	0,02
6	12	3	14	161	61704	14	0	330	0,3
		4	14	145	134837	14	0	533	0,03
		5	14	231	744012	14	0	419	0,03
		6	14	281	3118439	14	0	549	0,03
6	20	3	24,00419	413	612880	24,001723	**0,01**	934	0,13
		4	26,337028	1122	24687442	26,071833	**1,0**	1982	0,6
		5	LT	LT	LT	26,342728	-	2159	0,6
		6	LT	LT	LT	28	-	2890	0,5

As Tabelas 3.1 e 3.2 apresentam, para cada linha, os resultados encontrados na resolução do modelo restrito $\mathcal{F}_{P_4}$ sobre cada instância analisada, sendo testadas duas abordagens de resolução. Suas colunas são divididas em três grupos: o primeiro grupo descreve as instâncias utilizadas (*Instância*), que são detalhadas na seção 6.1.1. Estas colunas descrevem as dimensões $n = |V_1|$, $m = |V_2|$ do problema, e a coluna K representa o parâmetro dos algoritmos de separação, indicando até que nível foram inspecionadas as inequações recorrentes durante o processo de identificação de suas violações. O segundo grupo de colunas apresenta os resultados do Plano de Cortes que utiliza o MPI $\mathcal{F}_{BLG}$ para separação das inequações violadas (*MIP*), e o terceiro grupo apresenta os resultados do Plano de Cortes que utiliza a heurística de separação das inequações violadas (*HEU*). Estes dois últimos grupos são divididos nas colunas: *LB* que apresenta o melhor limite inferior encontrado; *cortes* indicando o número de inequações violadas que foram adicionadas ao modelo restrito pelo algoritmo de separação; e a coluna *tempo* que representa o tempo computacional, em segundos, da respectiva abordagem de Plano de Cortes. O conjunto *HEU* ainda possui uma coluna intitulada $\Delta(\%)$ que representa a diferença percentual entre os limites inferiores das duas abordagens. Seu valor foi calculado pela equação $\Delta = (LB_{MIP} - LB_{HEU})/LB_{MIP} * 100$. Colocamos um limite de tempo de dois dias de execução para cada instância e algumas execuções falharam por este motivo, sendo estes casos sinalizados com a marca LT.

Como todas as instâncias do grupos BC_2, abordados na Tabela 3.2, falharam no tempo limite com $K = 4$, esta tabela apresenta apenas os valores das instâncias para $K = 3$, sendo trocada a coluna K pela coluna p, que representa a densidade do grafo em relação ao número de arestas. A Tabela 3.1 abordou instâncias do grupo BC_1 apenas com valores de $p = 0,5$.

Tabela 3.2: Comparação entre o MPI $\mathcal{F}_{BLG}$ e a heurística de separação das inequações Fole (B_k), Escada (L_k) e Grid (G_k), instâncias do grupo BC_2.

Instância			**MPI**			**HEU**			
n	m	p	LB	cortes	*tempo* (s)	LB	$\Delta(\%)$	cortes	*tempo* (s)
10	16	0,5	32,016023	406	3068,286	32,007828	**0,025**	1196	0,24
10	16	0,6	38,408982	610	6438,998	38,401938	**0,02**	1552	0,36
12	17	0,5	42,802948	601	8840,037	42,80174	**0,003**	1776	0,49
12	17	0,6	46,807665	583	10304,1	46,801768	**0,01**	1921	0,5
12	20	0,5	50,00788	636	15958,473	50,002643	**0,01**	2350	0,69
12	20	0,6	56,000672	765	32847,864	56,000672	0	2435	0,9
16	20	0,5	65,600888	776	76314,84	65,600888	0	3193	1,367
16	20	0,6	LT	LT	LT	75,200766	-	3890	2,182

Em todas as instâncias analisadas, a heurística de separação apresentou uma boa eficiência comparada ao MPI $\mathcal{F}_{BLG}$. Das 25 execuções que o MPI obteve sucesso, a heurística de separação obteve o mesmo valor de LB para 18 instâncias, obtendo a maior diferença na instância $(n = 6; m = 20; K = 4)$ com $\Delta = 1\%$. Mesmo a heurística adicionando muita mais cortes que o MPI, o efeito no tempo computacional na resolução da raiz do modelo restrito é aceitável, não passando de 2,2 segundos em todas as instâncias analisadas.

Capítulo 4

Branch-and-price para o Problema de Edição de Biclusters

Neste capítulo propomos uma nova reformulação para o Problema de Edição de *Biclusters*. Este novo modelo possui o número de variáveis exponenciais, logo, o abordaremos com técnicas de Geração de Colunas a serem aplicadas ao *Branch-and-price* proposto. Ainda neste capítulo propomos um MIP e uma heurística para o subproblema de *pricing*.

4.1 Problema Mestre para o PEB

Seja $\mathcal{B} = \{B_1, B_2, ..., B_l\}$ o conjunto de todos os *biclusters* viáveis, onde um *bicluster* $B_k \in \mathcal{B}$ consiste de um subconjunto de vértices em $V_1 \cup V_2$. Uma variável binária λ_k é usada para decidir se o *bicluster* $B_k \in \mathcal{B}$ é parte da solução. O coeficiente a_{vk} assume valor 1 se o vértice $v \in V_1 \cup V_2$ estiver no *bicluster* $B_k \in \mathcal{B}$ e 0 caso contrário. O custo c_k do *bicluster* B_k é definido como:

$$c_k = \left|(ijk)^-\right| + \frac{1}{2}\left|(ijk)^+\right|,$$

onde

- $(ijk)^- = \{\bar{E} | i \in B_k \text{ e } j \in B_k\}$ e
- $(ijk)^+ = \{E | (i \in B_k \text{ e } j \notin B_k) \text{ ou } (i \notin B_k \text{ e } j \in B_k)\}$.

Sendo assim, c_k computa as adições ($(ijk)^-$) realizadas dentro do *bicluster* e metade do custo das remoções realizadas entre o *bicluster* B_k e os outros que se ligavam através de arestas ($(ijk)^+$).

A seguinte formulação tem como objetivo selecionar um conjunto de *biclusters* viáveis que minimize a soma de seus custos.

$$(F_\lambda) \min \sum_{k=1}^{|\mathcal{B}|} c_k \lambda_k \tag{4.1}$$

$$\text{sujeito a} \quad \sum_{k=1}^{|\mathcal{B}|} a_{vk} \lambda_k = 1 \qquad \forall v \in V_1 \cup V_2 \tag{4.2}$$

$$\lambda_k \in \{0,1\} \qquad \forall k \in \{1,2,...,|\mathcal{B}|\}$$

A função objetivo (4.1) minimiza a soma dos custos dos *biclusters* na solução. As restrições (4.2) garantem que todos os vértices $v \in V_1 \cup V_2$ sejam cobertos pelo conjunto de *biclusters* que fazem parte da solução.

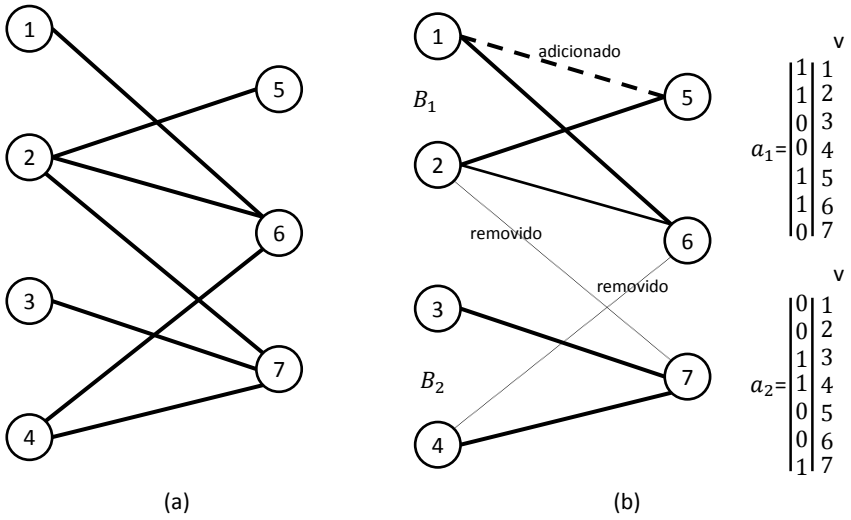


Figura 4.1: (a) Bigrafo de entrada. (b) Solução s contendo os *biclusters* B_1 e B_2.

A Figura 4.1(b) apresenta a solução s para o BEP com dois *biclusters* B_1 e B_2. É possível perceber que: $|(ijk_1)^+| = 2$; $|(ijk_1)^-| = 1$; $|(ijk_2)^+| = 2$; e $|(ijk_2)^-| = 0$. São apresentadas também as colunas a_1 e a_2 associadas a estes *biclusters*, donde é fácil perceber que ao fazer $\lambda_1 = \lambda_2 = 1$ teremos coberto todos os vértices do grafo de entrada obtendo uma solução viável. Sendo assim, o custo $z_{\mathcal{F}_\lambda}$ da solução s será dado por:

$$z_{\mathcal{F}_\lambda} = c_1\lambda_1 + c_2\lambda_2 = (1 + \frac{1}{2}2) + (0 + \frac{1}{2}2) = 3.$$

As colunas na formulação $\mathcal{F}_\lambda$ são associadas ao conjunto de todos os *biclusters* viáveis. Com o crescimento das instâncias, o número de colunas no modelo se torna muito

grande, ficando impossível resolvê-lo, já que depende diretamente da enumeração de todos os *biclusters* viáveis. Para tratar este problema, propomos resolvê-lo por técnicas de *Branch-and-Price*, onde a relaxação linear de $(\mathcal{F}_\lambda)$ é resolvida em cada nó da árvore de *Branch-and-Bound* por Geração de Colunas (GC). No algoritmo de GC, uma versão restrita de $\mathcal{F}_\lambda$ com poucas colunas, chamada de Problema Mestre Restrito (PMR), é inicialmente considerada e novas colunas são adicionadas através da resolução do subproblema de *pricing* até que novas colunas não sejam mais necessárias. A seguir apresentaremos o subproblema de *pricing*. Os algoritmos de GC serão detalhados na Seção 4.3.

4.2 Subproblema de *Pricing*

Podemos reescrever o modelo (F_λ) na forma padrão e particionar a matriz $A = (B\ N)$, onde B é uma matriz quadrada $m \times m$ e inversível. Desta forma (F_λ) poderá ser escrito:

$$(F_\lambda) \min z = c_B\lambda_B + c_N\lambda_N \tag{4.3}$$

$$\text{sujeito a} \quad B\lambda_B + N\lambda_N = b \tag{4.4}$$
$$\lambda \in \{0,1\}$$

Isolando $\lambda_B = B^{-1}b - B^{-1}N\lambda_N$ de (4.4) e substituindo em (4.3), temos:

$$\min z = c_B B^{-1}b + (c_N - C_B B^{-1}N)\lambda_N \tag{4.5}$$

Fazendo $\lambda_N = 0$ temos a solução $\lambda_B = B^{-1}b$. Tal solução é dita básica e será viável caso $B^{-1}b \geq 0$, assim, $\bar{z} = c_B B^{-1}b$ é o valor desta solução básica e sendo um problema de minimização, o custo reduzido $c_N - c_B B^{-1}N > 0$ indica que a solução $\bar{z}$ é ótima. Logo o objetivo do subproblema de *pricing* é encontrar uma coluna (representada por um *bicluster*) contida na matriz N de $\mathcal{F}_\lambda$ com custo reduzido mais negativo, procurando assim que a adição desta coluna ao PMR melhore sua solução corrente. Seja $\pi = c_B B^{-1}$ o vetor de valores das variáveis duais associadas às restrições (4.2), e seja a_k a k-ésima coluna contendo todos os vértices que fazem parte do *bicluster* B_k. Podemos definir o custo reduzido Cr_k da k-ésima coluna como

$$Cr_k = c_k - \sum_{v \in V_1 \cup V_2} \pi_v \; a_{vk}, \tag{4.6}$$

onde c_k é o custo associado à k-ésima coluna como definido na Seção 4.1. Note que a equação (4.6) pode ser reescrita como:

$$Cr_k = \sum_{i,j \in B_k} \bar{\mu}_{ij} + \sum_{j \in B_k} \bar{v}_j, \tag{4.7}$$

onde:

$$\bar{v}_j = \frac{1}{2} |N(j)| - \pi_j$$

$$\bar{\mu}_{ij} = \begin{cases} 1, \text{ se } \{i,j\} \notin E \\ -1, \text{ se } \{i,j\} \in E \end{cases}$$

Neste caso, $\bar{v}_j$ e $\bar{\mu}_{ij}$ são os custos reduzidos associados com o vértice j e a aresta $\{i,j\}$, respectivamente, lembrando que $|N(j)|$ denota o grau do vértice j. Para verificar que as equações (4.6) e (4.7) são equivalentes podemos reescrever o valor de c_k como se segue:

$$c_k = \sum_{i,j \in B_k} \bar{\mu}_{ij} + \sum_{j \in B_k} \frac{1}{2} |N(j)| \,.$$

Logo, iremos modelar o subproblema de *pricing* para construir uma coluna a_k avaliada pelo custo reduzido Cr_k.

Seja x'_v uma variável binária que toma valor 1 se o vértice $v \in V_1 \cup V_2$ faz parte da coluna (está no *bicluster*) ou valor 0, caso contrário. Sendo assim, o modelo do subproblema de *pricing* ($\mathcal{SP}$) é

$$(\mathcal{SP}) \min \sum_{i \in V_1 \cup V_2} \bar{v}_i x'_i + \sum_{i \in V_1, j \in V_2} \bar{\mu}_{ij} x'_i x'_j \tag{4.8}$$

$$\text{sujeito a} \quad x'_v \in \{0,1\} \qquad \forall v \in V_1 \cup V_2. \tag{4.9}$$

A solução de $\mathcal{SP}$ é um *bicluster* que representa uma coluna viável para o PMR com custo reduzido dado pela função objetivo. Qualquer combinação de vértices é solução viável para o subproblema, mesmo o *bicluster* vazio. Entretanto esta solução terá custo reduzido igual a 0 e será descartada pelo algoritmo de GC; por isto, as restrições (4.9) garantem apenas a integralidade da variável x'.

A Figura 4.2(a) apresenta um bigrafo de entrada para o subproblema de *pricing*. Já a Figura 4.2(b) ilustra uma solução s para o subproblema, representando uma coluna a_k para o PMR contendo os vértices $\{1; 2; 5; 6\}$. Na Figura 4.2, as linhas contínuas representam arestas do grafo de entrada, já as linhas pontilhadas representam pares de vértices que não se relacionam no grafo de entrada. O custo z_{SP} da solução s, obtida pela função objetivo (4.8), é dada por

$$\begin{aligned} z_{SP} &= \underbrace{((0,5-\pi_1)+(1,5-\pi_2)+(0,5-\pi_5)+(1,5-\pi_6))}_{\sum_{v\in V_1\cup V_2}\bar{v}_j x'_v} + \underbrace{(-1-1-1+1)}_{\sum_{i\in V_1, j\in V_2}\bar{\mu}_{ij}x'_i x'_j} \\ &= \underbrace{2}_{c_k} - \underbrace{(\pi_1+\pi_2+\pi_3+\pi_4)}_{\pi\cdot a_k}. \end{aligned}$$

Desta forma, observa-se que o custo reduzido dado por (4.8) é equivalente ao encontrado usando a expressão (4.6).

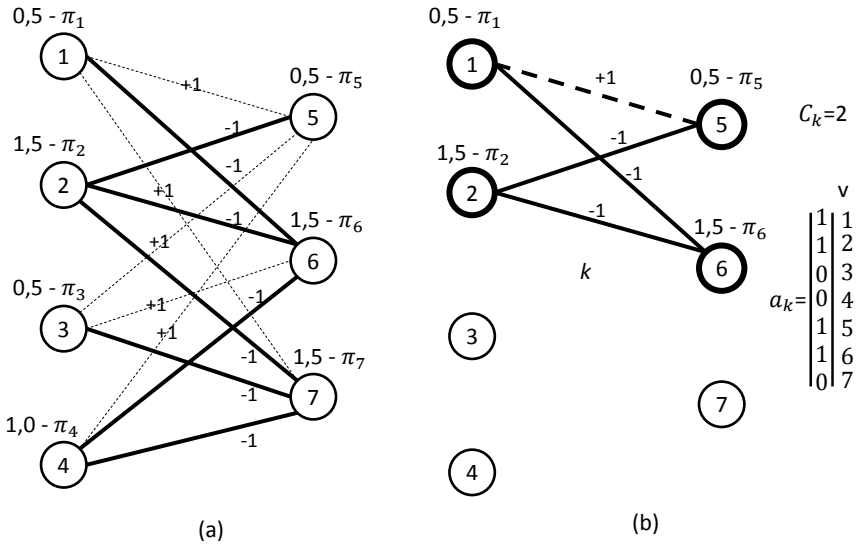


Figura 4.2: (a) Bigrafo G_{SP} de entrada para $\mathcal{SP}$. (b) Solução para $\mathcal{SP}$ e coluna viável a_k para o PMR.

Como o modelo $\mathcal{SP}$ é não linear, propomos duas linearizações. A primeira linearização é realizada por meio da adição do conjunto de variáveis binárias y'_{ij}. Para cada

par de vértices $i \in V_1, j \in V_2$, a variável y'_{ij} assume o valor 1 se a aresta $\{i, j\}$ estiver na coluna gerada, ou seja, ambos i e j estão na solução, caso contrário $y'_{ij} = 0$. A formulação $(\mathcal{SP}_{l1})$ pode ser expressa da seguinte forma:

$$(\mathcal{SP}_{l1}) \min \sum_{i \in V_1 \cup V_2} \bar{v}_i x'_i + \sum_{i \in V_1, j \in V_2} \bar{\mu}_{ij} y'_{ij} \tag{4.10}$$

$$\text{sujeito a} \quad y'_{ij} \leq x'_i \qquad \forall i \in V_1, j \in V_2 \tag{4.11}$$

$$y'_{ij} \leq x'_j \qquad \forall i \in V_1, j \in V_2 \tag{4.12}$$

$$y'_{ij} \geq x'_i + x'_j - 1 \qquad \forall i \in V_1, j \in V_2 \tag{4.13}$$

$$x'_i, y'_{ij} \in \{0, 1\} \qquad \forall i \in V_1, j \in V_2 \tag{4.14}$$

As restrições adicionais (4.11)–(4.13) garantem que a aresta $\{i, j\}$ terá seu custo computado na solução apenas se ambos os vértices i e j também estiverem.

A segunda linearização $\mathcal{SP}_{l2}$ é baseada na formulação para o problema da mochila quadrática proposta em [Billionnet & Éric Soutif, 2004]. A ideia desta formulação é definir variáveis continuas z_i, $\forall i \in V_1$, tal que $z_i = \sum_{j \in V_2} \bar{\mu}_{ij} x'_i x'_j$. Em outras palavras, para cada vértice i, z_i é igual a soma dos custo reduzidos associados as arestas $\{i, j\}$ que estão na solução. A formulação $\mathcal{SP}_{l2}$ pode ser escrita como se segue:

$$(\mathcal{SP}_{l2}) \min \sum_{i \in V_1} z_i + \sum_{i \in V_1 \cup V_2} \bar{v}_i x'_i \tag{4.15}$$

$$\text{sujeito a} \quad z_i \geq W_i^- x'_i \qquad \forall i \in V_1 \tag{4.16}$$

$$z_i \geq \sum_{j \in V_2} \mu_{ij} x'_j + W_i^+ (x'_i - 1) \qquad \forall i \in V_1 \tag{4.17}$$

$$|V_2| \sum_{i \in V_1} x'_i + 1 \geq \sum_{j \in V_2} x'_j \tag{4.18}$$

$$|V_1| \sum_{j \in V_2} x'_j + 1 \geq \sum_{i \in V_1} x'_i \tag{4.19}$$

$$(4.9), z_i \in \{0, 1\} \qquad \forall i \in V_1$$

Função objetivo (4.15) minimiza o custo reduzido da coluna em termos das variáveis z e x. Restrições (4.16) e (4.17) garantem que $z_i = \sum_{j \in V_2} \bar{\mu}_{ij} x'_i x'_j$. Para verificar a

corretude das restrições (4.16) e (4.17), considere um vértice $i \in \{1, 2, ..., |V_1|\}$. Se $x'_i = 0$, então $z_i = 0$. Para $x'_i = 0$, temos as seguinte restrições:

$$z_i \geq 0,$$
$$z_i \geq \sum_{j \in V_2} \mu_{ij} x'_j - W_i^+,$$

onde $W_i^+ = \sum_{j \in V_2 \wedge \mu_{ij} > 0} \mu_{ij}$ é suficientemente grande para garantir que $\sum_{j \in V_2} \mu_{ij} x'_j - W_i^+ \leq 0$. Note que z_i irá assumir o menor valor possível pois a variável z_i possui coeficiente positivo na função objetivo de um problema de minimização. Logo, a primeira restrição estará ativa e z_i irá assumir valor zero. Agora suponha que $x'_i = 1$. Neste caso, devemos verificar que $z_i = \sum_{j \in V_2} \mu_{ij} x'_j$. Para $x'_i = 1$ temos as seguinte restrições:

$$z_i \geq W_i^-,$$
$$z_i \geq \sum_{j \in V_2} \mu_{ij} x'_j,$$

onde $W_i^- = \sum_{j \in V_2 \wedge \mu_{ij} < 0} \mu_{ij}$ é suficientemente pequeno para que $W_i^- \leq \sum_{j \in V_2} \mu_{ij} x'_j$. Logo, a segunda restrição será ativa, significando que z_i assumirá o valor $\sum_{j \in V_2} \mu_{ij} x'_j$.

Finalmente, as restrições (4.18) e (4.19) proíbem que dois ou mais vértices da mesma partição façam parte da solução (coluna gerada) sem que não haja nenhum vértice da outra partição.

4.3 Técnicas de Geração de Colunas

Esta seção descreve a técnica de GC proposta para resolver o BEP. Devido ao grande número de variáveis da formulação $\mathcal{F}_\lambda$, sua solução direta por meio de um resolvedor de PLI se torna proibitiva à medida que as instâncias crescem. Os algoritmos de GC são usados para resolver sua relaxação linear a cada nó da árvore de *Branch-and-Bound*, resultando em um algoritmo de *Branch-and-Price*.

4.3.1 Geração de Colunas Padrão

O algoritmo de Geração de Colunas padrão proposto para o BEP é detalhado no Alg. 1. Ele começa com a inicialização do PMR com um subconjunto das colunas da relaxação linear de $\mathcal{F}_\lambda$. Tais colunas iniciais são fornecidas a partir da melhor solução gerada por uma meta-heurística GRASP proposta no Capítulo 5. A cada iteração, o PMR é resolvido e as variáveis duais π_v das restrições (4.2) são obtidas. O subproblema de *pricing* $\mathcal{SP}_{l2}$ é alimentado com estas variáveis duais e resolvido até a otimalidade por um resolvedor de PLI. As soluções do $\mathcal{SP}_{l2}$ representam *biclusters*, onde cada *bicluster* é uma coluna do PMR com custo reduzido dado pela função objetivo (4.15). Se o custo reduzido for negativo, esta coluna será adicionada ao PMR e o algoritmo prossegue para a próxima iteração. Em uma dada iteração, se as solução do subproblema de *pricing* gerar colunas com custo reduzido não negativo, significa que estas colunas não serão adicionada ao PMR. Além disso, significa que a adição de colunas ao PMR não é mais necessária e que a solução ótima da relaxação linear de $\mathcal{F}_\lambda$ foi encontrada. Então, o algoritmo para, retornando esta solução.

Algoritmo 1 Algoritmo de Geração de Colunas padrão.

```
1: ColsStandardCG(PMR)
2: sol ← ∅
3: sol_{SP_l2} ← ∅
4: Inicialize PMR
5: colAdded ← true
6: while colAdded do
7:     sol ← resolvedorPL(PMR)
8:     colAdded ← false
9:     Atualize SP_l2 com os valores da solução dual π
10:    lista_solucoes_SP_l2 ← resolvedorPLI(SP_l2(π))
11:    for all sol_{SP_l2} ∈ lista_solucoes_SP_l2 do
12:        if f(sol_{SP_l2}) < 0 then
13:            Insere coluna associada a sol_{SP_l2} para PMR
14:            colAdded ← true
15: return sol
```

4.3.2 Geração de Colunas por fases

Podemos perceber que o algoritmo GC padrão proposto para o problema na subseção anterior tem sua performance computacional dependente da performance do resolvedor de PLI usado no *pricing*. Por isto, com o objetivo de aliviar o peso computacional da GC padrão, propõe-se aplicar uma técnica similar à apresentada por [Kramer *et al.*, 2014] ao

Problema de *Clusterização* de *Software*.

Esta versão difere da anterior na adoção de duas fases de *pricing*, sendo uma heurística e outra exata, para a resolução do PMR. Primeiramente, o PMR é inicializado como previamente definido no Alg. 1. Na primeira fase (heurística) uma heurística construtiva é aplicada na resolução do subproblema de *pricing*. Mais detalhes desta heurística serão apresentados na Seção 4.4. A cada iteração, as colunas associadas às soluções obtidas pela heurística com o custo reduzido negativo serão adicionadas ao PMR. Se não houver colunas com custo reduzido negativo, nada será feito e a fase heurística é finalizada. Em seguida, inicia a fase exata, onde o subproblema $\mathcal{SP}_{l2}$ será resolvido na otimalidade pelo resolvedor de problemas de PLI. Havendo soluções da fase exata com colunas associadas de custo reduzido negativo, estas colunas são adicionadas ao PMR e o algoritmo itera, retornando à execução da fase heurística. Caso nenhuma solução da fase exata obtenha colunas de custo reduzido negativo então o algoritmo será finalizado, retornando a solução ótima do PMR.

4.3.3 *Branching*

O esquema de *branching* no algoritmo *Branch-and-Price* é: após resolver cada nó da árvore de *Branch-and-Bound* usando um dos algoritmos de GC apresentados, dois nós filhos são criados. Para cada um destes novos nós, restrições disjuntivas de *branching* são adicionadas ao PMR na forma apresentada por [Vanderbeck, 2011], que generaliza o esquema de [Ryan & Foster, 1981]. Para o BEP, estas restrições disjuntivas são:

$$\sum_{k \in \hat{K}} \lambda_k \leq 0 \quad \text{ou} \quad \sum_{k \in \hat{K}} \lambda_k \geq 1, \tag{4.20}$$

onde $\hat{K}$ é um subconjunto de colunas do PMR em que, para dois vértices $i \in V_1$ e $j \in V_2$, temos $a_{ik} = 1$ e $a_{jk} = 1$, ou seja, i e j fazem parte do mesmo *bicluster* nestas colunas.

Assim, adicionando as restrições disjuntivas da esquerda, as variáveis λ_k com $k \in \hat{K}$ são proibidas na solução do PMR, isto é, os vértices i e j não podem fazer parte do mesmo *bicluster*. Pela adição das restrições da direita, pelo menos uma variável λ_k com $k \in \hat{K}$ deve estar na solução do PMR; isto significa que, nesta solução, os vértices i e j devem estar no mesmo *bicluster*.

4.4 Heurística construtiva para o Subproblema de *Pricing*

Para resolver instâncias médias e grandes para o subproblema de *pricing* durante o algoritmo de GC, propomos uam heurística iterativa que consiste de duas fases: *construção* e *busca local*. A melhor solução obtida entre todas as iterações é considerada a solução final. Nesta seção apresentamos uma heurística construtiva e três movimentos de vizinhança adaptadas ao subproblema. Como entrada recebemos o bigrafo G_{SP} descrito na seção 4.2 e as variáveis duais π_v associadas as restrições do modelo $\mathcal{F}_\lambda$. Temos como representação da solução uma lista dos vértices que compõem um *bicluster*.

4.4.1 Fase de construção

A fase de construção é baseada na análise da vizinhança de vértices em G_{SP}. Ela inicia com um *bicluster* k vazio. Uma escolha gulosa de um par de vértices (i, j), $i \in V_1$ e $j \in V_2$, é realizada. Esta escolha é guiada com a ajuda de uma função gulosa $g(i, j)$. A cada execução constrói-se uma lista de candidatos (LC) contendo todos os pares de vértices de G_{SP} ordenados (crescentemente) pela função gulosa $g(i, j)$. Depois, uma lista restrita de candidatos (LRC) é construída pela seleção dos candidatos em LC com os menores valores de $g(i, j)$. Segue-se a definição da função gulosa:

$$g(i, j) = w_\pi(i, j) + in(i, j) + out(i, j)$$

onde:

- $w_\pi(i, j) = \frac{1}{2}(|N(i)| + |N(j)|) - (\pi_i + \pi_j) + w(i, j)$;
- $in(i, j)$ é a soma dos pesos $w_\pi(i, j)$ de todos os pares de vértices entre i e $N(i) \cap V_2$, e entre j e $N(j) \cap V_1$;
- $out(i, j)$ é a soma dos pesos positivos ($w(i, j) > 0$) de todos os pares de vértices entre i e $N^2(j) \cap V_2$, e entre j e $N^2(i) \cap V_1$;

Após a escolha aleatória de um par de vértices (i, j) contidos na LRC, um novo *bicluster* isolado k é criado, composto pelo subconjunto de vértices $N[i] \cup N[j]$. A Figura 4.3 ilustra a construção de um *bicluster* a partir da escolha do par de vértices $\{4, 7\}$.

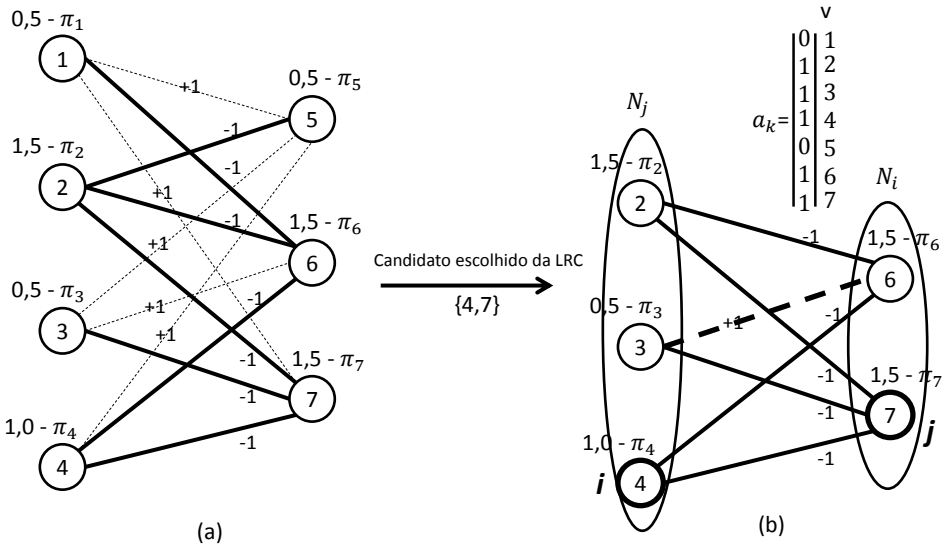


Figura 4.3: (a) Bigrafo de entrada para a fase de construção GRASP. (b) Coluna a_k construída a partir do isolamento dos vértices $N[i] \cup N[j]$.

4.4.2 Fase de busca local

Na fase de busca local, novas soluções próximas à solução obtida na fase de construção são geradas. O método de busca em descida, *Variable Neighborhood Descent* (VND) descrito em [Hansen *et al.*, 2010], é usado pela aplicação do conjunto de vizinhanças $\mathcal{N} = \{$*Remove-vértice, Adiciona-vértice, Troca-vértices*$\}$, na ordem em que aparecem. Para isto, são propostos os seguintes movimentos de vizinhança sobre um bicluster k: (a) ***Adiciona-vértice***: adiciona um vértice $v \notin k$ a k; (b) ***Remove-vértice***: remove um vértice $v \in k$ de k; (c) ***Troca-vértices***: remove um vértice $v_1 \in k$ e adiciona $v_2 \notin k$ a k.

Capítulo 5

Heurísticas para o Problema de Edição de Biclusters

Duas das abordagens tradicionalmente empregadas na implementação de algoritmos para o tratamento de Problemas de Otimização Combinatória (POC) são os métodos exatos e heurísticos. Métodos exatos permitem a obtenção da melhor solução possível para o problema (solução ótima). Porém, para muitas aplicações teóricas e práticas, a demanda computacional necessária para a execução desses métodos é freqüentemente impraticável. Por sua vez, métodos heurísticos possibilitam a produção de resultados sub-ótimos — eventualmente ótimos — consumindo tempo e recursos computacionais que normalmente atendem às limitações impostas pela aplicação.

Dentre os métodos heurísticos, as meta-heurísticas são largamente reconhecidas como ferramentas essenciais para o tratamento de problemas complexos em diversas áreas e, freqüentemente, consistem na única abordagem viável capaz de atender às demandas de determinadas aplicações [Blum *et al.*, 2005]. Originalmente cunhado por Fred Glover em [Glover, 1986], o termo meta-heurística é utilizado para descrever algoritmos que empregam metodologias gerais de alto nível que são utilizadas como estratégias orientadoras de heurísticas subjacentes para a resolução de POC [Talbi, 2009]. Em outras palavras, este conceito estabelece que as meta-heurísticas executam um processo de busca capaz de escapar de ótimos locais ao tempo em que realizam a varredura "inteligente" do espaço de soluções por meio de heurísticas especializadas.

Para resolver instâncias médias e de grande porte para o PEB, propomos heurísticas a serem adaptadas a meta-heurísticas clássicas. Sendo assim, neste capítulo, serão apresentados uma estrutura auxiliar de dados que acelera os movimentos de vizinhança, um método construtivo, movimentos de vizinhança aplicados à soluções viáveis e a des-

crição das meta-heurísticas adaptadas. Para isso, adotamos a seguinte terminologia: seja $G' = (V, E')$ uma solução viável (ou seja, G' é uma união disjunta de bicliques); se $(i, j) \in E'$ então (i, j) está *ativa* em G', caso contrário (i, j) está *inativa* em G'. O valor da solução viável G' é portanto dada por:

$$custo(G') = \sum_{+(i,j) \cap \overline{E'}} w(i,j) \quad + \sum_{-(i,j) \cap E'} |w(i,j)| \tag{5.1}$$

Nesta fórmula, a primeira soma refere-se as arestas positivas que se tornaram inativas (removidas), e a segunda refere-se às arestas negativas que se tornaram ativas (adicionadas).

5.1 Heurística Construtiva

Dado um grafo bipartido G para o PEB, a *Heurística construtiva* (HC) é baseada na análise das vizinhanças dos vértices em G. Ela inicia com um grafo vazio G'. Em cada iteração uma escolha gulosa de uma aresta (i, j), $i \in V_1$ e $j \in V_2$, é realizada. Esta escolha é guiada com a ajuda de uma função gulosa $g(i, j)$, definida para cada aresta (i, j) como segue:

$$g(i,j) = w(i,j) + in(i,j) + diff(i,j) - out(i,j) \tag{5.2}$$

onde:

- $w(i, j)$ representa o peso da aresta (i, j);
- $in(i, j)$ é a soma dos pesos das arestas positivas entre i e $N^2(j) \cap V_2$, e entre j e $N^2(i) \cap V_1$;
- $diff(i, j)$ é a soma dos pesos das arestas negativas entre i e $N^2(j) \cap V_2$, e entre j e $N^2(i) \cap V_1$;
- $out(i, j)$ é a soma dos pesos das arestas positivas entre i e $V_2 \backslash N^2(j)$, e entre j e $V_1 \backslash N^2(i)$.

A Figura 5.1 ilustra as arestas associadas a cada parâmetro da função $g(i, j)$ no grafo de entrada G ao analisar a aresta $\{3, 6\}$. Neste exemplo a função g(i,j), ao analisar a

aresta $\{3,6\}$, obteve o seguinte valor $g(3,6) = w(3,6) + in(3,6) + diff(3,6) - out(3,6) = -1 + 2 - 1 - (+1) = -1$.

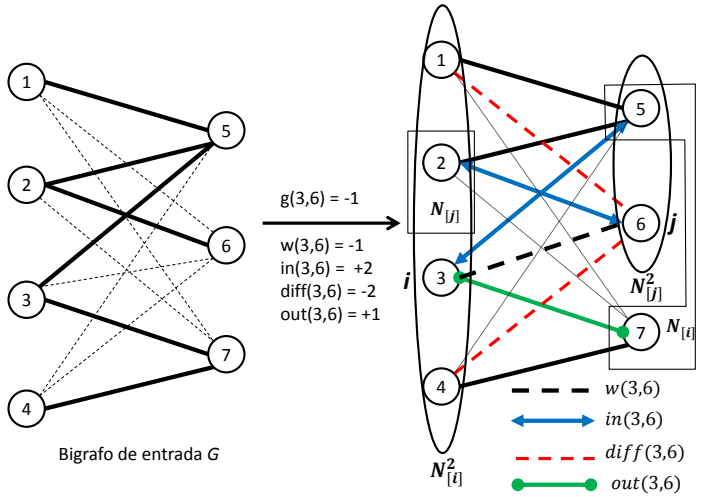


Figura 5.1: Parâmetros da função $g(i,j)$ analisando a aresta $\{3,6\}$ do Bigrafo G.

Após a escolha da aresta(i,j) com valor máximo $g(i,j)$, um novo *bicluster* isolado B, como ilustrado na Figura 5.2, composto pelo subconjunto de vértices $N(i) \cup N(j)$ é adicionado no grafo G'. Todas arestas originais do grafo G de entrada com pelo menos um ponto extremo em $N(i) \cup N(j)$ deixam de ser candidatas, e uma nova iteração será realizada enquanto houver arestas candidatas. Se ao final sobrar vértices sem arestas positivas para vértices da outra partição, cada vértice v isolado será colocado em um *bicluster* B composto apenas de v (*singleton*) e adicionado a solução G'.

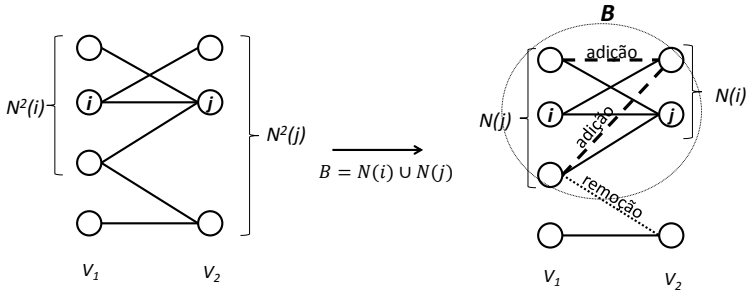


Figura 5.2: *Bicluster* B formado por uma iteração do HC.

5.2 Matriz de Edições

A avaliação de todos os movimentos de vizinhança propostos neste trabalho é composta pela avaliação de mover um vértice de seu atual *bicluster* B para outro *bicluster* B'. Assim, com o objetivo de acelerar a fase de busca local, definimos uma matriz auxiliar que permite

a avaliação deste movimento em tempo constante ($O(1)$). Seja G' uma solução formada pela coleção $\mathcal{B}$ de *biclusters*. A Matriz de Edições M para G' é definida do seguinte modo. Uma entrada $M(v, B)$, para um vértice $v \in V_1$ e um *bicluster* $B \in \mathcal{B}$, é dado por:

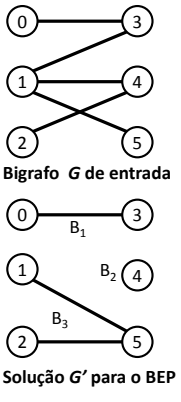
$$M(v,B) = \begin{cases} \sum_{j \in V(B) \cap V_2} w(v,j) & \text{, if } v \in V(B); \\ -\sum_{j \in V(B) \cap V_2} w(v,j) & \text{, if } v \notin V(B). \end{cases}$$

Similarmente, para $u \in V_2$ e $B \in \mathcal{B}$,

$$M(u,B) = \begin{cases} \sum_{i \in V(B) \cap V_1} w(i,u) & \text{, if } u \in V(B); \\ -\sum_{i \in V(B) \cap V_1} w(i,u) & \text{, if } u \notin V(B). \end{cases}$$

Note que $M(v, B)$ contém a variação na função objetivo após remover v de B se $v \in V(B)$, e a variação na função objetivo após inserir v em B se $v \notin V(B)$. A matriz M pode ser construída do seguinte modo. Inicialmente, todas as entradas de M são iguais a zero. Depois, para cada par de vértices (i, j) tal que $i \in V_1$ e $j \in V_2$, seja B_i *bicluster* contendo i e B_j o *bicluster* contendo j. Se $B_i = B_j$, então adiciona $w(i, j)$ para $M(i, B_i)$ e $M(j, B_j)$. Caso contrário, subtraia $w(i, j)$ de $M(i, B_i)$ e $M(j, B_j)$.

A Figura 5.3(a) apresenta um bigrafo de entrada G e uma solução G' formada pelos *biclusters* B_1, B_2 e B_3 com $valor(G') = 4$, enquanto a Figura 5.3(b) ilustra a matriz de edições para G'.



v	B		
	B_1	B_2	B_3
0	+1	+1	+1
1	-1	-1	+1
2	+1	-1	-1
3	+1	0	0
4	+1	0	-2
5	+1	0	0

Matriz de edições *M*

(b)

Figura 5.3: (a) Bigrafo de entrada G e solução G'; (b) Matriz de Edições para G'.

Na próxima seção, descrevemos como usar esta matriz na avaliação de todos os

movimentos de vizinhança propostos neste trabalho, assim como a atualização da matriz após a aplicação destes movimentos.

5.3 Movimentos de vizinhança

Nesta seção, apresentamos os movimentos de vizinhança propostos neste trabalho. Em todos os procedimentos de movimentação de vizinhança, é considerada uma solução prévia viável G' formada por um conjunto $\mathcal{B}$ de *biclusters*.

Move-Vértice: para um *bicluster* $B \in \mathcal{B}$ e para um vértice i em B, remova i de B e o mova para um outro *bicluster* $B' \in \mathcal{B}$; e adicione as arestas entre i e seus novos vizinhos em B', e remova suas arestas entre i e seus antigos vizinhos em B. Figura 5.4(a) apresenta este movimento.

Seja G'' a nova solução viável. Ao invés de usar a equação (2.5), que demanda mais cálculos, podemos calcular o valor da nova solução viável G'' em tempo constante $O(1)$, já que o $valor(G')$ é conhecido, usando a seguinte fórmula:

$$valor(G'') = valor(G') + M(v, B) + M(v, B') \tag{5.3}$$

Após mover um vértice $v \in V_1$ de um *bicluster* B para um *bicluster* B', a matriz de edição M pode ser atualizada em tempo $O(|V_2|)$, uma vez que temos que atualizar apenas os valores $M(v, B)$, $M(v, B')$, $M(u, B)$ e $M(u, B')$, $\forall u \in V_2$. Estes novos valores são computados do seguinte modo.

$$M(v, B) = -M(v, B)$$
$$M(v, B') = -M(v, B')$$

$$M(u, B) = \begin{cases} M(u, B) - w(v, u), & \forall u \in V(B) \cap V_2 \\ M(u, B) + w(v, u), & \forall u \in V_2 \backslash V(B) \end{cases}$$

$$M(u, B') = \begin{cases} M(u, B') + w(v, u), & \forall u \in V(B') \cap V_2 \\ M(u, B') - w(v, u), & \forall u \in V_2 \backslash V(B') \end{cases}$$

Analogamente, quando movemos um vértice $u \in V_2$, pode-se atualizar a matriz M em tempo $O(|V_1|)$.

União-Bicluster: para um par B, B' de *biclusters*, crie um novo *bicluster* B'' pela adição de todas as arestas entre $V_1 \cap V(B)$ e $V_2 \cap V(B')$, e entre $V_2 \cap V(B)$ e $V_1 \cap V(B')$ (onde

$V(B)$ e $V(B')$ são, respectivamente, o conjunto de vértices de B e B'); adiciona B'' a $\mathcal{B}$ e remove B', B'' de $\mathcal{B}$. Figura 5.4(b) apresenta este movimento; vértices em V_1 são coloridos de preto, enquanto vértices em V_2 são coloridos de branco.

Se G'' é a nova solução viável, $valor(G'')$ pode ser obtida a partir de $valor(G')$ em tempo $O(|V(B)|)$ usando a seguinte equação:

$$valor(G'') = valor(G') + \sum_{v \in V(B)} M(v, B') \tag{5.4}$$

Equivalentemente, $valor(G'')$ pode ser computado em tempo $O(|V(B')|)$ usando a seguinte fórmula:

$$valor(G'') = valor(G') + \sum_{v \in V(B')} M(v, B) \tag{5.5}$$

Após aplicar este movimento, uma nova coluna de M em relação a B'' é computada em tempo $O(|V_1| + |V_2|)$ do seguinte modo.

$$M(v, B'') = \begin{cases} M(v, B) - M(v, B'), & \forall v \in V(B) \\ M(v, B') - M(v, B), & \forall v \in V(B') \end{cases}$$
$$M(v, B'') = M(v, B) + M(v, B'), \forall v \in V_1 \cup V_2, v \notin V(B) \cup V(B')$$

Finalmente, as colunas relacionadas a B e B' são removidos de M.

Quebra-Bicluster: para um *bicluster* $B \in \mathcal{B}$, crie dois novos *biclusters* B' e B'' pelo particionamento dos vértices em $V(B) \cap V_1$ e em $V(B) \cap V_2$; adicione B' e B'' para $\mathcal{B}$ e remove B de $\mathcal{B}$. Vide Figura 5.4(c). Para realizar a divisão de B em dois *biclusters*, usamos a função $bind(v)$ definida a seguir, que examina como vértices $v \in V(B)$ estão atualmente conectados:

$$bind(v) = M(v, B) \tag{5.6}$$

Uma nova solução G'' é formada pelo particionamento do conjunto de vértices do *bicluster* B usando o seguinte algoritmo: cada vértice $v \in V(B)$ com valor $bind(v) < 0$ é movido de B para B'; os elementos restantes contidos em B formam o segundo *bicluster* B''. Note que podemos computar $bind(v)$ para todos os vértices $v \in V(B)$ em tempo $O(|\mathcal{B}||V(B)|)$ e $valor(G'')$ em tempo $O(|V(B)||V(B')|)$ usando as seguinte equações:

$$valor(G'') = valor(G') + \sum_{\substack{v \in B' \\ v \in V_1}} \sum_{\substack{u \in B'' \\ u \in V_2}} w(v,u) + \sum_{\substack{v \in B'' \\ v \in V_1}} \sum_{\substack{u \in B' \\ u \in V_2}} w(v,u) \tag{5.7}$$

Além disso, este movimento acarreta a criação de duas novas colunas em M respectivas aos dois novos *biclusters* B' e B'', onde a coluna de B' é computada do seguinte modo.

$$M(v,B') = M(v,B) \; - \sum_{\substack{u \in V(B'') \\ u \in V_2}} w(v,u), \forall v \in V(B') \cap V_1$$

$$M(v,B') = -M(v,B) \; + \sum_{\substack{u \in V(B'') \\ u \in V_2}} w(v,u), \forall v \in V(B'') \cap V_1$$

$$M(v,B') = M(v,B) \; + \sum_{\substack{u \in V(B'') \\ u \in V_2}} w(v,u), \forall v \in V_1, v \notin B' \cup B''$$

Os valores $M(u,B')$ para todo $u \in V(B') \cap V_2$ e a coluna de B'' são computados de modo similar. Finalmente, a coluna relacionada a B é removida de M.

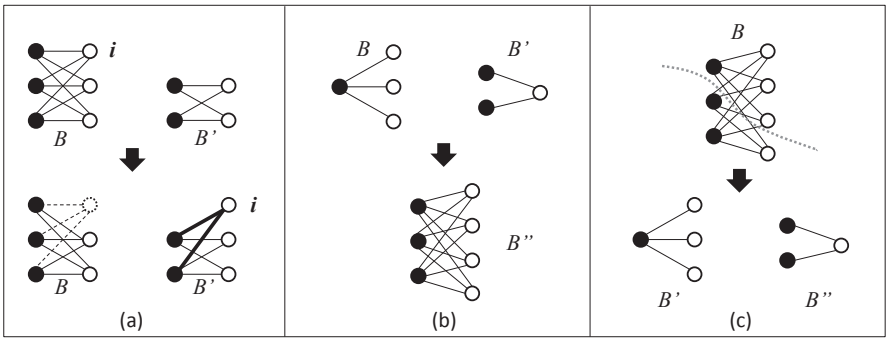


Figura 5.4: Movimentos de vizinhança.

5.4 Meta-heurística VNS para o PEB

Variable Neighbourhood Search (VNS) é uma meta-heurística baseada em mudanças sistemáticas de vizinhanças tanto na fase de descida, responsável por encontrar um mínimo local, quanto na fase de perturbação, responsável por emergir do correspondente vale.

O *framework* VNS adotado neste trabalho, GVNS, é apresentado no Algoritmo 2. É baseado no VNS geral descrito em [Hansen *et al.*, 2010]. A solução inicila é gerado pela Heurística Construtiva descrita na seção 5.1. Na fase de descida, o método *Variable*

Algoritmo 2 Algoritmo padrão do *General Variable Neigborhod Search.*

```
1: function GVNS(x, k_max)
2:     - x é uma solução viável inicial, a ser melhorada
3:     - k_max é o número de movimentos de vizinhança
4:     repeat
5:         k ← 1;  - k é o índice do movimento de vizinhança corrente
6:         repeat
7:             x' ← Shake(x, k);
8:             x'' ← VND(x', k_max);
9:             if f(x'') < f(x) then
10:                x ← x''; k ← 1;
11:            else
12:                k ← k + 1;
13:        until k = k_max
14:    until condição de parada cumprida
15:    return x
```

Neighborhood Descent (VND) é usado, pela aplicação do conjunto de vizinhanças $\mathcal{N}$ = {*União-Bicluster, Quebra-Bicluster, Move-Vertex*} (veja Figura 5.5). A fase de perturbação (método `Shake`) também trabalha com o conjunto de vizinhanças $\mathcal{N}$, aplicando para a vizinhança corrente um número aleatório de movimentos de perturbação, ou seja, o movimento é aplicado sem a necessidade de encontrar um vizinho de melhor qualidade, são aplicados um número aleatório entre 2 e 10 de movimentos de perturbação. Se nenhuma melhoria na solução for encontrada a cada iteração, o movimento de vizinhança a ser usado na próxima iteração é trocado. O uso desta técnica de VNS geral (VNS/VND) tem tido sucesso na resolução de algumas aplicações, como reportado por [Hansen *et al.*, 2010], p. 375.

procedure `VND`(x, k_{max})
1. Seja $\mathcal{N}_1 =$ ***Move-Vértice***, $\mathcal{N}_2 =$ ***União-Bicluster***, $\mathcal{N}_3 =$ ***Quebra-Bicluster***
2. $k \leftarrow 1$ `// k é o índice da estrutura de vizinhança corrente`
3. **repit**
4. $x' \leftarrow$ melhor vizinho de x na estrutura de vizinhnaça $\mathcal{N}_k$
5. **se** $f(x') < f(x)$
6. **então** $x \leftarrow x'$; $k \leftarrow 1$
7. **senão** $k \leftarrow k + 1$
8. **até** $k > k_{max}$
9. **retorne** (x)

end

Figura 5.5: Algoritmo do VND para o PEB.

5.5 Meta-heurística ILS para o PEB

Iterated Local Search (ILS) [Lourenço *et al.*, 2003] é um método que cria uma seqüência de soluções geradas por uma heurística embutida. Esta técnica torna-se dependente principalmente da escolha da busca local, as perturbações, e o critério de parada. Seu *framework* está apresentado no Algoritmo 3.

Algoritmo 3 *Framework* do *Iterated Local Search*.

```
1:  function IteratedLocalSearch(.)
2:      s_0 ← Gera_Solucao_Inicial();
3:      s* ← Busca_Local(s_0);
4:      s** ← s*;
5:      repeat
6:          s' ← Perturbacao(s**);
7:          s** ← Busca_Local(s');
8:          s* ← Criterio_Parada(s*, s**);
9:      until condição de parada cumprida
10:     return (s*);
```

5.5.1 Fase de Construção

Para a geração de soluções iniciais adotou-se o procedimento HC (Subseção 5.1).

5.5.2 Fase de Busca Local

Na fase de busca local, adotou-se o mesmo procedimento VND da linha 8 do Algoritmo 2. ILS escapa de um ótimo local pela aplicação de perturbações; assim propomos um número aleatório de movimentos de perturbação (entre 2 e 10), e para cada movimento uma escolha aleatória de um movimento de vizinhança contido no conjunto $\mathcal{N}$ é feito, permitindo assim caminhar em regiões mais amplas em torno de uma solução.

Todo progresso no espaço de busca de soluções é controlado por uma função `Critério_Aceitação`(s^*, s^{**}), na linha 7 do Algoritmo 3. Seja i_{last} a última iteração em que a melhor solução foi encontrada, e seja i o contador da iteração corrente. Então `Critério_Aceitação`(s^*, s^{**}) é definido como:

$$\texttt{Criterio\_Aceitacao}(s^*, s^{**}) = \begin{cases} s^{**} & \text{if } f(s^{**}) < f(s^*) \quad (\text{do } i_{last} \leftarrow i) \\ s^* & \text{if } f(s^{**}) \geq f(s^*) \text{ e } i - i_{last} < i_r \\ s^* & \text{caso contrário} \quad \triangleright \texttt{criterio parada cumprido} \end{cases}$$

onde: (i) s^* e s^{**} são soluções viáveis G' e G''; (ii) $f(s^*) = custo(G')$ e $f(s^{**}) = custo(G'')$ (vide equação (5.1)); (iii) i_r é um parâmetro indicando que o algoritmo deve parar se nenhuma melhora de solução foi encontrada em i_r iterações (definimos $i_r = 3$, por encontrar bons resultados empíricos).

5.6 Meta-heurística GRASP para o PEB

Greedy Randomized Adaptive Search Procedure (GRASP) [Resende, 2001] é um procedimento iterativo onde cada iteração consiste de duas fases: *construção* e *busca local*. A melhor solução obtida entre todas as iterações é considerada a solução final.

Na fase de construção, uma solução (vazia inicialmente) inclui interativamente um novo elemento ou parte dele, com o objetivo de formar uma nova solução viável. Durante inclusão de um novo elemento, dois aspectos são analisados: adaptação e aleatoriedade.

Soluções viáveis obtidas na fase de construção do GRASP não garantem um ótimo local, considerando uma dada estrutura de vizinhança. Sendo assim, o uso da segunda fase do GRASP é feito na tentativa de melhorar a solução obtida durante a fase de construção.

5.6.1 Fase de Construção

A fase de construção, apresentada no Algoritmo 4, inicia com um grafo vazio G'. A cada iteração, uma Lista de Candidatos LC é definida, consistindo de todas as arestas (i,j), sendo $i \in V_1$ e $j \in V_2$ pertencentes a G. Depois, uma Lista Restrita de Candidatos LRC é construída pela seleção dos melhores candidatos em LC, com a ajuda da função gulosa $g(i,j)$ definida na Subseção 5.1. Os melhores candidatos satisfazem a condição:

$$g(i,j) \geq g_{min} + \alpha(g_{max} - g_{min}), \tag{5.8}$$

onde $g_{min} = \min\{g(i,j) \mid (i,j) \in LC\}$, $g_{max} = \max\{g(i,j) \mid (i,j) \in LC\}$, e $\alpha \in (0,1)$. Então uma aresta (i,j) é escolhida aleatoriamente de LRC a fim de construir um *bicluster* B com o conjunto de vértices $N(i) \cup N(j)$. Ao final de uma iteração, B é adicionado a solução G', e os vértices em $N(i) \cup N(j)$ são removidos de G junto com todas as arestas que tenham algum ponto final sobre algum destes vértices removidos. Se ao final sobrar vértices sem arestas positivas para vértices da outra partição, cada vértice v isolado será colocado em um *bicluster* B composto apenas de v (*singleton*) e adicionado a solução G'. A condição de parada para a fase de construção é $V(G) = \emptyset$.

Algoritmo 4 Algoritmo GRASP: fase de construção.

1: **function** CONSTRUÇÃOGRASP($G = (V_1, V_2, E), \alpha, g(.)$)
2: $G' \leftarrow (\emptyset, \emptyset, \emptyset)$;
3: **while** $V(G) \neq \emptyset$ **do**
4: $LC \leftarrow E(G)$;
5: $LRC \leftarrow \{(i,j) \in CL \mid g(i,j) \geq g_{min} + \alpha(g_{max} - g_{min})\}$;
6: $(i,j) \leftarrow$ seleção aleatória em LRC;
7: $V(B) \leftarrow N(i) \cup N(j)$;
8: transforme B em um biclique isolado;
9: $G' \leftarrow G' \cup B$;
10: $G \leftarrow G[V(G) \backslash V(B)]$;
11: **return** G';

5.6.2 Fase de Busca Local

Na fase de busca local, novas soluções próximas da solução obtida na fase de construção são geradas pelo uso do mesmo procedimento VND da linha 8 do Algoritmo 2.

Capítulo 6

Resultados Computacionais

6.1 Configuração dos experimentos

Todos os algoritmos propostos neste trabalho foram desenvolvidos em C++ e com o auxílio da ferramenta matemática CPLEX 11. Os algoritmos de aproximação para o BEP 4-approx [Ailon *et al.*, 2012] e 11-approx [Amit, 2004] também foram desenvolvidos em C++. Utilizamos a implementação original das heurísticas EDH [Sun *et al.*, 2013] e Bi-Force [Sun *et al.*, 2014], que estão disponíveis em `http://biclue.mpi-inf.mpg.de/`. Os experimentos computacionais com as instâncias $G(n, m, p)$ e $P(G, q)$ (descritas na próxima Seção) foram realizados em um processador Intel Core 2 Quad 2,33 GHz com 4 GB de RAM, e executando o sistema operacional Linux Ubuntu 12.04. As instâncias grandes derivadas de dados biológicos (Seção 6.1.3) foram executados em um processador Intel Core 2 Quad 2,83 GHz com 16 GB de RAM, e executando sistema operacional Linux Ubuntu 14.04. Todas as nossas implementações e o conjunto de instâncias estão disponíveis em `http://sites.google.com/site/biclustereditingproblem/documents`.

6.1.1 Instâncias $G(n, m, p)$

Na Seção 2.2.1, foi observado que as instâncias $G(n, m, p)$ mais difíceis para o BEP são geradas com probabilidades $p \in \{0,5; 0,6; 0,7\}$. Em nossos experimentos computacionais, quatro grupos de instâncias aleatórias foram geradas, usando o algoritmo descrito na Seção 2.2.1. Cada grupo possui cinco pares da forma (n, m), onde $n = |V_1|$ e $m = |V_2|$, e para cada par os três valores de p são considerados. Os pares (n, m) usados em cada grupo BC_j são:

$BC_1 : (5, 7), (6, 8), (6, 12), (7, 11), (6, 20)$

$BC_2 : (10, 16), (20, 23), (16, 30), (20, 35), (24, 40)$
$BC_3 : (28, 46), (30, 41), (30, 50), (35, 45), (40, 40)$
$BC_4 : (37, 54), (30, 90), (40, 70), (50, 50), (40, 100)$

Instâncias com valores grandes para n e m e valor de p pequeno também foram geradas. Mais especificamente, para cada par (n, m) nos dois grupos seguintes, as probabilidades $p \in \{0,2; 0,3\}$ são consideradas:

$BC_5 : (74, 108), (60, 180), (80, 140), (100, 100), (80, 200)$
$BC_6 : (148, 216), (120, 360), (160, 280), (200, 200), (160, 400)$

Por fim, são consideradas 80 instâncias $G(n, m, p)$ em nosso experimentos.

6.1.2 Instâncias $P(G, q)$

Uma instância $P(G, q)$ para o BEP é derivada de um grafo biclusterizado G, cujas arestas são editadas com uma probabilidade q. Consideramos os grafos clusterizados correspondentes as soluções obtidas pela heurística GRASP (Seção 5.6) para as quinze instâncias $G(n, m, p)$ do grupo BC_4. Para cada solução G' obtida pelo GRASP, geramos três instâncias $P(G, q)$ usando G' e as probabilidades $q \in \{0,1; 0,2; 0,3\}$. Assim, consideramos 45 instâncias $P(G, q)$ em nossos experimentos.

6.1.3 Instâncias derivadas de dados biológicos

Como no trabalho de [Sun *et al.*, 2014], consideramos os dados de expressão genética disponível no conjunto de dados *Gene Expression Omnibus* (GEO) [Barrett *et al.*, 2013]. A Tabela 6.1 apresenta os conjuntos de dados adotados por [Sun *et al.*, 2014], cada um associado com um grafo bipartido ponderado onde o peso de uma aresta representa o nível da expressão de gene sobre uma certa condição. Informações detalhadas sobre estes conjuntos de dados podem ser encontrados em [Sun *et al.*, 2014].

Tabela 6.1: Conjunto de dados do GEO.

Dados	$\|V_1\|$	$\|V_2\|$	**Dados**	$\|V_1\|$	$\|V_2\|$	**Dados**	$\|V_1\|$	$\|V_2\|$
GDS181	12600	84	GDS589	8799	122	GDS1027	15923	154
GDS1319	22625	123	GDS1406	12488	87	GDS1490	12488	150
GDS3715	15923	6	GDS3716	22283	42	GDS3716	22283	42

Seja $G = (V_1, V_2, E)$ o grafo bipartido ponderado associado ao conjunto de dados mencionado acima. Como nossas soluções do PEB não atuam com grafos ponderados, transformamos G em um grafo bipartido não ponderado $G' = (V_1, V_2, E')$, onde $E' = \{(i,j) \mid i \in V_1 \text{ e } j \in V_2 \text{ e } w(i,j) \geq t\}$. Aqui, $w(i,j)$ significa o peso da aresta (i,j) e t é um limite usado para determinar se a aresta (i,j) pertence a E'. Para cada um dos 9 grafos associados ao conjunto de dados da Tabela 6.1, usamos como valores limites o primeiro, segundo e terceiro quartos (quadrantes) da lista formada por todos os pesos das arestas no grafo original. Portanto, tivemos uma total de 27 instâncias derivadas dos dados biológicos.

6.2 Comparação entre as regras de redução

A comparação das regras de redução apresentadas na Seção 2.2.2 foram conduzidas como se segue. Primeiro, a formulação linear $\mathcal{F}_{P_4}$ apresentada na Seção 6.3 foi implementada com o auxílio do software CPLEX 11. Em seguida, para cada instância, executamos três rodadas: na primeira, a instância original foi usada como entrada para a execução do modelo; na segunda rodada, aplicamos a **Regra 1** e **Regra 2** nas instâncias, antes de passá-las como entrada do modelo e na terceira rodada, aplicamos a **Regra 1** e **Nova Regra** nas instâncias. Nos experimentos, utiizamos 18 instâncias $G(n,m,p)$ cujos tamanhos foram escolhidos para que o software pudesse encontrar a solução ótima.

Tabela 6.2: Impacto das regras de redução.

Instância	***B&B***		**Regra 1 + Regra 2**			**Regra 1 + Nova Regra**		
$n*m$	e^*	*tempo* (ms)	$n*m$	*red* (%)	*tempo* (ms)	$n*m$	*red* (%)	*tempo* (ms)
35	7	96	35	0,00	118	35	0,00	100
35	8	72	35	0,00	83	**30**	**14,29**	55
35	7	39	35	0,00	49	**30**	**14,29**	27
48	8	87	**40**	**16,67**	51	42	12,50	91
48	11	158	48	0,00	161	48	0	166
48	12	137	48	0,00	141	**36**	**25,00**	75
72	14	501	72	0,00	520	**66**	**8,33**	406
72	17	1299	72	0,00	1281	**60**	**16,67**	456
72	20	1569	72	0,00	1582	72	0,00	1574
77	11	408	66	14,29	231	**56**	**27,27**	128
77	19	1723	77	0,00	1732	**70**	**9,09**	1118
77	21	1481	77	0,00	1486	77	0,00	1504
120	28	14230	114	5,00	16734	**102**	**15,00**	3938
120	30	11592	120	0,00	11595	**90**	**25,00**	2330
120	28	1204	120	0,00	1228	**78**	**35,00**	591
160	39	128060	160	0,00	130175	160	0,00	129442
160	47	721216	160	0,00	734261	160	0,00	693955
160	48	64119	160	0,00	65172	160	0,00	65944

Para cada linha da Tabela 6.2, a primeira coluna contém o tamanho das instâncias testadas, e as colunas restantes são divididas em três grupos: ***B&B***, **Regra 1 + Regra 2**, e **Regra 1 + Nova Regra**. No grupo ***B&B***, coluna e^* apresenta o valor ótimo (número

de edições), e coluna *tempo* indica o tempo (em milissegundos) gastos para resolver a instância original. Os grupos das colunas **Regra 1 + Regra 2** e **Regra 1 + Nova Regra**, têm a seguinte divisão: coluna $n * m$ indica a dimensão do problema aplicando a redução, *red* indica a porcentagem de redução com respeito ao tamanho original da instância, e coluna *tempo* indica o tempo de execução (em milissegundos) gastos para a redução da instância e resolução do problema reduzido.

As regras combinadas **Regra 1 + Regra 2** tiveram melhor resultados que **Regra 1 + Nova Regra** em apenas um caso. Este último, por sua vez, reduziu em média o tamanho das instâncias originais em 11,9%; podemos observar isto, na instância ($n = 6, m = 20, d = 0,7$), ela teve uma redução de 35%. Quando usamos **Regra 1 + Nova Regra**, as instâncias em que $red > 0$ levam, em média, para uma redução de 48,3% no tempo computacional, já as regras combinadas **Regra 1 + Regra 2**, em média, obtém uma redução de 28,2% no tempo computacional.

6.3 Comparação entre as formulações $\mathcal{F}_{P_4}$, $\mathcal{F}_{K_1}$ e $\mathcal{F}_{K_2}$

Para comparar os MPIs descritos no Capítulo 2, implementamos os três modelos no resolvedor matemático CPLEX, utilizando a técnica de *Branch-and-Bound*. São apresentados, na Tabela 6.3, os resultados dos modelos $\mathcal{F}_{P_4}$, $\mathcal{F}_{K_1}$ e $\mathcal{F}_{K_2}$ utilizando instâncias $G(n, m, p)$ com valor $p = \{0,5; 0,6\}$. A tabela é dividida em quatro grupos de colunas. O conjunto de colunas *Instância* é dividida em n, m, d e e^*, onde $n = |V_1|$, $m = |V_2|$, $d = \frac{|E|}{n*m}$ e e^* o valor ótimo relacionado a cada instância. As outras colunas possuem apenas o valor do tempo computacional de cada modelo em millisegundos.

Tabela 6.3: Comparação entre $\mathcal{F}_{P_4}$, $\mathcal{F}_{K_1}$ e $\mathcal{F}_{K_2}$ para o BEP.

Instância				$\mathcal{F}_{P_4}$	$\mathcal{F}_{K_1}$	$\mathcal{F}_{K_2}$
n	m	p	e^*	*tempo (ms)*	*tempo (ms)*	*tempo (ms)*
5	7	0,5	7	10	18	23
5	7	0,6	8	30	29	66
6	8	0,5	8	20	27	95
6	8	0,6	11	30	58	134
6	12	0,5	14	90	51	141
6	12	0,6	17	130	81	124
7	11	0,5	11	40	46	50
7	11	0,6	19	210	123	124
6	20	0,5	28	1990	105	127
6	20	0,6	30	1130	107	127
10	16	0,5	39	51560	1660	2292
10	16	0,6	47	62090	1716	1638
16	30	0,5	-	LT	LT	LT
16	30	0,6	162	LT	3182445	LT

Podemos observar na Tabela 6.3 que para as instâncias testadas os MPIS $\mathcal{F}_{K_1}$ e $\mathcal{F}_{K_2}$ obtiveram melhores tempos computacionais que o MIP $\mathcal{F}_{P_4}$. Reduziram o tempo em

mais de 95%, como podemos observar na instância $(n = 10, m = 16, p = 0,6)$, onde o $\mathcal{F}_{P_4}$ encontrou a solução ótima em 62 segundos enquanto $\mathcal{F}_{K_1}$ e $\mathcal{F}_{K_2}$ encontraram o ótimo, respectivamente, em 1,7 e 1,6 segundo.

6.4 Análise do algoritmo de $B\&C$ proposto

A implementação da abordagem de *Branch-and-Cut* ($B\&C$) proposta neste trabalho foi desenvolvida na linguagem C++ com o auxílio da ferramenta matemática CPLEX aplicado à formulação proposta por [Amit, 2004] e da heurística de separação das inequações Fole, Escada e Grid propostas na Seção 3.2. Comparamos nossos resultados com o algoritmo de *Branch-and-Cut* proposto por [Pinheiro *et al.*, 2016], código fornecido pelo autor, que utiliza uma solução baseada em programação dinâmica para identificar as inequações quadradas violadas durante o procedimento de Plano de Cortes.

Foram usadas 15 instâncias $G(n, m, p)$ durante os testes computacionais e como tratamos de algoritmos determinísticos, o algoritmo de $B\&C$ proposto por [Pinheiro *et al.*, 2016] executou apenas um vez a cada instância, enquanto o $B\&C$ proposto neste trabalho executou cada instância com um valor K (parâmetro da heurística se separação) variando entre 3 e $min\{|V_1|, |V_2|\}$, que representa o tamanho da menor partição da instância testada.

As Tabelas 6.4 e 6.5 apresentam para cada linha os resultados, associados à cada instância analisada, das estratégias de $B\&C$ comparadas nesta Seção. Suas colunas são divididas em três grupos. O primeiro grupo descreve as instâncias $G(n, m, p)$ utilizadas (**Instância**). Estas colunas descrevem as dimensões $n = |V_1|$, $m = |V_2|$ do problema e a coluna K, quando $K = 2^*$ representa os valores obtidos pela execução do $B\&C$ proposto por [Pinheiro *et al.*, 2016], já para K={3 .. min{n, m}} representa os valores obtidos pelas execuções do $B\&C$ proposto neste trabalho, além disso, o valor de K serve como parâmetro para a heurística de separação das inequações violadas que está embutida no algoritmo. O segundo grupo de colunas apresenta os resultados do Plano de Cortes até a resolução da raiz do modelo $\mathcal{F}_{P_4}$ restrito (**Raiz**) e o terceiro grupo apresenta os resultados dos algoritmos de $B\&C$ aqui comparados ($B\&C$). Estes dois últimos grupos são divididos nas colunas: LB que apresenta o melhor limite inferior encontrado; *cortes* indicando o número de inequações violadas que foram adicionadas ao modelo restrito e a coluna *tempo* que representa o tempo computacional, em segundos, da respectiva abordagem. O conjunto $B\&C$ ainda possui a coluna intitulada UB que apresenta o melhor limite superior encontrado e a coluna n° nós apresenta o número de nós resolvidos pela estratégia de

Tabela 6.4: Comparação entre o B&C de [Pinheiro *et al.*, 2016] com o B&C proposto, analisando todos os possíveis níveis das novas inequações. Instâncias do grupo BC_1.

Instância				**Raiz**			***B&C***				
n	m	p	K	LB	cortes	*tempo* (*s*)	LB	UB	cortes	n° nós	*tempo* (*s*)
5	7	0,5	2*	**6,4**	47	0,01	7	7	**49**	**3**	**0,01**
			3	**7**	100	0,01	7	7	**100**	**0**	**0,01**
			4	7	170	0,01	7	7	170	0	0,01
			5	7	213	0,01	7	7	213	0	0,01
5	7	0,6	2*	**7**	57	0,01	8	8	**75**	**7**	**0,02**
			3	**8**	92	0,01	8	8	**92**	**0**	**0,01**
			4	8	159	0,01	8	8	159	0	0,01
			5	8	174	0,01	8	8	174	0	0,01
6	8	0,5	2*	**8**	79	0,01	8	8	**79**	**0**	**0,01**
			3	**8**	169	0,02	8	8	**169**	**0**	**0,02**
			4	8	213	0,02	8	8	213	0	0,02
			5	8	394	0,02	8	8	394	0	0,02
			6	8	331	0,02	8	8	331	0	0,02
6	8	0,6	2*	**9,73**	108	0,02	11	11	**160**	**17**	**0,04**
			3	**11**	215	0,02	11	11	**215**	**0**	**0,02**
			4	11	358	0,02	11	11	358	0	0,02
			5	11	442	0,02	11	11	442	0	0,02
			6	11	509	0,02	11	11	509	0	0,02
7	11	0,5	2*	**11**	117	0,01	11	11	**117**	**0**	**0,01**
			3	**11**	208	0,02	11	11	**208**	**0**	**0,02**
			4	11	364	0,02	11	11	364	0	0,02
			5	11	416	0,02	11	11	416	0	0,02
			6	11	445	0,02	11	11	445	0	0,02
			7	11	506	0,02	11	11	506	0	0,02
7	11	0,6	2*	**15,4**	205	0,04	19	19	**329**	**24**	**0,17**
			3	**19**	545	0,04	19	19	**545**	**0**	**0,04**
			4	19	884	0,05	19	19	884	0	0,05
			5	19	973	0,05	19	19	973	0	0,05
			6	19	1189	0,06	19	19	1189	0	0,06
			7	19	1213	0,06	19	19	1213	0	0,06
6	12	0,5	2*	**12,09**	139	0,02	14	14	**181**	**8**	**0,04**
			3	**14**	330	0,02	14	14	**330**	**0**	**0,02**
			4	14	533	0,03	14	14	533	0	0,03
			5	14	419	0,03	14	14	419	0	0,03
			6	14	549	0,03	14	14	549	0	0,03
6	12	0,6	2*	**14,43**	202	0,03	17	17	**259**	**11**	**0,08**
			3	**17**	512	0,03	17	17	**512**	**0**	**0,03**
			4	17	875	0,04	17	17	875	0	0,04
			5	17	871	0,05	17	17	871	0	0,05
			6	17	1052	0,05	17	17	1052	0	0,05
6	20	0,5	2*	**20,14**	264	0,03	28	28	**733**	**146**	**1,3**
			3	24	934	0,12	28	28	1761	16	0,75
			4	26,07	1980	0,61	28	28	2307	3	0,71
			5	26,34	2160	0,61	28	28	2486	3	0,73
			6	**28**	2890	0,54	28	28	**2890**	**0**	**0,54**
6	20	0,6	2*	**24,04**	385	0,07	30	30	**973**	**107**	**1,6**
			3	28,26	1369	0,23	30	30	1583	3	0,27
			4	**30**	2156	0,5	30	30	**2156**	**0**	**0,5**
			5	30	2323	0,24	30	30	2323	0	0,24
			6	30	2799	0,26	30	30	2799	0	0,26

$B\&C$ até sua finalização.

É perceptível a melhora do algoritmo de $B\&C$ proposto neste trabalho comparado ao $B\&C$ proposto por [Pinheiro *et al.*, 2016] ($K = 2^*$). Na Tabela 6.5, que possui instâncias maiores, o ganho em média no tempo computacional da execução com $K = 7$ comparada a execução com $K = 2^*$ é de 97%, como exemplo, temos a instância $(n = 12; m = 20; p = 0,5)$ onde a execução $K = 2^*$ encontrou a solução ótima em 10.182 segundos e a execução $K = 7$ encontrou a solução ótima em apenas 87 segundos. Este ganho se deve a qualidade dos novos cortes adicionados no algoritmo de $B\&C$ proposto, levando com isto a um menor número de nós a serem resolvidos pelo algoritmo de *Branch-and-Bound* embutido.

A melhora obtida pela adição das inequações Fole, Escada e Grid também podem ser percebida pela resolução da raiz do modelo $\mathcal{F}_{P_4}$ restrito. A Tabela 6.5 apresenta um ganho médio de 29,6% no limite inferior (LB) quando comparadas as execuções $K = 7$ e $K = 2^*$. Sendo um problema de minimização quanto maior o limite inferior melhor a qualidade dos cortes adicionados.

6.5 Comparação entre $B\&B$, $B\&C$ e $B\&P$

Para a implementação das abordagens de *Branch-and-Price* ($B\&P$) propostas neste trabalho, utilizamos o *framework* de geração de colunas BaPCod[1] em conjunto com as bibliotecas *Boost C++ libraries* 1.54. As abordagens propostas são comparadas com o *Branch-and-Bound* do CPLEX aplicado à formulação proposta por [Amit, 2004] ($B\&B$) e com o algoritmo de *Branch-and-Cut* proposto na Seção 3.3.2 ($B\&C$).

Na Tabela 6.6 comparamos o resultado das estratégias $B\&B$, $B\&C$ e $B\&P$, sendo este último analisado em duas abordagens, o $B\&P$ - *1Col* que em sua fase de *pricing* exato constrói apenas uma coluna por iteração e o $B\&P$ - *MultiCol* que constrói múltiplas colunas por iteração, como descrito na Seção 4.3.2. Para isto utilizamos instâncias $G(n, m, p)$, com valores de $p = \{0,5; 0,6\}$. Nesta tabela temos cinco conjuntos de colunas. O conjunto *Instância*, dividido nas colunas $n = |V_1|$, $m = |V_2|$ e p. E as colunas $B\&B$, $B\&C$, $B\&P$ - *1Col* e $B\&P$ - *MultiCol* que são divididas nas colunas *Mint* que representa o melhor valor inteiro encontrado pelo procedimento, *tempo* que representa o tempo computacional em segundos para executar o procedimento e Δ que representa a razão entre a diferença dos valores de *upper bound* e *lower bound* sobre o valor do *upper*

[1]Mais informações em `https://realopt.bordeaux.inria.fr/?page_id=2`

Tabela 6.5: Comparação entre o B&C de [Pinheiro *et al.*, 2016] com o B&C proposto, analisando todos os possíveis níveis das novas inequações. Instâncias do grupo BC_2.

Instância				**Raiz**			***B&C***				
n	m	p	K	LB	cortes	*tempo* (*s*)	LB	UB	cortes	n° nós	*tempo* (*s*)
			2*	**26,74**	459	0,09	39	39	**1870**	**867**	**28,5**
			3	32	1196	0,24	39	39	4141	57	5
			4	36,55	2311	0,93	39	39	3537	7	2
			5	36,63	3162	1,11	39	39	4213	6	2,2
10	16	0,5	6	37,07	3298	1,23	39	39	4080	5	1,9
			7	**37,65**	3854	1,13	39	39	**3961**	**3**	**1,2**
			8	37,4	3438	1,21	39	39	3950	3	1,4
			9	37,29	4029	1,65	39	39	4350	3	1,9
			10	37,43	4400	1,91	39	39	4556	3	2,1
			2*	**32,03**	538	0,12	47	47	**3244**	**2191**	**141,9**
			3	38,4	1552	0,37	47	47	8592	142	29,6
			4	40	2844	0,98	47	47	9690	31	19,1
			5	41,37	3943	2,69	47	47	10354	28	19,2
10	16	0,5	6	42,28	4657	4,46	47	47	8841	15	14
			7	**43,75**	5189	4,45	47	47	**7106**	**8**	**8,1**
			8	43,44	5395	5,49	47	47	7918	12	13,4
			9	43,66	5915	5,93	47	47	8098	9	12,8
			10	43,08	5039	5,33	47	47	7387	13	13,5
			2*	**35,69**	636	0,16	56	56	**4598**	**9460**	**874,1**
			3	42,8	1776	0,49	56	56	25928	866	466,5
			4	49,77	3864	2,86	56	56	18390	81	75
			5	50,29	4697	3,96	56	56	16592	66	66,2
			6	50,7	4964	4,1	56	56	12972	35	31
12	17	0,5	7	**51,54**	5989	4,25	56	56	**10930**	**24**	**20,9**
			8	50,87	5842	4,6	56	56	12933	35	33,5
			9	51,71	7261	5,93	56	56	13121	30	33,5
			10	51,64	7676	8,4	56	56	13016	25	33,3
			11	50,99	7329	7,83	56	56	14542	31	39,8
			12	51,27	8002	8,24	56	56	15017	33	42,9
			2*	**39**	705	0,18	60	60	**6465**	**14373**	**2179,0**
			3	46,8	1921	0,51	60	60	29065	812	558,7
			4	50,5	3213	1,54	60	60	27750	90	141,9
			5	51,88	4659	6,13	60	60	22210	65	105
			6	54,42	6568	10,33	60	60	15291	29	41,4
12	17	0,6	7	**55,85**	7546	8,74	60	60	**14835**	**24**	**34,5**
			8	55,36	8057	7,4	60	60	15274	20	30,7
			9	55,79	8653	8,55	60	60	14132	20	33,6
			10	56,44	10319	14,96	60	60	14024	16	37,1
			11	56,2	9387	12,73	60	60	13701	14	31,6
			12	56,31	10844	17,69	60	60	14606	19	46,1
			2*	**41,69**	777	0,18	67	67	**7006**	**56848**	**10182,6**
			3	50	2350	0,69	67	67	57489	3358	4575,2
			4	58,58	4830	3,92	67	67	34755	173	322,9
			5	59,63	6821	7,52	67	67	31840	145	254
			6	60,48	7256	6,48	67	67	22157	81	123,172
12	20	0,5	7	**60,53**	8628	12,03	67	67	**20384**	**62**	**87,7**
			8	60,96	9069	13,85	67	67	22763	65	108,1
			9	61,04	9852	11,72	67	67	21063	67	109
			10	60,35	9324	15,57	67	67	23468	59	118,5
			11	60,85	12461	22,55	67	67	23272	62	143,7
			12	60,88	11543	21,24	67	67	23756	54	136,8

bound encontrado por cada procedimento.

Tabela 6.6: Comparação entre o $B\&B$, $B\&C$ e $B\&P$ para o BEP.

Instância			B&B				B&C				B&P			
n	m	p	*raiz*	*Mint*	tempo	Δ(%)	*raiz*	*Mint*	tempo	Δ(%)	*raiz*	*Mint*	tempo	Δ(%)
5	7	0,5	**7**	7	10	0	**7**	7	14	0	**7**	7	4	0
5	7	0,6	**8**	8	30	0	**8**	8	11	0	**8**	8	3	0
6	8	0,5	**8**	8	20	0	**8**	8	21	0	**8**	8	5	0
6	8	0,6	**11**	11	30	0	**11**	11	24	0	**11**	11	4	0
6	12	0,5	**14**	14	90	0	**14**	14	26	0	**14**	14	8	0
6	12	0,6	16,295	17	130	0	**17**	17	45	0	**17**	17	12	0
7	11	0,5	**11**	11	40	0	**11**	11	24	0	**11**	11	9	0
7	11	0,6	17,723	19	210	0	**19**	19	51	0	**19**	19	7	0
6	20	0,5	23,934	28	1990	0	26,343	28	729	0	**28**	28	24	0
6	20	0,6	27,322	30	1130	0	**30**	30	236	0	**30**	30	33	0
10	16	0,5	32,454	39	164620	0	37,648	39	620	0	**39**	39	50	0
10	16	0,6	37,923	47	62090	0	43,752	47	4083	0	**47**	47	113	0
16	30	0,5	80,987	168	TL	46,07	119,72	149	TL	15,44	**149**	149	40820	0
16	30	0,6	96,899	192	TL	44,81	139,193	162	TL	8,04	**162**	162	100210	0
20	23	0,5	77,913	167	TL	47,46	114,881	144	TL	15,56	**141,435**	143	508990	0
20	23	0,6	93,262	181	TL	42,6	135,91	157	TL	7,38	**157**	157	150600	0
20	35	0,5	117,46	271	TL	55,27	174,464	224	TL	19,88	**221,939**	224	1574340	0
20	35	0,6	140,858	275	TL	46,76	207,802	250	TL	14,76	**250**	250	1183560	0
24	40	0,5	160,973	411	TL	60,83	236,859	329	TL	27,99	**322,12**	326	TL	0,92
24	40	0,6	192	380	TL	49,47	282,69	337	TL	14,6	**337**	337	20303230	0

Podemos observar para estas instâncias, que os procedimentos $B\&P$ - *1Col* e $B\&P$ - *MultiCol* obtiveram os melhores tempos computacionais, encontrando 4 novas soluções ótimas. Os procedimentos $B\&B$ e $B\&C$ falharam nas 4 últimas instâncias devido a problemas com memória (EM) ou limite de tempo (LT) de seis horas ultrapassadas. O procedimento $B\&P$ - *MultiCol* se torna competitivo em termo de tempo computacional com o crescimento da dimensão da instância.

6.6 Comparação entre $B\&P$ e GRASP

Com o objetivo de identificarmos o número de iterações adotado da meta-heurística GRASP proposta, comparamos a mesma com o procedimento exato $Branch-and-Price$, cujos resultados foram apresentados anteriormente. Para cada instância, vinte execuções foram realizadas usando o GRASP, com a seguinte condição de parada: "encontrar o valor ótimo alcançado pelo procedimento exato". Para cada linha da Tabela 6.7, as primeiras três colunas apresentam os parâmetros das instâncias testadas, e as colunas restantes são divididas em dois grupos: $\boldsymbol{B\&P}$ e **GRASP**. No grupo $\boldsymbol{B\&P}$, coluna e^* apresenta o valor ótimo, e coluna *tempo* indica o tempo computacional (em milissegundos) gastos para resolver a instância. No grupo **GRASP**, coluna $iter_{avg}$ indica, em média, o número de iterações do algoritmo GRASP necessárias para encontrar o valor ótimo, e *tempo* representa o tempo de execução, em média, para encontrar esta solução.

Entre as 12 instâncias analisadas, GRASP obteve o valor ótimo em todas, requerendo em média um máximo de 25,15 iterações. A meta-heurística ainda apresentou uma boa eficiência computacional, como podemos observar na instância ($n = 10, m = 16, d =$

Tabela 6.7: Comparação entre $B\&P$ e GRASP.

Instância			***B&P***		**GRASP**	
p	n	m	e^*	*tempo* (ms)	$iter_{avg}$	*tempo* (ms)
0.5	5	7	7	20	4,85	0,15
	6	8	8	30	5,8	0,25
	6	12	14	50	3	0,2
	7	11	11	500	2,5	0,2
	6	20	28	40	**25,15**	2,7
	10	16	39	**1350**	9,15	**1,15**
0.6	5	7	8	30	9,3	0,3
	6	8	11	30	1,7	0,5
	6	12	17	80	5,75	0,35
	7	11	19	260	14,8	1
	6	20	30	120	3,3	0,3
	10	16	47	1195	24,1	2,85

$0,5$), onde uma média de tempo de 1,15 milissegundo foi necessária, enquanto o $B\&P$ gastou 1,35 segundo.

6.7 Comparação entre GVNS, ILS, GRASP e algoritmos da literatura

6.7.1 Resultados para as instâncias $G(n, m, p)$ e $P(G, q)$

Nesta Seção, comparamos nossas heurísticas com o resultado empírico dos métodos aproximativos e algoritmos heurísticos mencionados na Seção 2.3, usando as 80 instâncias $G(n, m, p)$ e 45 $P(G, q)$ mencionadas na Seção 6.1. Como o algoritmo 11-aproximado (11-Approx) [Amit, 2004] tem uma saída determinística, ele foi executado apenas uma vez para cada instância, enquanto o algoritmo 4-aproximado (4-Approx) [Ailon *et al.*, 2012], a heurística EDH [Sun *et al.*, 2013] e a heurística Bi-Force [Sun *et al.*, 2014] foram executadas vinte vezes para cada instância, bem como os métodos GVNS, ILS, e GRASP. Cada execução GRASP consiste de 30 iterações. A escolha para 30 iterações foi tirada da Tabela 6.7, ela mostra que o número médio máximo de iterações necessárias para o GRASP atingir a solução ótima em todos os cenários de entrada testados é $25,15$. Nos métodos GVNS e ILS, foram usados como critério de parada o tempo computacional médio do GRASP. O valor usado pelo parâmetro de aleatoriedade α na meta-heurística GRASP foi $0,5$, que apresentou os melhores resultados empíricos.

Para cada linha nas Tabelas 6.8-6.15, as colunas no grupo **Instâncias** apresentam as instâncias testadas. Note que nas Tabelas 6.13-6.15, que reportam os resultados para as instâncias $P(G, q)$, estas colunas descrevem as instâncias $G(n, m, p)$ usadas para obter as instâncias $P(G, q)$ correspondentes (veja Subseção 6.1.1). As colunas restantes são

Tabela 6.8: Comparação entre GVNS, ILS, GRASP, and os algoritmos da literatura usando instâncias $G(n, m, p)$ com $p = 0, 2$. Tempo computacional reportado em millisegundos.

Instância		**4-Approx**			**Bi-Force**			**GRASP**		**GVNS**		**ILS**		tempo para
n	m	e	e_{avg}	*tempo*	e	e_{avg}	*tempo*	e	e_{avg}	e	e_{avg}	e	e_{avg}	nossas heurísticas
74	108	1439	1481,0	0,5	1371	1371,0	992,2	1317	1317,0	1289	1321,6	**1273**	1284,7	717,9
100	100	1892	1931,9	0,7	1764	1764,0	1037,8	1685	1685,0	1677	1699,9	**1654**	1665,1	912,5
60	180	1953	1976,3	0,7	1879	1879,0	1189,2	1796	1796,0	1793	1819,2	**1744**	1757,9	1311,5
80	140	2055	2098,6	0,7	1992	1992,0	1104,6	1905	1905,0	1896	1913,6	**1848**	1864,2	1016,8
80	200	2918	2954,1	1,0	2835	2835,0	1452,9	2740	2740,0	2727	2740,0	**2661**	2673,9	2060,6
148	216	6137	6187,4	1,9	5982	5982,0	2149,2	5763	5763,0	5739	5770,3	**5674**	5693,8	4231,4
200	200	7673	7727,5	2,5	7452	7452,0	2588,2	7225	7225,0	7160	7216,0	**7124**	7153,5	5679,9
120	360	8154	8204,2	2,7	8115	8115,0	3254,8	7755	7755,0	7685	7768,8	**7632**	7651,1	10158,7
160	280	8541	8617,3	2,8	8357	8357,0	2948,9	8133	8133,0	8045	8101,5	**7987**	8003,8	8317,0
160	400	12375	12458,3	4,0	12209	12209,0	4366,8	11808	11808,0	11755	11839,5	**11679**	11695,4	16752,8

Tabela 6.9: Comparação entre GVNS, ILS, GRASP, e os algoritmos da literatura usando instâncias $G(n, m, p)$ com $p = 0,3$. Tempos computacionais reportados em milisegundos.

Instância		**4-Approx**			**Bi-Force**			**GRASP**		**GVNS**		**ILS**		tempo para
n	m	e	e_{avg}	*tempo*	e	e_{avg}	*tempo*	e	e_{avg}	e	e_{avg}	e	e_{avg}	nossas heurísticas
74	108	2297	2330,4	0,5	2159	2159,0	996,1	2023	2035,2	2012	2051,3	**2001**	2009,3	484,6
100	100	2931	2982,9	0,7	2806	2806,0	1041,5	2597	2610,5	2583	2617,5	**2575**	2583,8	561,7
60	180	3081	3135,8	0,7	3064	3064,0	1146,6	2787	2802,1	2792	2819,6	**2752**	2762,0	846,0
80	140	3248	3288,9	0,7	3113	3113,0	1147,4	2912	2931,2	2904	2951,9	**2894**	2902,9	688,2
80	200	4595	4647,5	1,0	4503	4503,0	1467,8	4211	4234,1	4207	4252,1	**4171**	4181,6	1430,8
148	216	9430	9485,2	2,0	9244	9244,0	2135,2	8741	8774,1	8727	8771,7	**8676**	8697,2	2998,6
200	200	11831	11897,0	2,6	11528	11528,0	2596,4	11009	11022,5	10976	11017,7	**10929**	10943,6	3716,1
120	360	12577	12660,5	2,8	12399	12399,0	3200,5	11809	11852,1	11779	11846,1	**11722**	11757,3	7415,3
160	280	13143	13215,8	2,9	12868	12868,0	2933,0	12304	12340,0	12312	12359,3	**12230**	12258,0	5903,4
160	400	18879	18975,7	4,3	18641	18641,0	4283,5	17865	17886,5	17864	17926,3	**17754**	17781,2	13085,5

Tabela 6.10: Comparação entre GVNS, ILS, GRASP, e os algoritmos da literatura usando instâncias $G(n, m, p)$ com $p = 0,5$. Tempos computacionais reportados em millisegundos.

Instância		**11-Approx**		**4-Approx**			**EDH**			**Bi-Force**			**GRASP**		**GVNS**		**ILS**		tempo para
n	m	e	*time*	e	e_{avg}	*tempo*	e	e_{avg}	*tempo*	e	e_{avg}	*tempo*	e	e_{avg}	e	e_{avg}	e	e_{avg}	nossas heurísticas
5	7	18	84,0	9	10,8	0,0	**7**	7,0	242,7	8	8,0	202,5	**7**	7,0	**7**	8,0	**7**	7,0	1,1
6	8	24	10,0	14	19,1	0,0	9	9,0	277,2	12	12,0	201,2	**8**	8,0	**8**	8,6	**8**	8,0	1,4
6	12	36	27,0	18	25,9	0,0	15	15,0	388,3	18	18,0	212,6	**14**	14,0	**14**	15,0	**14**	14,0	2,0
7	11	**11**	17,0	21	32,3	0,0	**11**	11,0	297,1	14	15,5	219,3	**11**	11,0	**11**	12,6	**11**	11,0	2,3
6	20	60	54,0	34	41,5	0,0	32	32,0	751,4	32	36,5	236,2	**28**	28,0	**28**	29,7	**28**	28,0	3,2
10	16	80	182,0	54	62,5	0,0	42	42,0	841,0	44	47,1	236,3	**39**	39,0	**39**	41,2	**39**	39,0	4,2
20	23	230	4424,0	198	208,1	0,1	153	153,0	10552,2	161	170,7	276,8	**143**	145,1	145	148,4	**143**	145,1	11,2
16	30	240	3707,0	202	212,9	0,1	159	159,0	14289,3	170	176,6	278,9	**149**	150,8	**149**	157,1	**149**	152,3	12,1
20	35	350	11229,0	299	315,1	0,1	250	250,0	49474,0	267	267,0	295,4	**224**	226,9	**224**	230,4	225	227,4	18,9
24	40	480	37337,0	423	444,0	0,1	340	340,0	147634,4	385	385,0	317,4	**326**	330,0	329	338,1	**326**	331,0	25,8
30	41	644	124221,0	549	579,8	0,1	450	450,0	312020,4	496	496,0	329,5	422	424,5	428	439,9	**419**	425,1	34,7
28	46	615	75970,0	577	606,7	0,1	478	478,0	551600,6	482	482,0	338,0	449	455,3	453	460,8	**447**	452,6	35,0
30	50	750	192353,0	672	693,5	0,1	579	579,0	719236,0	622	622,0	356,3	521	528,5	528	539,8	**520**	524,8	47,5
35	45	788	337226,0	724	744,1	0,1	590	590,0	705176,9	609	618,8	360,7	**565**	573,6	**565**	579,8	**565**	570,6	48,1
40	40	800	192658,0	751	773,0	0,1	616	616,0	643014,0	631	647,4	362,8	572	577,0	570	584,0	**569**	573,5	46,5
37	54	999	839420,0	928	952,3	0,2	770	770,0	1600766,2	784	797,0	402,8	736	741,7	744	756,7	**735**	742,2	59,3
50	50	*	*	1148	1194,8	0,2	1002	1002,0	4067372,7	1035	1035,0	429,9	928	935,3	932	940,8	**922**	930,3	75,4
30	90	*	*	1233	1265,6	0,2	1050	1050,0	6013013,1	1176	1176,0	515,9	998	1007,9	1004	1019,8	**991**	1003,5	94,7
40	70	*	*	1279	1329,1	0,2	1124	1124,0	5251013,9	1170	1170,0	478,0	1046	1055,7	1052	1065,5	**1043**	1050,6	89,8
40	100	*	*	1876	1910,4	0,3	1636	1636,0	20429977,9	1752	1752,0	613,2	1536	1546,3	1544	1562,3	**1528**	1540,3	144,9

Tabela 6.11: Comparação entre GVNS, ILS, GRASP, e os algoritmos da literatura usando as instâncias $G(n, m, p)$ com $p = 0,6$. Tempos computacionais reportados em millisegundos.

Instância		**11-Approx**		**4-Approx**			**EDH**			**Bi-Force**			**GRASP**		**GVNS**		**ILS**		tempo para
n	m	e	*time*	e	e_{avg}	*tempo*	e	e_{avg}	*tempo*	e	e_{avg}	*tempo*	e	e_{avg}	e	e_{avg}	e	e_{avg}	nossas heurísticas
5	7	21	50,0	12	13,3	0,0	10	10,0	259,6	9	9,3	197,5	**8**	8,0	**8**	9,1	**8**	8,4	1,1
6	8	29	17,0	17	21,0	0,0	**11**	11,0	318,0	**11**	11,0	200,2	**11**	11,0	**11**	12,0	**11**	11,0	1,4
6	12	43	27,0	25	31,9	0,0	20	20,0	448,7	**17**	17,0	211,7	**17**	17,0	**17**	17,9	**17**	17,0	1,9
7	11	46	43,0	28	31,7	0,0	23	23,0	437,8	20	21,1	209,0	**19**	19,0	**19**	19,3	**19**	19,0	2,0
6	20	72	49,0	45	49,3	0,0	**30**	30,0	832,9	**30**	30,6	235,5	**30**	30,0	**30**	31,8	**30**	30,0	3,0
10	16	96	177,0	66	75,9	0,0	52	52,0	1236,1	51	53,2	238,2	**47**	47,0	**47**	49,2	**47**	47,0	3,5
20	23	276	2934,0	223	242,1	0,1	175	175,0	27197,7	166	168,8	278,6	**157**	158,1	160	164,1	**157**	158,6	9,6
16	30	288	3368,0	236	248,6	0,1	172	172,0	42950,1	174	174,0	277,9	**162**	162,6	165	169,0	**162**	162,7	10,5
20	35	420	10891,0	347	366,4	0,1	280	280,0	152694,2	268	268,0	294,1	**250**	250,3	**250**	255,4	**250**	250,3	16,6
24	40	576	33725,0	483	506,2	0,1	347	347,0	385557,1	339	339,2	316,8	**337**	337,0	338	338,8	**337**	337,1	22,8
30	41	738	82766,0	606	652,9	0,1	470	470,0	1115061,2	459	459,0	329,2	**449**	449,4	**449**	453,8	**449**	450,5	28,2
28	46	773	69028,0	650	676,1	0,1	496	496,0	1354086,1	493	493,0	328,5	**474**	474,2	**474**	475,9	**474**	474,5	30,7
30	50	900	165799,0	777	796,5	0,1	557	557,0	1788833,3	551	551,0	356,7	**548**	548,0	**548**	549,5	**548**	548,3	35,6
35	45	945	213577,0	808	847,4	0,1	620	620,0	2353717,4	606	606,0	350,4	**600**	600,2	**600**	601,8	**600**	600,3	39,8
40	40	960	172651,0	799	855,3	0,1	640	640,0	2780579,2	612	612,0	350,1	**606**	606,0	**606**	607,7	**606**	606,0	38,3
37	54	1199	662523,0	1009	1072,8	0,2	795	795,0	6608135,0	792	792,0	383,3	**771**	772,7	773	774,8	**771**	772,4	49,3
50	50	*	*	1316	1368,0	0,2	998	998,0	13201877,4	988	988,0	426,8	**972**	972,8	**972**	976,8	**972**	972,1	58,0
30	90	*	*	1343	1430,4	0,2	1070	1070,0	20598297,2	1052	1052,0	502,9	**1026**	1026,5	**1026**	1037,4	**1026**	1026,0	78,2
40	70	*	*	1472	1511,7	0,2	1088	1088,0	14576875,3	1092	1092,0	458,3	**1071**	1071,0	**1071**	1072,0	**1071**	1071,0	74,4
40	100	*	*	2086	2139,3	0,3	*	*	*	1582	1582,0	582,9	**1562**	1562,0	**1562**	1565,1	**1562**	1562,0	125,8

Tabela 6.12: Comparação entre GVNS, ILS, GRASP, e os algoritmos da literatura usando as instâncias $G(n, m, p)$ com $p = 0,7$. Tempos computacionais reportados em millisegundos.

Instância		**11-Approx**		**4-Approx**			**EDH**			**Bi-Force**			**GRASP**		**GVNS**		**ILS**		tempo para
n	m	e	*time*	e	e_{avg}	*tempo*	e	e_{avg}	*tempo*	e	e_{avg}	*tempo*	e	e_{avg}	e	e_{avg}	e	e_{avg}	nossas heurísticas
5	7	**7**	601,0	10	14,0	0,0	**7**	7,0	293,2	**7**	7,0	194,3	**7**	7,0	10	10,0	**7**	7,0	1,0
6	8	34	11,0	14	19,0	0,0	13	13,0	357,7	**12**	12,0	198,6	**12**	12,0	**12**	12,0	**12**	12,0	1,1
6	12	50	52,0	30	34,0	0,0	22	22,0	471,0	22	22,0	211,1	**20**	20,0	**20**	21,3	**20**	20,0	1,6
7	11	54	72,0	27	37,0	0,0	22	22,0	597,4	22	22,0	209,6	**21**	21,0	**21**	21,9	**21**	21,0	1,7
6	20	84	57,0	39	53,3	0,0	32	32,0	1294,2	31	33,7	233,2	**28**	28,0	**28**	28,0	**28**	28,0	2,5
10	16	112	158,0	78	87,6	0,0	**48**	48,0	2272,7	**48**	48,0	228,8	**48**	48,0	**48**	48,0	**48**	48,0	2,8
20	23	322	2366,0	226	267,0	0,0	129	129,0	50597,2	129	129,0	271,3	**128**	128,0	**128**	128,8	**128**	128,0	7,9
16	30	336	2709,0	241	272,0	0,0	**140**	140,0	67906,0	**140**	140,0	276,1	**140**	140,0	**140**	141,4	**140**	140,0	9,0
20	35	490	7117,0	316	397,9	0,1	**186**	186,0	221932,1	**186**	186,0	289,5	**186**	186,0	**186**	186,5	**186**	186,0	13,5
24	40	672	20692,0	525	554,5	0,1	288	288,0	634627,8	288	288,0	306,2	**286**	286,0	**286**	286,0	**286**	286,0	15,9
30	41	861	60889,0	693	731,3	0,1	**350**	350,0	1363289,2	**350**	350,0	328,6	**350**	350,0	**350**	350,5	**350**	350,0	22,2
28	46	902	43665,0	727	766,8	0,1	**386**	386,0	1820064,3	**386**	386,0	328,7	**386**	386,0	**386**	386,0	**386**	386,0	20,9
30	50	1050	96989,0	814	881,5	0,1	**450**	450,0	2794693,6	**450**	450,0	344,4	**450**	450,0	**450**	450,0	**450**	450,0	25,2
35	45	1103	130104,0	882	929,8	0,1	**472**	472,0	3646996,5	**472**	472,0	348,9	**472**	472,0	**472**	472,0	**472**	472,0	25,8
40	40	1120	128531,0	906	968,6	0,1	**480**	480,0	3625476,3	**480**	480,0	346,2	**480**	480,0	**480**	480,0	**480**	480,0	24,3
37	54	1399	282157,0	975	1187,7	0,1	**593**	593,0	7982254,9	**593**	593,0	380,9	**593**	593,0	**593**	593,0	**593**	593,0	34,8
50	50	*	*	1396	1481,8	0,2	**748**	748,0	15653172,8	**748**	748,0	419,6	**748**	748,0	**748**	748,0	**748**	748,0	43,3
30	90	*	*	1453	1592,2	0,2	*	*	*	810	810,0	499,4	**808**	808,0	**808**	809,3	**808**	808,8	57,0
40	70	*	*	1601	1667,6	0,2	*	*	*	**840**	840,0	462,7	**840**	840,0	**840**	840,0	**840**	840,0	50,4
40	100	*	*	2189	2377,3	0,3	*	*	*	**1190**	1190,0	574,2	**1190**	1190,0	**1190**	1190,0	**1190**	1190,0	80,8

Tabela 6.13: Comparação entre GVNS, ILS, GRASP, e os algortimos da literatura usando instâncias $P(G, q)$ com $q = 0,1$. Tempos computacionais reportados em millisegundos.

Instância			**4-Approx**			**EDH**			**Bi-Force**			**GRASP**		**GVNS**		**ILS**		tempo para
n	m	p	e	e_{avg}	*tempo*	e	e_{avg}	*tempo*	e	e_{avg}	*tempo*	e	e_{avg}	e	e_{avg}	e	e_{avg}	nossas heurísticas
37	54	0,5	365	545,2	0,1	173	173	201286,5	248	248,0	406,9	**172**	172,0	**172**	186,7	**172**	172,0	60,5
37	54	0,6	555	1191,8	0,1	210	210	7490898,7	201	206,5	356,9	**198**	198,0	**198**	202,4	**198**	198,0	30,8
37	54	0,7	464	1503,3	0,1	189	189	15506319,3	**188**	188,0	346,5	**188**	188,0	**188**	188,0	**188**	188,0	21,5
50	50	0,5	507	705,7	0,2	237	237	425261,3	554	554,0	455,1	**231**	231,0	**231**	237,0	**231**	231,0	79,3
50	50	0,6	575	1429,9	0,1	244	244	13305566,2	**241**	241,0	395,3	**241**	241,0	**241**	241,0	**241**	241,0	41,8
50	50	0,7	1792	2004,9	0,1	*	*	*	**226**	226,0	422,0	**226**	226,0	**226**	226,0	**226**	226,0	20,7
30	90	0,5	511	704,4	0,2	240	240	902917,9	562	562,0	556,9	**238**	238,0	**238**	238,0	**238**	238,0	100,3
30	90	0,6	590	1537,6	0,1	*	*	*	**254**	260,3	498,0	**254**	254,0	**254**	254,0	**254**	254,0	47,2
30	90	0,7	632	2040,8	0,1	*	*	*	**260**	260,0	488,9	**260**	260,0	**260**	260,0	**260**	260,0	30,0
40	70	0,5	551	844,3	0,2	253	253	1491235,8	324	324,0	508,1	**245**	245,0	**245**	246,9	**245**	245,0	88,1
40	70	0,6	667	1768,6	0,1	*	*	*	264	264,9	410,9	**262**	262,0	**262**	262,1	**262**	262,0	37,3
40	70	0,7	2207	2296,7	0,1	*	*	*	**253**	253,0	444,6	**253**	253,0	**253**	253,0	**253**	253,0	24,4
40	100	0,5	844	992,5	0,2	**358**	358	2064662,5	735	735,0	657,5	**358**	358,0	**358**	400,0	**358**	358,0	141,8
40	100	0,6	866	2740,6	0,2	*	*	*	**349**	352,4	522,8	**349**	349,0	**349**	349,0	**349**	349,0	65,8
40	100	0,7	2915	3104,8	0,2	*	*	*	**351**	351,0	513,2	**351**	351,0	**351**	351,0	**351**	351,0	46,1

Tabela 6.14: Comparação entre GVNS, ILS, GRASP, e os algoritmos da literatura usando instâncias $P(G, q)$ com $q = 0,2$. Tempos computacionais em millisegundos.

Instância			**4-Approx**			**EDH**			**Bi-Force**			**GRASP**		**GVNS**		**ILS**		tempo para
n	m	p	e	e_{avg}	*tempo*	e	e_{avg}	*tempo*	e	e_{avg}	*tempo*	e	e_{avg}	e	e_{avg}	e	e_{avg}	nossas heurísticas
37	54	0,5	611	669,3	0,1	396	396,0	430444,4	560	560,0	402,3	**378**	378,1	**378**	386,8	**378**	378,0	75,8
37	54	0,6	719	1108,3	0,1	404	404,0	6511528,8	**386**	388,7	399,6	**386**	386,0	**386**	386,0	**386**	386,0	40,9
37	54	0,7	848	1290,4	0,1	*	*	*	**383**	383,0	361,2	**383**	383,0	**383**	386,4	**383**	383,0	30,3
50	50	0,5	764	828,5	0,2	453	453,0	813407,3	564	564,0	463,8	**438**	438,0	**438**	443,5	**438**	438,0	92,4
50	50	0,6	877	1365,1	0,1	469	469,0	11482608,2	451	451,0	436,2	**450**	450,0	**450**	450,3	**450**	450,0	47,8
50	50	0,7	1576	1776,2	0,1	*	*	*	**443**	443,0	422,2	**443**	443,0	**443**	443,0	**443**	443,0	30,6
30	90	0,5	802	893,2	0,2	502	502,0	1810867,8	751	751,0	572,0	**484**	484,1	**484**	491,1	**484**	484,0	119,2
30	90	0,6	1070	1589,3	0,2	*	*	*	**490**	490,0	520,1	**490**	490,0	**490**	490,0	**490**	490,0	58,8
30	90	0,7	1083	1767,4	0,1	*	*	*	**508**	508,0	489,1	**508**	508,0	**508**	508,0	**508**	508,0	43,3
40	70	0,5	790	955,0	0,2	523	523,0	2105320,7	780	780,0	510,6	**499**	500,5	500	505,5	**499**	499,4	101,2
40	70	0,6	1160	1658,2	0,2	*	*	*	515	515,9	431,9	**512**	512,0	**512**	512,0	**512**	512,0	51,5
40	70	0,7	1798	1944,2	0,1	*	*	*	**512**	512,0	450,1	**512**	512,0	**512**	512,0	**512**	512,0	35,3
40	100	0,5	1254	1321,5	0,3	761	761,0	5218230,4	1210	1210,0	690,5	**761**	761,0	**761**	790,7	**761**	761,0	191,2
40	100	0,6	2366	2531,6	0,2	*	*	*	**724**	724,0	570,8	**724**	724,0	**724**	724,0	**724**	724,0	84,5
40	100	0,7	2282	2701,5	0,2	*	*	*	**699**	699,0	540,0	**699**	699,0	**699**	699,0	**699**	699,0	65,1

Tabela 6.15: Comparação entre GVNS, ILS, GRASP, e os algoritmos da literatura usando instâncias $P(G, q)$ com $q = 0,3$. Tempos computacionais em millisegundos.

Instância			**4-Approx**			**EDH**			**Bi-Force**			**GRASP**		**GVNS**		**ILS**		tempo para
n	m	p	e	e_{avg}	*tempo*	e	e_{avg}	*tempo*	e	e_{avg}	*tempo*	e	e_{avg}	e	e_{avg}	e	e_{avg}	nossas heurísticas
37	54	0,5	727	757,1	0,1	561	561	583201,1	634	634,0	411,5	527	529,3	**526**	534,1	**526**	526,5	79,7
37	54	0,6	758	1090,9	0,1	523	523	5891264,7	511	511,0	415,6	**509**	509,0	**509**	510,0	**509**	509,0	42,8
37	54	0,7	924	1157,3	0,1	*	*	*	**554**	554,0	399,4	**554**	554,0	**554**	554,0	**554**	554,0	38,8
50	50	0,5	903	964,8	0,2	683	683	1252100,8	812	812,0	457,8	**655**	655,3	**655**	659,7	**655**	655,0	108,8
50	50	0,6	1049	1334,5	0,2	681	681	9980411,6	656	657,4	447,2	**653**	653,0	**653**	653,6	**653**	653,0	56,0
50	50	0,7	1427	1561,4	0,1	*	*	*	**668**	668,0	410,8	**668**	668,0	**668**	668,0	**668**	668,0	39,3
30	90	0,5	991	1031,4	0,2	751	751	2344404,7	978	978,0	576,7	**711**	715,2	**711**	728,3	**711**	711,3	133,1
30	90	0,6	1251	1496,6	0,2	717	717	21327270,2	775	775,0	510,9	**701**	701,0	**701**	701,8	**701**	701,0	71,8
30	90	0,7	1327	1575,8	0,2	*	*	*	**739**	739,0	496,7	**739**	739,0	**739**	739,8	**739**	739,0	59,0
40	70	0,5	1042	1094,4	0,2	788	788	2627842,5	909	909,0	493,6	753	756,0	754	759,9	**752**	753,6	114,8
40	70	0,6	1210	1538,4	0,2	738	738	18335133,7	731	731,0	486,9	**729**	729,0	**729**	730,8	**729**	729,0	55,8
40	70	0,7	1609	1714,3	0,2	*	*	*	**757**	757,0	448,4	**757**	757,0	**757**	757,0	**757**	757,0	53,2
40	100	0,5	1414	1466,6	0,3	1116	1116	6628335,4	1249	1249,0	677,8	**1033**	1034,4	**1033**	1044,6	**1033**	1033,0	217,3
40	100	0,6	1856	2237,8	0,2	*	*	*	1149	1149,0	582,9	**1082**	1082,0	**1082**	1085,6	**1082**	1082,0	102,3
40	100	0,7	2085	2404,3	0,2	*	*	*	**1080**	1080,0	572,8	**1080**	1080,0	**1080**	1080,0	**1080**	1080,0	86,6

divididas nos seguintes grupos: **11-Approx**, **4-Approx**, **EDH**, **Bi-Force**, **GVNS**, **ILS** e **GRASP**. No grupo **11-Approx**, a coluna e indica os valor da solução encontrada, e *tempo* o tempo computacional em millisegundos. Em todos os outros grupos, as colunas rotuladas e e e_{avg} indicam, respectivamente, o melhor e o valor médio das soluções encontradas para cada método. Nos grupos **4-Approx**, **EDH** e **Bi-Force**, as colunas rotuladas *tempo* representam o tempo computacional médio (em millisegundos) gastos por cada método correspondente. A última coluna, rotulada "tempo para nossas heurísticas", apresenta os tempos computacionais usados para GVNS, ILS, e GRASP, expressados em millisegundos. A melhor solução em uma linha é marcada em negrito. As colunas **11-Approx** e **EDH** não são apresentadas nas Tabelas 6.8-6.9 pois estes métodos não foram capazes de resolver nenhuma instância considerada nestas tabelas devido à quantidade de memória alocada ou ao tempo de execução excedido em mais de 6 horas. Pelas mesmas razões, a coluna **11-Approx** não é apresentada nas Tabelas 6.13-6.15.

Os algoritmos 11-aproximado e a heurística EDH falharam a execução para instâncias grandes (marcado com *), devido à quatidade de memória alocada ou ao tempo de execução excedido em mais de 6 horas. O algoritmo ILS obteve os melhores resultados médios na maioria das instâncias. Quando comparados à heurística Bi-Force, nossas metaheurísticas ILS, GVNS e GRASP conseguem obter melhorias de 8,8%, 6,91% e 8,08% nos valores médios de suas soluções, respectivamente, com tempo computaconal menor na maioria das instâncias — em detalhes, nossas heurísticas são mais lentas que Bi-Force apenas para instâncias $G(n, m, p)$ com $p \in \{0,2; 0,3\}$.

6.7.2 Resultados para as instâncias obtidas de dados biológicos

Esta Seção apresenta a comparação entre GRASP, GVNS, ILS e Bi-Force usando instâncias grandes obtidas de dados biológicos. Os resultados obtidos pelos quatro métodos são apresentados nas Tabelas 6.16-6.18. Nestas tabelas, a coluna **Instância** determina o tamanho das instâncias testadas, e as outras quatro colunas **Bi-Force**, **GRASP**, **VNS** e **ILS** apresentam o custo médio das soluções (e_{avg}) e o custo da melhor solução (e) encontrada por cada método em 20 execuções independentes. Para cada linha nestas tabelas, a coluna **tempo (s)** é o valor médio do tempo computacional gasto pelo Bi-Force para resolver a correspondente instância, valor este utilizado como condição de parada para nossas heurísticas.

Podemos observar nas Tabelas 6.16-6.18 que nossas heurísticas são muito superiores ao Bi-Force em relação ao melhor valor e valor médio nos custos das soluções encontradas.

Tabela 6.16: Comparação entre GVNS, ILS, GRASP e Bi-Force usando as instâncias obtidas de dados biológicos (limite no primeiro quadrante).

Instância		**Bi-Force**		**GRASP**		**VNS**		**ILS**		
n	m	e	e_{avg}	e	e_{avg}	e	e_{avg}	e	e_{avg}	tempo (s)
15923	6	23833	23833,0	**2127**	2127,0	**2127**	2127,0	**2127**	2127,0	1052,0
22283	42	233710	233710,0	**88064**	88064,0	**88064**	88064,0	**88064**	88064,0	2052,1
12600	84	261643	261643,0	**135621**	135621,0	144262	144284,0	**135621**	136810,7	909,3
8799	122	267547	267547,0	**122811**	122811,0	**122811**	126512,6	**122811**	123714,2	351,0
12488	87	268539	268539,0	**92396**	92396,0	138057	138057,0	**92396**	94883,8	909,7
12626	110	70988	70988,0	**34290**	34290,0	**34290**	34290,0	**34290**	34290,0	711,8
12488	150	456231	456231,0	**177887**	177887,0	185909	211488,6	**177887**	179423,9	621,5
15923	154	611991	611991,0	**148687**	148687,0	157894	158482,0	**148687**	156359,5	997,5
22625	123	**17**	17,0	**17**	17,0	**17**	17,0	**17**	17,0	1796,5

Tabela 6.17: Comparação entre GVNS, ILS, GRASP e Bi-Force usando as instâncias obtidas de dados biológicos (limite no segundo quadrante).

Instância		**Bi-Force**		**GRASP**		**VNS**		**ILS**		
n	m	e	e_{avg}	e	e_{avg}	e	e_{avg}	e	e_{avg}	tempo (s)
15923	6	**2668**	4593,6	**2668**	2668,0	**2668**	2668,0	**2668**	2668,0	1403,5
22283	42	330717	332350,2	**77451**	85578,5	95943	116664,3	**77451**	84972,3	3199,5
12600	84	265759	326351,8	182378	188630,1	182031	200966,8	**163153**	165282,9	1569,6
8799	122	194146	266366,4	143774	149601,2	147093	179673,2	**123790**	125250,2	681,5
12488	87	185357	293356,0	**87975**	91094,8	92233	98905,5	**87975**	89976,0	1172,5
12626	110	634949	634949,0	**59139**	60144,8	63935	72076,2	**59139**	63583,9	844,6
12488	150	344246	542305,0	220035	228473,8	216270	260418,0	**188762**	191428,7	1092,2
15923	154	636634	640659,6	**131820**	134833,1	137782	140326,9	136574	138644,2	1975,8
22625	123	788149	788149,0	**338575**	340226,2	350857	361979,5	**338575**	342347,7	3016,9

Tabela 6.18: Comparação entre GVNS, ILS, GRASP e Bi-Force usando as instâncias obtidas de dados biológicos (limite no terceiro quadrante).

Instância		**Bi-Force**		**GRASP**		**VNS**		**ILS**		
n	m	e	e_{avg}	e	e_{avg}	e	e_{avg}	e	e_{avg}	tempo (s)
15923	6	1912	3549,8	**664**	664,0	**664**	664,0	**664**	664,0	1536,6
22283	42	**55224**	55224,0	58652	59963,0	78481	80957,0	**55224**	55224,0	33469,6
12600	84	122359	122359,0	150970	153487,9	146342	159709,0	**121532**	122634,3	3733,9
8799	122	86650	86650,0	102824	106608,9	89157	110434,0	**86014**	86234,6	2617,5
12488	87	70143	70143,0	**70068**	70933,3	79835	83642,2	**70068**	70497,0	5163,6
12626	110	59263	59263,0	57061	57684,5	61774	65871,1	**56697**	57840,4	6176,2
12488	150	169082	169082,0	178405	190480,2	171841	193428,4	**162413**	162854,3	5060,3
15923	154	105397	105397,0	125563	130180,2	148124	148139,0	**105209**	107520,8	7717,8
22625	123	**194163**	194163,0	257383	262893,2	268614	299922,8	195198	199670,3	37198,3

As melhorias na qualidade da solução são especialmente notadas para as instâncias na Tabela 6.16 (limite no primeiro quadrante). Por exemplo, para as instâncias com $n = 12626$ e $m = 110$, GRASP reduziu em 52% o custo da solução encontrada pelo Bi-Force. Além disto, o Bi-Force encontra as melhores soluções conhecidas em apenas 4 instâncias de 27, e para muitas instâncias de péssima qualidade as soluções geradas são formadas por um único *bicluster* contendo todos os vértices (valores sublinhados na coluna **Bi-Force**). Finalmente, nossas heurísticas têm comportamento semelhante na Tabela 6.16, enquanto nas outras instâncias o algoritmo ILS é muito superior, como ocorre nas instâncias $G(n, m, p)$ e $P(G, q)$.

6.8 Análise de Sensibilidade

Nesta Seção apresentamos uma análise de sensibilidade dos principais componentes de nossas heurísticas. Foram estudados diferentes cenários com respeito as estruturas de vizinhança propostas, assim como o impacto da matriz de edições sobre o tempo computacional. Por uma questão de simplicidade, estas análises foram restritas a heurística GRASP descrita na Seção 5.6.

6.8.1 Impacto das estruturas de vizinhança

Para avaliar as estruturas de vizinhaça, comparamos três diferentes configurações para a heurística GRASP:

- **Configuração 1: Todas as vizinhanças.** GRASP como descrito na Seção 5.6, usando todas as vizinhanças na fase de busca local.
- **Configuração 2: Move-Vértice.** GRASP usando apenas Move-Vértice na fase de busca local, ou seja, executando o procedimento VND (Figura 5.5) com $k_{max} = 1$.
- **Configuração 3: União-Bicluster e Quebra-Bicluster.** GRASP usando apenas União-Bicluster e Quebra-Bicluster na fase de busca local. Para fazê-lo, basta trocar o comando $k \leftarrow 1$ por $k \leftarrow 2$ nas linhas 2 e 6 do procedimento VND, e executar este procedimento com $k_{max} = 3$.

A Tabela 6.19 apresenta a média dos resultados obtidos pela heurística GRASP nas três configurações descritas anteriormente, considerando todas as instâncias $G(n, m, p)$ e $P(G, q)$. Para cada linha desta tabela, a coluna **Grupo de Instâncias** determina o

Tabela 6.19: Impacto das estruturas de vizinhança no GRASP: média do número de edições e do tempo computacional em diferente configurações.

Grupo de Instâncias	***Todas vizinhanças***		***Move-Vértice***		***União-Bicluster e Quebra-Bicluster***	
	e_{avg}	*tempo* (ms)	e_{avg}	*tempo* (ms)	e_{avg}	*tempo* (ms)
$G(n,m,p), p = 0,2$	**5012,7**	5115,9	5492,5	213,8	5033,9	4537,7
$G(n,m,p), p = 0,3$	**7648,8**	3713,0	7824,9	2129,9	9834,4	861,0
$G(n,m,p), p = 0,5$	**440,3**	37,9	443,2	34,3	506,4	10,2
$G(n,m,p), p = 0,6$	**456,1**	31,5	460,4	26,6	462,3	10,1
$G(n,m,p), p = 0,7$	**359,7**	22,1	374,2	19,0	359,8	8,7
$P(G,q), q = 0,1$	**255,1**	55,7	278,5	39,7	260,3	25,8
$P(G,q), q = 0,2$	**511,3**	71,2	533,2	57,8	520,2	23,8
$P(G,q), q = 0,3$	**744,1**	83,9	762,4	70,7	821,9	22,7

grupo de instâncias consideradas na linha, enquanto as colunas ***Todas Vizinhanças***, ***Move-Vértice*** e ***União-Bicluster e Quebra-Bicluster*** apresentam o número médio de edições (colunas rotulados por e_{avg}) e a média do tempo computacional (colunas rotuladas por *tempo* (ms)) obtida pelo GRASP na configuração correspondente. A melhor média no número de edições em uma linha é marcado em negrito. Logo, é possível perceber que as melhores médias são obtidas pelo GRASP quando o procedimento usa todas as estruturas de vizinhança. O tempo de execução desta configuração é maior que as outras mas continua razoável, sendo a maior média de tempo abaixo de 6 segundos.

6.8.2 Impacto da Matriz de Edições

Para demonstrar a performance da estrutura de dados auxiliar proposta, a matriz de edições, comparamos o tempo de execução do GRASP com e sem o uso da matriz. Nesta comparação, consideramos todas a instâncias $G(n,m,p)$ com $p \in \{0,2; 0,3\}$, que são as maiores instâncias deste tipo. Quando não usamos a matriz, os movimentos de vizinhança são avaliados de maneira natural, sem o uso de qualquer memória auxiliar. Por exemplo, sem a matriz, o custo da nova solução G'' obtida de uma solução anterior G' pela movimentação de um vértice $v \in V_1$ do *bicluster* B para o *bicluster* B' pode ser computado da seguinte maneira:

$$custo(G'') = custo(G') + \sum_{\substack{u \in V(B) \\ u \in V_2}} w(v,u) - \sum_{\substack{u \in V(B') \\ u \in V_2}} w(v,u)$$

Os tempos computacionais são descritos na Figura 6.1. Nesta figura, o eixo X

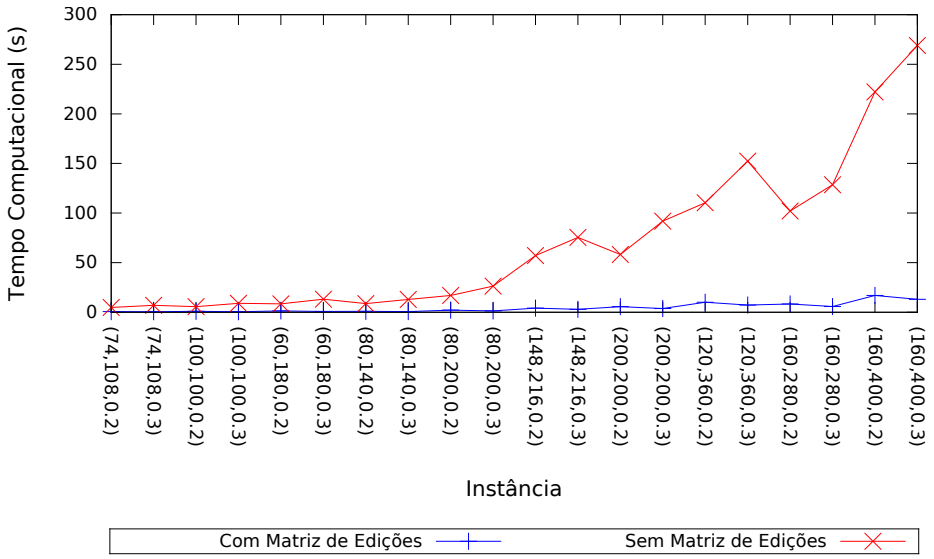


Figura 6.1: Impacto da matriz de edições no tempo computacional do GRASP.

contém as instâncias, cada uma representa pela tripla (n, m, p), e o eixo Y contém o tempo computacional em segundos. Fica claro na figura que os tempos são muito menores usando a matriz de edições. Por exemplo para a instância $(160; 400; 0, 3)$, os tempos computacionais com e sem a matriz são, respectivamente, 13 e 271 segundos. O impacto da matriz nas grandes instâncias é notável e aplicá-la em nossas heurísticas as torna competitivas em relação aos algoritmos existentes em termos de tempo computacional.

Capítulo 7

Considerações finais e trabalhos futuros

Neste trabalho abordamos o Problema de Edição de *Biclusters*, que procura transformar um grafo bipartido de entrada em uma união de subgrafos bipartidos completos disjuntos em vértices pela edição do menor número de arestas possíveis.

Devido à falta de experimentos na literatura, propomos e analisamos um algoritmo para geração de instâncias bipartidas aleatórias. Usando a métrica "número de arestas editadas para alcançar o ótimo", identificamos que as instâncias mais difíceis são geradas quando se usa a densidade $p \in \{0,5; 0,6; 0,7\}$, sobre o modelo de algoritmo aleatório de Gilbert. Também geramos grafos aleatórios com um modelo simples de perturbação, onde se realizam edições aleatórias em um grafo formado por *biclusters*. Finalmente, adaptamos dados biológicos do mundo real com o objetivo de criar instâncias de grande porte para o PEB.

Com o objetivo de reduzir a dificuldade das instâncias testadas, regras de redução foram definidas para PEB, aplicando reduções no tamanho do grafo de entrada sem comprometer a otimalidade da solução. Assim, propomos uma nova regra que utiliza o conceito de conjunto independente crítico. A regra consiste em colapsar vértices do conjunto independente crítico; as arestas correspondentes também são colapsadas e seus pesos acumulados. Comparamos a regra de redução proposta (Nova Regra) com Regra 2 (proposta em [Guo *et al.*, 2008]). Entre as 18 instâncias, **Regra 1 + Regra 2** reduziu apenas 3 instâncias, enquanto **Regra 1 + Nova Regra** foi capaz de reduzir 11 instâncias, alcançando uma redução de 35% em um dos casos.

Efetuamos um estudo poliédrico do modelo $\mathcal{F}_{P_4}$, identificando 3 novas famílias de inequações (Fole, Escada e Grid), provando sua viabilidade e que as mesmas são facetas deste poliedro. Propomos heurísticas de separação para estas novas inequações, além

de um algoritmo de *Branch-and-Cut* comparado a mesma estratégia exata proposta por [Pinheiro *et al.*, 2016]. Os resultados computacionais demonstraram que nossa estratégia obteve um ganho significativo no limite inferior após a resolução da raiz do modelo $\mathcal{F}_{P_4}$ restrito. Isto permitiu uma menor quantidade nós a serem resolvidos durante a estratégia de $Branch - and - Bound$, levando a um tempo muito menor para a resolução das instâncias testadas.

Dois novos modelos de programação linear inteira foram propostas para o PEB, $\mathcal{F}_{K_1}$ e $\mathcal{F}_{K_2}$, nos quais são criados variáveis de indexação do *bicluster*, indicando onde o vértice ficará na solução final. Também propomos restrições para a eliminação de soluções simétricas, melhorando assim o desempenho dos modelos. Em comparação ao modelo $\mathcal{F}_{P_4}$ proposto por [Amit, 2004], os dois novos modelos propostos, utilizando a técnica de *Branch-and-Bound*, obtiveram ganhos notáveis de até 95% no tempo computacional para encontrar as soluções ótimas. Além de encontrar uma nova solução ótima.

Propomos um terceiro modelo PLI para o PEB ($\mathcal{F}_\lambda$), onde este modelo apresenta um crescimento exponencial no número de variáveis com a dimensão da instância. Para tratar esta característica, propomos algoritmos de Geração de Colunas para resolver sua relaxação linear. Para os algoritmos de Geração de Colunas, propomos um modelo de PLI para o subproblema de *pricing*, que tem como objetivo encontrar o *bicluster* de menor custo reduzido. Como este subproblema deve ser resolvido durante várias iterações do algoritmo de GC, o procedimento se torna muito ineficiente. Para superar esta desvantagem propomos uma heurística para a resolução do subproblema de *pricing*.

Os resultados computacionais mostram que a abordagem *B&P - 1Col* é mais eficiente que as da literatura obtendo melhores tempos computacionais com o crescimento da dimensão e dificuldade das instâncias, provando 4 soluções ótimas que anteriormente não haviam sido provadas. O *B&P - MultiCol* apresenta uma boa escalabilidade para instâncias maiores, reduzindo o tempo computacional em mais de 90% para a instância $(n = 16, m = 30, d = 0,6)$.

A heurística proposta neste trabalho embutida na meta-heurística GRASP, quando comparada com um procedimento exato, foi capaz de encontrar as soluções ótimas para todas as pequenas instâncias resolvidas pelo procedimento exato, com tempo computacional razoável. O tempo gasto pelo procedimento GRASP ficou abaixo de 1,5 milissegundos em média.

Ainda realizamos uma comparação entre as três meta-heurísticas (GVNS, ILS, e GRASP) desenvolvidas para o PEB, com os dois algoritmos aproximativos e as duas

heurísticas da literatura. As meta-heurísticas propostas neste trabalho alcançaram os melhores resultados que os algoritmos existentes na maioria das instâncias testadas. O ILS proposto apresentou os melhores resultados, melhorando as soluções do Bi-Force (melhores resultados da literatura) para as instâncias $G(n, m, p)$ e $P(G, q)$ em 8,8%, em média. As melhorias na qualidade das soluções são notáveis para as instâncias derivadas de dados biológicos.

Como trabalhos futuros, propomos: (i) novos movimentos de vizinhança; (ii) hibridizações com procedimentos exatos, onde a relaxação linear do modelo de programação será guiado pelas informações identificadas pelas meta-heurísticas com o objetivo de encontrar melhores soluções; (iii) estudo poliedral dos modelos $\mathcal{F}_{K_1}$ e $\mathcal{F}_{K_2}$ para aplicar uma abordagem *Branch-and-cut*; (iv) novas regras de *branching* para o procedimento $B\&P$.

Referências

[Ailon *et al.*, 2012] Ailon, N., Avigdor-Elgrabli, N., Liberty, E., & van Zuylen, A. Improved approximation algorithms for bipartite correlation clustering. *SIAM Journal on Computing*, 41(5), 1110–1121, (2012).

[Amilhastre *et al.*, 1998] Amilhastre, J., Vilarem, M., & Janssen, P. Complexity of minimum biclique cover and minimum biclique decomposition for bipartite domino-free graphs. *Discrete Applied Mathematics*, 86(2–3), 125 – 144, (1998).

[Amit, 2004] Amit, N., (2004). The bicluster graph editing problem. Master's thesis, Tel Aviv University.

[Bansal *et al.*, 2004] Bansal, N., Blum, A., & Chawla, S. Correlation clustering. *Machine Learning*, 56, 89–113, (2004).

[Barrett *et al.*, 2013] Barrett, T., Wilhite, S. E., Ledoux, P., Evangelista, C., Kim, I. F., Tomashevsky, M., Marshall, K. A., Phillippy, K. H., Sherman, P. M., Holko, M., Yefanov, A., Lee, H., Zhang, N., Robertson, C. L., Serova, N., Davis, S., & Soboleva, A. Ncbi geo: archive for functional genomics data sets - update. *Nucleic Acids Research*, 41(Database-Issue), 991–995, (2013).

[Bastos *et al.*, 2014] Bastos, L., Ochi, L. S., Protti, F., Subramanian, A., Martins, I. C., & Pinheiro, R. G. S. Efficient algorithms for cluster editing. *Journal of Combinatorial Optimization*, (pp. 1–25)., (2014).

[Ben-Dor *et al.*, 2002] Ben-Dor, A., Chor, B., Karp, R., & Yakhini, Z., (2002). Discovering local structure in gene expression data: The order-preserving submatrix problem. In *Proceedings of the Sixth Annual International Conference on Computational Biology*, RECOMB '02 (pp. 49–57). New York, NY, USA: ACM.

[Bergmann *et al.*, 2003] Bergmann, S., Ihmels, J., & Barkai, N., (2003). Barkai n: Iterative signature algorithm for the analysis of large-scale gene expression data. In *Phys Rev E Stat Nonlin Soft Matter Phys 2003, 67(3 Pt 1):031902.*

[Billionnet & Éric Soutif, 2004] Billionnet, A. & Éric Soutif. Using a mixed integer programming tool for solving the 0–1 quadratic knapsack problem. *INFORMS Journal on Computing*, 16(2), 188–197, (2004).

[Blum *et al.*, 2005] Blum, C., Roli, A., & Alba, E., (2005). An introduction to metaheuristic techniques. In E. Alba (Ed.), *Parallel metaheuristics: a new class of algorithm* chapter 1, (pp. 3–42). Wiley-interscience.

[Bozdağ *et al.*, 2009] Bozdağ, D., Parvin, J. D., & Catalyurek, U. V., (2009). A biclustering method to discover co-regulated genes using diverse gene expression datasets. In

Proceedings of the 1st International Conference on Bioinformatics and Computational Biology, BICoB '09 (pp. 151–163). Berlin, Heidelberg: Springer-Verlag.

[Cheng & Church, 2000] Cheng, Y. & Church, G. M., (2000). Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology* (pp. 93–103).: AAAI Press.

[Gilbert, 1959] Gilbert, E. N., (1959). Random graphs. In *Annals of Mathematical Statistics*, volume 3 (pp. 1141–1144).

[Glover, 1986] Glover, F. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5), 533–549, (1986).

[Grötschel & Wakabayashi, 1990] Grötschel, M. & Wakabayashi, Y. Facets of the clique partitioning polytope. *Mathematical Programming*, 47(1), 367–387, (1990).

[Guo *et al.*, 2008] Guo, J., Hüffner, F., Komusiewicz, C., & Zhang, Y., (2008). Improved algorithms for bicluster editing. In *TAMC'08 - 5th International Conference on Theory and Applications of Models of Computation*, volume 4978 of *Lecture Notes in Computer Science* (pp. 445–456).

[Hansen *et al.*, 2010] Hansen, P., Mladenović, N., & Moreno Perez, J. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175, 367–407, (2010).

[Hochreiter *et al.*, 2010] Hochreiter, S., Bodenhofer, U., Heusel, M., Mayr, A., Mitterecker, A., Kasim, A., Khamiakova, T., Sanden, S. V., Lin, D., Talloen, W., Bijnens, L., Göhlmann, H. W. H., Shkedy, Z., & Clevert, D.-A. Fabia: factor analysis for bicluster acquisition. *Bioinformatics*, 26(12), 1520–1527, (2010).

[Huttenhower *et al.*, 2009] Huttenhower, C., Mutungu, K. T., Indik, N., Yang, W., Schroeder, M., Forman, J., & Coller, H. A., (2009). Detailing regulatory networks through large scale data integration.a. In *Bioinformatics 25(24)*, volume 25 (pp. 3267–3274).

[Köhler *et al.*, 2013] Köhler, V., Fampa, M., & Araújo, O. Mixed-integer linear programming formulations for the software clustering problem. *Computational Optimization and Applications*, 55(1), 113–135, (2013).

[Kluger *et al.*, 2003] Kluger, Y., Basri, R., Chang, J., & Gerstein, M. Spectral biclustering of microarray data: Coclustering genes and conditions. *Genome Research*, 13, 703–716, (2003).

[Kramer *et al.*, 2014] Kramer, H. H., Fampa, M., Köhler, V., Vanderbeck, F., & Uchoa, E., (2014). Column generation approaches for the software clustering problem. In *Anais do XLVI Simpósio Brasileiro de Pesquisa Operacional.*

[Lazzeroni & Owen, 2000] Lazzeroni, L. & Owen, A. Plaid models for gene expression data. *Statistica Sinica*, 12, 61–86, (2000).

[Li *et al.*, 2009] Li, G., Ma, Q., Tang, H., Paterson, A. H., & Xu, Y. Qubic: A qualitative biclustering algorithm for analyses of gene expression data. *Nucleic Acids Research*, 37(15), e101, (2009).

[Liu, 2013] Liu, P. Q. An exact algorithm for finding k-biclique vertex partitions of bipartites. *Advanced Materials Research*, 710(5), 687–691, (2013).

[Lourenço *et al.*, 2003] Lourenço, H., Martin, O., & Stutzle, T., (2003). Iterated local search. In F. Glover, G. Kochenberger, F. S. Hillier, & C. C. Price (Eds.), *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research and Management Science* (pp. 320–353). Springer New York.

[Madeira & Oliveira, 2004] Madeira, S. C. & Oliveira, A. L., (2004). Biclustering algorithms for biological data analysis: a survey. In *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, volume 1 (pp. 24–45). IEEE.

[Murali & Kasif, 2003] Murali, T. M. & Kasif, S., (2003). Extracting conserved gene expression motifs from gene expression data. In *Pac. Symp. Biocomput* (pp. 77–88).

[Pinheiro *et al.*, 2016] Pinheiro, R. G., Martins, I. C., Protti, F., Ochi, L. S., Simonetti, L. G., & Subramanian, A. On solving manufacturing cell formation via bicluster editing. *European Journal of Operational Research*, 254(3), 769–779, (2016).

[Prelić *et al.*, 2006] Prelić, A., Bleuler, S., Zimmermann, P., Wille, A., Bühlmann, P., Gruissem, W., Hennig, L., Thiele, L., & Zitzler, E. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9), 1122–1129, (2006).

[Protti *et al.*, 2009] Protti, F., da Silva, M. D., & Szwarcfiter, J. L. Applying modular decomposition to parameterized cluster editing problems. *Theory of Computing Systems*, 44, 91–104, (2009).

[Resende, 2001] Resende, M., (2001). Greedy randomized adaptive search procedures. In C. A. Floudas & P. M. Pardalos (Eds.), *Encyclopedia of Optimization* (pp. 913–922). Springer US.

[Ryan & Foster, 1981] Ryan, D. M. & Foster, B. A. An integer programming approach to scheduling. *Computer scheduling of public transport urban passenger vehicle and crew scheduling*, (pp. 269–280)., (1981).

[Shamir *et al.*, 2004] Shamir, R., Sharan, R., & Tsur, D. Cluster graph modification problems. *Discrete Applied Mathematics*, 144, 173–182, (2004).

[Subhashini & Kumar, 2010] Subhashini, R. & Kumar, V. J. S., (2010). Evaluating the performance of similarity measures used in document clustering and information retrieval. In *Integrated Intelligent Computing (ICIIC), 2010 First International Conference on* (pp. 27–31).

[Sun *et al.*, 2013] Sun, P., Guo, J., & Baumbach, J. Biclue - exact and heuristic algorithms for weighted bi-cluster editing of biomedical data. *BMC Proceedings*, 7(Suppl 7), S9, (2013).

[Sun *et al.*, 2014] Sun, P., Speicher, N. K., Röttger, R., Guo, J., & Baumbach, J. Bi-force: large-scale bicluster editing and its application to gene expression data biclustering. *Nucleic Acids Research*, (2014).

[Talbi, 2009] Talbi, E.-G., (2009). *Metaheuristics: From Design to Implementation.* Wiley Publishing.

[Tanay *et al.*, 2006] Tanay, A., Sharan, R., & Shamir, R., (2006). Biclustering algorithms: a survey. In S. Aluru (Ed.), *Handbook of Computational Molecular Biology.* Chapman Hall/CRC Press.

[Vanderbeck, 2011] Vanderbeck, F. Branching in branch-and-price: a generic scheme. *Mathematical Programming*, 130(2), 249–294, (2011).