Uma Abordagem de Programação Linear Inteira para o Problema da Clique Máxima com Peso nas Arestas

Elder Magalhães Macambira

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Uma Abordagem de Programação Linear Inteira para o Problema da Clique Máxima com Peso nas Arestas

Elder Magalhães Macambira[1]

Junho de 1997

Banca Examinadora:

- Prof. Dr. Cid Carvalho de Souza
  Instituto de Computação - Unicamp (Orientador)
- Profa. Dra. Yoshiko Wakabayashi
  Instituto de Matemática e Estatística - USP
- Prof. Dr. João Meidanis
  Instituto de Computação - Unicamp
- Prof. Dr. Ricardo Dahab
  Instituto de Computação - Unicamp

[1] O autor é Bacharel em Ciência da Computação pela Universidade Estadual do Ceará.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Macambira, Elder Magalhães

M118a Uma abordagem de programação linear inteira para o problema da clique máxima com peso nas arestas / Elder Magalhães Macambira -- Campinas, [S.P. :s.n.], 1997.

Orientador : Cid Carvalho de Souza

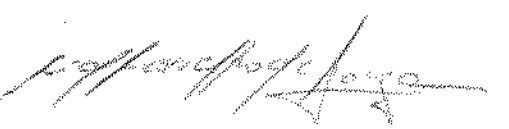
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Programação linear inteira. 2. Poliedros. 3. Algoritmos. 4. Otimização combinatória. 5. Programação matemática. I. Souza, Cid Carvalho de. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Uma Abordagem de Programação Linear Inteira para o Problema da Clique Máxima com Peso nas Arestas

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Elder Magalhães Macambira e aprovada pela Banca Examinadora.

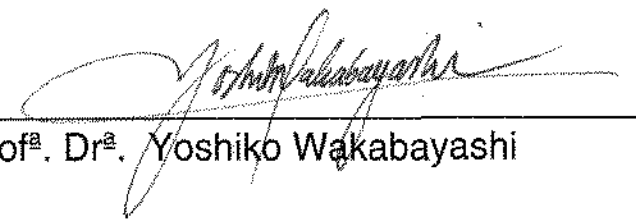
Campinas, 18 de junho de 1997.



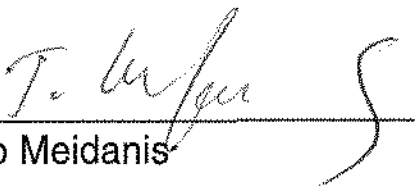
Prof. Dr. Cid Carvalho de Souza
Instituto de Computação - Unicamp
(Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

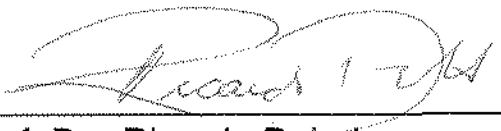
Tese de Mestrado defendida e aprovada em 06 de junho de 1997 pela Banca Examinadora composta pelos Professores Doutores



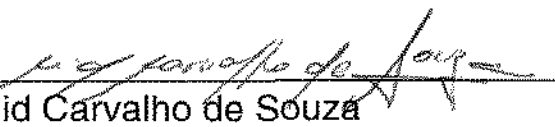
Profª. Drª. Yoshiko Wakabayashi



Prof. Dr. João Meidanis



Prof. Dr. Ricardo Dahab



Prof. Dr. Cid Carvalho de Souza

"Mais vale a pena ter a alma dolorida
de tanto buscar do que tê-la em paz
por haver renunciado a busca."

"É melhor tentar e falhar
que preocupar-se e ver a vida passar.
É melhor tentar ainda em vão,
que sentar-se fazendo nada até o final.
Eu prefiro na chuva caminhar,
que em dias tristes em casa me esconder.
Prefiro ser feliz, embora louco,
que em conformidade viver."

Martin Luther King

Resumo

Esta dissertação dá ênfase à abordagem poliedral para a resolução exata do Problema da Clique Máxima com Peso nas Arestas. Dado um grafo completo não-dirigido $K_n = (V_n, E_n)$, onde $|V_n| = n$, com um peso c_{ij} associado a cada aresta $(i, j) \in E_n$, e um inteiro b, onde $b \leq n$; procuramos uma clique C em K_n cuja soma dos pesos das arestas em C seja máxima e $|C| \leq b$.

São apresentadas e discutidas diferentes formulações de programação linear inteira para o problema. Investigamos ainda a estrutura facial do poliedro associado ao problema realizando uma revisão bibliográfica das desigualdades conhecidas e introduzindo novas famílias de facetas.

Por último, descrevemos os experimentos computacionais realizados com um algoritmo *branch-and-cut* e com uma metaheurística, ambos propostos neste trabalho. As maiores instâncias resolvidas de forma exata para este problema na literatura referem-se a grafos completos com no máximo 30 vértices. Neste trabalho, resolvemos exatamente instâncias para grafos com até 48 vértices e mostramos a força computacional para as novas desigualdades que introduzimos.

Abstract

Given a complete non-directed graph $K_n = (V_n, E_n)$ on n nodes with weights on the edges and an integer $b \leq n$, we look for a clique C in K_n whose sum of the weights of the edges in C is maximum and such that $|C| \leq b$. We discuss on different integer programming formulations and investigate the facial structure of the polyhedron associated to the problem. New families of facet defining inequalities are introduced.

Finally we describe our computational experiments with a *branch-and-cut* algorithm and a metaheuristic that we have proposed. The largest instances that are solved exactly in the literature refer to complete graphs with at most 30 nodes. In this work we solve to optimality instances for graphs with up to 48 nodes and we show the computational strength of the new inequalities we have introduced.

Agradecimentos

À Deus que, nos momentos de desencontro e angústia, estava sempre presente ao meu lado me dando apoio e conforto, o que possibilitou a realização deste trabalho.

Aos meus pais, Francisca e Edilson, que neste longo e árduo período de dois anos souberam suportar a ausência de um filho, e que por diversas vezes abdicaram dos seus sonhos a favor dos meus.

Aos meus irmãos, Felipe (*in memoriam*), Angélica e Natália, pelo carinho e amor demonstrado.

Ao Prof. Dr. Cid Carvalho de Souza que me orientou nos meus primeiros estudos em Combinatória Poliédrica, pela demonstração de profissionalismo na realização deste trabalho e pela amizade dispensada.

Ao Prof. Dr. Marcus Poggi pela ajuda e discussões feitas durante a realização deste trabalho. Aos Profs. Drs. Ulrich Faigle, da Universidade de Twente, e Sungsoo Park, da Universidade da Korea, por se mostrarem prestativos em enviar-me artigos referentes ao tema de nossa pesquisa.

Aos amigos, que tive o imenso prazer de conhecer, e sem os quais não seria possível transpor tantos obstáculos encontrados durante esta árdua caminhada: Aminadab, Célio Targa, Cereja, Cláudio, Cleidson, Cristiane Grando, Delano, Eduardo, Elbson, Gisele Craveiro, Gutemberg, Hugo Alexandre, Jerônimo, Marcelo de Jesus, Marcos André, Nivando, Patrícia Ropelatto, Ralph, Roberto e Socorrinha; e aos que não foram aqui mencionados, por falta de espaço, porém lembrados com muito, muito carinho.

Ao Victor pela acolhida, apoio e orientação durante o período inicial do programa de Mestrado.

Aos professores da Universidade Estadual do Ceará que sempre acreditaram em minha capacidade e empenho.

Aos professores do Instituto de Computação que direta ou indiretamente contribuíram ainda mais na minha formação profissional.

Aos funcionários, Luiz (*in memoriam*), Roseli, Vera e Daniel, pela competência e assistência nos serviços prestados.

Aos órgãos de fomento à pesquisa, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), pelo apoio financeiro.

Não poderia deixar de esquecer a uma pequena fatia do povo brasileiro que paga imposto, e sem saber, contribui de forma significativa para o progresso tecnológico, científico e cultural do país.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Nas últimas décadas tem sido crescente o emprego de programação linear na resolução de problemas provenientes do mundo real. Isto se deve ao fato de que tais problemas práticos apresentam características de problemas de otimização. Um problema de otimização caracteriza-se por minimizar ou maximizar uma função definida em um domínio. Este domínio representa o conjunto de soluções viáveis para o problema podendo ser infinito ou finito. Quando o domínio é finito o problema é denominado *problema de otimização combinatória.*

De uma maneira genérica um problema de otimização combinatória pode ser definido como abaixo (Jünger *et al.* [33]).

Problema de otimização combinatória. Dado um conjunto finito $\mathcal{I}$ de soluções viáveis, e uma função objetivo $f : \mathcal{I} \rightarrow \mathbb{R}$, deseja-se encontrar um elemento $I^* \in \mathcal{I}$, tal que, $f(I^*) = o\{f(I) : I \in \mathcal{I}\}$, onde $o = \max$ ou $o = \min$.

Uma classe mais restritiva de problemas de otimização combinatória pode ser vista quando exigimos que a função $f : \mathcal{I} \rightarrow \mathbb{R}$ seja linear. Tipicamente, para um problema de otimização combinatória cada instância I é um subconjunto de um conjunto finito S e a função linear f é obtida atribuindo-se pesos aos elementos de S. Ou seja, o valor de f para uma instância I é dada pela soma dos seus elementos, ponderada pelos pesos atribuídos a esses elementos.

Para uma melhor compreensão do que foi dito acima, damos em seguida a definição de um problema de otimização combinatória com essa restrição na função objetivo (Jünger *et al.* [33]).

Problema de otimização combinatória linear. Seja S um conjunto finito, $\mathcal{I}$ um conjunto finito de soluções viáveis, tal que, $\mathcal{I} \subseteq 2^S$ e $c : S \rightarrow \mathbb{R}$ uma função. Para

cada conjunto $F \subseteq S$ podemos definir $c(F) = \sum_{e \in F} c_e$. Deseja-se encontrar um conjunto $I^* \in \mathcal{I}$, tal que, $c(I^*) = o\{c(I) : I \in \mathcal{I}\}$.

Apesar da restrição imposta, veremos que diversos problemas interessantes podem ser modelados como problemas de otimização combinatória linear. Por exemplo, o problema do caixeiro viajante, o problema da árvore geradora mínima, o problema da mochila e vários outros. Em particular, podemos incluir a esta lista o problema que foi o objeto de estudo desta dissertação.

Assim, um problema de otimização combinatória linear fica bem caracterizado pela tripla (S,$\mathcal{I}$,c). Até o final desta dissertação, usaremos o termo problema de otimização combinatória ao invés de problema de otimização combinatória linear visto que o nosso problema, bem como a maioria dos assuntos que iremos apresentar lidam com esta restrição imposta à função objetivo.

Vimos que o processo de avaliação da função objetivo se torna bem mais fácil quando realizamos a sua linearização, porém surge uma pergunta: como obter o conjunto de soluções viáveis $\mathcal{I}$? Embora seja finito, a obtenção de uma descrição razoável do conjunto $\mathcal{I}$ para os problemas de otimização combinatória é impraticável posto que seria necessário a enumeração de todas as soluções viáveis do problema.

Essa dificuldade muitas vezes se reflete no projeto de algoritmos com tempo polinomial para problemas de otimização combinatória. Tal aspecto despertou o interesse de vários pesquisadores em estudar esses tipos de problemas sob um outro ângulo, e assim tornar possível o desenvolvimento de algoritmos.

O crescente desenvolvimento de algoritmos para problemas de otimização combinatória favoreceu o surgimento da área de Combinatória Poliédrica. A Combinatória Poliédrica lida com a aplicação de vários aspectos da Teoria Poliedral e sistemas lineares com o intuito de encontrar um sistema de desigualdades lineares para descrição de poliedros associados problemas combinatórios. Em seguida faremos uma breve descrição do desenvolvimento dessa área, divididos em períodos, baseados no trabalho de Pulleyblank [43].

O primeiro período iniciou-se realmente em 1950 com a descoberta do algoritmo simplex para a programação linear. Nesse período, constatou-se que vários problemas de otimização combinatória, principalmente problemas de fluxos em redes, poderiam ser formulados como programas lineares inteiros e serem resolvidos através do algoritmo simplex.

O segundo período teve início na metade da década de 60 com os trabalhos de Jack Edmonds na resolução do problema de emparelhamento em grafos. Ele obteve uma descrição linear completa, do conjunto de restrições associado ao problema. Essa descoberta possibilitou o desenvolvimento de algoritmos com tempo polinomial para o problema de emparelhamento em grafos, e incentivou o estudo de outros problemas.

O terceiro período iniciou-se em 1979 com o desenvolvimento do método elipsóide

para programação linear. O sucesso do método elipsóide dependia da possibilidade de se verificar se ou não um dado ponto pertencia ao conjunto solução do sistema linear. Caso o ponto não viesse a pertencer ao conjunto solução, o método deveria ser capaz de produzir uma desigualdade violada. Através dos trabalhos de Grötschel, Lovász e Schrijver [24, 25] foi demonstrado que problemas de otimização são equivalentes a problemas de separação, ou seja se é possível encontrar a solução ótima de forma eficiente então também é possível resolver eficientemente o problema de separação. Isto provocou um novo impulso no estudo de algoritmos de planos-de-corte cuja base é o estudo da Combinatória Poliédrica.

Todos esses resultados permitiram a resolução de instâncias de médio e grande porte para vários problemas combinatórios difíceis. Estimulados pelo sucesso obtido na resolução destes problemas, decidimos investigar a possibilidade de tratar o problema abordado nesta dissertação usando as técnicas de programação linear inteira e Combinatória Poliédrica.

1.1 Principais Objetivos

Como visto anteriormente, em Combinatória Poliédrica certos problemas de otimização combinatória podem ser reduzidos a problemas de programação linear sobre um poliedro inteiro. No entanto, devido à complexidade inerente desses problemas combinatórios, em sua maioria problemas NP-difíceis, muitos deles ainda não foram resolvidos de forma satisfatória.

Do ponto de vista da Combinatória Poliédrica, procura-se atender a este objetivo encontrando-se desigualdades válidas que consigam melhorar a descrição linear dada pelo conjunto de desigualdades associado ao problema, e fazendo-se uso de alguma técnica algorítmica adequada, possibilitar a geração de novas soluções viáveis.

Um dos objetivos principais com esse trabalho é resolver de forma exata, ou pelo menos obter bons limitantes superiores, para o Problema da Clique Máxima com Peso nas Arestas.

Seguindo a nomenclatura utilizada por Garey e Johnson [21], o problema de decisão definido abaixo caracteriza o Problema da Clique Máxima com Peso nas Arestas:

Instância: Seja $K_n = (V_n, E_n)$ um grafo completo não-dirigido, onde $|V_n| = n$, com um peso c_{ij} associado a cada aresta $(i, j) \in E_n$, b um inteiro, onde $1 \leq b \leq n$, e W um outro inteiro.

Questão: Existe uma clique $C = (U, F)$ em K_n, tal que, $|U| \leq b$ e $\sum_{e \in E} w_e \geq W$?

É bem conhecido na literatura que o tradicional Problema da Clique Máxima é NP-

Completo (veja Garey e Johnson [21]). Portanto, é possível demonstrar, através de uma redução polinomial, que o Problema da Clique Máxima com Peso nas Arestas também é *NP-Completo*.

A motivação inicial para a resolução exata do Problema da Clique Máxima com Peso nas Arestas, doravante denominado PCMPA, deve-se aos trabalhos realizados por Faigle e Dijkhuizen [14]. Estes autores realizaram um estudo poliédrico do PCMPA. Neste estudo eles propuseram uma formulação inteira tomando-se as arestas do grafo como variáveis de decisão e definiram novas classes de desigualdades válidas. Além disso, empregaram algoritmos de planos-de-corte para a resolução do problema.

Sugerindo que este tipo de algoritmo não era adequado para resolver o problema em questão, os autores apontaram como possível causa para isso, o fato de que a estrutura facial do poliedro associado ao problema ainda era pouco conhecida.

Baseado nesses aspectos foi possível então levantar dois pontos de pesquisa: um estudo poliédrico mais aprofundado do Problema da Clique Máxima com Peso nas Arestas que permitisse uma melhor caracterização da estrutura facial do poliedro através de novas desigualdades lineares e o uso de uma outra técnica algorítmica que verificasse a qualidade dessas desigualdades.

Tendo em vista esses dois pontos, nós propomos um novo estudo do politopo associado ao Problema da Clique Máxima com Peso nas Arestas no espaço definido por vértices e arestas, e o uso de algoritmos *branch-and-cut* como a técnica algorítmica a ser empregada para a resolução do problema.

Como será possível constatar nos capítulos seguintes, o estudo poliédrico do Problema da Clique Máxima com Peso nas Arestas no espaço definido por vértices e arestas, e o uso de algoritmos *branch-and-cut* possibilitou a resolução de forma mais satisfatória do problema em questão, mesmo para instâncias de tamanho moderado.

1.2 Organização da Dissertação

Esta dissertação está organizada em oito capítulos. Os tópicos a serem cobertos em cada um deles são como descritos em seguida.

Neste primeiro capítulo, foram apresentados o problema que trataremos, os principais objetivos deste trabalho, o que nos motivou para o seu desenvolvimento e a forma de abordagem empregada na resolução do problema. No segundo capítulo, apresentaremos alguns conceitos preliminares provinientes da Teoria dos Grafos e Teoria Poliedral, bem como formas de modelar problemas de otimização combinatória. Tais conceitos serão úteis para um melhor entendimento do trabalho.

No terceiro capítulo, apresentaremos trabalhos relacionados com o PCMPA. No quarto capítulo, apresentaremos algumas técnicas algorítmicas utilizadas em programação linear

inteira para a resolução de problemas de otimização combinatória. Em particular, nos detalheremos apenas àquelas que estão relacionadas com o nosso trabalho.

No quinto capítulo, serão apresentados resultados sobre a estrutura poliedral do Problema da Clique Máxima com Peso nas Arestas em ambos os modelos - espaço definido por arestas e o definido por vértices e arestas - bem como novas famílias de facetas encontradas neste trabalho.

No sexto capítulo, apresentaremos a descrição e os resultados computacionais obtidos com o emprego da metaheurística *GRASP* na obtenção de limitantes inferiores. No sétimo capítulo, descreveremos o algoritmo *branch-and-cut* e os resultados computacionais obtidos com o emprego desta técnica na resolução do Problema da Clique Máxima com Peso nas Arestas.

No oitavo, e último capítulo, serão apresentadas as conclusões desse trabalho e algumas considerações finais que poderão levar a direções futuras do trabalho.

Finalizaremos este capítulo expondo algumas aplicações práticas do Problema da Clique Máxima com Peso nas Arestas.

1.3 Problema da Clique Máxima com Peso nas Arestas: aplicações práticas

O Problema da Clique Máxima com Peso nas Arestas possui várias aplicações típicas, em especial, problemas que envolvam a seleção e localização de facilidades e no projeto de redes de telecomunicações.

1.3.1 Seleção e Localização de Facilidades

Vários problemas da teoria de localização lidam com a escolha de facilidades em uma região. Nestes tipos de problemas deseja-se minimizar ou maximizar a distância entre as facilidades, ou entre as facilidades e os outros pontos localizados na região.

Quando se deseja *minimizar* o valor da função objetivo estamos trabalhando com um problema de localização de facilidades desejáveis. Exemplos de facilidades neste caso seriam: armazéns, hospitais, postos policiais, quartéis de bombeiros, postos de sáude. Os trabalhos de Späth [47, 48] retratam esses tipos de aplicações.

Entretanto, existem situações onde deseja-se *maximizar* a distância entre os pontos em uma região que representam as facilidades. Tais problemas de localização são denominados *problemas de dispersão* porque eles modelam situações nas quais a proximidade das facilidades é indesejável (veja Ravi *et al.* [44]). Os exemplos de facilidades aqui seriam: *franchises* de uma empresa, usinas nucleares, refinarias de petróleo e aterros sanitários.

1.3.2 Projeto de Redes de Telecomunicações

Uma outra aplicação prática do Problema da Clique Máxima com Peso nas Arestas surge no projeto de redes de telecomunicações (veja Park *et al.* [42]). Uma rede de telecomunicações pode ser vista em dois níveis. O primeiro é composto por um conjunto de nós, denominados nós *hubs*, que servem como pontos de concentração da demanda de tráfegos para comunicação entre regiões. O segundo nível é composto por um ou dois nós *hubs* e um conjunto de nós ordinários. Portanto, podemos estabelecer que o tráfego de telecomunicação entre duas regiões pode ser transportado via nós *hubs* de cada subregião.

O problema no projeto de uma rede de telecomunicações consiste em encontrar um *clustering* de pontos que minimize o tráfego entre os *clusters*. Cada *cluster* (subregião) deve ainda satisfazer restrições de conectividade, ou seja, se um ponto do *cluster* falhar deve-se automaticamente selecionar um nó ordinário que passará a ser *hub*. Além disso, os pontos pertencentes a um mesmo *cluster* devem ser compatíveis. A fase de formação dos *clusters* é resolvida modelando o problema como um PCMPA.

Capítulo 2

Conceitos Preliminares

Neste capítulo procuramos deixar claro alguns conceitos e notações que empregaremos comumente no decorrer desta dissertação. Os primeiros conceitos a serem apresentados são referentes à Teoria dos Grafos. Em seguida, apresentaremos alguns modelos de otimização utilizados na resolução de problemas de otimização. Por último, daremos algumas noções de termos presentes em Teoria Poliedral.

2.1 Definições em Teoria dos Grafos

As definições apresentadas aqui foram compiladas de Ferreira e Wakabayashi [20] (veja também Bondy e Murty [4] e Bollobás [3]). Vale lembrar que os conceitos apresentados são referentes a grafos não-dirigidos.

Definição 2.1 *Um* grafo $G = (V, E)$ *consiste de um conjunto finito não-vazio* V *de elementos chamados* vértices *e um conjunto de pares não-ordenados de elementos distintos de* V*, chamados* arestas*, isto é,* $E \subseteq \{\{u, v\} : u, v \in V \text{ com } u \neq v\}$.

Definição 2.2 *Se* $e = \{u, v\}$ *é uma aresta dizemos que* e incide *em* u *e* v*; que* u *e* v *são seus* extremos*; e que* u *e* v *são* adjacentes.

Definição 2.3 *Seja* $v \in V$ *um vértice qualquer em* G*. O* grau *do vértice* v *corresponde ao número de arestas incidentes em* v.

Definição 2.4 *Sejam* $G = (V, E)$ *e* $H = (W, F)$ *dois grafos. Se* $V \subseteq W$ *e* $E \subseteq F$ *então* G *é dito um* subgrafo *de* H.

Definição 2.5 *Se* $A \subseteq E$ *então o subgrafo de* G *com conjunto de arestas* E *e vértices consistindo dos extremos das arestas* E *é o subgrafo* gerado *ou* induzido *por* E*. Se* $W \subseteq V$*, então* $G = (W, E(W))$ *é o subgrafo gerado por* W.

Definição 2.6 *Seja $G = (V, E)$ um grafo. Se para quaisquer dois vértices $v, u \in V$, com $u \neq v$, temos u e v adjacentes então G é* completo. *O grafo completo com n vértices é denotado por $K_n = (V_n, E_n)$.*

Definição 2.7 *Uma* clique *$C = (U, F)$ em um grafo G corresponde a um subgrafo completo de G.*

Vale ressaltar que ao longo deste texto, uma clique $C = (U, F)$ estará sendo caracterizada pelo seu conjunto de arestas F.

Definição 2.8 *Um* caminho *em um grafo $G = (V, E)$ é uma seqüência não-vazia $\mathcal{P} = \{e_1, e_2, \ldots, e_k\}$ de arestas, onde $e_i = (v_i, v_{i+1}) \in E$, e $v_i \neq v_j$ para $i \neq j$. Um* ciclo *é um caminho onde $v_k = v_1$; com $e_1, e_k \in \mathcal{P}$, $e_k = (v_{k-1}, v_k)$ e $e_1 = (v_1, v_2)$.*

Definição 2.9 *Um grafo é* conexo *se possui um caminho entre quaisquer dois de seus vértices; caso contrário é* desconexo.

Definição 2.10 *Seja $G = (V, E)$ um grafo. Dizemos que H é um* componente *de G se H é um subgrafo conexo maximal em G.*

Definição 2.11 *Uma* árvore *$T = (V, E(T))$ é um grafo conexo sem ciclos.*

Definição 2.12 *Uma* floresta *$F = (V, E(F))$ é um grafo onde cada componente é uma árvore.*

2.2 Modelos de Otimização

Como vimos no capítulo anterior problemas de otimização combinatória procuram maximizar ou minimizar uma função. Devido a essa característica, tais problemas podem ser associados a dois modelos de otimização: programação linear e programação linear inteira.

Um *problema de programação linear* pode ser definido como um problema de otimizar uma função linear sobre uma região viável descrita por um conjunto de desigualdades e igualdades lineares. Utilizando a notação matricial um problema de programação linear pode ser visto como (veja Jünger *et al.* [33]).

Problema de programação linear. Dada uma matriz $A \in \mathbb{R}^{(m,n)}$, e os vetores $q \in \mathbb{R}^m$ e $c \in \mathbb{R}^n$, deseja-se encontrar um vetor $x^* \in \mathbb{R}^n$, tal que, $cx^* = \max \{cx : Ax \leq q\}$.

A função $cx : \mathbb{R}^n \rightarrow \mathbb{R}$ é denominada *função objetivo* e as desigualdades no sistema $Ax \leq q$ são denominadas *restrições*. Um problema de programação linear é também denomindado programa linear (PL). Até o final deste capítulo nós assumiremos que a função objetivo é para ser maximizada (note que maximizar cx é equivalente a minimizar $-cx$).

Problemas de programação linear podem ser resolvidos com a utilização de versões do algoritmo simplex que, embora sendo exponencial, se mostra bastante eficiente na prática.

Alguns problemas podem ser formulados como PLs quando acrescentadas restrições que forçam a integralidade, ou seja $x \in \mathbb{Z}^n$ ao invés de $x \in \mathbb{R}^n$. Tal problema é denominado *problema de programação linear inteira*. As soluções viáveis em problemas de programação linear inteira correpondem a pontos cujas coordenadas são inteiras e que satisfazem as desigualdades lineares.

Se for exigido que apenas algumas variáveis assumam valores inteiros então o problema de programação linear inteira é denominado *problema de programação linear inteira mista*. No contrário, se todas as variáveis assumem valores inteiros temos um *problema de programação linear inteira pura*.

No modelo de programação linear inteira quando as variáveis inteiras são utilizadas para representar relações lógicas os seus valores ficam restritos a 0-1. Assim nós obtemos um problema de programação linear inteira mista (pura) 0-1. Por exemplo, para um problema que possui um grafo $G = (V, E)$ como estrutura poderíamos dizer que para cada variável associada a uma aresta $e \in E$ ela valerá 1 se a aresta e pertence a solução e 0 caso contrário.

Em geral, problemas de programação linear inteira são NP-difíceis. Os algoritmos usados para resolver estes tipos de problemas são baseados na resolução de relaxações lineares. Uma relaxação linear pode ser obtida quando substituímos as restrições de integralidade por restrições do tipo $0 \leq x \leq 1$ ou $x \in \mathbb{R}^n$ em programação linear inteira 0-1 e programação linear inteira pura, respectivamente.

2.3 Definições em Teoria Poliedral

Nessa subseção apresentaremos alguns resultados básicos da Teoria Poliedral necessários ao bom entendimento do texto. Estes resultados foram compilados de Nemhauser e Wolsey [36] e Schrijver [45] (veja também Ferreira e Wakabayashi [20]).

Nas definições que se seguem denotaremos por A uma matriz real de dimensão $m \times n$ ($A \in \mathbb{R}^{m \times n}$) e por q um vetor real de dimensão n ($q \in \mathbb{R}^n$).

Definição 2.13 *Um conjunto de pontos $x^1, x^2, \ldots, x^k \in \mathbb{R}^n$ é* linearmente independente *se a solução única para $\sum_{i=1}^{k} \lambda_i x^i = 0$ é $\lambda_i = 0$ para $i = 1, 2, \ldots, k$.*

Definição 2.14 *Um conjunto de pontos* $x^1, x^2, \ldots, x^k \in \mathbb{R}^n$ *é* afim independente *se a solução única para* $\sum_{i=1}^k \lambda_i x^i = 0$ *e* $\sum_{i=1}^k \lambda_i = 0$ *é* $\lambda_i = 0$ *para* $i = 1, 2, \ldots, k$.

Definição 2.15 *Seja* $S = \{x^1, \ldots, x^k\}$ *um conjunto de pontos em* $\mathbb{R}^n$. *A* envoltória convexa *de* S, *denotada por* $conv(S)$, *é o conjunto de pontos dado por* $conv(S) = \{\sum_{i=1}^k \lambda_i x^i : \sum_{i=1}^k \lambda_i = 1,\ x^i \in S,\ \lambda_i \in \mathbb{R}$ *e* $\lambda_i \geq 0\ \forall i = 1, 2, \ldots, k\}$.

Definição 2.16 *Um* poliedro $P \subseteq \mathbb{R}^n$ *é um conjunto de pontos que satisfaz um número finito de desigualdades lineares, isto, é* $P = \{x \in \mathbb{R}^n : Ax \leq q\}$.

Definição 2.17 *Dado um poliedro* $P = \{x \in \mathbb{R}^n : Ax \leq q\}$, *o* fecho inteiro *de* P, *denotado por* $\overline{P}$, *consiste da envoltória convexa dos vetores inteiros de* P, *isto é,* $\overline{P} = conv(\{x \in P : x$ *é inteiro*$\})$.

Definição 2.18 *Um poliedro* $P \subseteq \mathbb{R}^n$ *é dito ser* limitado *se existem vetores* ℓ *e* $u \in \mathbb{R}^n$, *tal que,* $\ell \leq x \leq u$ *para todo* $x \in P$. *Um poliedro limitado é denominado um* politopo.

Definição 2.19 *A desigualdade* $\pi x \leq \pi_0$ *é denominada uma* desigualdade válida *para o poliedro* $P \subseteq \mathbb{R}^n$ *se ela é satisfeita por todos os pontos de* P.

Definição 2.20 *O* suporte *de uma desigualdade* $\pi x \leq \pi_0$ *válida para o poliedro* $P \subseteq \mathbb{R}^n$ *é dado pelo conjunto de índices* $\{j \in \{1, 2, \ldots, n\} : \pi_j \neq 0\}$.

Definição 2.21 *Um conjunto* $\mathcal{F} \subseteq P = \{x \in \mathbb{R}^n : Ax \leq q\}$ *é uma* face *de* P, *se somente se, para algum subsistema* $A^{(=)}x \leq q^{(=)}$ *de* $Ax \leq q$ *nós temos* $\mathcal{F} = \{x \in P : A^{(=)}x = q^{(=)}\}$. *Uma face* $\mathcal{F}$ *é dita ser* própria *se* $\mathcal{F} \neq \emptyset$ *e* $\mathcal{F} \neq P$.

Definição 2.22 *Seja* $P = \{x \in \mathbb{R}^n : Ax \leq q\}$ *e* $v \in P$. *Então* v *é um* vértice *de* P, *se somente se,* v *não pode ser escrito como uma combinação convexa de vetores em* $P - \{v\}$.

Definição 2.23 *Um conjunto* $S \subseteq P$ *possui* dimensão d, *denotado por* $dim(S) = d$, *se o número máximo de pontos afim independentes em* S *é* $d + 1$. *Em particular,* $dim(\mathbb{R}^n) = n$.

Definição 2.24 *Seja* $\mathcal{F}$ *uma face própria de um poliedro* $P \subseteq \mathbb{R}^n$. *Então* $\mathcal{F}$ *é uma* faceta, *se somente se,* $dim(\mathcal{F}) = dim(P) - 1$.

Facetas são de interesse especial em Combinatória Poliédrica, pois cada faceta corresponde a uma desigualdade distinta no sistema linear que define o poliedro P. O teorema enunciado abaixo justifica esta importância.

Teorema 2.1 *Dados $A \in \mathbb{R}^{m\times n}$, $D \in \mathbb{R}^{p\times n}$, $q \in \mathbb{R}^n$ e $s \in \mathbb{R}^p$. Seja P o poliedro descrito por $\{x \in \mathbb{R}^n : Ax \leq q \text{ e } Dx = s\}$, onde $Ax \leq q$ não contém implicitamente uma igualdade, e D possui posto completo (isto é, número de colunas linearmente independente em D é dado por $min\{p,n\}$). Então essa descrição do poliedro é não-redundante (isto é a remoção de uma desigualdade ou de uma igualdade resulta em um poliedro diferente de P), se somente se, toda desigualdade em $Ax \leq q$ define uma faceta em P.*

Em algumas situações é possível obter desigualdades válidas mais fortes, ou até mesmo facetas, a partir de faces de um poliedro P. Um processo comumente utilizado para isso é o *lifting*. Antes de conceituarmos o que venha a ser um *lifting*, vamos conceituar desigualdades válidas fortes.

Definição 2.25 *Seja $x \in \mathbb{R}^n_+$. Dizemos que uma desigualdade $\gamma x \leq \gamma_0$* domina *outra desigualdade $\pi x \leq \pi_0$, ou seja, $\gamma x \leq \gamma_0$ é* mais forte *do que $\pi x \leq \pi_0$, se existir um $\alpha > 0$, tal que, $\gamma \geq \alpha\pi$ e $\gamma_0 \leq \alpha\pi_0$.*

Procuremos agora entender o significado de *lifting*. Considere a desigualdade válida $\pi x \leq \pi_0$ com respeito ao poliedro $P \subseteq \mathbb{R}^n$, e $\mathcal{F}_{(\pi,\pi_0)}$ a face definida por essa desigualdade em P. Considere ainda a existência de uma outra desigualdade $\beta x \leq \beta_0$, também válida com respeito a P, que define a face $\mathcal{F}_{(\beta,\beta_0)}$. Podemos dizer que a desigualdade $\beta x \leq \beta_0$ é um *lifting* da desigualdade $\pi x \leq \pi_0$ se as duas condições abaixo ocorrem:

1. $\mathcal{F}_{(\pi,\pi_0)} \subset \mathcal{F}_{(\beta,\beta_0)}$;
2. $\dim(\mathcal{F}_{(\pi,\pi_0)}) < \dim(\mathcal{F}_{(\beta,\beta_0)}) \leq \dim(P) - 1$.

Claramente, se uma desigualdade define uma faceta para um poliedro P então ela não pode sofrer o processo de *lifting*.

Definição 2.26 *Seja $S = \{e_1, ..., e_n\}$ um conjunto com n elementos, e U um subconjunto de S. O* vetor de incidência *de U é o vetor $\mathcal{X}^U \in \mathbb{R}^n$, tal que, a i-ésima componente de $\mathcal{X}^U$ é igual a 1 se $e_i \in U$, e 0 caso contrário.*

2.3.1 Caracterização de Facetas

Descreveremos aqui algumas técnicas comumente utilizadas na prova de que desigualdades válidas definem facetas para um dado poliedro.

Considere o poliedro $P = \{x \in \mathbb{R}^n : Ax \leq q, Dx = s\}$; onde $A \in \mathbb{R}^{m\times n}$, $D \in \mathbb{R}^{p\times n}$, $q \in \mathbb{R}^n$ e $s \in \mathbb{R}^p$; e $\pi x \leq \pi_0$ uma desigualdade válida para P. Seja $\mathcal{F}$ a face definida por $\pi x \leq \pi_0$ em P.

Construção direta

Nesta primeira técnica, deve-se mostrar inicialmente que existe $x \in P$, tal que, $\pi x \leq \pi_0$ (isto é, $\{x \in P : \pi x = \pi_0\}$ é uma face própria de P). Neste caso, pode-se dizer que $\dim(\mathcal{F}) \leq \dim(P) - 1$. Em seguida, procura-se por um conjunto de vetores $S \subseteq P$ afim independentes com cardinalidade igual a $\dim(P)$, e tal que, cada elemento de S satisfaz a $\pi x = \pi_0$. Se for possível encontrar este conjunto S, a dimensão de $\mathcal{F}$ é igual a $\dim(P) - 1$ e, então, $\pi x \leq \pi_0$ define uma faceta para P.

Verificação da maximalidade

A segunda técnica consiste em provar que a face $\mathcal{F}$ não está contida em qualquer outra face (*maximalidade*) de P.

Seja $\mathcal{F} = \{x \in P : \pi x = \pi_0\}$. Assuma que $\gamma x \leq \gamma_0$ é uma desigualdade válida qualquer para P, tal que, $\mathcal{F} \subseteq \{x \in P : \gamma x = \gamma_0\}$. Se $\gamma x \leq \gamma_0$ pode ser expresso como uma soma de uma combinação linear das equações em $Dx = s$ e da desigualdade $\pi x \leq \pi_0$ multiplicada por um escalar positivo então $\pi x \leq \pi_0$ é uma faceta para P, ou seja $(\gamma, \gamma_0) = (\beta\pi + \alpha D, \beta\pi_0 + \alpha s)$ para algum $\beta \in \mathbb{R}^n_+$ e $\alpha \in \mathbb{R}^p$.

A caracterização de facetas através da verificação de maximalidade é o método mais comumente utilizado por diversos autores em suas provas de desigualdades que induzem facetas. Nós também escolhemos esse método para a caraterização das novas famílias de facetas encontradas para politopo do Problema da Clique Máxima com Peso nas Arestas nesse trabalho.

Capítulo 3

Trabalhos Relacionados

Neste capítulo apresentaremos alguns problemas de otimização combinatória que podem ser caracterizados como uma generalização ou uma especialização do Problema da Clique Máxima com Peso nas Arestas.

Dividiremos este estudo segundo duas formas de abordagem que podem ser empregadas na resolução de problemas de otimização combinatória: heurística e poliédrica.

3.1 Abordagem Heurística

Uma *heurística* é um algoritmo projetado para encontrar uma boa solução, mas para a qual não se pode dar uma garantia da qualidade em relação à solução ótima. As heurísticas são empregadas para encontrar soluções razoáveis para problemas NP-difíceis.

Uma heurística é única, ou seja, só pode ser aplicada a um problema específico. Dois aspectos devem ser considerados no desenvolvimento de heurísticas: a rapidez e a estratégia de obtenção das soluções. Quando nos referimos a rapidez, exigimos da heurística que ela consiga produzir uma solução "satisfatória" em tempo polinomial na entrada do problema. Já as estratégias costumam ser classificadas como estratégias construtivas e de melhoria (ou de busca local). As heurísticas que descreveremos nesta seção fazem uso de busca local.

Späth [48] projetou duas heurísticas, *Greedy First* (G1) e *Greedy All* (GA), com o intuito de resolver um problema prático: seleção de facilidades. Para isto ele modelou o problema de seleção de facilidades como um problema em grafos que consistia em encontrar uma clique com uma dada cardinalidade, tal que, ela apresentasse peso mínimo.

Problema da clique com peso nas arestas e com uma dada cardinalidade. Seja $K_n = (V_n, E_n)$ um grafo completo não-dirigido com pesos positivos c_{ij} para cada aresta $(i, j) \in E_n$, onde $|V_n| = n$, e b um inteiro também positivo com $1 < b < n$. Deseja-se

encontrar um subconjunto $C \subset V_n$, com $|C| = b$, tal que, $D(C) = \sum_{i \in C} \sum_{j \in C} c_{ij}$ seja mínimo.

Este problema é um caso especial do Problema da Clique Máxima com Peso nas Arestas, pois exige-se que a cardinalidade na clique seja *exatamente* igual a b. Uma vez que os pesos associados às arestas são todos eles positivos é possível realizar o processo descrito abaixo e resolver o problema de seleção de facilidades como o PCMPA:

1. $\mathcal{M} := max\{c_{ij} \text{ com } (i,j) \in E_n\}$;
2. $\mathcal{M} := \mathcal{M} + 1$;
3. alterar os pesos atuais por $\tilde{c}_{ij} := \mathcal{M} - c_{ij}$;
4. a função objetivo passa a ser expressa agora por $Z = max \sum \tilde{c}_{ij}$

Logo, deduz-se que $Z - \mathcal{M}\ b(b-1)/2$ é igual a $min \sum c_{ij}$.

As duas heurísticas se caracterizam por iniciarem a sua execução a partir de uma solução inicial, e procurarem através de melhoramentos sucessivos (busca local) atingir uma solução com peso mínimo. Os melhoramentos sucessivos são obtidos através de um procedimento de perturbação bastante simples: escolhe-se dois vértices quaisquer p e q, tal que, $p \in C$ e $q \notin C$, e realiza-se uma troca entre os dois. Essa troca possibilita a formação de uma nova solução expressa por $\{C \setminus p\} \cup q$.

A diferença básica entre as duas heurísticas é que enquanto a *Greedy First* atualiza a solução corrente à medida que se encontra uma solução melhor na sua vizinhança, a *Greedy All* realiza o processo de atualização somente após todas as possíveis soluções vizinhas terem sido geradas e testadas.

Em seguida apresentamos uma breve discussão das heurísticas.

3.1.1 Descrição da Heurística *Greedy All*

A cada iteração t, consideram-se todas as $b(n-b)$ possíveis trocas que podem ser realizadas pelo procedimento de perturbação. Essas trocas permitem gerar soluções vizinhas da clique atual C^t. Sejam $p \in C^t$ e $q \notin C^t$ vértices, tais que, o par $< p, q >$ corresponde à clique $\tilde{C} = (\{C^t \setminus p\} \cup q)$ que apresenta o menor custo dentre todas as soluções vizinhas a C^t após todas as trocas. Se o custo de $\tilde{C}$ for inferior ao custo de C^t, então a clique atual passa a ser $C^{t+1} = \tilde{C}$, atualiza-se o valor de D, e inicia-se a próxima iteração $t+1$. Por outro lado, se o custo de $\tilde{C}$ for maior do que o da clique atual, a heurística GA pára e retorna a clique atual C^t como solução.

3.1.2 Descrição da Heurística *Greedy First*

A cada iteração t os vértices do grafo são considerados estarem dispostos segundo uma ordem lexicográfica para eventuais trocas (consideraremos como ordem lexicográfica, a ordem dos índices dos vértices no grafo, ou seja, $v_1 < v_2 < \ldots < v_n$). A exemplo da heurística GA são geradas soluções vizinhas $\tilde{C}$ da clique atual C^t, porém sempre que a clique $\tilde{C}$ possuir um custo menor do que a clique atual C^t, C^t é atualizada pela clique $\tilde{C}$ e $D(C^t)$ por $D(\tilde{C})$. Em seguida, continuamos com os procedimentos de pertubação em C^t escolhendo o próximo vértice com maior índice em C^t ainda não considerado para eventuais trocas. Se ao final da iteração t, após realizarmos todas as trocas, obtivermos alguma clique $\tilde{C}$ com custo menor do que o da clique atual C^t atualizamos $C^{t+1} = \tilde{C}$, e inicia-se a próxima iteração $t + 1$. Em caso contrário, a execução de G1 é finalizada, e retornamos a clique atual C^t como solução.

3.2 Abordagem Poliédrica

Nesta seção apresentamos uma pequena noção poliédrica de alguns problemas de otimização combinatória, e traçamos uma relação deles com o PCMPA. Através disto demonstraremos que as soluções viáveis do PCMPA estão contidas no conjunto de soluções viáveis destes problemas, e portanto as desigualdades válidas definidas para aqueles problemas poderão ser empregadas no estudo da caracterização da estrutura facial do PCMPA.

Iniciamos este estudo com problemas de particionamento de grafos, em seguida apresentamos dois problemas quadráticos: o booleano e o da mochila.

3.2.1 Problemas de Partição em um Grafo

Um problema de particionamento de um grafo $G = (V, E)$, com peso associado às arestas, consiste em agrupar os vértices de V em blocos, tal que, o somatório dos pesos das arestas que ligam vértices em blocos distintos seja mínimo.

Este problema é conhecido ser NP-difícil (veja Garey e Johnson [21]) e, tanto métodos heurísticos quanto abordagem poliédrica já foram empregados para a sua resolução. Em particular, nos deteremos na abordagem poliédrica.

Chopra e Rao [7, 8, 9] estudaram o problema de particionamento de um grafo e definiram esse problema como abaixo

Problema de particionamento de um grafo. Dado um grafo conexo $G = (V, E)$ com pesos c_e para toda aresta $e \in E$, pretende-se particionar o conjunto dos vértices V em

r subconjuntos não-vazios, tal que, o somatório dos pesos das arestas com extremos em blocos distintos (arestas de corte) seja mínimo.

De acordo com a definição anterior, pode-se deduzir que minimizar o somatório dos pesos das arestas de corte é equivalente a maximizar o somatório dos pesos das arestas em um mesmo bloco. Em seguida, enumeramos alguns casos do problema de particionamento em grafos relatados por aqueles autores. Antes porém, vamos definir uma r-partição.

Definição 3.1 *Considere que B_i represente um bloco de vértices. Dizemos que $\mathcal{T} = (B_1, B_2, \ldots, B_r)$ define uma* r-partição *do conjunto de vértices V se*

(i) $B_i \neq \emptyset$ *para todo* $i = 1, 2, \ldots, r$;
(ii) $B_i \cap B_j = \emptyset$ *para todo* $i \neq j$;
(iii) $\cup_{i=1}^{r} B_i = V$.

De acordo com o tipo de restrição que se impõe quanto ao número de subconjuntos, o problema de particionamento em grafos pode apresentar algumas modificações. Considere os casos abaixos, tal que, p seja um inteiro:

(a) deseja-se particionar o conjunto de vértices V em r subconjuntos, tal que, $r \leq p$;
(b) deseja-se particionar o conjunto de vértices V em r subconjuntos, tal que, $r \geq p$;
(c) deseja-se particionar o conjunto de vértices V em exatamente r subconjuntos.

Chopra e Rao [8] deduziram algumas equivalências entre os problemas definidos em (a), (b) e (c), e apresentaram casos em que o problema de particionamento de um grafo pode ser resolvido em tempo polinomial. Em particular, podemos estabelecer uma relação entre o problema definido no item (c) e o PCMPA:

Afirmação. Seja $K_n = (V_n, E_n)$ um grafo completo não-dirigido. O problema definido em (c) pode ser visto como uma generalização do PCMPA se estabelecermos que: o valor de r é fixado em $r = n - b + 1$, e adicionarmos a restrição de que nenhum dos r blocos pode conter mais do que b vértices.

A Figura 3.1 ilustra um exemplo bastante simples que irá facilitar a compreensão da relação estabelecida acima. Deseja-se particionar um grafo completo $K_n = (V_n, E_n)$ não-dirigido, com $|V_n| = 7$, tendo cada aresta $e \in E_n$ um peso $c_e = 1$ associado. Pretendemos encontrar uma clique com cardinalidade $b = 4$ e com peso máximo. Como dito anteriormente minimizar o somatório das arestas de corte corresponde a maximizar o somatório

das arestas em um bloco. O conjunto de vértices V_n foi particionado em quatro blocos, como indicado na figura, sendo que três blocos possuem cardinalidade igual a 1 e o último cardinalidade igual a b.

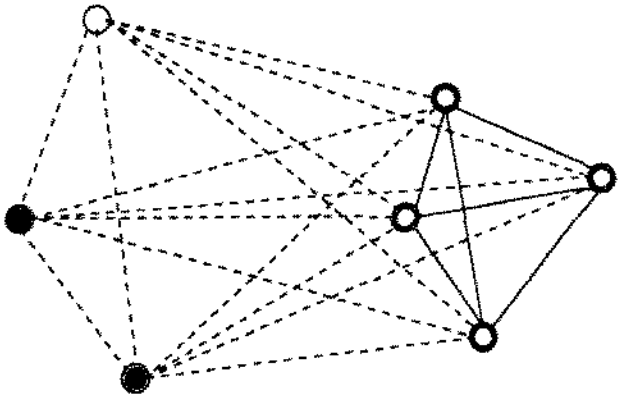


Figura 3.1: Particionamento de um grafo completo em $n - b + 1$ blocos.

O problema de particionamento de grafos foi estudado por vários pesquisadores. Em particular, citaremos neste trabalho apenas aqueles estudos que possuem alguma relação com o Problema da Clique Máxima com Peso nas Arestas.

Particionamento em Cliques

O problema de particionamento de um grafo $G = (V, E)$ em cliques foi estudado por Grötschel e Wakabayashi [26, 27]. Estes autores definiram o problema como segue

Problema de particionamento em cliques. Um conjunto de arestas $\mathcal{E}$ em um grafo $G = (V, E)$ é dito ser um particionamento em cliques de G se existir uma partição $\mathcal{T} = (B_1, B_2, \ldots, B_r)$ do conjunto de vértices V, tal que, $\mathcal{E} = \cup_{i=1}^{r} E(B_i)$ com $E(B_i) = \{(t, u) : t \in B_i \text{ e } u \in B_i\}$, e o subgrafo induzido por B_i seja uma clique para todo $i = 1, 2, \ldots, r$.

Atribuindo-se pesos $c_e \in \mathbb{R}$ para toda aresta $e \in E$, o problema de particionamento em cliques consiste em encontrar uma partição $\mathcal{E}$ de G, tal que, $\sum_{e \in \mathcal{E}} c_e$ seja mínimo.

Consideremos agora o estudo poliédrico deste problema. Associado a cada conjunto de arestas $\mathcal{E} \subseteq E$ pode-se definir o seguinte vetor de incidência: $\mathcal{X}_e^{\mathcal{E}}$ é igual a 1, se a aresta $e \in \mathcal{E}$; e $\mathcal{X}_e^{\mathcal{E}}$ é igual a 0, em caso contrário. Grötschel e Wakabayashi propuseram a seguinte formulação inteira para o problema de particionamento de um grafo em cliques.

$$\min \sum_{i,j,i<j} c_{ij} y_{ij}$$

s.a.:

$$y_{ij} + y_{jk} - y_{ik} \leq 1 \quad \forall i \forall j \forall k \; i < j < k$$
$$y_{ij} \in \{0,1\} \quad \forall i \forall j \; i < j$$

Pode-se verificar que uma solução inteira (0-1) desta formulação corresponde a uma partição em cliques em um grafo completo não-dirigido K_n. Sendo assim, pode-se deduzir que toda solução viável do PCMPA é uma solução viável do particionamento de um grafo em cliques, uma vez que uma solução do PCMPA corresponde à uma partição do grafo completo em $(n-b)$ cliques com tamanho igual a 1 e uma única clique com cardinalidade igual a b.

O problema de particionamento de grafos que mostraremos em seguida foi estudo por Faigle *et al.* [16], e se aproxima um pouco mais do Problema da Clique Máxima com Peso nas Arestas, pois restringe-se a cardinalidade máxima em cada bloco da partição.

Particionamento b-restrito

O problema de partição estudado por Faigle *et al.* [16] distingue daquele estudado por Grötschel e Wakabayashi em dois aspectos: deseja-se maximizar o somatório dos pesos das arestas em um mesmo bloco, e impõe-se uma cardinalidade máxima para cada bloco. Os autores definiram o problema de particionamento b-restrito de um grafo como sendo

Problema de particionamento b-restrito. Dado um grafo $G = (V, E)$ com pesos $c_e \geq 0$ para cada aresta $(i,j) \in E$ e um inteiro $b \geq 1$, deseja-se encontrar uma partição $\mathcal{T}$ do conjunto de vértices V, tal que, $|B_i| \leq b$ e o somatório dos pesos das arestas de corte seja mínimo.

Caso o grafo não seja completo associa-se $c_e = 0$ para toda as arestas $e \notin E$. Apesar de terem definido o problema como sendo de minimização, os autores optaram por maximizar o somatório das arestas em um mesmo bloco (devido a função objetivo dos problemas práticos que eles se propuseram a resolver ser de maximização).

Considere a formulação inteira abaixo proposta por Faigle *et al.* para o problema

$$\max \sum_{i,j,i<j} c_{ij} y_{ij}$$

s.a.:

$$\begin{aligned} & y_{ij} + y_{jk} - y_{ik} \leq 1 && \forall i \forall j \forall k \ i < j < k \\ & \sum_{j,j>i} y_{ij} + \sum_{j,j<i} y_{ji} \leq b - 1 && \forall i \in V_n \\ & y_{ij} \in \{0,1\} && \forall i \forall j \ i < j \end{aligned}$$

onde $y_{ij} = 1$, se a aresta $(i,j) \in E$ não pertence ao corte; e $y_{ij} = 0$, em caso contrário.

Como pode-se observar qualquer solução inteira (0-1) define uma partição em cliques com no máximo b vértices. Portanto, o PCMPA é um caso especial deste problema, pois para chegarmos a uma solução viável do PCMPA basta acrescentarmos desigualdades que impeçam a formação de subcliques.

3.2.2 Problema Quadrático Booleano

Padberg [38] estudou o problema quadrático não-restrito 0-1 a partir do ponto de vista poliédrico do problema quadrático booleano. O problema quadrático não-restrito 0-1 pode ser definido como abaixo:

$$max\{cx + x^T Q x : x \in \{0,1\}\} \qquad \text{(P1)}$$

onde c é um vetor de tamanho n e Q é uma matriz de tamanho $n \times n$.

Através de um processo de linearização bastante simples, Padberg conseguiu transformar o programa quadrático 0-1 anterior em um problema de programação linear inteira 0-1. Considere novas variáveis y_{ij} definidas para o problema, tais que, $y_{ij} = x_i x_j$ e satisfaçam as desigualdades abaixo

$$\begin{aligned} y_{ij} &\leq x_i && \forall i \forall j \ i < j \\ y_{ij} &\leq x_j && \forall i \forall j \ i < j \\ x_i + x_j - y_{ij} &\leq 1 && \forall i \forall j \ i < j \\ x_i &\in \{0,1\} \\ y_{ij} &\in \{0,1\} \end{aligned}$$

Com base nessas novas informações o problema (P1) pode ser formulado como o seguinte programa linear (0-1).

$$\max\ cx + \sum_{i,j,i<j} q_{ij} y_{ij}$$

s.a.:

$$\begin{aligned}
& y_{ij} \leq x_i && \forall i \forall j\ i < j \quad \text{(I)} \\
& y_{ij} \leq x_j && \forall i \forall j\ i < j \quad \text{(II)} \\
& x_i + x_j - y_{ij} \leq 1 && \forall i \forall j\ i < j \quad \text{(III)} \\
& y_{ij} \in \{0,1\} && \forall i \forall j\ i < j \\
& x_i \in \{0,1\} && \forall i
\end{aligned}$$

O problema anterior é denominado *problema quadrático booleano*. Através deste conjunto de desigualdades é possível garantir que a formulação acima contém todas as soluções inteiras (0-1) viáveis do problema de particionamento em cliques. Isto porque cada desigualdade $y_{ij} + y_{jk} - y_{ik} \leq 1$ pode ser obtida pela soma das desigualdades (I), (II) e (III). Portanto, o PCMPA pode ser visto como uma versão linearizada do problema quadrático booleano se ignorarmos a restrição de cardinalidade $\sum_i x_i \leq b$.

Outras abordagens para o problema quadrático restrito 0-1 podem ser encontradas nos trabalhos de Boros e Hammer [5] e De Simone [46].

3.2.3 Problema Quadrático da Mochila

Johnson *et al.* [30] estudaram o problema do *min-cut clustering* que é um problema de particionamento em grafos (não necessariamente completo) com pequenas diferenças: o acréscimo de uma desigualdade que limita a capacidade de cada bloco e pesos w_i associado a cada vértice $i \in V$.

Considere as seguintes variáveis de decisão: $x_i^k = 1$, se vértice $i \in B_k$; e $x_i^k = 0$, em caso contrário; $y_{ij}^k = 1$, se a aresta $(i,j) \in E$ pertence ao bloco B_k; e $y_{ij}^k = 0$, no contrário.

Baseado nesse vetor de incidência Johnson *et al.* propuseram a seguinte formulação inteira para o *min-cut clustering*.

$$\max \sum_{k=1}^{K} \sum_{i,j,i<j} c_{ij} y_{ij}^k$$

s.a.:

$$\begin{array}{lll}
y_{ij}^k \leq x_i^k & \forall i \forall j \ i<j & \forall k = 1,\ldots,K \\
y_{ij}^k \leq x_j^k & \forall i \forall j \ i<j & \forall k = 1,\ldots,K \\
x_i^k + x_j^k - y_{ij}^k \leq 1 & \forall i \forall j \ i<j & \forall k = 1,\ldots,K \\
\sum_i x_i^k = 1 & \forall k = 1,2,\ldots,K & \\
\sum_i x_i^k \geq 1 & \forall k = 1,2,\ldots,K & \\
\sum_i w_i x_i^k \leq W & \forall k = 1,2,\ldots,K & \text{(IV)} \\
y_{ij}^k \in \{0,1\} & \forall i \forall j \ i<j & \forall k = 1,\ldots,K \\
x_i^k \in \{0,1\} & \forall i & \forall k = 1,\ldots,K
\end{array}$$

onde K é o maior número de blocos em uma solução viável.

A desigualdade (IV) é denominada *desigualdade da mochila* e garante que a capacidade em cada bloco não exceda um certo valor inteiro $W > 0$. Johnson *et al.* propuseram uma formulação mais forte para o *min-cut clustering* baseado no processo de decomposição Dantizg-Wolf. Nessa decomposição foi criado um problema mestre e um subproblema de otimização para a geração de colunas.

Para o problema mestre foi proposta uma formulação onde havia a necessidade da geração de blocos (*clusters*) viáveis, ou seja, geração de colunas que correspondiam a soluções viáveis. Porém, devido ao grande número de colunas que poderiam ser geradas foi necessário a criação de um subproblema, denominado *problema quadrático da mochila*. A formulação inteira para esse subproblema é composto pelas restrições $y_{ij} \leq x_i$, $y_{ij} \leq x_j$ e $\sum_i w_i x_i \leq W$, pois desde que $c_e \geq 0$ temos $x_i + x_j - y_{ij} \leq 1$ satisfeita na otimalidade. Com isto soluções viáveis para o problema do *min-cut clustering* são obtidas. Fica evidente, que se $\sum_i w_i x_i \leq W$ for eliminada da formulação inteira proposta para o subproblema de otimização temos o problema quadrático booleano, e portanto, o PCMPA é um caso especial do problema quadrático da mochila.

Capítulo 4

Técnicas Algorítmicas em Programação Linear Inteira

No capítulo 2 foi mostrado que um problema de otimização combinatória pode ser formulado como um problema de programação linear, o que torna possível a obtenção de uma descrição linear parcial do politopo associado ao problema através de um conjunto de desigualdades lineares. Além do mais, tendo em mãos essas desigualdades podemos utilizar técnicas algorítmicas comuns em programação linear na resolução do problema.

Embora programação linear inteira seja um problema pertencente à classe dos problemas NP-difíceis (veja Garey e Johnson [21]), o crescente desenvolvimento de algoritmos empregados na resolução desses tipos de problemas possibilitou que problemas de otimização combinatória, formulados como problemas de programação linear inteira, pudessem ser resolvidos de forma satisfatória.

Nesse capítulo, descreveremos algumas das técnicas algorítmicas comumente empregadas na resolução de problemas de programação linear inteira. As duas primeiras técnicas são clássicas: algoritmos *cutting-planes* (planos-de-corte) e procedimentos *branch-and-bound*. Enquanto que a terceira, algoritmos *branch-and-cut*, é recente e obtida a partir de uma combinação das técnicas tradicionais citadas.

4.1 Algoritmos de Planos-de-Corte

Os algoritmos de planos-de-corte foram introduzidos por Gomory na década de 50 para a resolução de problemas de programação linear inteira mista ou pura. Em um algoritmo de planos-de-corte, novas desiguldades são acrescidas iterativamente à relaxação linear. Tais desigualdades produzem uma relaxação linear mais forte e ao mesmo tempo não eliminam qualquer solução inteira.

O princípio de um algoritmo de planos-de-corte é baseado em um processo sistemático

bastante simples. Considere a formulação inteira max $\{cx : x \in P, x \text{ é inteiro}\}$ para um poliedro P, e um conjunto de desigualdades que são válidas para todas as soluções inteiras. Resolvendo-se a relaxação linear do programa anterior suponha que $\overline{x}^*$ seja a solução obtida. Se $\overline{x}^*$ é um vetor de incidência viável então $\overline{x}^*$ é a solução ótima inteira para o problema de otimzação combinatória. Caso contrário, procura-se uma desigualdade pertencente a uma classe de desigualdades válidas conhecidas para o poliedro P que seja violada por $\overline{x}^*$, isto é, $\pi\overline{x}^* > \pi_0$ (embora $\pi x \leq \pi_0$ para todo x inteiro). Adiciona-se então a desigualdade violada ao programa linear e volta-se a resolver a nova relaxação. Esse processo se repete até que se encontre uma solução ótima inteira ou caso não seja mais possível encontrar desigualdades válidas violadas.

Este é o princípio básico dos algoritmos de planos-de-cortes cujo nome é originário do fato de que as novas desigualdades adicionadas à relaxação linear eliminam (*cut-off*) a solução fracionária obtida. Assim, um algoritmo de planos-de-corte pode ser visto como um procedimento para reescrever ou reformular o problema original de tal maneira que o conjunto de soluções inteiras seja preservado e a relaxação linear fique mais justa. Ou seja, o acréscimo dos planos-de-corte procura trazer a relaxação linear mais próxima da envoltória convexa formada pelas soluções inteiras de P (fecho inteiro).

A Figura 4.1 ilustra o funcionamento de um algoritmo de planos-de-corte. Considere que o problema é um problema de maximização. As linhas cheias representam desigualdades válidas que descrevem o poliedro inicial (relaxação linear), a seta indica a direção da função objetivo, os círculos indicam pontos inteiros, os círculos cheios indicam soluções viáveis, e por último, a linha tracejada representa uma desigualdade adicionada à relaxação linear que elimina a solução fracionária sem eliminar qualquer solução viável.

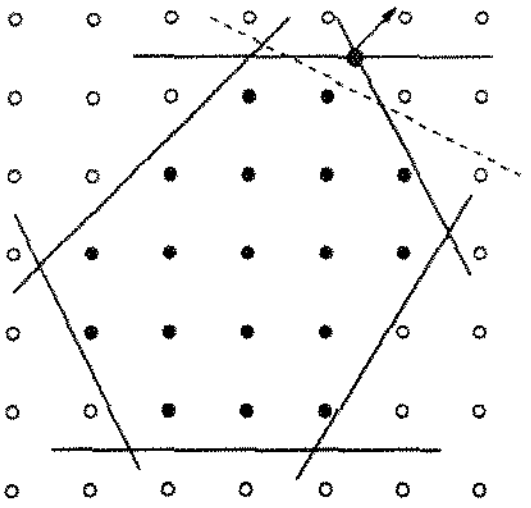


Figura 4.1: Funcionamento de um algoritmo de planos-de-corte.

Note que o fato importante em algoritmos de planos-de-corte reside em termos um método apropriado para identificar aquelas desigualdades que são válidas para o problema original, mas violadas pela solução fracionária atual. Assim, se conhecemos uma classe de desigualdades válidas, devemos ser capazes de verificar se uma desigualdade dessa classe é violada, ou seja devemos resolver um problema de separação. Nos detalharemos melhor com problemas de separação no final da seção onde estabeleceremos uma relação com problemas de otimização.

Citaremos agora alguns tipos de planos-de-corte comumente utilizados quando trabalhamos com algoritmos de planos-de-corte.

4.1.1 Planos-de-Corte Gerais

Os planos-de-corte que apresentaremos aqui podem ser aplicados a qualquer problema de programação linear inteira pura ou mista. Como esses tipos de planos-de-corte não são específicos a um dado problema eles recebem o nome de planos-de-corte gerais ou de propósito geral.

Gomory [22] introduziu os primeiros algoritmos de planos-de-corte. Naquele trabalho, ele provou que o algoritmo termina após um número finito de iterações e que o programa inteiro pode ser resolvido através da geração de um número também finito de planos-de-corte.

Infelizmente, os planos-de-corte de Gomory não se mostraram eficazes em experimentos práticos. Os planos-de-corte eram "fracos" e causavam instabilidade númerica, e apenas em problemas de pequeno porte era possível atingir a otimalidade.

Os planos-de-corte gerais são muitos limitados quando os utilizamos em problemas de otimização combinatória. Para esses tipos de problemas recomenda-se o emprego de planos-de-corte projetados de acordo com a estrutura do problema que se deseja resolver. Na próxima subseção, abordaremos esses tipos de planos-de-corte denominados planos-de-corte faciais.

4.1.2 Planos-de-Corte Faciais

Os planos-de-corte faciais são desigualdades válidas definidoras de facetas para um poliedro P. Como as facetas não são dominadas por quaisquer outras desigualdades válidas e ainda fornecem uma descrição completa e não-redundante do poliedro, elas podem ser consideradas como sendo os melhores tipos de plano-de-corte.

Entretanto, dois aspectos devem ser considerados: não é trivial encontrarmos desigualdades válidas definidoras de facetas, e segundo, mesmo que tenhamos uma classe de desigualdades válidas devemos ser capazes de projetar algoritmos de separação para encontrá-las.

A Figura 4.2 ilustra a idéia de um algoritmo de planos-de-corte baseado no uso de facetas. A região limitada pelas linhas mais fraca representa o fecho inteiro do poliedro de um problema de maximização, e a limitada pelas linhas mais forte a relaxação linear. Na Figura 4.2(a) a solução fracionária obtida na relaxação linear é cortada pela faceta representada pela linha pontilhada. Na Figura 4.2(b) mais uma vez a solução fracionária obtida na relaxação linear (fortalecida) é cortada por outra faceta. Embora não tenhamos na Figura 4.2(c) uma descrição completa do fecho inteiro do poliedro P, a solução obtida pertence ao fecho inteiro, e portanto representa a solução ótima inteira para o problema.

4.1.3 Problemas de Separação e Otimização

Como pode-se verificar, o ponto crucial em algoritmos de planos-de-corte é a possibilidade de identificarmos a cada iteração desigualdades que estejam sendo violadas. O problema de gerar tais desigualdades violadas é denominado problema de separação.

Grötschel, Lovász e Schrijver [24, 25] estabeleceram uma relação entre o problema de separação e o problema de otimização, antes porém vamos definir cada um desses problemas.

Problema de separação. Dado um ponto $y \in \mathbb{R}^n$ e um poliedro $P \subseteq \mathbb{R}^n$, devemos decidir se y pertence ou não a P. Se y não pertence a P, deve-se encontrar uma desigualdade $\pi x \leq \pi_0$ que é válida para P e tal que $\pi y > \pi_0$.

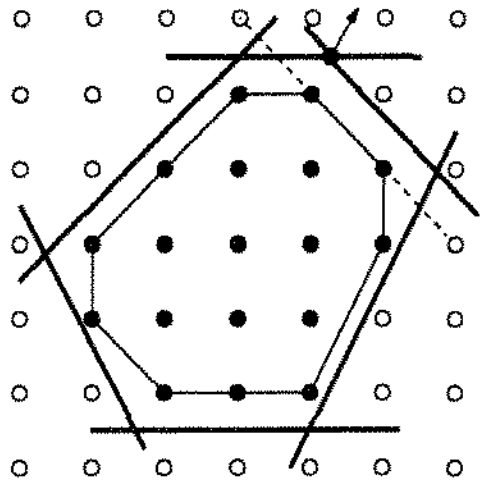
Problema de otimização. Dado um poliedro $P \subseteq \mathbb{R}^n$ e um vetor $c \in \mathbb{R}^n$, ou encontramos um $x^* \in P$, tal que maximize cx para todo $x \in P$, ou concluímos que P é vazio.

Teorema 4.1 *(Grötschel, Lovász e Schrijver) Para qualquer classe de poliedros, existe um algoritmo polinomial para o problema de separação, se e somente se, existe um algoritmo de tempo também polinomial para o problema de otimização.*

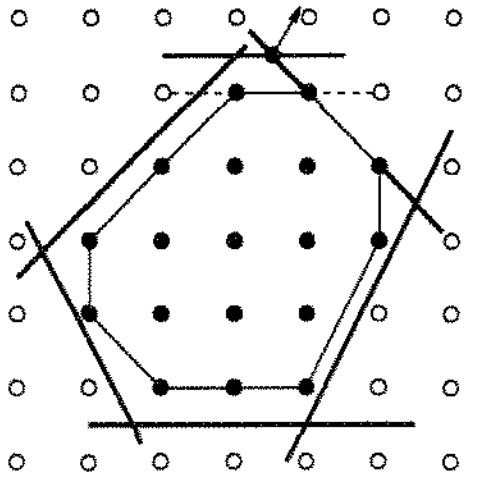
A partir do teorema enunciado acima pode-se deduzir que (de Souza [13]):

1. se a envoltória convexa de um poliedro P pode ser completamente descrita usando somente desigualdades válidas, e o problema de programação linear inteira é resolvido em tempo polinomial então o problema de separação também é resolvido em tempo polinomial;

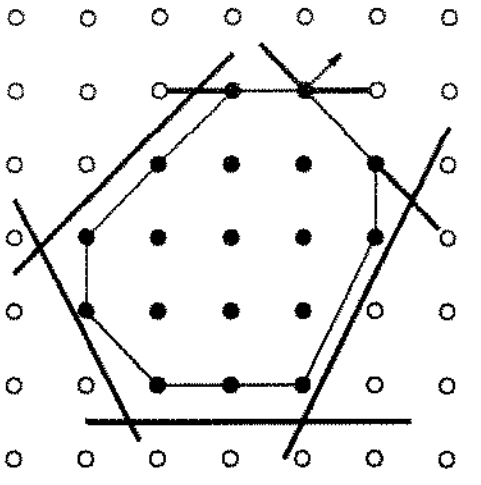
2. entretanto, se o problema é NP-difícil existirão apenas algumas desigualdades válidas que poderão ser geradas em tempo polinomial. No entanto, existirão outras desigualdades válidas para as quais os problemas de separação são NP-difíceis.



(a)



(b)



(c)

Figura 4.2: Funcionamento de um algoritmo de planos-de-corte faciais.

Em (1) podemos deduzir que a solução fracionária será sempre cortada por alguma desigualdade válida, já em (2) pode-se deduzir que para algumas instâncias do problema de programação linear inteira o algoritmo de planos-de-corte sempre terminará com uma solução fracionária.

4.2 Procedimentos *Branch-and-Bound*

Na seção anterior observamos que algoritmos de planos-de-corte nem sempre conseguem atingir uma solução ótima inteira para um problema de otimização combinatória. Nesse aspecto podemos empregar os procedimentos *branch-and-bound*. Para um melhor entendimento dessa técnica sugere-se Ceria [6] e Nemhauser e Wolsey [36].

Os procedimentos *branch-and-bound* realizam uma enumeração implícita das soluções do problema de otimização combinatória através do particionamento sucessivo do espaço de soluções em diversos subespaços na tentativa de se resolver o problema original. Essa forma recursiva de particionamento do problema pode ser visualizada como uma árvore denominada árvore *branch-and-bound*, onde cada nó dessa árvore corresponderia a um subproblema com o seu subespaço definido.

Nas próximas linhas explicaremos de uma maneira geral o uso de procedimentos *branch-and-bound* em programação linear inteiral; antes, explicaremos a notação que será utilizada. A variável Z_{IP} denotará o valor da melhor solução inteira conhecida no momento, Z^i_{RL} o valor da relaxação linear obtida em cada subproblema i, e por último, P denotará o conjunto de soluções (inteiras) viáveis para o problema de otimização combinatória a ser resolvido.

Suponha que se esteja resolvendo um problema de maximização. Seja i um nó da árvore *branch-and-bound*. A cada nó i podemos associar um *bound* que representa um limite para as soluções viáveis do subespaço correspondente ao nó i. Esse limite é obtido resolvendo-se a relaxação linear RL^i associado ao nó i e dada por:

$$Z^i_{RL} = \max\{cx : x \in P^i_{RL}\}$$

onde $P^i_{RL} = \{x \in \mathbb{R}^n_+ : A^i x \leq b^i\}$.

Se o valor Z^i_{RL} for menor do que Z_{IP}, então a exploração do nó i é finalizada, pois as chances de encontrarmos uma solução viável com valor maior do que o valor Z_{IP} nos subespaços gerados a partir de i são nulas. Caso o valor da solução ótima Z^i_{RL} seja inteiro, então é possível realizar duas operações: verificar se Z_{IP} pode ser atualizado e em seguida finalizar a exploração sob o nó i, pois encontramos uma solução inteira viável para o

problema em questão. Em ambos os casos citados anteriormente o nó i é considerado estar *maduro*.

Por outro lado, se o valor Z^i_{RL} for maior do que o valor Z_{IP}, então é possível realizar um *branching* sob o nó i. *Branching* é um processo que permite gerar novos subproblemas. Esse processo aumenta o número de desigualdades da relaxação linear gerado para cada novo subproblema. Em particular, para um problema de programação linear 0-1, o que se estabelece é a criação de um subproblema P_{j0} obtido quando fixamos uma variável x_j igual a 0, e um outro subproblema P_{j1} quando a variável x_j recebe o valor 1.

O procedimento *branch-and-bound* finalizará quando todos os nós da árvore *branch-and-bound* estiverem maduros.

4.3 Algoritmos *Branch-and-Cut*

Uma das técnicas recentes nos últimos anos e comumente empregada na resolução de problemas de otimização combinatória é o algoritmo *branch-and-cut*. Nessa seção, explicaremos o princípio de funcionamento de um algoritmo *branch-and-cut* e alguns aspectos computacionais que podem ser utilizados quando projetamos algoritmos dessa natureza.

Em um algoritmo *branch-and-cut* além do uso de relaxações lineares em cada nó da árvore *branch-and-bound*, utiliza-se também planos-de-corte específicos ao problema. Portanto, um algoritmo *branch-and-cut* é constituído de uma fase com planos-de-corte dentro do procedimento *branch-and-bound*.

Os primeiros trabalhos que fizeram uso dessa técnica foram: Miliotis [35] que utilizou pela primeira vez desigualdades válidas e procedimentos *branch-and-bound*; Grötschel *et al.* [23] passaram a gerar automaticamente planos-de-corte definidoras de facetas em combinação com procedimentos *branch-and-bound*; e Padberg e Rinaldi [39] que introduziram a nomenclatura *branch-and-cut* e o uso de planos-de-corte não apenas no nó raiz, mas em todos os nós da árvore.

Descreveremos agora o princípio de funcionamento de um algorimo *branch-and-cut*. Considere que o problema a ser resolvido seja de maximização. Em uma iteração típica do algoritmo nós estamos em um nó i da árvore *branch-and-bound* e $P^i = \{x \in \mathbb{R}^n_+ : A^i x \leq b \text{ e } 0 \leq x \leq 1\}$ é o poliedro correspondente a relaxação linear RL^i. Se $\overline{x}_i$ é a solução obtida nessa relaxação linear, e ela tem valor fracionário devemos então tentar resolver esse problema de separação, ou seja procurar por uma desigualdade $\pi x \leq \pi_0$ que corte fora a solução fracionária. Caso tenha sido possível gerar uma desigualdade do tipo $\pi x \leq \pi_0$, essa desigualdade será adicionada ao sistema de desigualdades que definem P^i, e RL^i será resolvida novamente. Esse processo se repete até que: ou $\overline{x}_i$ possua valor inteiro, ou a solução para todo nó i é menor do que o *bound* calculado, ou não foi possível encontrar uma desigualdade válida $\pi x \leq \pi_0$. A regra de *branching* é aplicada nessa última situação,

enquanto que nas outras duas situações o nó é considerado estar *maduro* e parte-se para a seleção de um novo nó i.

Durante a execução do algoritmo *branch-and-cut* pode ocorrer que o acréscimo de planos-de-corte não traga qualquer aumento no valor da função objetivo ou este aumento pode ser muito pequeno. Nesse caso, opta-se por realizar um *branching* mesmo que futuros cortes possam ser encontrados. Tal fenômeno é conhecido como *tailing-off* e foi detectado por Padberg e Rinaldi [39] em trabalhos com o problema do caixeiro viajante.

Algoritmos de *branch-and-cut* só passaram a obter sucesso quando se percebeu que a eficiência da fase de planos-de-corte é fortemente dependente da capacidade das desigualdades válidas utilizadas aproximarem a relaxação linear do fecho inteiro do poliedro associado a um problema de otimização. Isso fortaleceu o uso de desigualdades válidas definidoras de facetas uma vez que já vimos que elas são obtidas de forma única, e portanto computacionalmente interessantes.

Finalizaremos este capítulo apresentando alguns dos aspectos computacionais presentes durante o projeto de um algoritmo *branch-and-cut*.

4.3.1 Aspectos Computacionais

Para maiores detalhes sobre os tópicos que serão apresentados aqui sugere-se Ferreira e Wakabayashi [20] e Jünger *et al.* [33]. Assuma que o problema que estamos resolvendo seja de maximização.

1. *Limitantes.* O cálculo de bons limitantes é um fator preponderante, pois isso pode diminuir o número de nós da árvore *branch-and-bound* e torna mais eficiente o processo de busca nessa árvore. O limitante inferior inicial pode ser obtido através de heurísticas que forneçam boas soluções para o problema. Já o limitante superior inicial pode ser obtido através de uma relaxação linear inicial para o problema que possua um número de desigualdades não muito grande.

2. *Seleção de variáveis.* Quando um nó não pode ser rotulado como maduro devemos selecionar uma variável e gerar novos descendentes na árvore *branch-and-bound*. O processo escolhido para a seleção de uma variável influencia de forma significativa no tamanho da árvore *branch-and-bound*.

3. *Seleção de nós.* A etapa de seleção de nós pode ser vista como uma maneira de se estabelecer uma ordem dos nós da árvore *branch-and-bound* ainda não explorados. A exemplo da seleção de variáveis, a estratégia escolhida para a seleção do nó influencia no tamanho da árvore *branch-and-bound*.

4. *Rotinas de separação eficientes.* O sucesso de um algoritmo *branch-and-cut* está ligado à escolha de boas desigualdades válidas que funcionarão como planos-de-corte, uma vez que essas desigualdades irão tornar a relaxação linear mais justa (próxima do fecho inteiro), e conseqüentemente a obtenção de melhores limitantes superiores. Porém, é necessário resolver o problema de separação, ou seja, projetar algoritmos eficientes. A eficiência aqui se refere no aspecto dos algoritmos serem rápidos e, em boa parte da execução, encontrar planos-de-corte que eliminem a solução fracionária obtida na relaxação linear.

5. *Remoção e inserção de desigualdades.* Durante o processo de inserção de novas desigualdades deve-se procura estabelecer uma hierarquia entre as classes de desigualdades válidas e um número máximo de desigualdades que poderão ser geradas a cada iteração da fase de planos-de-corte. Já no processo de remoção procura-se encontrar aquelas desigualdades que não estão mais ativas em um nó da árvore *branch-and-bound.* Porém, um cuidado deve ser tomado no processo de remoção, pois uma desigualdade inativa em um nó pode estar ativa em outro nó. Para isto, sugere-se a implementação de um *pool* de desigualdades que armazenará as desigualdades inativas, a fim de que, ao se escolher um novo nó, as desigualdades existentes neste *pool* possam ser testadas e eventualmente reinseridas à relaxação linear.

6. *Heurísticas primais.* As soluções fracionárias obtidas a partir da relaxação linear podem nos fornecer informações utéis na determinação de soluções viáveis boas. Ou seja, tendo em mãos uma solução fracionária com algumas variáveis com valores iguais a 0 ou 1, ou até mais próximos desses valores, é possível projetar heurísticas de melhoria, denominadas heurísticas primais, baseadas nesses valores para a obtenção de bons limitantes inferiores.

7. *Fixação de variáveis.* Dada uma solução fracionária, é possível em determinados instantes, verificar se uma variável pode ser incluída ou excluída da solução ótima. Assim, poderíamos fixar o valor daquela variável em 0 ou 1. Através do processo de fixação de variáveis pode-se diminuir o tamanho da relaxação linear o que proporcionará um ganho significativo no desempenho do algoritmo *branch-and-cut.*

Capítulo 5

Estudo Poliédrico do Problema da Clique Máxima com Peso nas Arestas

Um dos principais objetivos deste trabalho consiste em estudar e comparar diferentes formulações do Problema da Clique Máxima com Peso nas Arestas como um problema de programação linear inteira. Para isto, realizamos neste capítulo um estudo do politopo associado ao problema.

São apresentadas duas formulações inteiras, cada uma definida sob um modelo, e as famílias de facetas conhecidas para os politopos associados a ambos os modelos. Em seguida, são apresentadas novas famílias de facetas que foram encontradas ao longo deste trabalho.

5.1 Formulações de Programação Linear Inteira

Nesta seção apresentaremos duas formulações inteiras para o PCMPA e uma caracterização parcial do politopo associado ao problema obtida a partir destas formulações. A primeira formulação trabalha com as variáveis de decisão definidas apenas sobre as arestas (*modelo natural*). Já a segunda, além das variáveis naturais (arestas), utiliza-se também das variáveis definidas sobre os vértices (*modelo estendido*).

5.1.1 Modelo Natural

Estamos considerando o estudo do Problema da Clique Máxima com Peso nas Arestas do ponto de vista poliedral. Seja (K_n, c, b) uma instância do problema, onde $K_n = (V_n, E_n)$ é um grafo completo não-dirigido, com $|V_n| = n$ e $|E_n| = m = n(n-1)/2$, c é uma função

que atribui peso a cada aresta $e \in E_n$ e b indica a cardinalidade da clique procurada. Vale lembrar que para o modelo em questão não é permitido a presença de cliques triviais, ou seja assume-se $b \geq 2$. Considerando-se a instância acima é possível introduzirmos as seguintes variáveis de decisão

$$y_{ij} = \begin{cases} 1 & \text{se a aresta } (i,j) \text{ pertence a clique.} \\ 0 & \text{caso contrário.} \end{cases}$$

O politopo associado ao Problema da Clique Máxima com Peso nas Arestas, denotado por PCN_b, corresponde então à envoltória convexa de todas as soluções viáveis (inteiras) para o problema. A princípio, o Problema da Clique Máxima com Peso nas Arestas poderia ser resolvido como

$$max\{cy : y \in PCN_b\}.$$

Entretanto, para que seja possível o uso de técnicas de programação linear inteira na resolução do Problema da Clique Máxima com Peso nas Arestas, é preciso que tenhamos uma descrição do PCN_b através de um conjunto de desigualdades lineares. Iniciamos essa caracterização parcial do PCN_b através da formulação inteira proposta por Faigle *et al.* [15]. Em seguida, apresentaremos outras desigualdades válidas definidoras de facetas para o PCN_b que podem ser empregadas na obtenção de uma melhor caracterização deste politopo.

Faigle *et al.* [15] formularam o Problema da Clique Máxima com Peso nas Arestas da seguinte forma

(PL1)

$$\max \sum_{i,j,i<j} c_{ij} y_{ij}$$

sujeito a:

$$y_{ij} + y_{jk} - y_{ik} \leq 1 \qquad \forall i,j,k \in V_n \ i < j < k \qquad \text{(I)}$$

$$y_{ij} + y_{jk} + y_{k\ell} - y_{ik} - y_{j\ell} \leq 1 \qquad \forall i,j,k,\ell \in V_n \ i < j < k < \ell \qquad \text{(II)}$$

$$\sum_{j,j>i}^{n} y_{ij} + \sum_{j,j<i}^{n} y_{ji} \leq b - 1 \qquad \forall i \in V_n \qquad \text{(III)}$$

$$y_{ij} \in \{0,1\}^m$$

As desigualdades em (I) são denominadas *desigualdades triangulares* e estabelecem que se as arestas (i,j) e (j,k) pertencem a clique então a aresta (i,k) também deverá pertencer. As desigualdades da classe (I) são suficientes para garantir que toda solução inteira corresponde à uma partição do grafo em cliques (veja Grötschel e Wakabayashi [27]). Entretanto, no problema que estamos estudando, uma solução viável é uma partição do grafo onde somente uma clique de tamanho no máximo b é não unitária, ou em outras palavras, só uma clique pode conter arestas. As desigualdades em (II) evitam a formação de mais de duas cliques não unitárias na partição. Estas desigualdades são denominadas *desigualdades Z*. Por último, as desigualdades em (III) estabelecem que a clique representada pelo vetor de incidência (0-1) não deve conter mais do que $b-1$ arestas incidentes em cada vértice. Tais desigualdades são chamadas *desigualdades grau*.

Tomando-se a formulação (PL1) é possível definir o politopo PCN_b como sendo $PCN_b = \text{conv}\,\{y \in \mathbb{R}^m : y \text{ satisfaz (I), (II), (III) e } y \text{ seja inteiro}\}$. Vale ressaltar ainda que, a formulação (PL1) possui um número muito grande de desigualdades. Isto se deve por que o número de desigualdades Z é $O(n^4)$.

Pode-se demonstrar que o politopo PCN_b possui dimensão cheia. Antes porém, vamos fazer algumas considerações que simplificam a prova desta afirmação (veja Faigle [17]). Para $2 \leq b \leq n$, denotaremos C^b como sendo a coleção de todas as cliques com exatamente b vértices, e $C(b)$ o conjunto de todas as cliques com no máximo b vértices, isto é, $C(b) = C^2 \cup C^3 \cup \ldots \cup C^b$. Considere que $C = (U,F)$ seja uma clique, com $C \in C(b)$, e que $\mathcal{X}^C \in \mathbb{R}^m$ corresponda ao vetor de incidência (0-1) definido para C. O politopo PCN_b poderia ser definido como $PCN_b = \text{conv}\,\{\mathcal{X}^C \in \mathbb{R}^m : C \in C(b)\}$.

Teorema 5.1 *(Faigle [17]) A dimensão do politopo PCN_b satisfaz*

$$dim(PCN_b) = \begin{cases} m-1 & se\ b = 2, \\ m & se\ 3 \leq b \leq n. \end{cases}$$

Prova:

Para $b = 2$, as desigualdades em (I)-(III) fazem com que em qualquer solução viável exista exatamente uma aresta. Ou seja, o politopo PCN_2 está contido no hiperplano $\sum_{y_{ij}} = 1$, e portanto, $dim(PCN_2) \leq m-1$. Para provar a igualdade basta observar que todos os m vetores de incidência de cliques C^2 estão em PCN_2, e são afins independentes. Isto implica que $dim(PCN_2) \geq m-1$ que, combinado com a desigualdade anterior, leva a $dim(PCN_2) = m-1$.

Para $b \geq 3$, basta notar que todos os vetores de incidência de cliques C^2 estão em PCN_b e são afins independentes. Além disso, o vetor de incidência de qualquer clique C^3

também está em PCN_b, e pode-se mostrar facilmente que ele forma um conjunto afim independente de tamanho $m+1$ com os vetores de incidência de todas as cliques em C^2. Logo, $dim(PCN_b) \geq m$, mas como $PCN_b \subseteq \mathbb{R}^m$, temos que $dim(PCN_b) = m$.

□

Em seguida apresentamos uma formulação inteira para o PCMPA no modelo estendido e o politopo obtido a partir desta formulação.

5.1.2 Modelo Estendido

O novo politopo associado ao Problema da Clique Máxima com Peso nas Arestas está definido agora no espaço que contém vértices e arestas. A interpretação das variáveis de decisão y_{ij} associadas às arestas continua a mesma, e a definição das novas variáveis de decisão $x_i \in \mathbb{R}^n$ para os vértices é a seguinte

$$x_i = \begin{cases} 1 & \text{se o vértice } i \text{ pertence a clique.} \\ 0 & \text{caso contrário.} \end{cases}$$

Um outro aspecto que deve ser levantado quando trabalhamos com este novo modelo é o fato de passarmos a aceitar a existência de cliques triviais, ou seja $1 \leq b \leq n$. Conseqüentemente, isto acarretará uma alteração na definição de $C(b)$ que passará a ser expresso por $C(b) = C^1 \cup C^2 \cup C^3 \cup \ldots \cup C^b$. Assim, o politopo associado ao Problema da Clique Máxima com Peso nas Arestas no modelo estendido, denotado por PCE_b, pode ser definido como $PCE_b = \text{conv}\{X^C \in \mathbb{R}^{n+m} : C \in C(b)\}$.

Uma caracterização inicial deste politopo pode ser obtida através de uma formulação inteira proposta por Faigle *et al.* [15]. A formulação de Faigle é descrita como se segue

(PL2)

$$\max \sum_{i,j,i<j} c_{ij} y_{ij}$$

sujeito a:

$$\begin{aligned}
& y_{ij} \leq x_i && \forall (i,j) \in E_n \ i<j && \text{(IV)} \\
& y_{ij} \leq x_j && \forall (i,j) \in E_n \ i<j && \text{(V)} \\
& x_i + x_j - y_{ij} \leq 1 && \forall (i,j) \in E_n \ i<j && \text{(VI)} \\
& \sum_{e \in \delta(i)} y_e - (b-1)x_i \leq 0 && \forall i \in V_n && \text{(VII)} \\
& y_{ij} \in \{0,1\}^m \\
& x_i \in \{0,1\}^n
\end{aligned}$$

A primeira classe de desigualdades, representadas em (IV) e (V), estabelecem que se um vértice não pertence à clique então a aresta incidente nele também não pertencerá. Por outro lado, as desigualdades em (VI) obrigam que uma aresta esteja na clique sempre que as suas extremidades estiverem na clique. A última classe, representada pelas desigualdades em (VII), estabelece que o número de arestas incidentes em um vértice deve ser igual a zero se o vértice não pertence à clique; e no máximo $b-1$, caso contrário. As desigualdades desta classe são denominadas *desigualdades estrelas.*

A formulação inteira (PL2) será a empregada por nós nos experimentos computacionais. Ela apresenta da $O(n+m)$ variáveis e da $O(n^2)$ desigualdades.

Uma outra formulação inteira para o PCMPA, no modelo estendido, foi proposta por Park *et al.* [40]. Essa nova formulação utiliza-se das classes de desigualdades (IV)-(VII) e de uma classe adicional descrita abaixo

$$\alpha \sum_{i \in V_n} x_i - \sum_{e \in E_n} y_e \leq \alpha(\alpha+1)/2 \qquad \text{para todo } \alpha = 1, \ldots, n-2 \qquad \text{(VIII)}$$

As desigualdades em (VIII) são denominadas *desigualdades clique* e estabelecem uma relação entre o número (ponderado) de vértices e o número de arestas de K_n que podem estar na clique. Esta classe de desigualdades será exposta com mais clareza na seção 5.3.

A exemplo do politopo PCN_b do modelo anterior, o politopo PCE_b também possui dimensão cheia. O teorema abaixo enuncia este fato.

Teorema 5.2 *A nova dimensão do politopo associado ao Problema da Clique Máxima com Peso nas Arestas (PCE_b) satisfaz*

$$dim(PCE_b) = \begin{cases} n & se\ b = 1, \\ n+m & se\ 2 \leq b \leq n. \end{cases}$$

Prova:

Para $b = 1$, temos $y_{ij} = 0$ para toda aresta $(i,j) \in E_n$. Logo, $dim(PCE_b) \leq n$. Segue-se ainda que, o vetor de incidência (0-1) de cliques com esta cardinalidade seria definido por C^1, isto é, seria constituído por elementos onde a posição i no vetor teria valor igual a 1, e as outras posições teriam entradas iguais a 0. É fácil de verificar que os

n elementos em C^1 são afins independentes. Assim, $dim(PCE_b) \geq n-1$. Além do mais, desde que o vetor nulo pertence a PCE_b, então $dim(PCE_b) \geq n-1+1 \geq n$. Portanto, $dim(PCE_b) = n$.

Para $b \geq 2$, o politopo PCE_b é definido pelas cliques do conjunto $C(b)$, e portanto através dos vetores de incidência (0-1) de cliques em $C^1, C^2, \ldots, C^b$. Sabendo-se que C^1 e C^2 são conjuntos afins independentes, pode-se facilmente verificar que $C^1 \cup C^2$ forma um conjunto com $n+m$ vetores afins independentes. Como já existem no mínimo $n+m$ vetores afins independentes, então $dim(PCE_b) \geq n+m-1$. Porém, como dito anteriormente o vetor nulo pertence a PCE_b. Logo, $dim(PCE_b) \geq n+m-1+1 \geq n+m$. Portanto, PCE_b possui dimensão cheia, pois $PCE_b \subseteq \mathbb{R}^{n+m}$.

□

Nas duas seções seguintes são apresentadas desigualdades definidoras de facetas para ambos os politopos. Na seção 5.2, apresenta as desigualdades para o politopo PCN_b enquanto que na seção 5.3 aquelas definidas para o PCE_b.

Para as desigualdades já conhecidas na literatura apresentaremos apenas as provas de validade das mesmas com o intuito de tornar mais fácil a compreensão destas desigualdades para o leitor. Já para as desigualdades encontradas neste trabalho serão apresentadas as provas de validade e faceta.

Durante a explicação das desigualdades, faremos uso do conceito de *grafo suporte* de uma desigualdade. Entende-se por grafo suporte de uma desigualdade o subgrafo induzido pelas arestas (e vértices) cujos coeficientes na desigualdade são não-nulos. Será adotada ainda a seguinte convenção na apresentação daquelas:

- Modelo natural
  1. linha pontilhada indicará que o coeficiente da aresta é negativo;
  2. linha cheia indicará que o coeficiente da aresta é positivo.
- Modelo estendido
  1. círculo com linha pontilhada indicará que o coeficiente do vértice é negativo;
  2. círculo hachurado indicará que o coeficiente do vértice é nulo;
  3. círculo com linha cheia indicará que o coeficiente do vértice é positivo;
  4. linha pontilhada indicará que o coeficiente da aresta é negativo;
  5. linha cheia indicará que o coeficiente da aresta é positivo.

5.2 Desigualdades Definidoras de Facetas para o PCN_b

Nesta seção, apresentaremos algumas desigualdades definidoras de facetas para o modelo natural. A princípio, nós iniciaremos com um *survey* das desigualdades conhecidas na literatura, e em seguida, com as desigualdades obtidas no estudo da caracterização facial do politopo PCN_b.

5.2.1 Trabalhos Anteriores

Como o politopo $PCN_b \subseteq \mathbb{R}^m$ e possui dimensão cheia para $b \geq 3$, então cada faceta do PCN_b é definida de forma única, a menos de uma multiplicação por um escalar positivo. Apresentaremos em seguida algumas desigualdades encontradas na literatura que definem facetas para o politopo PCN_b.

Sejam i, j, k, ℓ, r e s vértices quaisquer em V_n.

Proposição 5.1 *Se $n \geq 4$ então cada* desigualdade trivial

$$y_{ij} \geq 0$$

define uma faceta para o PCN_b.

Proposição 5.2 *Se $n \geq 3$ e $3 \leq b \leq n$ então cada* desigualdade triangular

$$y_{ij} + y_{jk} - y_{ik} \leq 1$$

define uma faceta para o PCN_b.

Prova: A desigualdade triangular nos diz que as arestas (i, j) e (j, k), com $y_{ij} = y_{jk} = 1$, não podem estar na clique sem que a aresta (i, k) também esteja $(y_{ik} = 1)$. Portanto é válida, pois isto implica que o lado esquerdo da desigualdade vale no máximo 1.

□

Proposição 5.3 *Se $n \geq 4$ e $4 \leq b \leq n$ então cada* desigualdade Z

$$y_{ij} + y_{jk} + y_{k\ell} - y_{ik} - y_{j\ell} \leq 1$$

define uma faceta para o PCN_b.

Prova: A desigualdade Z é trivialmente satisfeita se nenhuma, ou apenas uma das arestas do conjunto $\{(i, j), (j, k), (k, \ell)\}$ estão na clique. Para somente duas destas arestas estarem na clique, elas devem ter um vértice em comum (caso contrário a outra também

estaria na clique). Suponha que (i,j) e (j,k) estejam na clique, mas (k,ℓ) não, ou seja $y_{ij} = y_{jk} = 1$ e $y_{k\ell} = 0$. Nesse caso, a desigualdade reduz-se ao caso da desigualdade triangular $y_{ij} + y_{jk} - y_{ik}$ que deve ser menor ou igual a 1.

Resta apenas o caso em que as três arestas $(i,j),(j,k)$ e (k,ℓ) estão na clique, mas neste caso, as arestas (i,k) e (j,ℓ) também devem estar na clique. Portanto, a desigualdade é válida.

□

Proposição 5.4 *Se $n \geq 5$ e $4 \leq b \leq n$ então cada* desigualdade W

$$y_{ij} + y_{jk} + y_{k\ell} + y_{r\ell} - y_{ik} - y_{j\ell} - y_{kr} \leq 1$$

define uma faceta para o PCN_b.

Prova: Para ver que a desigualdade é válida, devemos novamente nos concentrar sobre as arestas com coeficientes positivos. Se nenhuma, ou apenas uma, ou todas estas arestas estão na clique, a desigualdade é trivialmente válida. A presença destas quatro arestas na clique obriga as três arestas com coeficientes negativo a estarem na clique, e a desigualdade se verifica. Por outro lado, se exatamente duas destas arestas estão na clique então: ou elas são consecutivas no caminho $\{(i,j),(j,k),(k,\ell),(\ell,r)\}$, e neste caso as desigualdades triangulares para $\{i,j,k\},\{j,k,\ell\}$ e $\{k,\ell,r\}$ garantem a validade, ou as arestas (i,j) e (ℓ,r) estão na clique, o que obriga a aresta (j,ℓ) a também estar, garantindo a validade da desigualdade.

Para exatamente três arestas no caminho $\{(i,j),(j,k),(k,\ell),(\ell,r)\}$ estarem na clique, estas devem ser: ou as três primeiras ou as três últimas do caminho. Neste caso, as desigualdades Z para $\{i,j,k,\ell\}$ e $\{j,k,\ell,r\}$ garantem a validade de W, completando a prova.

□

Proposição 5.5 *Se $n \geq 6$ e $4 \leq b \leq n$ então cada* desigualdade S

$$y_{ij} + y_{jk} + y_{k\ell} + y_{r\ell} + y_{rs} - y_{ik} - y_{j\ell} - y_{kr} - y_{s\ell} - y_{is} \leq 1$$

define uma faceta para o PCN_b.

Prova: Considera-se as arestas no caminho $\{(i,j),(j,k),(k,\ell),(\ell,r),(r,s)\}$ que são aquelas com coeficientes não-negativos na desigualdade. Novamente, a validade da desigualdade S pode ser facilmente verificada se nenhuma, ou apenas uma, ou todas as arestas do caminho estão em S. No caso de haver exatamente duas arestas do caminho

na clique, a validade pode ser verificada pelo fato de que para cada par de arestas do caminho, existe uma aresta com coeficiente negativo na desigualdade cujos vértices estão cada um em uma aresta do par escolhido. Portanto, esta aresta deverá também estar na clique tornando o lado esquerdo da desigualdade menor ou igual a 1.

Havendo exatamente três arestas da clique no caminho os seguintes casos podem ocorrer. As três arestas são consecutivas e neste caso recai-se nos argumentos que provam a validade da desigualdade Z. Duas arestas são consecutivas e estão na extremidade do caminho e a terceira aresta está na outra extremidade do caminho. Isto faz com que três arestas de coeficientes negativos devam estar na clique, e a validade se verifica.

Resta o último caso onde exatamente quatro arestas do caminho estão na clique. Estas arestas devem ser consecutivas e a validade fica provada pois os argumentos são os mesmos que provam a validade da desigualdade W.

□

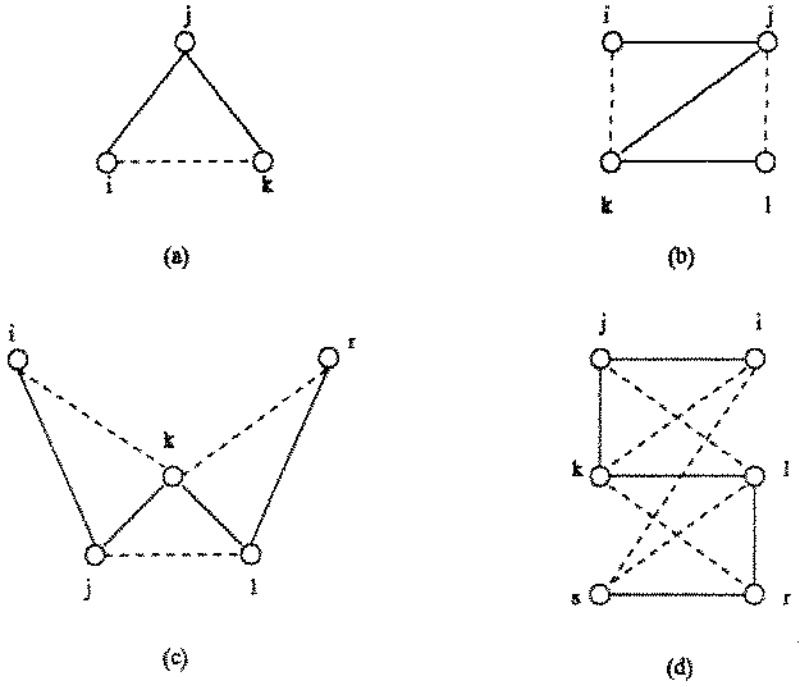


Figura 5.1: (a) Desigualdade triangular. (b) Desigualdade Z. (c) Desigualdade W. (d) Desigualdade S.

As desigualdades apresentadas acima foram propostas por Faigle e Dijkhuizen [14]. Como pode ser visto na Figura 5.1, as desigualdades das Proposições (5.2)-(5.5) têm seus nomes devido a forma dos grafos suporte a elas associados.

Neste mesmo trabalho, Faigle e Dijkhuizen [14] propuseram a desigualdade 1-p-partição. Seja $T \subseteq V_n$ um subconjunto de vértices em K_n, tal que, $|T| = p \geq 2$, e i um vértice

qualquer, tal que, $i \in V_n - T$. Considere ainda que $E(T)$ denota o conjunto de arestas em E_n com ambos os pontos extremos em T.

Proposição 5.6 *Se $n \geq 3$ e $3 \leq b \leq n$ então cada* desigualdade 1-p-partição

$$\sum_{t \in T} y_{it} - \sum_{e \in E(T)} y_e \leq 1$$

define uma faceta para o PCN_b.

Prova: Seja α o número de arestas ligando i a vértices em T que pertencem a clique. O número de arestas de $E(T)$ que pertencem a clique é expresso por $\alpha(\alpha-1)/2$. Com isso, o lado esquerdo da desigualdade reduz-se a $\alpha - \alpha(\alpha-1)/2$ que pode ser facilmente verificado ser menor ou igual a 1 para todo α inteiro não-negativo.

□

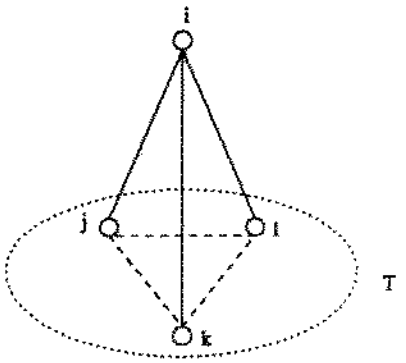


Figura 5.2: Grafo suporte de uma desigualdade 1-p-partição com $p = 3$ e $T = \{j, k, \ell\}$. A desigualdade 1-p-partição correspondente é $y_{ij} + y_{ik} + y_{i\ell} - y_{k\ell} - y_{jk} - y_{j\ell} \leq 1$.

O grafo suporte de uma desigualdade 1-p-partição para $p = 3$ é mostrada na Figura 5.2. É fácil observar que para $p = 2$ a desigualdade 1-p-partição corresponde à desigualdade triangular.

5.2.2 Facetas do PCN_b Encontradas neste Trabalho

Os primeiros estudos computacionais realizados neste trabalho restringiram-se ao modelo natural com variáveis somente sobre as arestas. Foi possível constatar que mesmo para instâncias consideradas pequenas, por exemplo $n = 6$ e $b = 3$ e $n = 7$ e $b = 3$, a solução ótima obtida através da fomulação (PL1) poderia ser fracionária.

Na busca por desigualdades válidas que cortassem estas soluções fracionárias, obtidas computacionalmente, realizamos estudos poliédricos que nos levaram à obtenção de duas classes de facetas para o politopo PCN_b que acreditavámos serem desconhecidas. Entretanto, em artigo bastante recente de Park *et al.* [40] constatamos que as desigualdades por nós obtidas se tornavam casos particulares das desigualdades apresentadas por aqueles autores através de argumentos teóricos.

O artigo de Park *et al.* [40] traz a prova deste resultado. No entanto, com o intuito de explicar o que foi feito em nosso trabalho, apresentaremos os resultados obtidos seguidos das respectivas provas.

Sejam S e T dois subconjuntos distintos de vértices em K_n, tal que, $S = \{r, s\}$ e $|T| = p$ com $2 \leq p \leq n-2$. Considere ainda que $E(S)$ represente o conjunto das arestas com ambas as extremidades em S, e $E(S:T)$ o conjunto das arestas com uma extremidade em S e a outra em T.

Para $b \geq 4$, considere a *desigualdade 2-p-partição* dada por

$$\sum_{e \in E(S:T)} y_e - \sum_{e \in E(T)} y_e - 2y_{rs} \leq 1 \tag{5.1}$$

Proposição 5.7 *Se $n \geq 4$ e $b \geq 4$, a* desiguadade 2-*p*-partição *define uma faceta para o politopo PCN_b.*

Prova: (a) *Validade.* Seja C uma clique viável e $\mathcal{X}^C$ o vetor de incidência (0-1) associado a C. Assuma que α_r e α_s o número de arestas ligando r e s, respectivamente, a vértices de T tais que estas arestas estejam na clique C. Considere os casos abaixo.

Caso 1. Assuma que a aresta (r, s) possua entrada igual a 0 em $\mathcal{X}^C$, ou seja $y_{rs} = 0$.

Caso 1.1. $\alpha_r = 0$ ou $\alpha_s = 0$. Suponha, sem perda de generalidade, que $\alpha_r \neq 0$ (com $\alpha_s = 0$). Neste caso, recai-se na prova de validade da desigualdade 1-*p*-partição.

Caso 1.2. $\alpha_r = \alpha_s = 0$. Trivialmente é válida, pois nenhuma aresta com coeficiente positivo estará na clique.

Caso 2. Assuma agora que aresta (r, s) possua entrada igual a 1 em $\mathcal{X}^C$, ou seja $y_{rs} = 1$.

Com isso temos $\alpha_r = \alpha_s = \alpha$ onde α denota o número de vértices em T na clique. Portanto, o lado esquerdo da desigualdade resultaria em $2\alpha - \alpha(\alpha-1)/2 - 2$ que é menor ou igual a 1 para qualquer α inteiro não-negativo.

(b) *Faceta.* Considere a face $\mathcal{F} = \{\mathcal{X}^C \in PCN_b : \mathcal{X}^C$ satisfaz 2-*p*-partição na igualdade$\}$, e a desigualdade $\pi y \leq \pi_0$ válida para PCN_b que define a face $\mathcal{F}_\pi = \{y \in PCN_b : \pi y = \pi_0\}$, com $\mathcal{F} \subseteq \mathcal{F}_\pi$. Vamos provar que $\pi y \leq \pi_0$ é um múltiplo escalar positivo da desigualdade (5.1). Assuma os seguintes casos abaixo.

Caso 1. $C = \{u, i\}$ com $u \in \{r, s\} = S$ e $i \in T$.

Desde que $\mathcal{X}^C \in \mathcal{F}$ tem-se que $\pi_{ui} = \pi_0$ para toda aresta $(u,i) \in E(S:T)$.

Caso 2. $C = \{u,i,j\}$ com $u \in S$ e $i,j \in T$, tal que $i \neq j$.

Como $\mathcal{X}^C \in \mathcal{F}$ segue-se que $\pi_{ui} + \pi_{uj} + \pi_{ij} = \pi_0$. Sabendo-se que $\pi_{ui} = \pi_{uj} = \pi_0$, conclui-se que $\pi_{ij} = -\pi_0$ para toda aresta $(i,j) \in E(T)$.

Caso 3. $C = \{r,s,i,j\}$ com $i,j \in T$, tal que $i \neq j$.

Como $\mathcal{X}^C \in \mathcal{F}$ isto implica em $\pi_{ri} + \pi_{si} + \pi_{rj} + \pi_{sj} + \pi_{ij} + \pi_{rs} = \pi_0$. Sabendo-se que $\pi_{sj} = \pi_{rj} = \pi_{si} = \pi_{ri} = \pi_0$, a partir do caso (1), e que $\pi_{ij} = -\pi_0$, a partir do caso (2), conclui-se que $\pi_{rs} = -2\pi_0$.

Caso 4. $C = \{u,i,z\}$ com $u \in S = \{r,s\}$, $i \in T$ e $z \in V_n - (S \cup T)$.

Desde que $\mathcal{X}^C \in \mathcal{F}$ segue-se que $\pi_{iu} + \pi_{iz} + \pi_{uz} = \pi_0$. Logo, $\pi_{iz} + \pi_{uz} = 0$ devido o caso (1).

Caso 5. $C = \{u,i,j,z\}$ com $u \in S = \{r,s\}$, $i,j \in T$ e $z \in V_n - (S \cup T)$.

Segue-se que $\mathcal{X}^C \in \mathcal{F}$, logo $\pi_{iu} + \pi_{ij} + \pi_{iz} + \pi_{ju} + \pi_{jz} + \pi_{uz} = \pi_0$. Sabendo-se que $\pi_{iz} + \pi_{uz} = 0$ e que $\pi_{iu} = \pi_{ju} = -\pi_{ij} = \pi_0$ conclui-se que $\pi_{jz} = 0$. Portanto, $\pi_{uz} = 0$ a partir do caso (4).

Caso 6. $C = \{r,i,w,z\}$ com $i \in T$ e $w,z \in V_n - (S \cup T)$.

Como $\mathcal{X}^C \in \mathcal{F}$, isto implica em $\pi_{ri} + \pi_{rw} + \pi_{rz} + \pi_{iw} + \pi_{iz} + \pi_{wz} = \pi_0$. Substituindo, $\pi_{rw} = \pi_{rz} = \pi_{iw} = \pi_{iz} = 0$ e $\pi_{ri} = \pi_0$, segue-se que $\pi_{wz} = 0$.

□

O grafo suporte de uma desigualdade (5.1) é mostrado na Figura 5.3(a).

A próxima classe de desigualdades, chamada de 3-*p*-partição, é uma extensão possível das desigualdades de 1-*p*-partição e 2-*p*-partição para o caso em que $|S| = 3$. Suponha novamente S e T dois subconjuntos de V_n, com $S = \{r,s,z\}$ e $|T| = p$ com $2 \leq p \leq n-3$.

A *desigualdade 3-p-partição* é expressa por

$$\sum_{e \in E(S:T)} y_e - \sum_{e \in E(T)} y_e - 2(y_{rs} + y_{rz} + y_{sz}) \leq 1 \qquad (5.2)$$

Proposição 5.8 *Se $n \geq 5$ e $b \geq 4$, a* desiguadade 3-*p*-partição *define uma faceta para o politopo PCN_b.*

Prova: (a) *Validade.* Seja C uma clique viável e $\mathcal{X}^C$ o vetor de incidência (0-1) associado a C. Assuma que α_r, α_s e α_z o número de arestas ligando r, s e z, respectivamente, aos vértices de T tais que estas arestas estejam na clique. Apenas dois casos podem ocorrer.

Caso 1. Suponha que apenas uma das arestas (r,s), (r,z) e (s,z) possua entrada igual em $\mathcal{X}^C$, ou seja $y_{rs} + y_{rz} + y_{sz} \leq 1$.

Com isto recai-se no caso da desigualdade 2-*p*-partição que já se sabe ser válida.

Caso 2. Assuma agora que tanto a aresta (r,s) como (r,z) e a aresta (s,z) possuam entrada igual a 1 χ^C, ou seja $y_{rs} = y_{rz} = y_{sz} = 1$.

Neste caso tem-se que $\alpha_r = \alpha_s = \alpha_z = \alpha$, onde α denota o número de vértices de T na clique. Logo, o lado esquerdo da desigualdade torna-se $3\alpha - \alpha(\alpha-1)/2 - 6$ que sempre é menor ou igual a 1 para α inteiro não-negativo.

(b) *Faceta.* Análoga à prova da Proposição 5.7, considerado-se apenas que a face $\mathcal{F}$ seja expressa por $\mathcal{F} = \{y \in PCN_b : \sum_{e\in E(S:T)} y_e - \sum_{e\in E(T)} y_e - 2(y_{rs} + y_{rz} + y_{sz}) = 1\}$, e que o par $(r,s) \in S$ deverá ser substituído por qualquer par (k,t), com $k,t \in S$.

□

O grafo suporte de uma desigualdade (5.2) é visto na Figura 5.3(b).

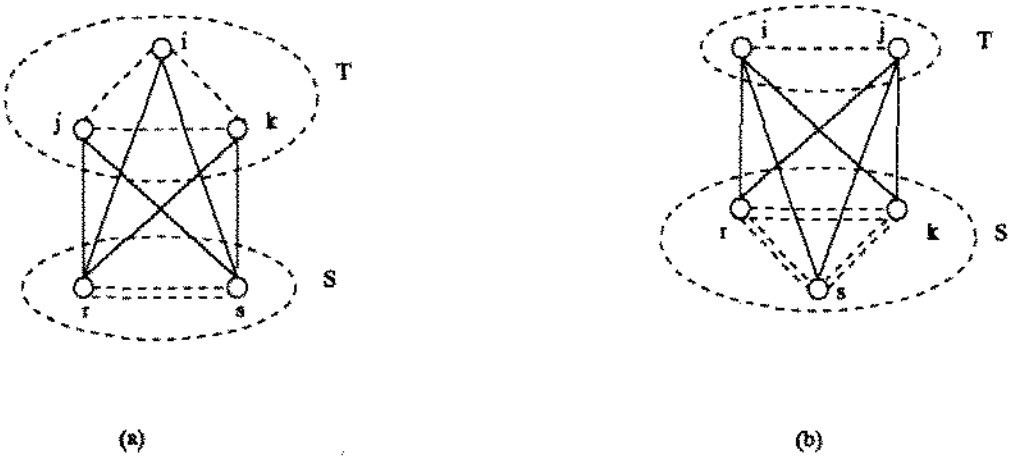


Figura 5.3: (a) Grafo suporte de uma desigualdade 2-p-partição para $p = 3$ e $S = \{r,s\}$. A desigualdade 2-p-partição correspondente é $y_{ir} + y_{jr} + y_{kr} + y_{is} + y_{js} + y_{ks} - y_{ij} - y_{ik} - y_{jk} - 2y_{rs} \leq 1$. (b) Grafo suporte de uma desigualdade 3-p-partição para $p = 2$ e $S = \{r,s,k\}$. A desigualdade 3-p-partição correspondente é $y_{ir} + y_{is} + y_{ik} + y_{jr} + y_{js} + y_{jk} - y_{ij} - 2y_{rs} - 2y_{rk} - 2y_{ks} \leq 1$.

Na seção 5.3 mostraremos como Park *et al.* [40] chegaram à uma classe mais geral de desigualdades que incluem 1-p-partição, 2-p-partição e 3-p-partição.

5.3 Desigualdades Definidoras de Facetas para o PCE_b

Nesta seção, apresentaremos algumas desigualdades definidoras de facetas para o modelo estendido. A exemplo da seção anterior, primeiro apresentamos um *survey* das desigualdades já conhecidas na literatura, e em seguida, as encontradas neste trabalho.

5.3.1 Trabalhos Anteriores

Como foi dito no Capítulo 3, o Problema da Clique Máxima com Peso nas Arestas está intimamente relacionado ao Problema Quadrático Booleano. Este último, foi estudado em Padberg [38] onde vários resultados relativos ao politopo associado àquele problema são apresentados. Não é surpreeendente portanto, que vários destes resultados possam ser estendidos para o problema que estamos estudando.

A partir de algumas desigualdades obtidas por Padberg para o Problema Quadrático Booleano, Park *et al.* [40] estabeleceram condições segundo as quais aquelas passam a definir facetas para o PCE_b.

As Proposições 5.9, 5.10 e 5.11 a seguir fixam as condições, propostas por Park *et al.*, para que as desigualdades (IV)-(VIII) definam facetas para o politopo PCE_b. Vale lembrar que apenas as provas de validade serão apresentadas, para a prova de que as desigualdades definem facetas veja o trabalho daqueles autores.

Proposição 5.9 *Seja $n \geq 3$. Para quaisquer dois vértices distintos $i, j \in V_n$, as desigualdades abaixo*

$$y_{ij} - x_i \leq 0 \tag{5.3}$$

e

$$x_i + x_j - y_{ij} \leq 1 \tag{5.4}$$

definem facetas para o PCE_b, se somente se, $b \geq 3$.

Prova: Demonstramos primeiro a validade da desigualdade $y_{ij} - x_i \leq 0$. Considere que o vértice i pertença a clique, ou seja $x_i = 1$. Neste caso, a aresta (i, j) incidente no vértice i pode ser igual a zero ou a 1. Para $y_{ij} = 0$ temos o lado esquerdo igual a -1. Já para $y_{ij} = 1$ o lado esquerdo será igual a zero. Portanto, a desigualdade mantém-se válida. Assuma agora que o vértice i não pertença a clique, ou seja, $x_i = 0$. Para este caso, a aresta (i, j) só poderá ser igual a zero. Com isso, o lado esquerdo da desigualdade será igual a zero, e a desigualdade mantém-se válida.

Demonstramos agora a validade da desigualdade $x_i + x_j - y_{ij} \leq 1$. Assuma, sem perda de generalidade, que um dos vértices esteja na clique, ou seja $x_i = 0$ e $x_j \neq 0$. Neste caso, a aresta (i, j) será igual a zero devido $x_i = 0$. Logo, o lado esquerdo da desigualdade será igual a 1. Segue-se que a desigualdade é válida. Considere agora que $x_i = x_j = 0$. Com isso $y_{ij} = 0$, e portanto a desigualdade mantém-se válida. Por último, façamos $x_i = x_j = 1$. Isto implica em $y_{ij} = 1$ e a validade da desigualdade.

□

Proposição 5.10 *Para qualquer vértice $i \in V_n$, a* desigualdade estrela

$$\sum_{e \in E(\{i\}:V_n-\{i\})} y_e - (b-1)x_i \leq 0 \tag{5.5}$$

define uma faceta para o PCE_b, se somente se, $b \leq n-1$.

Prova: Considere que $x_i = 0$. Neste caso o vértice i não pertence à clique, e portanto todas as arestas que incidem em i não pertencem à clique, ou seja $\sum y_e = 0$. Segue-se que a desigualdade é válida. Assuma agora que o vértice i pertença à clique, isto é, $x_i = 1$. Para este caso temos $\sum y_e \leq b-1$. Logo, o lado esquerdo da desigualdade será no máximo igual a zero. Portanto, a desigualdade mantém-se válida.

□

Proposição 5.11 *Seja $S \subseteq V_n$ um subconjunto de vértices. Para $|S| \geq 3$ e $1 \leq \alpha \leq |S|-2$, a desigualdade*

$$\alpha \sum_{i \in S} x_i - \sum_{e \in E(S)} y_e \leq \alpha(\alpha+1)/2 \tag{5.6}$$

define uma faceta para o PCE_b, se somente se, $S = V_n$ ou $\alpha \leq b-2$.

Prova: A validade da desigualdade (5.6) pode ser verificada a partir dos seguintes argumentos. Seja p o número de vértices na clique solução. O lado esquerdo da desigualdade torna-se $\alpha p - p(p-1)/2$. Para que a desigualdade seja válida, é preciso que $\alpha p - p(p-1)/2 - \alpha(\alpha+1)/2$ seja menor ou igual a zero. Ou ainda, que $-\frac{1}{2}[p^2-(2\alpha+1)p+\alpha(\alpha+1)]$ seja menor ou igual a zero para α inteiro com $\alpha \geq 0$. Ocorre que este polinômio tem raízes em α e $\alpha+1$, portanto inteiras. Conseqüentemente, temos que $-\frac{1}{2}[p^2-(2\alpha+1)p+\alpha(\alpha+1)] \leq 0$ para todo p inteiro, e sempre que $\alpha \in \{1,2,\ldots,n-2\}$. Logo, a desigualdade é válida.

□

Um caso especial da desigualdade (5.6), de interesse para este trabalho, são as chamadas *desigualdades clique-triangulares* obtidas quando $|S| = 3$ e $\alpha = 1$. O grafo suporte de uma desigualdade clique-triangular é mostrado na Figura 5.4(a).

Na Proposição 5.12 abaixo, apresentamos as desigualdades conhecidas por *desigualdades corte* e as condições segundo as quais elas definem facetas para o politopo PCE_b.

Proposição 5.12 *Sejam $S \subseteq V_n$ e $T \subseteq V_n - S$ dois subconjuntos de vértices. Para $|S| = s \geq 1$ e $|T| = t \geq 2$, a desigualdade*

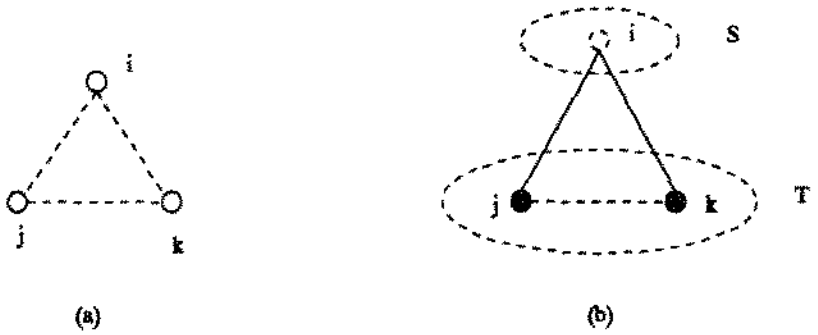


Figura 5.4: (a) Grafo suporte de uma desigualdade clique-triangular com $S = \{i, j, k\}$ e $\alpha = 1$. A desigualdade clique-triangular correspondente é $x_i + x_j + x_k - y_{ij} - y_{ik} - y_{jk} \leq 1$. (b) Grafo suporte de uma desigualdade corte-triangular com $S = \{i\}$ e T $= \{j, k\}$. A desigualdade corte-triangular correspondente é $y_{ij} + y_{ik} - y_{jk} - x_i \leq 0$.

$$\sum_{e \in E(S:T)} y_e - \sum_{e \in E(S)} y_e - \sum_{e \in E(T)} y_e - \sum_{i \in S} x_i \leq 0 \tag{5.7}$$

define uma faceta para o PCE_b, se somente se, $|S| = 1$ e $b \geq 3$ ou $|S| \geq 2$ e $b \geq 4$.

Prova: Assuma que C seja uma clique viável qualquer e que $|C \cap S| = \tilde{s}$ e $|C \cap T| = \tilde{t}$. Isto implica em $\tilde{t}\tilde{s} - \tilde{s}(\tilde{s}-1)/2 - \tilde{t}(\tilde{t}-1)/2 - \tilde{s} \leq 0$, ou ainda fatorando-se o polinômio anterior obtemos $-\frac{1}{2}[\tilde{t}^2 - (2\tilde{s}+1)\tilde{t} + \tilde{s}(\tilde{s}+1)] \leq 0$ com $0 \leq \tilde{s} \leq s$ e $0 \leq \tilde{t} \leq t$. Como $-\frac{1}{2}[(\tilde{t}-\tilde{s})(\tilde{t}-\tilde{s}-1)] \leq 0$ para qualquer $\tilde{s}$ e $\tilde{t}$ inteiros, segue-se que a desigualdade é válida. □

Quando $|S| = 1$ e $|T| = 2$ na Proposição 5.12 tem-se as chamadas *desigualdades corte-triangulares* que, como veremos adiante, tem uma enorme importância do ponto de vista computacional. O grafo suporte de uma destas desigualdades é visto na Figura 5.4(b).

Antes de termos conhecimento do trabalho de Park *et al.* [40], dois casos particulares da desigualdade (5.7) haviam sido obtidos por nós usando técnicas de *lifting*. De certa forma, estávamos realizando o caminho inverso daquele feito por Park. No nosso caso, estávamos fazendo *lifting* do modelo natural para o modelo estendido. Já Park *et al.*, usando técnicas de projeção, obtiveram algumas facetas do modelo natural.

A título de exemplo, sobre como poderiam ser obtidas facetas do modelo estendido a partir do modelo natural, mostraremos os *liftings* realizados nas desigualdades triangular e 2-2-partição, e que levam a desigualdades como as definidas na Proposição (5.12) para o modelo estendido.

Considere a desigualdade triangular $y_{ij} + y_{ik} - y_{jk} \leq 1$ e a face definida por ela em PCE_b expressa por $\mathcal{F} = \{(x, y) \in PCE_b : y_{ij} + y_{ik} - y_{jk} = 1\}$. Todos os pontos sobre esta face satisfazem $x_i = 1$ e, portanto $\mathcal{F}$ não pode definir uma faceta de PCE_b. No

entanto, existe um $\alpha \in \mathbb{R}_+$, tal que a desigualdade $\alpha(1 - x_i) + y_{ij} + y_{ik} - y_{jk} \leq 1$ é válida, e preferencialmente, definidora de faceta. Quando $x_i = 1$ esta desigualdade recai no caso original, portanto devemos escolher α de forma que a desigualdade seja válida e mais apertada possível para quando $x_i = 0$. Logo, se $x_i = 0$, $y_{ij} = y_{ik} = 0$ o problema resume-se a encontrar o maior valor de α tal que, $x_i = 0$ e $\alpha \leq 1 + y_{jk}$. Pode-se verificar que o valor de α deve ser igual a 1, e a desigualdade torna-se $y_{ij} + y_{ik} - y_{jk} - x_i \leq 0$. Tal desigualdade corresponde à desigualdade corte-triangular.

Assuma agora a desigualdade 2-2-partição dada por $y_{ik} + y_{i\ell} + y_{jk} + y_{j\ell} - y_{ij} - 2y_{k\ell} \leq 1$ e a face $\mathcal{F}$ definida por ela em PCE_b. Todas as cliques cujos vetores de incidência (0-1) estão em $\mathcal{F}$ contêm o vértice k ou o vértice ℓ, mas não ambos. Isto faz com que a desigualdade $x_k + x_\ell - y_{k\ell} \leq 1$ da formulação (PL2) sempre seja satisfeita com igualdade. Desde que $x_k + x_\ell - y_{k\ell} \leq 1$ só pode assumir valores 0 ou 1, como no caso anterior, o problema é encontrar o maior valor de α, quando $x_k = x_\ell = 0$, para que a desigualdade $\alpha(1 - x_k - x_\ell + y_{k\ell}) + y_{ik} + y_{jk} + y_{j\ell} + y_{i\ell} - y_{ij} - 2y_{k\ell} \leq 1$ se torne válida. Este valor é obtido para $\alpha = 1$ e a desigualdade tornar-se $y_{ik} + y_{jk} + y_{i\ell} + y_{j\ell} - y_{ij} - y_{k\ell} - x_k - x_\ell \leq 0$ que é uma desigualdade válida. Tal desigualdade corresponde à desigualdade corte quando fazemos $|S| = 2$ e $|T| = 2$.

Dando continuidade a apresentação das desigualdades definidoras de facetas no PCE_b, a próxima desigualdade é originária do Problema do Subgrafo de Peso Máximo com restrição de Capacidade estudado por Johnson *et al.* [30]. Quando o grafo é completo e a restrição de capacidade reduz-se à uma restrição de cardinalidade, o problema tratado por Johnson *et al.* é exatamente o PCMPA.

Algumas das desigualdades propostas por Johnson *et al.* foram generalizadas em de Souza [13]. A Proposição a seguir foi extraída de Park *et al.* [40] que mostraram que as condições para uma desigualdade árvore definir uma faceta de PCE_b eram mais brandas do que aquelas propostas por Johnson *et al.*

Proposição 5.13 *Seja $\hat{T}$ uma árvore no grafo $K_n = (V_n, E_n)$ com $V(\hat{T})$ representando o conjunto de vértices em $\hat{T}$, tal que, $|V(\hat{T})| = b + 1$, e o conjunto de arestas $E(\hat{T})$. Considere ainda que $n \geq 3$. Então cada* desigualdade árvore

$$\sum_{e \in E(\hat{T})} y_e - \sum_{i \in V(\hat{T})} (d_i - 1)x_i \leq 0$$

onde d_i é o grau do vértice i na árvore $\hat{T}$, define uma faceta para o PCE_b, se somente se, $b = n - 1$ ou $\hat{T}$ não é uma estrela.

Prova:

Assuma que D seja um conjunto dependente (veja Johnson *et al.* [30]), isto é, $\sum_{i \in D} x_i > b$. Em particular, $|D| = b + 1$. Considere que as arestas pertencentes a

$E(D) \cap E(\hat{T})$ induzem uma floresta F. Seja $\tilde{T} = (V(\tilde{T}), E(\tilde{T}))$ um componente qualquer da floresta F, com $V(\tilde{T}) = \tilde{S}$. Portanto, o componente $\tilde{T}$ corresponde a uma árvore.

Para $i \in \tilde{S}$, seja $\tilde{\delta}(i) = \{j : (i,j) \in E(\tilde{T})\}$. Como $|\tilde{S}| = |D|$, existe uma aresta $(i,j) \in E(\hat{T}) - E(\tilde{T})$ com $j \in \tilde{S}$, tal que, $|\delta(j)| > |\tilde{\delta}(j)|$. Desde que $y_e = 1$, para $e \in E(\tilde{T})$, segue-se que

$$\sum_{e \in E(\tilde{T})} y_e = |E(\tilde{T})| = |V(\tilde{T})| - 1 = |\tilde{S}| - 1.$$

Expressemos $|\tilde{S}| - 1$ como sendo $2(|\tilde{S}| - 1) - |\tilde{S}| + 1$. Cada aresta $(i,j) \in E(\tilde{T})$ é contada duas vezes para cada vértice $i \in \tilde{S}$, logo

$$2(|\tilde{S}| - 1) - |\tilde{S}| + 1 = \sum_{i \in \tilde{S}} |\tilde{\delta}(i)| - |\tilde{S}| + 1.$$

Porém, como o grau de um vértice é expresso pelo número de arestas incidentes nele menos 1, aplicando esta definição para todo $i \in \tilde{S}$ obtemos

$$[\sum_{i \in \tilde{S}} |\tilde{\delta}(i)| - |\tilde{S}|] + 1 = [\sum_{i \in \tilde{S}} (|\tilde{\delta}(i)| - 1)] + 1.$$

Sabendo-se que $|\delta(j)| > |\tilde{\delta}(j)|$

$$[\sum_{i \in \tilde{S}} (|\tilde{\delta}(i)| - 1)] + 1 \leq \sum_{i \in \tilde{S}} (|\delta(i)| - 1) = \sum_{i \in \tilde{S}} (|\delta(i)| - 1) x_i,$$

pois $x_i = 1$ para todo $i \in \tilde{S}$.

Adicionando-se estas desigualdades para todo componente em F mostra-se que a desigualdade árvore é válida para qualquer solução $(x, y) \in PCE_b$.

□

Uma outra família de facetas para o PCE_b foi sugerida por Park *et al.* [41] e pode ser descrita da seguinte forma. Assuma que P seja um caminho no grafo K_n, com $|V(P)| = b + 1$, e $V_2(P)$ os vértices não extremos do caminho, isto é, os vértices com grau igual a 2 em P. Considere um conjunto $K \subseteq V(P)$, tal que, K induz um subgrafo conexo em P com $|K| \leq b - 1$, e o politopo definido abaixo

$$P_\beta = \{\beta \in \mathbb{R}^{b+1} : \sum_{i \in K} \beta_i \leq |V_2(P) \cap K| - |K| + 1, \text{ para todo K}\}$$

Proposição 5.14 *Para $b \geq 3$, a desigualdade*

$$\sum_{e \in E(P)} y_e - \sum_{i \in V_2(P)} x_i + \sum_{i \in V(P)} \beta_i y_{it} \leq 0 \tag{5.8}$$

onde $t \in V_n - V(P)$, define uma faceta para o PCE_b, se somente se, β é um ponto extremo de P_β.

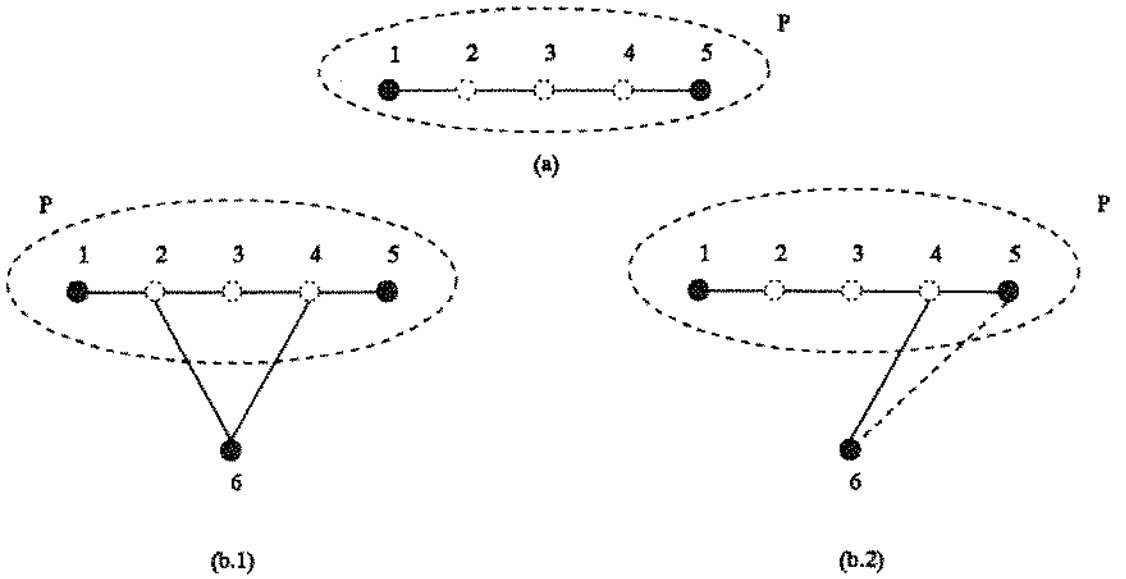


Figura 5.5: (a) Grafo suporte de uma desigualdade caminho. A desigualdade correspondente é $y_{12} + y_{23} + y_{34} + y_{45} - x_2 - x_3 - x_4 \leq 0$. (b) Grafo suporte de duas desigualdades pára-quedas. Em (b.1) a desigualdade correspondente é $y_{12} + y_{23} + y_{34} + y_{45} + y_{26} + y_{46} - x_2 - x_3 - x_4 \leq 0$. Em (b.2) temos $y_{12} + y_{23} + y_{34} + y_{45} + y_{46} - y_{56} - x_2 - x_3 - x_4 \leq 0$.

A desigualdade (5.8) é denominada *desigualdade pára-quedas*. Alguns exemplos de desigualdades de grafos suporte de desigualdades pára-quedas são vistos na Figura 5.5.

Um caso particular, interessante para o nosso trabalho, das desigualdades (5.8) são as chamadas *desigualdades de caminho*, obtidas quando $\beta_i = 0$ para todo $i \in K$. Note que, como $0 \in P_\beta$, a desigualdade define faceta para PCE_b.

5.3.2 Relação entre Facetas dos Politopos PCN_b e PCE_b

Nessa subseção procuraremos traçar um paralelo entre o modelo natural e o modelo estendido. Para isto mostraremos como desigualdades no modelo natural podem ser obtidas a partir de desigualdades no modelo estendido, e será possível verificar que os

bounds obtidos na formulação inteira definida no modelo estendido são melhores do que os obtidos no modelo natural.

A Projeção de PCE_b em PCN_b

Estudos realizados por Park *et al.* [40] possibilitaram a descoberta de novas famílias de facetas para o politopo associado ao Problema da Clique Máxima com Peso nas Arestas em ambos os modelos.

Nesta seção, daremos uma breve descrição dos resultados apresentados por Park *et al.* que permitirão a obtenção de novas desiguladades para o PCMPA através de técnicas de projeção aplicadas ao Problema Quadrático Booleano (veja Padberg [38]).

Antes de prosseguirmos daremos algumas notações e definições que serão úteis para a compreensão desta seção. Seja Ψ um subconjunto arbitrário de $\mathbb{R}^m$ e o espaço

$$Q = \{(x, y) \in \mathbb{R}^{n+m} : Ax + By \leq d, x \geq 0, y \in \Psi\}$$

onde A, B e d são matrizes $m \times n$, $m \times m$ e $m \times 1$, respectivamente, tal que $Q \neq \emptyset$. A *projeção* do poliedro Q dentro do espaço y, denotada por $Pr_y(Q)$, é definida como

$$Pr_y(Q) = \{y \in \mathbb{R}^m : (x, y) \in Q, \text{ para algum } x \in \mathbb{R}^n\}.$$

Nós estamos interessados em descrever o conjunto $Pr_y(Q)$ de uma maneira similar a Q, isto é através de um conjunto de desigualdades lineares, e claro, com a condição $y \in \Psi$. O resultado abaixo define $Pr_y(Q)$

Teorema 5.3 *(Balas e Pulleyblank [1]) Seja $W = \{w \in \mathbb{R}^m : wA = 0 \text{ e } w \geq 0\}$. Se $w^1, w^2, \ldots, w^s$ são os raios extremos de W, então $Pr_y(Q) = \{y \in \mathbb{R}^m : (w^k B)y \leq w^k d$ para todo $k = 1, 2, \ldots, s; y \in \Psi\}$.*

Tendo em mente este resultado, e as notações e definições apresentadas, retornemos a questão de projetar o politopo PCE_b no espaço $\mathbb{R}^m$ (politopo PCN_b). Seja $Q_{p,q}$ um conjunto de soluções viáveis para o sistema de desigualdades lineares abaixo

$$\sum_{t \in S} y_{it} - \sum_{e \in E(S)} y_e - x_i \leq 0 \quad \forall S \subseteq V_n - \{i\}, \text{ com } 1 \leq |S| \leq p \quad \forall i \in V_n. \tag{5.9}$$

$$\sum_{i\in S} x_i - \sum_{e\in E(S)} y_e \leq 1 \quad \forall S \subseteq V_n, \text{ com } 1 \leq |S| \leq q. \tag{5.10}$$

$$y_e \geq 0 \quad \forall e \in E_n. \tag{5.11}$$

Note que as desigualdades em (5.9) e (5.10) são casos especiais de desigualdades (5.7) e (5.6), respectivamente.

Vamos definir $P_{p,q}$ como sendo igual a $Pr_y(Q_{p,q})$ e

$$W_{p,q} = \{\alpha \in I\!R^r_+, \beta \in I\!R^s_+ : \sum_{S\subseteq V_n-\{i\}, 1\leq|S|\leq p} \alpha_{iS} - \sum_{i\in S\subseteq V_n, |S|\leq q} \beta_S = 0, \text{ para todo } i \in V_n\}.$$

onde $r = n\sum_{k=1}^{p}(n-1)C_k$ e $s = \sum_{k=1}^{q} nC_k$.

Em seguida caracterizamos o conjunto de raios extremos de $W_{p,q}$.

Proposição 5.15 *Seja (α, β) um raio de $W_{p,q}$. Então (α, β) é um raio extremo, se somente se, ele é equivalente a seguinte forma*

$$\begin{aligned}
\beta_S &= 1 \quad \text{para algum } S \subseteq V_n, 1 \leq |S| \leq q;\\
\beta_T &= 0 \quad \text{caso contrário;}\\
\alpha_{iS_i} &= 1 \quad \text{para algum } S_i \subseteq V_n - \{i\}, 1 \leq |S_i| \leq p \quad \forall i \in S;\\
\alpha_{iT} &= 0 \quad \text{caso contrário.}
\end{aligned}$$

Prova: [Park *et al.* [40]].

□

Portanto, usando o Teorema 5.3 nós obtemos

Corolário 5.1 $P_{p,q} = \{y \in I\!R^m_+ : \sum_{i\in S}\sum_{t\in S_i} y_{it} - \sum_{i\in S}\sum_{e\in E(S_i)} - \sum_{e\in E(S)} y_e \leq 1, \forall S \subseteq V_n, 1 \leq |S| \leq q \text{ e } S_i \subseteq V_n - \{i\}, 1 \leq |S_i| \leq p \quad \forall i \in S\}$.

Mostraremos agora alguns casos especiais dos resultados acima. Primeiro considere $Q_{1,2}$ que consiste das desigualdades (5.3) e (5.4). Note que $Q_{1,2}$ com a restrição de cardinalidade dá uma relaxação linear para a formulação estendida.

Corolário 5.2 *$P_{1,2}$ é o conjunto de soluções viáveis para o seguinte sistema de desigualdades lineares*

$$\begin{aligned} y_e &\geq 0 \quad \forall e \in E_n && (5.12)\\ y_e &\leq 1 \quad \forall e \in E_n && (5.13)\\ y_{ik} + y_{jk} - y_{ij} &\leq 1 \quad \text{para três vértices distintos } i,j,k \in V_n && (5.14)\\ y_{ik} + y_{j\ell} - y_{ij} &\leq 1 \quad \text{para quatro vértices distintos } i,j,k,\ell \in V_n && (5.15) \end{aligned}$$

A formulação $Q_{1,2}$ requer $n+m$ variáveis com $O(n^2)$ restrições, e a formulação $P_{1,2}$ requer somente m variáveis mas $O(n^4)$ restrições.

Considere agora $Q_{2,2}$.

Corolário 5.3 *$P_{2,2}$ é o conjunto de soluções viáveis para o sistema de desigualdades lineares composto pelas desigualdades de (5.12)-(5.15) e por*

$$\begin{aligned} y_{ik} + y_{j\ell} + y_{jt} - y_{ij} - y_{\ell t} &\leq 1 && (5.16)\\ \forall i,j,k,\ell,t \text{ com } i \neq j, \ell \neq t, j \neq \ell, j \neq t, i \neq k.& \\ y_{ik} + y_{i\ell} + y_{jt} + y_{ju} - y_{ij} - y_{k\ell} - y_{tu} &\leq 1 && (5.17)\\ \forall i,j,k,\ell,t,u \text{ com } i \neq j, i \neq k, i \neq \ell, k \neq \ell, j \neq t, j \neq u, t \neq u.& \end{aligned}$$

Note que se fizermos $k = t$ em (5.17), a desigualdade corresponde a desigualdade Z. Note também que se $\ell = t$ em (5.17), a desigualdade nada mais é do que a desigualdade W. A formulação $Q_{2,2}$ requer $n+m$ variáveis com $O(n^3)$ restrições, enquanto que a formulação $P_{2,2}$ requer somente m variáveis, mas $O(n^6)$ restrições.

Fica evidente então que se $Q_{2,2}$ conjuntamente com a restrição de cardinalidade for utilizada como relaxação linear, o limitante superior resultante da formulação estendida será mais justo do que o obtido usando a formulação natural com todas as desigualdades triangulares, Z e W.

Os resultados acima são suficientes para mostrar que a abordagem no modelo estendido é superior a abordagem no modelo natural, uma vez que com um número menor de restrições o valor obtido na relaxação linear do modelo estendido é mais justo do que a formulação no modelo natural que possui bem mais restrições.

Desigualdades Obtidas por Projeção

A primeira desigualdade descrita nesta seção é uma generalização das desigualdades triangular e W. Essa família de desigualdades pode ser obtida através da projeção das

desigualdades clique e corte-triangular válidas para o politopo do Problema Quadrático Booleano. Sejam S e T_i dois conjuntos disjuntos em V_n, com $|S| \geq 1$, e $i \in S$ e $t(i)$ e $u(i) \in T_i$. Então a desigualdade

$$\sum_{i \in S} (y_{it(i)} + y_{iu(i)} - y_{t(i)u(i)}) - \sum_{e \in E(S)} y_e \leq 1$$

é denominada *desigualdade sunflower*.

Park *et al.* [41] conseguiram demonstrar que a desigualdade anterior pode ser obtida através da adição das seguintes desigualdades

$$\sum_{i \in S} x_i - \sum_{e \in E(S)} y_e \leq 1,$$

e

$$y_{it(i)} + y_{iu(i)} - y_{t(i)u(i)} - x_i \leq 0 \quad \forall i \in S.$$

Isto facilmente atesta o fato de que a desigualdade *sunflower* é válida para o PCN_b. De uma maneira geral, Park *et al.* [40] definiram a desigualdade *sunflower* como abaixo

$$\sum_{i \in S} \sum_{t \in T_i} y_{it} - \sum_{i \in S} \sum_{e \in E(T_i)} y_e - \sum_{e \in E(S)} y_e \leq 1 \tag{5.18}$$

A desigualdade (5.18) pode ser obtida a partir do Corolário 5.1 quando fazemos $p = n - 1$ e q $= n$. A Proposição abaixo estabelece que a desigualdade (5.18) define faceta para o politopo PCN_b.

Proposição 5.16 *Se $b \geq 4$ e as condições são satisfeitas*

$$\begin{aligned} T_i &\subseteq V_n - S \quad \forall i \in S; \\ Se\ |T_i| &= 1 \quad \textit{para algum } i \in S \textit{ então } T_i \subseteq T_j \quad \forall j \in S; \\ |T_i \cap T_j| &\leq 1 \quad \forall i, j \in S. \end{aligned}$$

então a desigualdade sunflower *define uma faceta para o PCN_b.*

Prova:

Considere as desigualdades cortes obtidas para cada $i \in S$ e o conjunto T_i. Considere ainda que a desigualdade clique obtida com o conjunto S e $\alpha = 1$. Somando-se todas estas desigualdades pode-se trivialmente ver que a desigualdade *sunflower* é válida para o PCN_b.

□

Pode-se demonstrar que todas as desigualdades triangulares, Z, W e 1-p-partição são casos especiais da desigualdade *sunflower*. A Figura 5.6 mostra exemplos de desigualdades *sunflower*. Em particular, na Figura 5.6(a) note que a desigualdade *sunflower* para $|S| = 1$ e $|T_i| = 2$ corresponde à desigualdade triangular, enquanto que se fizermos $S = \{i, j\}$ e $|T_i| = |T_j| = 2$, tal que, $|T_i \cap T_j| = 1$, obteremos a desigualdade W.

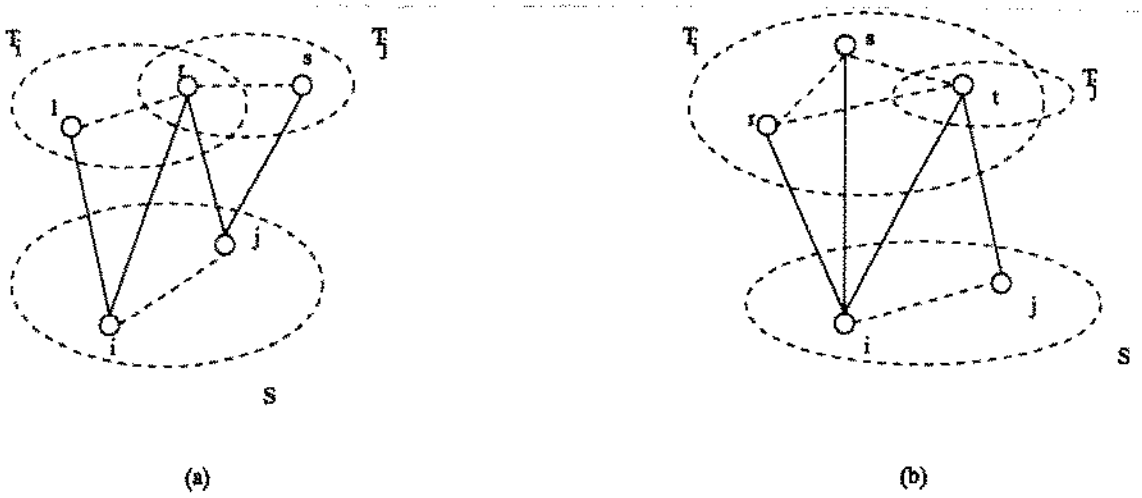


Figura 5.6: (a) Grafo suporte de uma desigualdade *sunflower* para $S = \{i, j\}$, $T_i = \{r, \ell\}$ e $T_j = \{r, s\}$. A desigualdade *sunflower* correspondente é $y_{it} + y_{ir} + y_{jr} + y_{js} - y_{r\ell} - y_{rs} - y_{ij} \leq 1$. (b) Grafo suporte de uma desigualdade *sunflower* para $S = \{i, j\}$, $T_i = \{r, s, t\}$ e $T_j = \{t\}$. A desigualdade correspondente é $y_{ir} + y_{is} + y_{it} + y_{jt} - y_{rs} - y_{rt} - y_{st} - y_{ij} \leq 1$.

Entretanto, Park *et al.* [40] deduziram que a desigualdade (5.18) nem sempre define uma faceta para o PCN_b. Eles constataram que se $T_i = T$ para todo $i \in S$, com $T \subseteq V_n - S$, é possível encontrar uma desigualdade que domina a desigualdade (5.18). Tal desigualdade é expressa em seguida

$$\sum_{e \in E(S:T)} y_e - \sum_{e \in E(S)} y_e - 2 \sum_{e \in E(T)} y_e \leq 1 \tag{5.19}$$

A desigualdade acima é denominada *desigualdade 2-partição*.

Proposição 5.17 *Se $b \geq 4$ e $|S| \geq 3$, a* desigualdade 2-partição *define uma faceta para o politopo PCN_b.*

Prova: Considere que C seja uma clique qualquer e que $|C \cap S| = \tilde{s}$ e $|C \cap T| = \tilde{t}$. Com isso obtemos $\tilde{t}\tilde{s} - \tilde{s}(\tilde{s}-1)/2 - \tilde{t}(\tilde{t}-1) \leq 1$. Ou ainda, $f(\tilde{t}, \tilde{s}) = \frac{1}{2}(-\tilde{s}^2 + \tilde{s} + 2\tilde{t}\tilde{s} - 2\tilde{t}^2 + 2\tilde{t}) \leq 1$. É fácil demonstrar que para $\tilde{s} = 2$ e $\tilde{t} = 1$ e $\tilde{s} = 2$ e $\tilde{t} = 2$, a função $f(\tilde{t}, \tilde{s})$ atingi o valor máximo, e portanto temos a validade.

□

Note que quando $|T| = 1$, a desigualdade (5.19) reduz-se à desigualdade 1-p-partição. Além disso, a desigualdade (5.19) engloba as desigualdades 2-p-partição e 3-p-partição propostas por nós. Para isto basta fazermos $|T| = 2$ e $|T| = 3$, respectivamente.

A última família de facetas proposta por Park *et al.* [41] no modelo natural generaliza as desigualdades Z. Essa família de facetas, a exemplo das desigualdades *sunflower*, é obtida a partir da projeção de desigualdades válidas para o politopo do Problema Quadrático Booleano.

Sejam u e $v \in V_n$ dois vértices, com $u \neq v$ e S e T dois subconjuntos de vértices em V_n. Considere ainda que $S \cap T \neq \emptyset$. Então a desigualdade abaixo

$$\sum_{s \in S} y_{us} + \sum_{t \in T} y_{vt} - \sum_{e \in E(S)} y_e - \sum_{e \in E(T)} y_e - y_{uv} \leq 1 \tag{5.20}$$

é denominada *desigualdade Z generalizada do tipo 1.*

Proposição 5.18 *Para* $b \geq 4$, *a* desigualdade Z generalizada do tipo 1 *define uma faceta para o politopo* PCN_b.

Prova:

A desigualdade (5.20) pode ser obtida através da adição das seguintes desigualdades válidas para o politopo PCE_b

$$x_u + x_v - y_{uv} \leq 1,$$

$$\sum_{s \in S} y_{us} - \sum_{e \in E(S)} y_e - x_u \leq 0$$

e

$$\sum_{t \in T} y_{vt} - \sum_{e \in E(T)} y_e - x_v \leq 0.$$

o que facilmente atesta a validade da desigualdade Z generalizada do tipo 1.

□

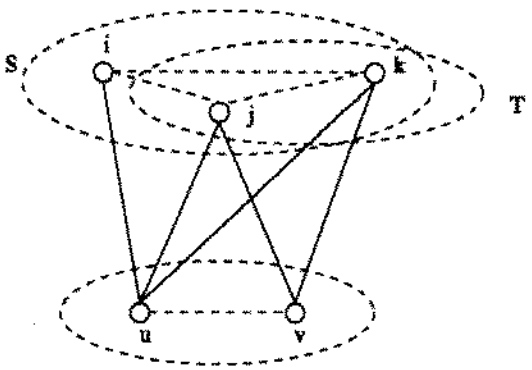


Figura 5.7: Grafo suporte de uma desigualdade Z generalizada do tipo 1 para $S = \{i, j, k\}$ e $T = \{j, k\}$. A desigualdade correspondente é $y_{iu}+y_{ju}+y_{ku}+y_{jv}+y_{kv}-y_{ij}-y_{jk}-y_{ik}-y_{uv} \leq 1$.

A Figura 5.7 mostra um exemplo de desigualdade (5.20). Note que se fizermos $S = \{i, j\}$ e $T = \{j\}$ a desigualdade (5.20) nada mais é do que a desigualdade Z.

Um outro tipo de desigualdade Z generalizada é a *desigualdade Z generalizada do tipo 2*. Seja $r \in V_n$ um vértice qualquer, $S \subseteq V_n - \{r\}$ um subconjunto de vértices com $|S| \geq 1$, e $T_i \subseteq V_n - \{r\}$ um outro subconjunto de vértices para todo $i \in S$. Assuma ainda que $T_i \cap T_j = \emptyset$ para quaisquer dois vértices distintos $i, j \in S$. Seja $U_i = \{r\} \cup T_i$. Então a desigualdade

$$\sum_{i \in S} \sum_{j \in U_i} y_{ij} - \sum_{i \in S} \sum_{e \in E(U_i)} y_e - \sum_{e \in E(S)} y_e \leq 1 \qquad (5.21)$$

é denominada desigualdade Z generalizada do tipo 2.

A Figura 5.8 mostra um exemplo de desigualdade (5.21). Se fizermos $S = \{i, j\}$, $T_j = \{\ell\}$ é possível obter a desigualdade Z a partir da desigualdade (5.21).

Proposição 5.19 *Para $b \geq 4$, a* desigualdade Z generalizada do tipo 2 *define uma faceta para o politopo do PCN_b.*

Prova:

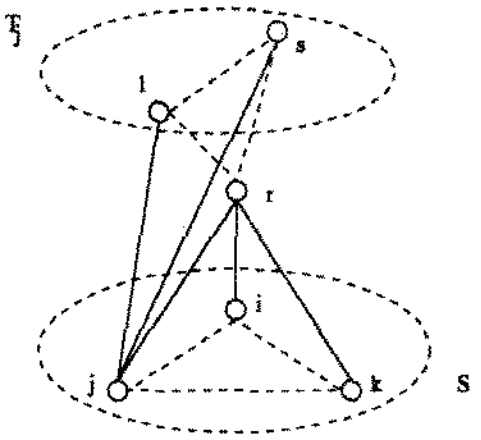


Figura 5.8: Grafo suporte de uma desigualdade Z generalizada do tipo 2 para $S = \{i, j, k\}$ e $T_j = \{\ell, s\}$. A desigualdade correspondente é $y_{j\ell} + y_{js} + y_{jr} + y_{ir} + y_{kr} - y_{s\ell} - y_{r\ell} - y_{rs} - y_{ij} - y_{ik} - y_{jk} \leq 1$.

Novamante pode-se fazer uso da adição de desigualdades válidas para o politopo PCE_b para a obtenção da desigualdade Z generalizada do tipo 2. Essas desigualdades estão expressas abaixo

$$\sum_{e \in E(S:U_i)} y_e - \sum_{e \in E(U_i)} y_e - x_i \leq 0 \quad \forall i \in S$$

e

$$\sum_{i \in S} x_i - \sum_{e \in E(S)} y_e \leq 1.$$

Portanto, a desigualdade Z generalizada do tipo 2 é válida para o politopo PCN_b.

□

Nas seções 5.2, 5.3.1 e 5.3.2 enumeramos algumas famílias de facetas para o Problema da Clique Máxima com Peso nas Arestas para ambos os modelos. Como já mencionamos, algumas das famílias por nós propostas já existiam na literatura, apesar de terem sido obtidas de forma independente. Na próxima seção, apresentaremos novas famílias de facetas por nós propostas para o Problema da Clique Máxima com Peso nas Arestas no modelo estendido e que acreditamos serem originais.

5.3.3 Novas Famílias de Facetas

Nesta seção apresentaremos cinco novas famílias de desigualdades válidas no modelo estendido que foram obtidas dos estudos realizados no politopo PCE_b neste trabalho.

Na oportunidade mostramos condições necessárias para que estas desigualdades sejam definidoras de facetas.

A primeira desigualdade a ser apresentada é uma generalização da desigualdade corte e foi obtida por Padberg [38] com o emprego de argumentos de "simetria". Esta desigualdade sob certas condições, como veremos em seguida, também é uma desigualdade válida para o politopo PCE_b.

Lema 5.1 *Sejam $S \subseteq V_n$ e $T \subseteq V_n - S$ conjuntos quaisquer de vértices , com $|S| = s \geq 1$ e $|T| = t \geq 2$. Para $b \geq 1$ a* desigualdade $[s : t]$-corte

$$\sum_{e \in E(S:T)} y_e - \sum_{e \in E(T)} y_e - \sum_{e \in E(S)} y_e + (s-t)\sum_{i \in S} x_i + (t-s-1)\sum_{i \in T} x_i \leq \frac{1}{2}(t-s)(t-s-1) \quad (5.22)$$

é válida para o politopo PCE_b.

Prova: Considere que $t > s$ e C seja uma clique viável. Assuma ainda que $|C \cap S| = \tilde{s}$ e $|C \cap T| = \tilde{t}$, tal que, $0 \leq \tilde{s} \leq s$ e $0 \leq \tilde{t} \leq t$. Com isso, obtemos para a desigualdade (5.22) $\tilde{t}\tilde{s} - \tilde{t}(\tilde{t}-1)/2 - \tilde{s}(\tilde{s}-1)/2 + (s-t)\tilde{s} + (t-s-1)\tilde{t} \leq (t-s)(t-s-1)/2$. Ou ainda,

$$\begin{aligned}
\frac{1}{2}[2\tilde{t}\tilde{s} - \tilde{t}^2 + \tilde{t} - \tilde{s}^2 + \tilde{s} + 2(s-t)\tilde{s} + 2(t-s-1)\tilde{t} - (t-s)(t-s-1)] &\leq 0 \\
\therefore \; -\frac{1}{2}[\tilde{t}^2 - 2\tilde{t}\tilde{s} - \tilde{t} - 2(t-s-1)\tilde{t} + \tilde{s}^2 - \tilde{s} - 2(s-t)\tilde{s} + (t-s)(t-s-1)] &\leq 0 \\
\therefore \; -\frac{1}{2}[\tilde{t}^2 - 2\tilde{t}\tilde{s} - \tilde{t} - 2(t-s-1)\tilde{t} + \tilde{s}^2 + \tilde{s}(-2(s-t)-1) + (t-s)(t-s-1) &\leq 0
\end{aligned}$$

Pode-se fatorar o polinômio $\tilde{s}^2 + \tilde{s}(-2(s-t)-1) + (t-s)(t-s-1)$ obtendo:

$$\begin{aligned}
-\frac{1}{2}[\tilde{t}^2 - 2\tilde{t}\tilde{s} - \tilde{t} - 2(t-s-1)\tilde{t} + (\tilde{s} + (t-s))(\tilde{s} + (t-s-1))] &\leq 0 \\
\therefore \; -\frac{1}{2}[\tilde{t}^2 - \tilde{t}(2\tilde{s} + 1 + 2t - 2s - 2) + (\tilde{s} + t - s)(\tilde{s} + t - s - 1))] &\leq 0 \\
\therefore \; -\frac{1}{2}[\tilde{t}^2 - \tilde{t}(2\tilde{s} + 2t - 2s - 1) + (\tilde{s} + t - s)(\tilde{s} + t - s - 1))] &\leq 0 \quad (5.23)
\end{aligned}$$

Agora fatorando-se o polinômio em $\tilde{t}$ do lado esquerdo de (5.23) tem-se:

$$-\frac{1}{2}[(\tilde{t}-(\tilde{s}+t-s))(\tilde{t}-(\tilde{s}+t-s-1))] \leq 0$$
$$\therefore \quad -\frac{1}{2}(\tilde{t}-\tilde{s}-t+s)(\tilde{t}-\tilde{s}-t+s+1) \leq 0 \qquad (5.24)$$

De fato, como $\tilde{t}-\tilde{s}-t+s$ e $\tilde{t}-\tilde{s}-t+s+1$ são números inteiros consecutivos, o lado esquerdo da desigualdade (5.24) é sempre menor ou igual a zero. Logo, a desigualdade $[s:t]$-corte é válida.

□

Antes de provarmos que a desigualdade $[s:t]$-corte é definidora de faceta para o PCE_b caracterizemos as *raízes* da equação correspondente. Para isto, considere o lema enunciado abaixo.

Lema 5.2 *Seja C uma clique viável e $S \subseteq V_n$ e $T \subseteq V_n - S$ dois subconjuntos quaisquer no grafo suporte de uma desigualdade do tipo $[s:t]$-corte, tal que, $|S| = s$ e $|T| = t$. Considere ainda que $\tilde{s} = |C \cap S|$ e $\tilde{t} = |C \cap T|$, com $0 \leq \tilde{s} \leq s$ e $0 \leq \tilde{t} \leq t$.*

(1) Se $t > s$ então tomando-se $\tilde{t} = (t-s)+\tilde{s}$ ou $\tilde{t} = (t-s)+\tilde{s}-1$, obtém-se uma raiz *da desigualdade (5.22). Além disso, se $\tilde{s} = 0$ e $\tilde{t} = (t-s)$ tem-se uma outra* raiz *de (5.22).*

(2) Considere agora $s > t$. Fazendo-se $\tilde{s} = (s-t+1)+\tilde{t}$ ou $\tilde{s} = (s-t+1)+\tilde{t}-1$, obtém-se uma raiz *da desigualdade (5.22). Além do mais, se fizermos $\tilde{t} = 0$ e $\tilde{s} = (s-t+1)$ tem-se uma outra* raiz *de (5.22).*

Prova: A prova será realizada apenas para o caso (1), ou seja, quando $t > s$. Porém , a mesma pode ser estendida quando $s > t$. A prova consiste em demonstrar que o lado esquerdo da desigualdade (5.22) é igual ao lado direito quando substituímos os valores de $\tilde{s}$ e $\tilde{t}$ em (5.22).

Considere os casos abaixo:

(i) $\tilde{s} = 0$ e $\tilde{t} = (t-s)$. Substituindo em (5.22) obtemos

$$-\frac{1}{2}\tilde{t}(\tilde{t}-1)+\tilde{t}(t-s-1)$$
$$\therefore \quad -\frac{1}{2}(t-s)(t-s-1)+(t-s)(t-s-1)$$

$$\therefore \ \frac{1}{2}(t-s)(t-s-1)$$

(ii) $\tilde{t} = (t-s) + \tilde{s}$. Substituindo em (5.22) obtemos

$$\tilde{s}(t-s+\tilde{s}) - \frac{1}{2}\tilde{s}(\tilde{s}-1) - \frac{1}{2}(t-s+\tilde{s})(t-s+\tilde{s}-1) + \tilde{s}(s-t) + (t-s-1)(t-s+\tilde{s})$$

$$\therefore \ \tilde{s}(t-s)+\tilde{s}^2-\frac{1}{2}(t-s)(t-s-1)-\frac{1}{2}\tilde{s}(t-s)-\frac{1}{2}\tilde{s}^2-\frac{1}{2}\tilde{s}(t-s-1)+(t-s-1)(t-s)+\tilde{s}(t-s-1)$$

$$\therefore \ \tilde{s}^2-\frac{1}{2}\tilde{s}^2+\frac{1}{2}\tilde{s}-\frac{1}{2}(t-s)(t-s-1)-\frac{1}{2}\tilde{s}t+\frac{1}{2}\tilde{s}s-\frac{1}{2}\tilde{s}^2-\frac{1}{2}\tilde{s}t+\frac{1}{2}\tilde{s}s+\frac{1}{2}\tilde{s}+(t-s-1)(t-s)+\tilde{s}t-\tilde{s}s-\tilde{s}$$

$$\therefore \ \frac{1}{2}(t-s)(t-s-1)$$

(iii) $\tilde{t} = (t-s) + \tilde{s} - 1$. Substituindo em (5.22) obtemos

$$\tilde{s}(t-s+\tilde{s}-1)-\frac{1}{2}\tilde{s}(\tilde{s}-1)-\frac{1}{2}(t-s+\tilde{s}-1)(t-s+\tilde{s}-1-1)+\tilde{s}(s-t)+(t-s-1)(t-s+\tilde{s}-1)$$

$$\therefore \ -\tilde{s}^2-\tilde{s}-\frac{1}{2}\tilde{s}^2+\frac{1}{2}\tilde{s}+\frac{1}{2}(t-s)(t-s-1)+\frac{1}{2}(\tilde{s}-1)(t-s-1)-\frac{1}{2}\tilde{s}^2+\tilde{s}+\frac{1}{2}-\frac{1}{2}(t-s)(\tilde{s}-1)$$

$$\therefore \ \frac{1}{2}(t-s)(t-s-1)$$

□

Temos agora condições de enunciar o teorema que caracteriza a desigualdade $[s:t]$-corte como definidora de faceta para o politopo PCE_b.

Teorema 5.4 *Sejam $S \subseteq V_n$ e $T \subseteq V_n - S$, com $|S| = s \geq 1$ e $|T| = t \geq 2$, conjuntos quaisquer de vértices. Para $b \geq 3$ a* desigualdade $[s:t]$-corte *define uma faceta para o politopo PCE_b.*

Prova:

Assuma que $S = \{1, 2, \ldots, s\}$. Seja $\mathcal{F}$ a face expressa pela desigualdade (5.22) e $\pi x + \beta y \leq \alpha_0$ uma desigualdade que define uma face para o politopo PCE_b, tal que, $\mathcal{F} \subseteq \mathcal{F}_{(\pi,\beta)} = \{(x,y) \in PCE_b : \pi x + \beta y = \alpha_0\}$. Então, desde que $\mathcal{F} \neq \emptyset$ poderemos concluir que a desigualdade (5.22) define uma faceta para o politopo PCE_b.

Durante a prova assuma que $\mathcal{X}^C$ representa o vetor de incidência de uma solução viável e que a construção de χ^C segue o processo de encontrar as raízes da desigualdade (5.22) como enunciado no Lema 5.2.

Afirmação 1. $\beta(a,b) = -\alpha$ e $\beta(a,c) = \alpha$ para todo $a, b \in S$ e $c \in T$.
Prova. Considere que $\tilde{T} \subseteq T$, tal que, $|\tilde{T}| = t - s$. Sejam $a \in S$, $b \in S - \{a\}$ e $c \in T - \tilde{T}$ vértices quaisquer. Sabendo-se que $\chi^{\tilde{T}} \in \mathcal{F}$, $\chi^{\tilde{T}\cup\{a\}} \in \mathcal{F}$ e $\chi^{\tilde{T}\cup\{b\}} \in \mathcal{F}$ obtemos, respectivamente, que

$$(\pi,\beta)\tilde{T} = \alpha_0 \quad (5.25)$$
$$(\pi,\beta)\tilde{T} + \pi_a + \beta(a,\tilde{T}) = \alpha_0 \quad (5.26)$$
$$(\pi,\beta)\tilde{T} + \pi_b + \beta(b,\tilde{T}) = \alpha_0 \quad (5.27)$$

Através de (5.25) e (5.26), e (5.25) e (5.27) é possível deduzir que $\pi_a + \beta(a,\tilde{T}) = 0$ e $\pi_b + \beta(b,\tilde{T}) = 0$. Com base nestes resultados, e sabendo-se que $\chi^{\tilde{T}\cup\{a\}\cup\{c\}} \in \mathcal{F}$, pode-se concluir que $\pi_c + \beta(c,\tilde{T}) + \beta(a,c) = 0$. Além do mais, $\chi^{\tilde{T}\cup\{a\}\cup\{b\}\cup\{c\}} \in \mathcal{F}$, logo segue-se que $\pi_a + \pi_b + \pi_c + \beta(a,\tilde{T}) + \beta(b,\tilde{T}) + \beta(c,\tilde{T}) + \beta(a,b) + \beta(a,c) + \beta(b,c) = 0$. Portanto, $\beta(a,b) + \beta(b,c) = 0$, ou ainda, $\beta(a,b) = -\beta(b,c)$.

Considere agora que $\tilde{T} \subseteq T$, tal que, $|\tilde{T}| = t - s - 1$. Sejam $a \in S$, $c \in T - \tilde{T}$ e $d \in T - (\tilde{T} \cup \{c\})$ vértices quaisquer, e $\chi^{\tilde{T}} \in \mathcal{F}$, $\chi^{\tilde{T}\cup\{a\}} \in \mathcal{F}$ e $\chi^{\tilde{T}\cup\{b\}} \in \mathcal{F}$ vetores de incidência. É possível deduzir que $\pi_c + \beta(c,\tilde{T}) = 0$ e $\pi_d + \beta(d,\tilde{T}) = 0$. Sabendo-se ainda que $\chi^{\tilde{T}} \in \mathcal{F}$, $\chi^{\tilde{T}\cup\{a\}\cup\{c\}} \in \mathcal{F}$ e $\chi^{\tilde{T}\cup\{a\}\cup\{d\}} \in \mathcal{F}$, segue-se que $\pi_a + \beta(a,\tilde{T}) + \beta(a,c) = 0$ e $\pi_a + \beta(a,\tilde{T}) + \beta(a,d) = 0$, respectivamente. Portanto, com base nos resultados anteriores temos que $\beta(a,c) = \beta(a,d) = \alpha_a$. Aplicando este mesmo raciocínio para $b \in S - \{a\}$ obtemos $\beta(b,c) = \beta(b,d) = \alpha_b$. Portanto, desde que $\beta(a,b) = -\beta(b,c)$ temos $\beta(a,b) = -\beta(a,c) = -\alpha_a$ e $\beta(a,b) = -\beta(b,c) = -\alpha_b$, ou seja, $\beta(a,b) = -\alpha_a = -\alpha_b = -\alpha$. Assim, para todo $i \in S$ temos $\alpha_i = \alpha$, e portanto, $\beta(a,c) = \beta(b,c) = \alpha$.

(d.c.a.)

A partir deste ponto considere que $\tilde{T} \subseteq T$ seja um conjunto de vértices quaisquer, tal que, $|\tilde{T}| = t - s$.

Afirmação 2. $\beta(z,w) = 0$ para todo $z, w \in V_n - (S \cup T)$.
Prova. Seja $\tilde{T} \subseteq T$ um conjunto de vértices, tal que, $|\tilde{T}| = t - s$, e $z, w \in V_n - (S \cup T)$ vértices quaiquer. É possível deduzir que $\pi_z + \beta(z,\tilde{T}) = 0$ e $\pi_w + \beta(w,\tilde{T}) = 0$ desde que $\chi^{\tilde{T}} \in \mathcal{F}$, $\chi^{\tilde{T}\cup\{z\}} \in \mathcal{F}$ e $\chi^{\tilde{T}\cup\{w\}} \in \mathcal{F}$. Com base nestes resultados e sabendo-se que $\chi^{\tilde{T}} \in \mathcal{F}$ e $\chi^{\tilde{T}\cup\{z\}\cup\{w\}} \in \mathcal{F}$ obtemos $\beta(z,w) = 0$.

(d.c.a.)

Afirmação 3. $\beta(u,z) = 0$ e $\beta(v,z) = 0$ para todo $u \in S$, para todo $v \in T - \tilde{T}$ e para todo $z \in V_n - (S \cup T)$.

Prova. Sejam $u \in S$, $v \in T - \tilde{T}$ e $z \in V_n - (S \cup T)$ vértices quaisquer. Através dos vetores de incidência $\chi^{\tilde{T}} \in \mathcal{F}$, $\chi^{\tilde{T}\cup\{u\}} \in \mathcal{F}$, e $\chi^{\tilde{T}\cup\{z\}} \in \mathcal{F}$ e $\chi^{\tilde{T}\cup\{u\}\cup\{z\}} \in \mathcal{F}$ é possível deduzir que $\beta(u,z) = 0$.

Considerando agora os vetores de incidência $\chi^{\tilde{T}} \in \mathcal{F}$, $\chi^{\tilde{T}\cup\{z\}} \in \mathcal{F}$, $\chi^{\tilde{T}\cup\{v\}\cup\{u\}} \in \mathcal{F}$ e $\chi^{\tilde{T}\cup\{v\}\cup\{u\}\cup\{z\}} \in \mathcal{F}$, e sabendo-se que $\beta(u,z) = 0$, pode-se deduzir que $\beta(v,z) = 0$.

(d.c.a.)

Afirmação 4. $\pi_z = 0$ para todo $z \in V_n - (S \cup T)$.
Prova. Seja $z \in V_n - (S \cup T)$ um vértice qualquer. Através dos vetores de incidência $\chi^{\tilde{T}} \in \mathcal{F}$ e $\chi^{\tilde{T}\cup\{z\}} \in \mathcal{F}$ pode-se deduzir que $\pi_z = 0$, pois $\beta(z,\tilde{T}) = 0$ com base nos resultados acima.

(d.c.a.)

Afirmação 5. $\beta(b,d) = -\alpha$ para todo $b, d \in T$.
Prova. Sejam $a \in S$, $c \in S - \{a\}$, $b \in T - (\tilde{T} \cup \{b\})$ e $d \in \tilde{T}\{b\}$. Através dos vetores de incidência $\chi^{\tilde{T}} \in \mathcal{F}$, $\chi^{\tilde{T}\cup\{a\}\cup\{b\}} \in \mathcal{F}$ e $\chi^{\tilde{T}\cup\{c\}\cup\{d\}} \in \mathcal{F}$ é possível deduzir que $\pi_a + \pi_b + \beta(a,\tilde{T}) + \beta(b,\tilde{T}) + \beta(a,b) = 0$ e $\pi_c + \pi_c + \beta(c,\tilde{T}) + \beta(d,\tilde{T}) + \beta(c,d) = 0$. Logo, como $\chi^{\tilde{T}} \in \mathcal{F}$ e $\chi^{\tilde{T}\cup\{a\}\cup\{b\}\cup\{c\}\cup\{d\}} \in \mathcal{F}$, $\beta(a,d) = \beta(b,c) = \alpha$ e $\beta(a,c) = \beta(b,d) = -\alpha$, obtemos $\beta(b,d) = -\alpha$.

(d.c.a.)

Afirmação 6. $\pi_a = \alpha(s-t)$ para todo $a \in S$.
Prova. Seja $a \in S$ um vértice qualquer. Desde que $\chi^{\tilde{T}} \in \mathcal{F}$ e $\chi^{\tilde{T}\cup\{a\}} \in \mathcal{F}$ é possível deduzir que $\pi_a + \beta(a,\tilde{T}) = 0$. Como $\beta(a,\tilde{T}) = -\alpha(t-s)$, pode-se concluir que $\pi_a = \alpha(t-s)$.

(d.c.a.)

Afirmação 7. $\pi_c = \alpha(t-s-1)$ para todo $c \in T$.
Prova. Sejam $a \in S$ e $c \in T - \tilde{T}$ vértices quaisquer. Através dos vetores de incidência $\chi^{\tilde{T}} \in \mathcal{F}$, $\chi^{\tilde{T}\cup\{a\}} \in \mathcal{F}$ e $\chi^{\tilde{T}\cup\{a\}\cup\{c\}} \in \mathcal{F}$ obtemos $\pi_c + \beta(c,\tilde{T}) + \beta(a,c) = 0$. Sabendo-se que $\beta(c,\tilde{T}) = -\alpha(t-s)$ e $\beta(a,c) = \alpha$, conclui-se que $\pi_c = \alpha(t-s-1)$.

(d.c.a.)

□

Na verdade a desigualdade $[s:t]$-corte é um caso particular da desigualdade *flor-do-sul*

$$\sum_{e\in E(S:T)} y_e - \sum_{e\in E(T)} y_e - \sum_{e\in E(S)} y_e - \alpha\sum_{i\in S} x_i + \beta\sum_{i\in T} x_i \leq \frac{1}{2}\alpha\beta \qquad (5.28)$$

para $\alpha - \beta = 1$, com α e $\beta \in \mathbb{Z}$.

A seguir, daremos a prova de validade da desigualdade (5.28).

Lema 5.3 *Sejam $S \subseteq V_n$ e $T \subseteq V_n - S$ dois subconjuntos quaisquer de vértices, com $|S| = s \geq 1$ e $|T| = t \geq 2$. Para $b \geq 1$ e $\alpha - \beta = 1$, com α e $\beta \in \mathbb{Z}$, a desigualdade* flor-do-sul *é válida para o politopo PCE_b.*

Prova: Considere que C seja uma clique viável, e $|C \cap S| = \tilde{s}$ e $|C \cap T| = \tilde{t}$. Substituindo em (5.28) obtemos

$$\begin{aligned}
\tilde{s}\tilde{t} - \frac{\tilde{s}(\tilde{s}-1)}{2} - \frac{\tilde{t}(\tilde{t}-1)}{2} - \alpha\tilde{s} + \beta\tilde{t} &\leq \frac{1}{2}\alpha\beta \\
\therefore \frac{1}{2}[2\tilde{s}\tilde{t} - \tilde{s}^2 + \tilde{s} - \tilde{t}^2 + \tilde{t} - 2\alpha\tilde{s} + 2\beta\tilde{t} - \alpha\beta] &\leq 0 \\
\therefore \frac{1}{2}[-\tilde{t}^2 + \tilde{t} + 2\tilde{t}(\alpha-1) + 2\tilde{s}\tilde{t} - \tilde{s}^2 + \tilde{s} - 2\tilde{s}\alpha - \alpha(\alpha-1)] &\leq 0 \\
\therefore \frac{1}{2}[-\tilde{t}^2 + \tilde{t} + 2\tilde{t}(\alpha-1) + 2\tilde{s}\tilde{t} - (\tilde{s}^2 + \tilde{s}(2\alpha-1) + \alpha(\alpha-1))] &\leq 0
\end{aligned}$$

Fatorando-se o polinômio $\tilde{s}^2 + \tilde{s}(2\alpha-1) + \alpha(\alpha-1)$ obtemos

$$\begin{aligned}
\frac{1}{2}[-\tilde{t}^2 + \tilde{t} + 2\tilde{t}(\alpha-1) + 2\tilde{s}\tilde{t} - (\tilde{s}+\alpha)(\tilde{s}+\alpha-1)] &\leq 0 \\
\therefore \frac{1}{2}[-\tilde{t}^2 + (2\alpha + 2\tilde{s} - 1)\tilde{t} - (\tilde{s}+\alpha)(\tilde{s}+\alpha-1)] &\leq 0 \qquad (5.29)
\end{aligned}$$

Agora fatorando-se o polinômio em $\tilde{t}$ do lado esquerdo de (5.29) obtemos

$$\begin{aligned}
-\frac{1}{2}[(\tilde{t} - (\tilde{s}+\alpha))(\tilde{t} - (\tilde{s}+\alpha-1))] &\leq 0 \\
\therefore -\frac{1}{2}[(\tilde{t} - \tilde{s} - \alpha)(\tilde{t} - \tilde{s} - \alpha + 1)] &\leq 0 \qquad (5.30)
\end{aligned}$$

Como $\tilde{t} - \tilde{s} - \alpha$ e $\tilde{t} - \tilde{s} - \alpha + 1$ são números inteiros consecutivos, o lado esquerdo da desigualdade (5.30) é sempre menor ou igual a zero. Logo, a desigualdade *flor-do-sul* é válida.

□

Enunciamos em seguida dois casos da desigualdade *flor-do-sul* que definem facetas para o politopo PCE_b. Considere a desigualdade *flor-do-sul*

$$\sum_{e \in E(S:T)} y_e - \sum_{e \in E(T)} y_e - \sum_{e \in E(S)} y_e - 2\sum_{i \in S} x_i + \sum_{i \in T} x_i \leq 1 \tag{5.31}$$

definida para $\alpha = 2$ e $\beta = 1$. A partir deste ponto faremos referência a desigualdade (5.31) como sendo a *desigualdade [2:1]-flor-do-sul.*

Teorema 5.5 *Sejam $S \subseteq V_n$ e $T \subseteq V_n - S$ dois subconjuntos de vértices, com $|S| = s \geq 1$ e $|T| = t \geq 3$. Para $b \geq 4$, a* desigualdade [2:1]-flor-do-sul *define uma faceta para o politopo PCE_b.*

Prova:

Seja $\mathcal{F}$ a face expressa pela desigualdade (5.31) e $\pi x + \beta y = \alpha_0$ uma desigualdade que define uma face para o politopo PCE_b, tal que, $\mathcal{F} \subseteq \mathcal{F}_{(\pi,\beta)} = \{(x,y) \in PCE_b : \pi x + \beta y = \alpha_0\}$. Então, como $\mathcal{F} \neq \emptyset$ poderemos concluir que a desigualdade (5.31) define uma faceta para o politopo PCE_b se $\pi x + \beta y \leq \alpha_0$ for múltiplo escalar não-negativo de (5.31).

Considere que $\mathcal{X}^C$ represente o vetor de incidência da clique viável C.

Caso 1. Como $\mathcal{X}^{\{j\}} \in \mathcal{F}$, para qualquer $j \in T$, isto implica que $\pi_j = \alpha_0$.

Caso 2. Como $\mathcal{X}^{\{j,k\}} \in \mathcal{F}$, para qualquer $j,k \in T$, temos $\pi_j + \pi_k + \beta_{jk} = \alpha_0$. Como $\pi_j = \pi_k = \alpha_0$ isto implica que $\beta_{jk} = -\alpha_0$.

Caso 3. Como $\mathcal{X}^{\{i,j,k\}} \in \mathcal{F}$, para qualquer $i \in S$ e $j,k \in T$, segue-se que $\pi_j + \pi_k + \pi_i + \beta_{ij} + \beta_{ik} + \beta_{jk} = \alpha_0$. Com base nos resultados anteriores podemos deduzir

$$\pi_i + \beta_{ik} + \beta_{ij} = 0 \tag{5.32}$$

Caso 4. Como $\mathcal{X}^{\{i,j,k,\ell\}} \in \mathcal{F}$, para qualquer $i \in S$ e $j,k,\ell \in T$, segue-se então que $\pi_i + \pi_j + \pi_k + \pi_\ell + \beta_{ij} + \beta_{ik} + \beta_{i\ell} + \beta_{jk} + \beta_{j\ell} + \beta_{k\ell} = \alpha_0$. Pelos resultados anteriores, obtemos $\beta_{i\ell} = -\beta_{jk} = \alpha_0$. Logo, $\beta_{ij} = \beta_{ik} = \alpha_0$. Pode-se concluir ainda que $\pi_i = -2\alpha_0$ através de (5.32).

Caso 5. Como $\mathcal{X}^{\{i,r,j,k,\ell\}} \in \mathcal{F}$, para qualquer $i,r \in S$ e $j,k,\ell \in T$, segue-se então que

$\pi_i+\pi_j+\pi_k+\pi_\ell+\pi_r+\beta_{ij}+\beta_{ik}+\beta_{i\ell}+\beta_{ir}+\beta_{jk}+\beta_{j\ell}+\beta_{jr}+\beta_{k\ell}+\beta_{kr}+\beta_{r\ell}=\alpha_0$. Sabendo-se que $\pi_j=\pi_k=\pi_\ell=\alpha_0$, $\pi_i=\pi_r=-2\alpha_0$, $\beta_{ij}=\beta_{ik}=\beta_{i\ell}=\beta_{jr}=\beta_{r\ell}=\beta_{rk}=\alpha_0$ e $\beta_{jk}=\beta_{j\ell}=\beta_{k\ell}=-\alpha_0$, deduz-se que $\beta_{ir}=-\alpha_0$.

Caso 6. Como $\mathcal{X}^{\{j,k,z\}}\in\mathcal{F}$, para qualquer $j,k\in T$ e $z\in V_n-(S\cup T)$, segue-se que $\pi_j+\pi_k+\pi_z+\beta_{jk}+\beta_{jz}+\beta_{kz}=\alpha_0$. Isto implica em

$$\pi_z+\beta_{jz}+\beta_{kz}=0 \tag{5.33}$$

Caso 7. Como $\mathcal{X}^{\{j,z\}}\in\mathcal{F}$, para qualquer $j\in T$ e $z\in V_n-(S\cup T)$, obtemos

$$\pi_z+\beta_{jz}=0 \tag{5.34}$$

Substituindo (5.34) em (5.33) deduz-se que $\beta_{kz}=0$. Portanto, $\beta_{jz}=0$. Com isto, $\pi_z=0$ através de (5.34).

Caso 8. Como $\mathcal{X}^{\{i,j,k,z\}}\in\mathcal{F}$, para qualquer $i\in S$, $j,k\in T$ e $z\in V_n-(S\cup T)$, deduz-se que $\pi_i+\pi_j+\pi_k+\pi_z+\beta_{jk}+\beta_{ji}+\beta_{jz}+\beta_{ki}+\beta_{kz}+\beta_{iz}=\alpha_0$. Isto implica em $\beta_{iz}=0$ com base nos resultados anteriores.

Caso 9. Como $\mathcal{X}^{\{i,j,k,z,w\}}\in\mathcal{F}$, para qualquer $i\in S$, $j,k\in T$ e $z,w\in V_n-(S\cup T)$, isto implica que $\pi_i+\pi_j+\pi_k+\pi_z+\pi_w+\beta_{jk}+\beta_{ji}+\beta_{jz}+\beta_{jw}+\beta_{ki}+\beta_{kz}+\beta_{kw}+\beta_{iz}+\beta_{iw}+\beta_{zw}=\alpha_0$. Segue-se que $\beta_{zw}=0$ com base nos resultados anteriores.

□

Considere agora a desigualdade *flor-do-sul* obtida para $\alpha=3$ e $\beta=2$.

$$\sum_{e\in E(S:T)}y_e-\sum_{e\in E(T)}y_e-\sum_{e\in E(S)}y_e-3\sum_{i\in S}x_i+2\sum_{i\in T}x_i\leq 3 \tag{5.35}$$

Denominaremos a desigualdade (5.35) como sendo a *desigualdade [3:2]-flor-do-sul*. O teorema abaixo menciona as condições nas quais esta desigualdade define faceta para o politopo PCE_b.

Teorema 5.6 *Considere que* $|S|=s\geq 1$ *e* $|T|=t\geq 4$. *Para* $b\geq 5$, $\alpha=3$ *e* $\beta=2$, *a* desigualdade [3:2]-flor-do-sul *define uma faceta para o* PCE_b.

Prova: Seja $\mathcal{F}=\{(x,y)\in PCE_b:\sum_{e\in E(S:T)}y_e-\sum_{e\in E(T)}y_e-\sum_{e\in E(S)}y_e-3\sum_{i\in S}x_i+2\sum_{i\in T}x_i=3\}$ uma face para PCE_b. Suponha que exista uma desigualdade $\pi x+\beta y\leq\alpha_0$ válida para o politopo PCE_b, tal que, cada ponto $(x,y)\in\mathcal{F}$ satisfaça $(\pi,\beta)(x,y)=\alpha_0$. Ou seja, a face $\mathcal{F}_{(\pi,\beta)}$ definida por $\pi x+\beta y\leq\alpha_0$ está contida em $\mathcal{F}$.

Considere que C seja uma clique viável e que $\mathcal{X}^C$ represente o seu vetor de incidência.

Caso 1. Como $\mathcal{X}^{\{j,k\}}$ para todo $j,k \in T$, segue-se que $\pi_j + \pi_k + \beta_{jk} = \alpha_0$.
Caso 2. Como $\mathcal{X}^{\{j,k,\ell\}}$ para todo $j,k,\ell \in T$, segue-se que $\pi_j+\pi_k+\pi_\ell+\beta_{jk}+\beta_{j\ell}+\beta_{k\ell} = \alpha_0$.
Caso 3. Como $\mathcal{X}^{\{i,j,k,\ell\}}$ para todo $j,k,\ell \in T$ e $i \in S$, segue-se que $\pi_j + \pi_k + \pi_\ell + \pi_i + \beta_{jk} + \beta_{j\ell} + \beta_{ji} + \beta_{k\ell} + \beta_{ki} + \beta_{i\ell} = \alpha_0$.
Caso 4. Como $\mathcal{X}^{\{i,j,k,\ell,u\}}$ para todo $j,k,\ell,u \in T$ e $i \in S$, segue-se que $\pi_j + \pi_k + \pi_\ell + \pi_i + \pi_u + \beta_{jk} + \beta_{j\ell} + \beta_{ij} + \beta_{ju} + \beta_{k\ell} + \beta_{ki} + \beta_{ku} + \beta_{i\ell} + \beta_{iu} + \beta_{u\ell} = \alpha_0$.

Como j,k,ℓ e u são vértices quaisquer em T pode-se concluir que $\beta_{jk} = \beta_{j\ell} = \beta_{ju} = \beta_{k\ell} = \beta_{ku} = \beta_{u\ell}$ e $\pi_j = \pi_k = \pi_\ell = \pi_u$ através dos casos acima. Com base nisso, combinando os resultados dos casos (1) e (2) é possível concluir que

$$\beta_{jk} + \pi_j = \alpha_0/3 \tag{5.36}$$

Portanto, substituindo a equação (5.36) em $\pi_j + \pi_k + \beta_{jk} = \alpha_0$ obtemos $\pi_k = 2\alpha_0/3$. Agora substituindo este valor em (5.36) tem-se $\beta_{jk} = -\alpha_0/3$.

É possível deduzir ainda, a partir dos casos (1)-(4), que $\beta_{ji} = \beta_{ki} = \beta_{i\ell}$ uma vez que os vértices $j,k,\ell \in T$ e $i \in S$ são quaisquer. Através disto, e sabendo-se dos resultados deduzidos anteriormente, pode-se concluir que $\beta_{iu} = \alpha_0/3$ e $\pi_i = -\alpha_0$, para todo $i \in S$ e para todo $u \in T$.

Em seguida, provaremos que $\beta_{jz} = \beta_{iz} = \pi_z = 0$ para todo $j \in T$, $i \in S$ e $z \in V_n - (S \cup T)$. Para isso considere os casos abaixo.

Caso 5. Como $\mathcal{X}^{\{j,k,z\}}$ para todo $j,k,\ell \in T$ e $z \in V_n - (S \cup T)$, segue-se que $\pi_j + \pi_k + \pi_z + \beta_{jk} + \beta_{jz} + \beta_{kz} = \alpha_0$. Sabendo-se que $\pi_j = \pi_k = 2\alpha_0/3$ e que $\beta_{jk} = -\alpha_0/3$ obtemos

$$\pi_z + \beta_{jz} + \beta_{kz} = 0 \tag{5.37}$$

Caso 6. Como $\mathcal{X}^{\{j,k,\ell,z\}}$ para todo $j,k,\ell \in T$ e $z \in V_n - (S \cup T)$, segue-se que $\pi_j + \pi_k + \pi_\ell + \pi_z + \beta_{jk} + \beta_{j\ell} + \beta_{jz} + \beta_{kz} + \beta_{k\ell} + \beta_{z\ell} = \alpha_0$. Através de (5.37) e dos resultados anteriores deduz-se que $\beta_{z\ell} = \pi_z = 0$. Assim, $\pi_z = 0$ a partir de (5.37).
Caso 7. Como $\mathcal{X}^{\{i,j,k,\ell,z\}}$ para todo $i \in S$, $j,k \in T$ e $z \in V_n - (S \cup T)$, segue-se que $\pi_j + \pi_k + \pi_\ell + \pi_i + \pi_z + \beta_{jk} + \beta_{j\ell} + \beta_{jz} + \beta_{ij} + \beta_{kz} + \beta_{k\ell} + \beta_{ki} + \beta_{i\ell} + \beta_{z\ell} + \beta_{zi} = \alpha_0$ é possível deduzir que $\beta_{iz} = 0$ a partir dos resultados anteriores.
Caso 8. Como $\mathcal{X}^{\{i,j,k,\ell,z,w\}}$ para todo $i \in S$, $j,k \in T$ e $z,w \in V_n - (S \cup T)$, segue-se que $\pi_j + \pi_k + \pi_\ell + \pi_i + \pi_z + \pi_w + \beta_{jk} + \beta_{j\ell} + \beta_{jz} + \beta_{jw} + \beta_{ij} + \beta_{kz} + \beta_{k\ell} + \beta_{ki} + \beta_{kw} + \beta_{i\ell} + \beta_{iw} + \beta_{z\ell} + \beta_{zi} + \beta_{zw} + \beta_{\ell w} = \alpha_0$. Com base nos resultados anteriores pode-se deduzir que $\beta_{zw} = 0$.
Caso 9. Como $\mathcal{X}^{\{i,j,k,\ell,u\}}$ para todo $j,k,\ell,u \in T$ e $i,r \in S$, segue-se que $\pi_j+\pi_k+\pi_\ell+\pi_i+\pi_u+\pi_r+\beta_{jk}+\beta_{j\ell}+\beta_{ij}+\beta_{ju}+\beta_{jr}+\beta_{k\ell}+\beta_{ki}+\beta_{ku}+\beta_{kr}+\beta_{i\ell}+\beta_{iu}+\beta_{ir}+\beta_{u\ell}+\beta_{ur}+\beta_{\ell r} = \alpha_0$. Sabendo-se que $\beta_{ij} = \beta_{ik} = \beta_{i\ell} = \beta_{iu} = \beta_{rj} = \beta_{rk} = \beta_{r\ell} = \beta_{ru} = \alpha_0/3$, $\beta_{jk} = \beta_{j\ell} = \beta_{ju} =$

$\beta_{k\ell} = \beta_{ku} = \beta_{\ell u} = -\alpha_0/3$, $\pi_i = \pi_r = -\alpha_0$ e $\pi_j = \pi_k = \pi_\ell = \pi_u = -2\alpha_0/3$ pode-se concluir que $\beta_{ir} = -\alpha_0/3$.

□

A próxima família de facetas é definida baseada nos trabalhos de de Souza [13] para alguns problemas de particionamento em grafos, em particular o Problema de *Cluster* Único. No caso de grafos completos, um *cluster* se confunde com uma clique e os resultados obtidos para o politopo daquele problema se aplicam diretamente ao PCE_b.

Como mencionado anteriormente, Johnson em et al. [30] também estudaram o Problema de *Cluster* Único e propuseram a desigualdade árvore definida na Proposição 5.13. de Souza [13] enunciou o lema seguinte que comprova a validade da desigualdade árvore e caracteriza os pontos na face correspondente do politopo associado ao Problema de *Cluster* Único.

Lema 5.4 *(de Souza [13]) Seja o grafo $G = (V, E)$, $\widehat{T}$ uma árvore em G e (x, y) uma solução viável do Problema de* Cluster *Único. Considere que A seja um subconjunto de vértices em V definido como $A = \{v \in V(\widehat{T}) : x_v = 1\}$. Se $c(A)$ é o número de componentes conectados induzidos por A em $\widehat{T}$ e $\delta_{\widehat{T}}(A)$ é o conjunto das arestas de corte em $(V(\widehat{T}), \widehat{T})$, então pode-se definir $w(\widehat{T})(x, y) = c(A) - |\delta_{\widehat{T}}(A)|$.*

Prova: [de Souza [13]]

□

Claramente é possível observar que $c(A)$ é sempre menor ou igual a $|\delta_{\widehat{T}}(A)|$, exceto no caso em que $A = V(\widehat{T})$, pois neste caso temos $c(A) = 1$ e $|\delta_{\widehat{T}}(A)| = 0$. Com base nestes resultados considere o lema enunciado abaixo.

Lema 5.5 *Seja $\widehat{T}$ uma árvore e $V(\widehat{T}) \subseteq V_n$ um conjunto de vértices, tal que, $|V(\widehat{T})| = b$. Então para $b \geq 1$*

$$w(\widehat{T})(x, y) = \sum_{e \in E(\widehat{T})} y_e - \sum_{i \in V(\widehat{T})} (d_i - 1)x_i \leq 1.$$

Prova: Considere o conjunto $A \subset V_n$, tal que, $A = \{i \in V(\widehat{T}) : x_i = 1\}$, $c(A)$ como definido no Lema 5.4 e $\delta_{\widehat{T}}(A)$ o conjunto das arestas de corte na árvore $\widehat{T}$. Sabendo-se que $w(\widehat{T})(x, y) = c(A) - |\delta_{\widehat{T}}(A)|$, assuma que $|A| = b$, logo $|\delta_{\widehat{T}}(A)| = 0$ e $c(A) = 1$, pois só existe um componente conexo com b vértices. Isto implica que $w(\widehat{T})(x, y) = 1$. Assuma agora que $|A| = k$ com $1 \leq k < b$. Segue-se que $c(A) \leq |\delta_{\widehat{T}}(A)|$, e portanto $w(\widehat{T})(x, y) = c(A) - |\delta_{\widehat{T}}(A)| \leq 0$.

□

O lema acima demonstra que a desigualdade árvore original com $b+1$ vértices é faceta. No entanto, se retirarmos um vértice esta desigualdade deixa de ser válida, pois o lado esquerdo passa a ser igual a 1. Entretanto, é possível recuperarmos a validade. Para isto, observemos uma clique externa. Realizando um *lifting* nos componentes desta clique, e lembrando que para um conjunto $S \subseteq V_n$ a desigualdade clique para $\alpha = 1$ é dada por

$$\tilde{w}(S)(x,y) = \sum_{i \in S} x_i - \sum_{e \in E(S)} y_e \leq 1$$

podemos definir a desigualdade *árvore-clique* como abaixo.

Lema 5.6 *Seja $S \subseteq V_n$ um conjunto de vértices, com $|S| \geq 2$, e $\hat{T} \subseteq S - V_n$ um conjunto de vértices que compõe uma árvore, com $|V(\hat{T})| = b$. Para $b \geq 2$ a desigualdade* árvore-clique

$$w(\hat{T})(x,y) + (\tilde{w}(S)(x,y)) = \sum_{e \in E(\hat{T})} y_e - \sum_{i \in V(\hat{T})} (d_i - 1)x_i + (\sum_{i \in S} x_i - \sum_{e \in E(S)} y_e) \leq 1 \quad (5.38)$$

é válida para o PCE_b.

Prova: Por contradição, assuma que a desigualdade (5.38) não é válida. Como $w(\hat{T})(x,y)$ e $\tilde{w}(S)(x,y)$ são quantidade inteiras menores ou iguais a 1, e a desigualdade não é válida, existe $(\tilde{x},\tilde{y}) \in PCE_b$, tal que, $w(\hat{T})(\tilde{x},\tilde{y}) = 1$ e $w(S)(\tilde{x},\tilde{y}) = 1$. Pelo Lema 5.4, $w(\hat{T})(\tilde{x},\tilde{y}) = 1$, se somente se, todos os vértices de $\hat{T}$ estão na clique. Como $|\hat{T}| = b$, isso implica $\tilde{x}_i = 0$ para todo $i \in V_n - \hat{T}$, ou seja, $w(S)(\tilde{x},\tilde{y})$ tem que ser igual a zero, contrariando a hipótese.

□

Embora não tenha sido possível encontrar as condições necessárias e suficientes para que a desigualdade (5.38) defina uma faceta de PCE_b, no teorema abaixo este resultado é mostrado no caso em que $\hat{T}$ é um caminho.

Teorema 5.7 *Seja $S \subseteq V_n$ um subconjunto de vértices quaisquer e $P \subseteq V_n - S$ um caminho, tal que, $|S| \geq 3$ e $|V(P)| = b$. Para $b \geq 2$ a desigualdade* caminho-clique

$$\sum_{e \in E(P)} y_e - \sum_{i \in V(P)} (d_i - 1)x_i + \sum_{i \in S} x_i - \sum_{e \in E(S)} y_e \leq 1 \quad (5.39)$$

define uma faceta para o PCE_b.

Prova: Seja $\mathcal{F}$ a face expressa pela desigualdade (5.39) e $\pi x+\beta y \leq \alpha_0$ uma desigualdade que define uma face para o politopo PCE_b, tal que, $\mathcal{F} \subseteq \mathcal{F}_{(\pi,\beta)} = \{(x,y) \in PCE_b : \pi x + \beta y = \alpha_0\}$. Então, desde que $\mathcal{F} \neq \emptyset$ poderemos concluir que a desigualdade (5.39) define uma faceta para o politopo PCE_b.

Como anteriormente mencionado assuma que $\mathcal{X}^C$ representa o vetor de incidência de uma clique viável C. Iniciemos a prova encontrando o valor dos coeficientes das variáveis que pertencem ao suporte da desigualdade clique e daquelas que não estão no suporte da desigualdade (5.39), mas que se ligam a variáveis do suporte da desigualdade clique. Considere os casos abaixo.

Caso 1. Desde que $\mathcal{X}^{\{i\}} \in \mathcal{F}$ para todo $i \in S$, segue-se que $\pi_i = \alpha_0$.

Caso 2. Desde que $\mathcal{X}^{\{i,j\}} \in \mathcal{F}$ para todo $i,j \in S$, segue-se que $\pi_i + \pi_j + \beta_{ij} = 0$. Com os resultados do caso (1) pode-se afirmar que $\beta_{ij} = -\alpha_0$.

Caso 3. Desde que $\mathcal{X}^{\{i,z\}} \in \mathcal{F}$ para todo $i \in S$ e $z \in V_n - (S \cup P)$. Isto implica em $\pi_i + \pi_z + \beta_{iz} = \alpha_0$. Portanto, $\pi_z + \beta_{iz} = 0$ desde que $\pi_i = \alpha_0$.

Caso 4. Desde que $\mathcal{X}^{\{i,j,z\}} \in \mathcal{F}$ para todo $i,j \in S$ e $z \in V_n - (S \cup P)$. Isto implica que $\pi_i + \pi_j + \pi_z + \beta_{ij} + \beta_{iz} + \beta_{jz} = \alpha_0$. Como $\pi_i = \pi_j = \alpha_0$ e $\beta_{ij} = -\alpha_0$, obtemos $\pi_z + \beta_{iz} + \beta_{jz} = 0$. No entanto, $\pi_z + \beta_{iz} = 0$ a partir do caso (3). Assim, $\beta_{jz} = 0$ e $\pi_z = 0$.

Finalizada esta primeira parte da prova, demos início a prova dos coeficientes das variáveis que pertencem ao suporte da desigualdade caminho e das que não pertencem ao suporte desta desigualdade, mas estão ligadas a ela.

Caso 5. Desde que $\mathcal{X}^{\{u_1,i\}} \in \mathcal{F}$ para todo $i \in S$ e $u_1 \in P$. Isto implica em $\pi_i + \pi_{u_1} + \beta_{iu_1} = \alpha_0$. Segue-se que $\pi_{u_1} + \beta_{iu_1} = 0$.

Caso 6. Desde que $\mathcal{X}^{\{i,j,u_1\}} \in \mathcal{F}$ para todo $i,j \in S$ e $u_1 \in P$, segue-se que $\pi_{u_1} + \pi_i + \pi_j + \beta_{u_1 i} + \beta_{u_1 j} + \beta_{ij} = \alpha_0$. Sabendo-se que $c_{u_1} + d_{iu_1} = 0$ pode-se deduzir que $\beta_{ju_1} = 0$. Logo, pelo caso (5) obtemos $\pi_{u_1} = 0$.

Caso 7. Desde que $\mathcal{X}^{\{u_1,i,z\}} \in \mathcal{F}$ para todo $i \in S$, $u_1 \in P$ e $z \in V_n - (S \cup P)$. Isto implica em $\pi_{u_1} + \pi_i + \pi_z + \beta_{u_1 z} + \beta_{u_1 i} + \beta_{iz} = \alpha_0$. Como $\pi_{u_1} = \pi_z = \beta(i,z) = 0$, $\pi_i = \alpha_0$ e $\pi_{u_1} + \beta_{iu_1} = 0$, obtemos $\beta_{u_1 z} = 0$.

Por simetria, podemos aplicar os casos (5)-(7) ao vértice u_b. Isto permite deduzir $\pi_{u_b} = \beta_{iu_b} = \beta_{ku_b} = 0$.

Caso 8. Desde que $\mathcal{X}^{\{u_1,u_2,i\}} \in \mathcal{F}$ para todo $i \in S$ e u_1, $u_2 \in P$. Isto implica em $\pi_{u_1} + \pi_{u_2} + \pi_i + \beta_{iu_1} + \beta_{iu_2} + \beta_{u_1 u_2} = \alpha_0$. Com base nos resultados anteriores obtemos $\pi_{u_2} + \beta_{iu_2} + \beta_{u_1 u_2} = 0$.

Caso 9. Desde que $\mathcal{X}^{\{u_1,u_2,i,j\}} \in \mathcal{F}$ para todo $i,j \in S$ e $u_1,u_2 \in P$. Isto implica em

$\pi_{u_1} + \pi_{u_2} + \pi_i + \pi_j + \beta_{iu_1} + \beta_{ju_1} + \beta_{iu_2} + \beta_{ju_2} + \beta_{ij} + \beta_{u_1u_2} = \alpha_0$. Sabendo-se que $\pi_{u_2} + \beta_{iu_2} + \beta_{u_1u_2} = 0$, podemos deduzir que $\beta_{ju_2} = 0$, e $\pi_{u_2} + \beta_{u_1u_2} = 0$ a partir do caso (8).

Caso 10. Desde que $\mathcal{X}^{\{u_1,u_2,i,j,z\}} \in \mathcal{F}$ para todo $i, j \in S$, $z \in V_n - (S \cup P)$ e $u_1, u_2 \in P$, segue-se que $\pi_{u_1} + \pi_{u_2} + \pi_i + \pi_j + \pi_z + \beta_{iu_1} + \beta_{ju_1} + \beta_{zu_1} + \beta_{iu_2} + \beta_{ju_2} + \beta_{zu_2} + \beta_{ij} + \beta_{iz} + \beta_{jz} + \beta_{u_1u_2} = \alpha_0$. É possível deduzir que $\beta_{zu_2} = 0$ uma vez que $\pi_{u_2} + d_{u_1u_2} = 0$.

Mantendo-se os vértices $i, j \in S$ e $z \in V_n - (S \cup P)$, ao se aplicar os casos (8)-(10) aos conjuntos $\{u_1, u_2, u_3\}$, $\{u_1, u_2, u_3, u_4\}$, ..., $\{u_1, u_2, u_3, \ldots, u_{b-3}, u_{b-2}\}$ obtemos

$$\sum_{r=1}^{p-1} \beta_{u_r u_p} + \pi_{u_p} = 0 \quad \text{para todo } p = 2, 3, \ldots, b-3, b-2. \tag{5.40}$$

e assim deduzir que $\beta_{iu_p} = \beta_{ku_p} = 0$ para todo $p = 2, 3, \ldots, b-3, b-2$. Através de simetria, ou seja, aplicando-se os casos (8)-(10) aos conjuntos $\{u_b, u_{b-1}\}$, $\{u_b, u_{b-1}, u_{b-2}\}$, ..., $\{u_b, u_{b-1}, \ldots, u_4, u_3\}$ obteríamos

$$\sum_{r=p+1}^{b} \beta_{u_r u_p} + \pi_{u_p} = 0 \quad \text{para todo } p = b-1, b-2, \ldots, 4, 3. \tag{5.41}$$

e $\beta_{iu_p} = \beta_{ku_p} = 0$ para todo $p = b-1, b-2, \ldots, 4, 3$.

Com base nos resultados anteriores falta deduzirmos os coeficientes de $\pi_{u_{p+1}}$, $\beta_{u_p u_{p+1}}$ e das cordas do caminho P para todo $p = 1, 2, 3, \ldots, b-1$.

Caso 11. Desde que $\mathcal{X}^{\{u_1,u_b,i\}} \in \mathcal{F}$ para todo $i \in S$ e $u_1, u_b \in P$. Isto implica em $\pi_{u_1} + \pi_{u_b} + \pi_i + \beta_{u_1u_b} + \beta_{iu_1} + \beta_{iu_b} = \alpha_0$. Segue-se que $\beta_{u_1u_b} = 0$.

Caso 12. Desde que $\mathcal{X}^{\{u_1,u_b,u_{b-1},i\}} \in \mathcal{F}$ para todo $i \in S$ e u_1, u_b, $u_{b-1} \in P$, segue-se que $\pi_{u_1} + \pi_{u_b} + \pi_{u_{b-1}} + \pi_i + \beta_{u_1i} + \beta_{u_1u_{b-1}} + \beta_{u_1u_b} + \beta_{u_{b-1}i} + \beta_{u_{b-1}u_b} + \beta_{u_bi} = \alpha_0$. Isto implica em $\beta_{u_1u_{b-1}} = 0$, pois $\pi_{u_{b-1}} + \beta_{u_1u_{b-1}} = 0$ a partir da equação (5.41).

Caso 13. Desde que $\mathcal{X}^{\{u_1,u_b,u_{b-1},u_{b-2},i\}} \in \mathcal{F}$ para todo $i \in S$ e $u_1, u_b, u_{b-1}, u_{b-2} \in P$, segue-se que $\pi_{u_1} + \pi_{u_b} + \pi_{u_{b-1}} + \pi_{u_{b-2}} + \pi_i + \beta_{u_1u_b} + \beta_{u_1u_{b-1}} + \beta_{u_1u_{b-2}} + \beta_{iu_1} + \beta_{u_bi} + \beta_{iu_{b-1}} + \beta_{iu_{b-2}} + \beta_{u_bu_{b-1}} + \beta_{u_bu_{b-2}} + \beta_{u_{b-1}u_{b-2}} = \alpha_0$. Por (5.41), $\pi_{u_{b-2}} + \beta_{u_{b-1}u_{b-2}} + \beta_{u_bu_{b-1}} = 0$, e usando-se dos resultados anteriores obtemos $\beta_{u_1u_{b-2}} = 0$.

Repetindo-se os casos (11)-(13) para os vértices $\{u_{b-3}, u_{b-4}, \ldots, u_{\lfloor \frac{b}{2} \rfloor + 1}, \ldots, u_4, u_3\}$, e fazendo-se uso da equação (5.41), obtemos $\beta_{u_1u_{b-3}} = \beta_{u_1u_{b-4}} = \ldots = \beta_{u_1u_{\lfloor \frac{b}{2} \rfloor + 1}} = \ldots = \beta_{u_1u_2} = 0$.

Considere agora os seguintes conjuntos $P_1 = \{u_2, u_3, \ldots, u_i\}$ e $P_2 = \{u_b, u_{b-1}, \ldots, u_j\}$ para todo $i = 2, 3, \ldots, b-2$ e para todo $j = b, b-1, \ldots, i+2$. Aplicando a mesma seqüência dos casos (11)-(13), ou seja, $\{u_1, u_2, u_b\}$, $\{u_1, u_2, u_b, u_{b-1}\}$, ..., $\{u_1, u_2, u_b, u_{b-1}, \ldots, u_4\}$,

com o auxílio das equações (5.40) e (5.41) obteremos $\beta_{u_r u_s} = 0$ para todo $r \in P_1$ e $s \in P_2$ com $r \neq s$.

Com o emprego dos casos (11)-(13) aos elementos dos conjuntos P_1 e P_2 é possível deduzir que

$$\pi_{u_{r+1}} = -\beta_{u_r u_{r+1}} \tag{5.42}$$

para todo $r = 2, 3, \ldots, b-2$ com o auxílio da equação (5.40). Por argumentos de simetria, fazendo-se uso da equação (5.41) obtemos

$$\pi_{u_r} = -\beta_{u_r u_{r+1}} \tag{5.43}$$

para todo $r = 2, 3, \ldots, b-1$.

Com isso, através de (5.42) e (5.43) conclui-se que todos os π_i são iguais, bem como todos os $\beta_{u_i u_{i+1}}$ para $i = 2, 3, \ldots, b-1$.

Caso 14. Desde que $\mathcal{X}^{\{u_1, u_2, \ldots, u_{b-2}, u_{b-1}, u_b\}} \in \mathcal{F}$ com $u_1, u_2, \ldots, u_{b-1}, u_b \in P$. Segue-se que $\pi_{u_1} + \pi_{u_2} + \ldots + \pi_{u_{b-2}} + \pi_{u_{b-1}} + \pi_{u_b} + \beta_{u_1 u_2} + \beta_{u_1 u_3} + \ldots + \beta_{u_1 u_{b-1}} + \beta_{u_1 u_b} + \ldots + \beta_{u_{b-2} u_{b-1}} + \beta_{u_{b-1} u_b} = \alpha_0$. Como $\pi_{u_{r+1}} = -\beta_{u_r u_{r+1}}$ para todo $r = 2, 3, \ldots, b-2$, e $\sum_{p=r+2}^{b} \beta_{u_p u_{p+1}} = 0$ para todo $p = 1, 2, \ldots, b-2$, pode-se concluir que $\beta_{u_1 u_2} = \alpha_0$ e ainda que $\pi_{u_2} = -\alpha_0$.

Portanto, a partir do caso (14) pode-se concluir $\pi_i = \alpha_0$ e $\beta_{u_i u_{i+1}} = -\alpha_0$ para todo $i = 2, 3, \ldots, b-1$.

□

Ainda trabalhando com a desigualdade árvore enunciada no Lema 5.5 é possível realizar um *lifting* na desigualdade $[s:t]$-corte quando $2 \leq |T| \leq |S| + k$, onde $k = 1$ ou $k = 2$; e recuperarmos a validade daquela desigualdade. O lema abaixo enuncia este resultado.

Lema 5.7 *Sejam $S \subseteq V_n$ e $T \subseteq V_n - S$ dois subconjuntos de vértices. Considere a árvore $\widehat{T}$ formada pelos vértices em $V(\widehat{T}) \subseteq V_n - (S \cup T)$. Para $b \geq 2$, $|S| \geq 1$ e $2 \leq |T| \leq |S| + k$, a desigualdade* árvore-corte

$$w(\widehat{T})(x,y) + (\hat{w}(S,T)(x,y)) = \sum_{e \in E(\widehat{T})} y_e - \sum_{i \in V(\widehat{T})} (d_i - 1)x_i \quad +$$

$$\Big(\sum_{e \in E(S:T)} y_e - \sum_{e \in E(S)} y_e - \sum_{e \in E(T)} y_e + (s-t) \sum_{i \in S} x_i + (t-s-1) \sum_{i \in T} x_i \Big) \quad \leq \quad 1 \tag{5.44}$$

é válida para o PCE_b.

Prova: Se $|T| = |S|$ ou $|T| = |S| + 1$ então o lado direito da desigualdade $[s:t]$-corte é igual zero, e portanto a desigualdade (5.44) é trivialmente válida devido o Lema 5.5. Se não, para $|T| = |S| + 2$ assuma, por contradição que (5.44) não é válida. Assim, existe um $(\tilde{x}, \tilde{y}) \in PCE_b$, tal que, $w(\hat{T})(x,y) = 1$ e $\hat{w}(S,T)(x,y) = 1$. Porém, $w(\hat{T})(x,y) = 1$, se somente se, todos os vértices de $\hat{T}$ estão na clique. Como $|V(\hat{T})| = b$, isso implica que $\tilde{x}_i = 0$ para todo $i \in V_n - \hat{T}$, ou seja, $\hat{w}(S,T)(x,y) = 0$, contrariando a hipótese.

□

O teorema a seguir fornece uma condição necessária para a desigualdade (5.44) definir uma faceta de PCE_b.

Teorema 5.8 *Sejam $S \subseteq V_n$ e $T \subseteq V_n - S$ dois subconjuntos de vértices. Considere o caminho P formado pelos vértices $V(P) \subseteq V_n - (S \cup T)$. Para $b \geq 2$, $|S| \geq 1$ e $|T| = |S| + 2$, a desigualdade* caminho-corte *define uma faceta para o PCE_b.*

Prova:

Seja $\mathcal{F} = \{(x,y) \in PCE_b : \sum_{e \in E(\hat{T})} y_e - \sum_{i \in V(\hat{T})} (d_i - 1)x_i + (\sum_{e \in E(S:T)} y_e - \sum_{e \in E(S)} y_e - \sum_{e \in E(T)} y_e + (s-t)\sum_{i \in S} x_i + (t-s-1)\sum_{i \in T} x_i) = 1\}$ uma face para PCE_b. Suponha que exista uma desigualdade $\pi x + \beta y \leq \alpha_0$ válida para o politopo PCE_b, tal que, cada ponto $(x,y) \in \mathcal{F}$ satisfaça $(\pi, \beta)(x,y) = \alpha_0$. Ou seja, a face $\mathcal{F}_{(\pi,\beta)}$ definida por $\pi x + \beta y \leq \alpha_0$ está contida em $\mathcal{F}$.

Denotaremos por $\mathcal{X}^C$ o vetor de incidência de uma clique viável. Em seguida, considere os casos abaixo. Iniciamos a prova demonstrando os coeficientes das variáveis que pertencem ao suporte da desigualdade $[s:t]$-corte e das que não estão no suporte da desigualdade (5.44), mas estão ligadas a elementos no suporte da desigualdade $[s:t]$-corte.

Caso 1. Desde que $\mathcal{X}^{\{j\}} \in \mathcal{F}$ para todo $j \in T$. Segue-se que $\pi_j = \alpha_0$.

Caso 2. Desde que $\mathcal{X}^{\{j,k\}} \in \mathcal{F}$ para todo $j,k \in T$. Isto implica em $\pi_j + \pi_k + \beta_{jk} = \alpha_0$. Como $\pi_j = \pi_k = \alpha_0$, obtemos $\beta_{jk} = -\alpha_0$.

Caso 3. Desde que $\mathcal{X}^{\{j,z\}} \in \mathcal{F}$ para todo $j \in T$ e $z \in V_n - (S \cup T \cup P)$. Com isso obtemos $\pi_j + \pi_z + \beta_{jz} = \alpha_0$. Logo, como $\pi_j = \alpha_0$ temos $\pi_z + \beta_{jz} = 0$.

Caso 4. Desde que $\mathcal{X}^{\{j,k,z\}} \in \mathcal{F}$ para todo $j,k \in T$ e $z \in V_n - (S \cup T \cup P)$. Segue-se que $\pi_j + \pi_k + \pi_z + \beta_{jk} + \beta_{jz} + \beta_{kz} = \alpha_0$. Assim, $\beta_{kz} = 0$, e a partir do caso (3) pode-se concluir que $\pi_z = 0$.

Caso 5. Desde que $\mathcal{X}^{\{i,j,k\}} \in \mathcal{F}$ para todo $i \in S$ e $j,k \in T$. Segue-se que $\pi_i + \pi_j + \pi_k + \beta_{ij} + \beta_{ik} + \beta_{jk} = \alpha_0$. Como $\pi_j = \pi_k = -\beta_{jk} = \alpha_0$, obtemos $\pi_i + \beta_{ij} + \beta_{ik} = 0$.

Caso 6. Desde que $\mathcal{X}^{\{i,j,k,\ell\}} \in \mathcal{F}$ para todo $i \in S$ e $j,k,\ell \in T$. Isto implica em $\pi_i + \pi_j + \pi_k + \pi_\ell + \beta_{ij} + \beta_{ik} + \beta_{i\ell} + \beta_{jk} + \beta_{j\ell} + \beta_{k\ell} = \alpha_0$. Com base nos resultados anteriores deduz-se que $\beta_{i\ell} = \alpha_0$, e a partir do caso (5) que $\pi_i = -2\alpha_0$.

Caso 7. Desde que $\mathcal{X}^{\{i,j,k,z\}} \in \mathcal{F}$ para todo $i \in S$, $j,k \in T$ e $z \in V_n - (S \cup T \cup P)$.

Segue-se que $\pi_i + \pi_j + \pi_k + \pi_z + \beta_{ij} + \beta_{ik} + \beta_{iz} + \beta_{jk} + \beta_{jz} + \beta_{kz} = \alpha_0$. Como $\pi_i = -2\alpha_0$, $\pi_j = \pi_k = \beta_{ij} = \beta_{ik} = -\beta_{jk} = \alpha_0$ e $\pi_z = \beta_{jz} = \beta_{kz} = 0$, obtemos $\beta_{iz} = 0$.
Caso 8. Desde que $\mathcal{X}^{\{i,j,k,\ell,r\}} \in \mathcal{F}$ para todo $i, r \in S$, $j, k, \ell \in T$. Segue-se que $\pi_i + \pi_r + \pi_j + \pi_k + \pi_\ell + \beta_{ij} + \beta_{ik} + \beta_{i\ell} + \beta_{ir} + \beta_{jk} + \beta_{j\ell} + \beta_{jr} + \beta_{k\ell} + \beta_{kr} + \beta_{\ell r} = \alpha_0$. Com base nos resultados anteriores deduz-se que $\beta_{ir} = -\alpha_0$.

Concluída esta primeira parte, iniciemos agora a prova dos coeficientes das variáveis que pertencem ao suporte da desigualdade caminho e daquelas que se ligam ao caminho.

Caso 9. Desde que $\mathcal{X}^{\{j,u_1\}} \in \mathcal{F}$ para todo $j \in T$ e $u_1 \in P$. Isto implica em $\pi_j + \pi_{u_1} + \beta_{ju_1} = \alpha_0$. Segue-se que $\pi_{u_1} + \beta_{ju_1} = 0$.
Caso 10. Desde que $\mathcal{X}^{\{j,z,u_1\}} \in \mathcal{F}$ para todo $j \in T$, $z \in V_n - (S \cup T \cup P)$ e $u_1 \in P$. Isto implica em $\pi_{u_1} + \pi_j + \pi_z + \beta_{u_1z} + \beta_{u_1j} + \beta_{jz} = \alpha_0$. Segue-se que $\pi_{u_1} + \beta_{u_1j} + \beta_{u_1z} = 0$. Entretanto, do caso (9) $\pi_{u_1} + \beta_{ju_1} = 0$, Logo, $\beta_{u_1r} = 0$.
Caso 11. Desde que $\mathcal{X}^{\{j,k,u_1\}} \in \mathcal{F}$ para todo $j, k \in T$ e $u_1 \in P$. Isto implica em $\pi_{u_1} + \pi_j + \pi_k + \beta_{u_1j} + \beta_{u_1k} + \beta_{jk} = \alpha_0$. Sabendo-se que $\pi_{u_1} + \beta_{ju_1} = 0$ a partir do caso (9), obtemos que $\beta_{ku_1} = 0$ e $\pi_{u_1} = 0$.
Caso 12. Desde que $\mathcal{X}^{\{i,j,k,u_1\}} \in \mathcal{F}$ para todo $i \in S$, para todo $j, k \in T$ e $u_1 \in P$. Isto implica em $\pi_{u_1} + \pi_i + \pi_j + \pi_k + \beta_{u_1i} + \beta_{u_1j} + \beta_{u_1k} + \beta_{ij} + \beta_{ik} + \beta_{jk} = \alpha_0$. Segue-se, com base nos resultados anteriores, que $\beta_{u_1i} = 0$.

Usando a seqüência de casos (9)-(12) para o vértice u_b, ao invés de u_1 ,obtemos $\pi_{u_b} = \beta_{u_bi} = \beta_{u_br} = \beta_{u_bk} = 0$ para todo $i \in S$, para todo $k \in T$ e $z \in V_n - (S \cup T \cup P)$.

Os casos seguintes irão permitir encontrar os coeficientes de π_p, $\beta_{u_pu_{p+1}}$, β_{u_pj} e β_{u_pi} para todo $p = 2, 3, \ldots, b-1$, para todo $i \in S$ e para todo $j \in T$; e $\beta_{u_ru_q}$ para todo $r = 1, 2, 3, \ldots, b-2$ e para todo $q = r+2, r+3, \ldots, b$.

Caso 13. Desde que $\mathcal{X}^{\{j,u_1,u_2\}} \in \mathcal{F}$ para todo $j \in T$ e $u_1, u_2 \in P$. Isto implica em $\pi_{u_1} + \pi_{u_2} + \pi_j + \beta_{ju_1} + \beta_{ju_2} + \beta_{u_1u_2} = \alpha_0$. Com os resultados anteriores obtemos $\pi_{u_2} + \beta_{ju_2} + \beta_{u_1u_2} = 0$.
Caso 14. Desde que $\mathcal{X}^{\{j,k,u_1,u_2\}} \in \mathcal{F}$ para todo $j, k \in T$ e $u_1, u_2 \in P$. Isto implica em $\pi_{u_1} + \pi_{u_2} + \pi_j + \pi_k + \beta_{ju_1} + \beta_{ku_1} + \beta_{ju_2} + \beta_{ku_2} + \beta_{jk} + \beta_{u_1u_2} = \alpha_0$. Sabendo-se que $\pi_{u_2} + \beta_{ju_2} + \beta_{u_1u_2} = 0$ do caso (13), podemos deduzir que $\beta_{ku_2} = 0$ e $\pi_{u_2} + \beta_{u_1u_2} = 0$.
Caso 15. Desde que $\mathcal{X}^{\{i,j,k,u_1,u_2\}} \in \mathcal{F}$ para todo $i \in S$, $j, k \in T$ e $u_1, u_2 \in P$. Segue-se que $\pi_{u_1} + \pi_{u_2} + \pi_i + \pi_j + \pi_k + \beta_{iu_1} + \beta_{ju_1} + \beta_{ku_1} + \beta_{iu_2} + \beta_{ju_2} + \beta_{ku_2} + \beta_{ij} + \beta_{ik} + \beta_{jk} + \beta_{u_1u_2} = \alpha_0$. É possível deduzir que $\beta_{iu_2} = 0$ uma vez que $\pi_{u_2} + \beta_{u_1u_2} = 0$.
Caso 16. Desde que $\mathcal{X}^{\{j,k,z,u_1,u_2\}} \in \mathcal{F}$ para todo $j, k \in T$, $z \in V_n - (S \cup T \cup P)$ e $u_1, u_2 \in P$. Isto implica em $\pi_{u_1} + \pi_{u_2} + \pi_j + \pi_k + \pi_z + \beta_{ju_1} + \beta_{ku_1} + \beta_{u_1z} + \beta_{ju_2} + \beta_{ku_2} + \beta_{zu_2} + \beta_{jk} +$

$\beta_{jz} + \beta_{kz} + \beta_{u_1u_2} = \alpha_0$. Segue-se que $\beta_{u_2z} = 0$ uma vez que $\pi_{u_2} + \beta_{u_1u_2} = 0$ do caso (14).

Aplicando os casos (13)-(16) aos conjuntos de vértices $\{u_1, u_2, u_3\}$, $\{u_1, u_2, u_3, u_4\}, \ldots,$ $\{u_1, u_2, \ldots, u_{b-3}\}$ obtemos $\beta_{u_pi} = \beta_{u_pj} = \beta_{u_pz} = 0$ para todo $i \in S$, $j \in T$ e $z \in V_n - (S \cup T \cup P)$.

$$\sum_{r=1}^{p-1} \beta_{u_pu_r} + \pi_{u_p} = 0 \qquad \text{para todo } p = 2, 3, 4, \ldots, b-4, b-3. \tag{5.45}$$

Por simetria, é possível deduzirmos

$$\sum_{r=s+1}^{b} \beta_{u_su_r} + \pi_{u_s} = 0 \qquad \text{para todo } s = b-3, b-4, \ldots, 3, 4. \tag{5.46}$$

para os conjuntos de vértices $\{u_b, u_{b-1}\}$, $\{u_b, u_{b-1}, u_{b-2}\}$, $\ldots$, $\{u_b, u_{b-1}, \ldots, u_5, u_4\}$.

Encontraremos agora os valores dos coeficientes das variáveis que pertencem apenas ao suporte da desigualdade caminho. Se fizermos $S = \emptyset$ na desigualdade (5.44) esta desigualdade torna-se idêntica a desigualdade caminho-clique, e portanto, os pontos afins independentes de ambas serão iguais. Logo, a dedução do restante dos coeficientes pode ser feita usando-se a mesma seqüência de passos dos casos (11)-(14) do Teorema 5.7 tomando-se apenas o cuidado de trocar $i \in S$ por $j \in T$.

□

Capítulo 6

Limitantes Inferiores

Em um algoritmo *branch-and-cut*, o cálculo dos limitantes inferiores e superiores é muito importante e merece uma atenção especial. Para um problema de maximização o valor da relaxação linear fortalecida pelos planos-de-corte adicionados à formulação inteira (inicial) corresponde ao limitante superior. Já o limitante inferior pode ser obtido através de duas maneiras distintas: a primeira na fase de inicialização do algoritmo *branch-and-cut* e a segunda durante a execução deste.

Neste capítulo, apresentaremos uma metaheurística utilizada na fase de inicialização do algoritmo *branch-and-cut* para a obtenção de bons limitantes inferiores (iniciais) para o nosso problema. No próximo capítulo, iremos expor a segunda maneira de determinarmos tais limitantes inferiores a partir de informações fornecidas pela relaxação linear.

6.1 Metaheurística *GRASP*

No Capítulo 3 vimos que a heurística *Greedy All* proposta por Späth [48] pode ser aplicado na resolução do PCMPA. Porém, os valores obtidos com esta heurística, apesar de serem expressivos, podem ser melhorados. Nesta seção, descrevemos o *GRASP* - *Greedy Randomized Adaptive Search Procedure* - proposto para o Problema da Clique Máxima com Peso nas Arestas. A escolha desta metaheurística deve-se à simplicidade da implementação e de estarmos utilizando este método apenas para a obtenção de limitantes inferiores.

Uma metaheurística consiste de várias heurísticas de caráter genérico que se adaptam facilmente às estruturas de arquiteturas paralelas e são direcionadas à uma otimização global de um problema, podendo conter diferentes procedimentos heurísticos de busca local em sua estrutura. Nas últimas décadas, surgiram vários procedimentos, enquadrados como metaheurísticas, empregados na resolução de problemas de otimização combinatória. Dentre elas, destacam-se: algoritmos genéticos, busca tabu, *simulated annealing* e, mais

recentemente, *A-Teams* e *GRASP*.

Um *GRASP* é um procedimento iterativo que combina várias propriedades favoráveis de outras heurísticas (Resende e Feo [18]). Mais especificamente, cada iteração do *GRASP* consiste de dois estágios: uma fase de construção da solução e uma fase de busca local. Em cada iteração, uma solução é encontrada e a melhor solução obtida dentre todas as iterações é considerada a solução final.

A Figura 6.1 apresenta um algoritmo genérico para o *GRASP*. No passo 1, temos a leitura dos dados de entrada. Nos passos 2-6, temos o *GRASP* propriamente dito. No passo 3, ocorre a fase de construção da solução. No passo 4, é realizada a fase de busca local. Por último, no passo 5, temos a atualização da solução caso algum melhoramento tenha sido obtido. O critério de parada comumente utilizado é estipular um número máximo de iterações.

```
Algoritmo GRASP()
  início
    1. Leitura dos dados.
    2. enquanto critério de parada não for satisfeito faça
    3.    Construa uma solução aleatória baseada em um critério guloso.
    4.    Realize uma busca local.
    5.    Atualize a solução.
    6. fim-enquanto
  fim
```

Figura 6.1: Algoritmo informal para um *GRASP* genérico.

Na fase de construção de uma solução, iniciamos com um conjunto vazio que iterativamente recebe um elemento até formar uma solução viável. Em cada iteração da fase de construção, dois aspectos são analisados: a aleatoriedade e a adaptação. A aleatoriedade se deve ao fato de que o próximo elemento a ser escolhido para compor a solução não será necessariamente o melhor, segundo o critério guloso utilizado. Na verdade a escolha é feita de forma aleatória, a partir de uma lista, denominada *Lista Restrita de Candidatos* (LRC), que contém os elementos candidatos. Por outro lado, a adaptatividade do processo se mostra evidente à medida que se escolhe um elemento que irá compor a solução inicial, pois os próximos elementos a serem inseridos em LRC dependerão dos já incluídos na solução inicial.

As soluções obtidas na fase de construção do *GRASP* não são garantidas como ótimos locais considerando uma dada vizinhança (uma solução s é dita ser *localmente ótima* se

não existe uma solução melhor na vizinhança de s). Portanto, o emprego da segunda fase do *GRASP* é feita, então, com o intuito de se melhorar a solução obtida na fase de construção. Um algoritmo de busca local seria utilizado para, sucessivamente, substituir a solução atual por uma solução melhor, encontrada na vizinhança da solução atual. O algoritmo terminaria quando nenhuma solução melhor fosse encontrada na vizinhança da solução atual ou após algum outro critério de parada.

Em seguida, descrevemos com detalhes cada uma das fases para o *GRASP* proposto ao PCMPA.

6.1.1 Fase de Construção

Na fase de construção, iniciamos com uma solução $C = \{s\}$. A escolha deste vértice s pode ser feita de duas maneiras: gulosa ou aleatória. Na escolha gulosa, o vértice $s \in V_n$ será aquele que apresentar o maior somatório dos pesos das arestas incidentes nele. A escolha aleatória é bastante simples e consiste em escolher um vértice qualquer $s \in V_n$.

Um outro aspecto que deve ser levantado nesta fase é a obtenção de LRC. A cada iteração são escolhidos os β melhores vértices candidatos a comporem a solução. Este valor β limita o tamanho de LRC e é denominado *restrição de cardinalidade*. A lista LRC é obtida em duas etapas. Na primeira, realiza-se o seguinte cálculo: $soma_i := \sum_{j \in C} c_{ij}$ para todo $i \notin C$. Este cálculo nos indicará quais vértices contribuem mais no peso da clique atual caso venham a ser adicionados a clique atual. Na segunda etapa, utilizamos o algoritmo Estatística de Ordem (c.f. Cormen *et al.* [10]) para obter os β primeiros vértices cujas as somas são as maiores sem a necessidade de ordená-las a cada iteração, o que demandaria um maior tempo computacional.

Por último, após a obtenção de LRC, selecionamos um vértice $s \in$ LRC de forma aleatória, e realizamos o processo de adaptação, ou seja, $V_n := V_n - \{s\}$. Na Figura 6.2 temos o algoritmo informal para a fase de construção descrita anteriormente. No passo 1, inicializamos a solução (inicial) C^0. Nos passos 2-4, escolhemos o vértice inicial, de forma gulosa, que irá compor C^0. Nos passos 5-18, são selecionados os outros vértices de C^0. Nos passos 6-12, são calculadas as somas que irão permitir a obtenção de LRC. No passo 13, seleciona-se os β melhores vértices que irão compor LRC. Nos passos 14-15 o vértice s é escolhido, de forma aleatória, e adicionado a C^0. Nos passos 16-17, é realizado o processo de adaptação.

Vale ressaltar que, para o algoritmo informal da Figura 6.2, o passo 3 pode ser substituído por

3. Escolha um vértice $s \in V_n$ de forma aleatória.

Algoritmo GRASP-PCMPA-CONSTRUÇÃO()

entrada: um grafo completo $K_n = (V_n, E_n)$ não-dirigido com pesos c_{ij} associado a cada aresta $(i,j) \in E_n$, e um inteiro $b > 0$.
saída: solução inicial C^0.

início

1. $C^0 := \emptyset$.
2. */* escolha do vértice inicial */*
3. Escolha $s \in V_n$, tal que, o somatório dos pesos das arestas incidentes em s seja máximo.
4. $C^0 := C^0 \cup \{s\}$.
5. **enquanto** não se atingiu a cardinalidade b **faça**
6. */* construção da lista LRC */*
7. **para** cada $j \in V_n$ **faça**
8. $Soma[j] := 0$.
9. **para** cada $i \in C^0$ **faça**
10. $Soma[j] := Soma[j] + c_{ij}$.
11. **fim-para**
12. **fim-para**
13. Selecione os β melhores vértices para compor LRC através do algoritmo Estatística de Ordem.
14. Escolha um vértice $s \in LRC$ de forma aleatória.
15. $C^0 := C^0 \cup \{s\}$.
16. */* adaptação */*
17. $V_n := V_n - \{s\}$.
18. **fim-enquanto**

fim

Figura 6.2: Algoritmo informal para a fase de construção do *GRASP*.

Esta alteração possibilita a implementação da segunda maneira (abordagem aleatória) de se obter uma solução inicial na fase de construção do *GRASP*.

6.1.2 Fase de Busca Local

Na fase de busca local, as soluções vizinhas à solução obtida na fase de construção serão geradas utilizando-se do mesmo princípio de pertubação empregado na heurística *Greedy All*. Os passos envolvidos nesta fase estão apresentados no algoritmo informal da Figura 6.3.

Algoritmo GRASP-PCMPA-BUSCA-LOCAL()

entrada: uma solução inicial C^0, um grafo completo $K_n = (V_n, E_n)$ não-dirigido com pesos c_{ij} associado a cada aresta $(i,j) \in E_n$ e um inteiro $b > 0$.
saída: uma solução $C \in N(C^0)$.

início

1. Construa o conjunto $\overline{C}^0 := V_n - C^0$.
2. $Ótimo := 0$.
3. **para** cada $i := 1$ até b **faça**
4. **para** cada $j := 1$ até $n - b$ **faça**
5. Escolha um par de vértcies $< p, q >$, com $p := C^0[i]$ e $q := \overline{C}^0[j]$.
6. **se** (f$((C^0 - \{p\}) \cup \{q\}) > Ótimo$)
7. **então** $Ótimo := f((C^0 - \{p\}) \cup \{q\})$.
8. $Ind1 := i$.
9. $Ind2 := j$.
10. **fim-se**
11. **fim-para**
12. **fim-para**
13. $C := (C^0 - \{C^0[Ind1]\}) \cup \{C^0[Ind2]\}$.

fim

Figura 6.3: Algoritmo informal para a fase de busca local do *GRASP*.

Na Figura 6.4 temos o *GRASP* proposto ao PCMPA. Na próxima seção apresentaremos os resultados computacionais obtidos com a execução da heurística *Greedy All* e do *GRASP* para algumas instâncias do PCMPA testadas.

Algoritmo GRASP-PCMPA()

entrada: um grafo completo $K_n = (V_n, E_n)$ não-dirigido com pesos c_{ij} associado a cada aresta $(i,j) \in E_n$ e um inteiro $b > 0$.
saída: limitante inferior.

início

1. **enquanto** número de iterações não for atingido **faça**
2. GRASP-PCMPA-CONSTRUÇÃO(C^0).
3. GRASP-PCMPA-BUSCA-LOCAL(C^0,K_n,C).
4. **se** C apresenta alguma melhoria com relação a C^0
5. **então** atualize o limitante inferior inicial.
6. **fim-se**
6. **fim-enquanto**

fim

Figura 6.4: Algoritmo informal para o *GRASP* proposto ao PCMPA.

6.2 Resultados Computacionais

Para investigarmos o desempenho da heurística e da metaheurística foram gerados 30 grafos completos $K_n = (V_n, E_n)$ de forma aleatória. Este conjunto de instâncias foi dividido em 6 grupos conforme mostrado na Tabela 6.1 . A primeira coluna se refere ao grupo da instância, a segunda coluna indica o número de vértices, a terceira coluna o número de arestas, e a quarta, e última, coluna a cardinalidade da clique.

Nós testamos a heurística e a metaheurística em dois casos. No primeiro caso os pesos associados as arestas são todos positivos. No segundo caso, eles são gerados para assumirem valores positivos e negativos. A obtenção destes valores é feita através de um processo semelhante ao descrito por Späth [48]: sejam k e w dois inteiros, com $k > 0$ e $w > 0$, e c_{ij} o peso associado a aresta $(i,j) \in E_n$ expresso por

- $1 \leq c_{ij} \leq \lfloor 10^{w+1} r^k \rfloor$, no caso de pesos positivos;
- $-\lfloor 10^{w+1} r^k \rfloor \leq c_{ij} \leq \lfloor 10^{w+1} r^k \rfloor$, no caso de pesos positivos e negativos;

onde r^k é a k-ésima potência de r, com $r \in \ ]0,1]$.

Os valores estipulados para k e w foram $k = 1$ a 5 e $w = 2$. Através dos valores de k foi possível obter 5 grafos por grupo. O critério de parada utilizado foi o mesmo tanto na heurística como na metaheurística: número máximo de iterações. Estabeleceu-se que esse número máximo de iterações, denotado por NUM_MAX_ITER, fosse igual ao número de arestas do grafo que estivesse sendo resolvido.

Grupo	n	m	b
Grafo I	40	780	20
Grafo II	42	861	21
Grafo III	44	946	22
Grafo IV	45	990	22
Grafo V	46	1035	23
Grafo VI	48	1128	24

Tabela 6.1: Instâncias utilizadas como testes.

O critério utilizado para investigar o desempenho dos métodos foi o quão distante está a melhor solução encontrada (valor obtido) por um dos métodos da solução ótima (ótimo inteiro) obtida nos experimentos descritos no próximo capítulo. Este valor, denotado por GAP, representará uma diferença percentual e será expresso por GAP = 100 × (ótimo inteiro - valor obtido) / ótimo inteiro.

6.2.1 Pesos Positivos

Neste caso, os pesos associados às arestas assumem valores pertencentes ao intervalo $1 \leq c_{ij} \leq 1000$.

Greedy All

A Tabela 6.2 mostra os resultados obtidos com o emprego da heurística *Greedy All* na resolução das 30 instâncias. A primeira coluna da tabela denota os grupos de instâncias. A segunda e terceira colunas indicam os valores de k e w, respectivamente. A quarta coluna, o valor obtido para o limitante inferior e a quinta coluna, o valor do GAP.

Observando a Tabela 6.2, verifica-se que não foi possível atingir o ótimo inteiro em nenhuma das instâncias. O menor valor obtido para o GAP foi observado para o grupo de instâncias Grafo V com $k = 1$ enquanto que o maior GAP no grupo Grafo II para $k = 5$.

GRASP

Os testes computacionais realizados com o *GRASP* foram divididos em dois experimentos de acordo com o tipo de abordagem empregado para a escolha do vértice inicial.

Grupo	k	w	Limitante	GAP
Grafo I ($n = 40$)	1	2	104266	4.65%
	2	2	75403	8.55%
	3	2	64737	5.85%
	4	2	56633	6.83%
	5	2	56048	7.38%
Grafo II ($n = 42$)	1	2	115552	3.95%
	2	2	83761	4.61%
	3	2	71424	6.70%
	4	2	64399	7.32%
	5	2	59901	10.97%
Grafo III ($n = 44$)	1	2	126524	7.33%
	2	2	90374	7.96%
	3	2	78305	7.52%
	4	2	67498	10.33%
	5	2	63842	8.19%
Grafo IV ($n = 45$)	1	2	129809	6.41%
	2	2	90502	7.95%
	3	2	76784	7.20%
	4	2	69812	9.92%
	5	2	62050	10.80%
Grafo V ($n = 46$)	1	2	137980	3.50%
	2	2	99838	7.76%
	3	2	86864	8.43%
	4	2	72094	8.45%
	5	2	66371	8.37%
Grafo VI ($n = 48$)	1	2	154560	5.41%
	2	2	107242	7.13%
	3	2	88564	8.38%
	4	2	80024	9.81%
	5	2	73932	9.97%

Tabela 6.2: Resultados obtidos com a heurística *Greedy All* para pesos positivos.

No primeiro, a escolha do vértice inicial que irá compor a clique obtida na fase de construção é feita de forma aleatória. No segundo experimento, o vértice inicial é obtido de forma gulosa. Em ambos os experimentos, utilizamos o mesmo número de iterações, NUM_MAX_ITER, e o mesmo tamanho da lista LCR expresso por β. Os valores de β foram estipulados em função do número de vértices. Os valores escolhidos foram $\beta = 0.5 \times n$, $0.6 \times n$, $0.7 \times n$, $0.8 \times n$ e $0.9 \times n$.

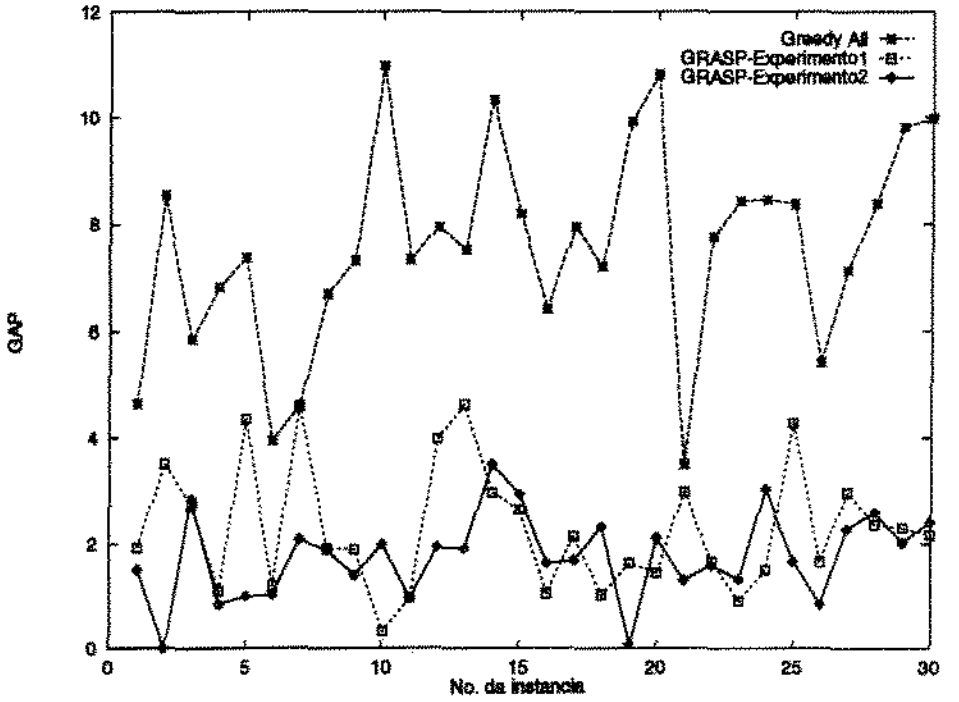


Figura 6.5: Comparação entre os métodos empregados para pesos com valores positivos.

Foi possível observar em ambos os experimentos que, para valores de β igual a $0.5 \times n$ e $0.6 \times n$, obtêm-se valores expressivos para o limitante inferior na grande maioria das instâncias. Porém, para algumas outras instâncias, ainda seria melhor o emprego da heurística *Greedy All*. No entanto, a partir de $\beta = 0.7 \times n$, todos os valores encontrados pela heurística Greedy All foram superados pela metaheurística *GRASP* (em ambos os experimentos). A Figura 6.5 ilustra esta superioridade.

A Tabela 6.3 apresenta os resultados obtidos nos Experimentos 1 e 2 para $\beta = 0.9 \times n$. Na tabela, a primeira coluna indica o grupo das instâncias. A segunda e a terceira colunas, os valores de k e w, respectivamente. A quarta e quinta colunas indicam, respectivamente, o limitante inferior e o GAP obtidos no Experimento 1. As duas últimas colunas, o limitante inferior e o GAP para o Experimento 2.

No Experimento 1, foi possível encontrar o ótimo inteiro em 17 das 30 instâncias testadas. Portanto, o valor mínimo obtido para o GAP foi de 0.00%, enquanto que

Grupo	k	w	Experimento 1 (Aleatório) Limitante	GAP	Experimento 2 (Guloso) Limitante	GAP
Grafo I ($n = 40$)	1	2	109346	0.00%	109346	0.00%
	2	2	82451	0.00%	82451	0.00%
	3	2	67763	1.45%	68579	0.00%
	4	2	60569	0.35%	60782	0.00%
	5	2	59958	0.92%	60513	0.00%
Grafo II ($n = 42$)	1	2	119400	0.75%	119901	0.31%
	2	2	86901	1.04%	87810	0.00%
	3	2	76554	0.00%	76554	0.00%
	4	2	69482	0.00%	69482	0.00%
	5	2	67383	0.00%	67383	0.00%
Grafo III ($n = 44$)	1	2	136525	0.00%	136525	0.00%
	2	2	96975	1.23%	98186	0.00%
	3	2	83660	1.20%	84675	0.00%
	4	2	75274	0.00%	75274	0.00%
	5	2	69540	0.00%	69540	0.00%
Grafo IV ($n = 45$)	1	2	138694	0.00%	138689	0.0036%
	2	2	98321	0.00%	97999	0.33%
	3	2	82644	0.12%	82743	0.00%
	4	2	77500	0.00%	77500	0.00%
	5	2	69563	0.00%	69519	0.06%
Grafo V ($n = 46$)	1	2	142985	0.00%	142985	0.00%
	2	2	108243	0.00%	108243	0.00%
	3	2	94859	0.00%	94859	0.00%
	4	2	78607	0.18%	78747	0.00%
	5	2	72290	0.19%	72375	0.08%
Grafo VI ($n = 48$)	1	2	163088	0.19%	163397	0.00%
	2	2	113130	2.03%	115350	0.10%
	3	2	96479	0.00%	96469	0.20%
	4	2	88197	0.60%	88108	0.70%
	5	2	82117	0.00%	82117	0.00%

Tabela 6.3: Resultados obtidos com o *GRASP* para os pesos positivos.

o máximo foi de 2.03%. No Experimento 2 foi possível atingir o ótimo em 22 das 30 instâncias, e o GAP diminui consideravelmente. Dessa forma, o valor mínimo para o GAP continuou em 0.00% e o máximo diminui para 0.70%.

A Tabela 6.3 nos mostra que o *GRASP* empregado na forma gulosa consegue resolver todas as instâncias do grupo Grafo I e III de forma ótima. Por outro lado, se utilizarmos os dois experimentos conjuntamente para a determinação do limitante inferior (inicial), é possível atingir o ótimo inteiro nos grupos Grafo I, III e IV.

6.2.2 Pesos Positivos e Negativos

Neste caso, os pesos associados às arestas passam a pertencer ao intervalo $-1000 \leq c_{ij} \leq 1000$. Na realização dos testes computacionais para este caso, utilizamos os mesmos parâmetros do caso de pesos positivos. Ou seja, os valores atribuídos a NUM_MAX_ITER e β permaneceram os mesmos.

Greedy All

Foi possível observar que a execução da heurística *Greedy All*, mais uma vez, não atingiu o ótimo inteiro em nenhuma da instâncias testadas. Pelo contrário, os valores do GAP só aumentaram, atingindo pontos que oscilam entre 10% e 35%, como pode ser visto na Tabela 6.4.

GRASP

O desempenho da metaheurística se manteve estável neste caso. Continuou-se a encontrar o ótimo inteiro na grande maioria das instâncias testadas, como pode ser visto na Tabela 6.5. Um fato relevante é que a partir de $\beta = 0.6 \times n$ todos os valores encontrados pela heurística *Greedy All* para o limitante inferior foram superados pela metaheurística *GRASP* em ambos os experimentos. Tal fato pode ser visualizado através do gráfico apresentado na Figura 6.6.

Através dos resultados apresentados em ambos os casos, fica evidente a superioridade do *GRASP* sobre a heurística *Greedy All* na obtenção de limitantes inferiores (iniciais), principalmente, quando trabalhamos com pesos cujos valores sejam positivos e negativos. No próximo capítulo veremos como usar estes bons limitantes inferiores na tentativa de melhorar o desempenho do algoritmo *branch-and-cut*.

Grupo	k	w	Limitante	GAP
Grafo I ($n = 40$)	1	2	60897	13.43%
	2	2	36450	19.72%
	3	2	25865	24.13%
	4	2	20408	26.48%
	5	2	22940	17.97%
Grafo II ($n = 42$)	1	2	72899	10.70%
	2	2	37082	20.81%
	3	2	28722	21.71%
	4	2	27821	22.69%
	5	2	24643	27.68%
Grafo III ($n = 44$)	1	2	83058	8.34%
	2	2	48921	14.11%
	3	2	33922	16.65%
	4	2	24555	24.68%
	5	2	22340	24.03%
Grafo IV ($n = 45$)	1	2	87049	14.90%
	2	2	42401	23.05%
	3	2	31193	27.31%
	4	2	23470	30.95%
	5	2	23678	23.56%
Grafo V ($n = 46$)	1	2	84679	14.94%
	2	2	47822	18.06%
	3	2	33595	23.50%
	4	2	26973	18.18%
	5	2	21145	31.79%
Grafo VI ($n = 48$)	1	2	100496	11.44%
	2	2	48656	21.23%
	3	2	34304	25.33%
	4	2	24010	34.94%
	5	2	25894	17.41%

Tabela 6.4: Resultados obtidos com a heurística *Greedy All* para os pesos positivos e negativos.

Grupo	k	w	Experimento 1 (Aleatório) Limitante	GAP	Experimento 2 (Guloso) Limitante	GAP
Grafo I ($n = 40$)	1	2	70348	0.00%	70348	0.00%
	2	2	44032	3.02%	45404	0.00%
	3	2	30972	9.15%	34091	0.00%
	4	2	27653	0.38%	27758	0.00%
	5	2	27080	3.17%	27967	0.00%
Grafo II ($n = 42$)	1	2	80541	1.34%	81633	0.00%
	2	2	45676	2.46%	46718	0.23%
	3	2	36689	0.00%	36689	0.00%
	4	2	35987	0.00%	35987	0.00%
	5	2	35460	0.00%	35460	0.00%
Grafo III ($n = 44$)	1	2	90620	0.00%	90620	0.00%
	2	2	56960	0.00%	56960	0.00%
	3	2	40697	0.00%	40967	0.00%
	4	2	32601	0.00%	31570	3.16%
	5	2	28679	2.48%	29407	0.00%
Grafo IV ($n = 45$)	1	2	102096	0.19%	102259	0.00%
	2	2	54528	1.04%	55103	0.00%
	3	2	43914	0.00%	43914	0.00%
	4	2	33990	0.00%	33517	1.39%
	5	2	30994	0.00%	30729	0.79%
Grafo V ($n = 46$)	1	2	98731	0.82%	98707	0.85%
	2	2	58361	0.00%	57871	0.84%
	3	2	43915	0.00%	43755	0.36%
	4	2	32968	0.00%	32951	0.05%
	5	2	31000	0.00%	31000	0.00%
Grafo VI ($n = 48$)	1	2	113425	0.19%	113458	0.02%
	2	2	60871	1.45%	61768	0.00%
	3	2	45941	0.00%	45941	0.00%
	4	2	36894	0.02%	36903	0.00%
	5	2	30937	1.32%	31081	0.86%

Tabela 6.5: Resultados obtidos com o *GRASP* para os pesos positivos e negativos.

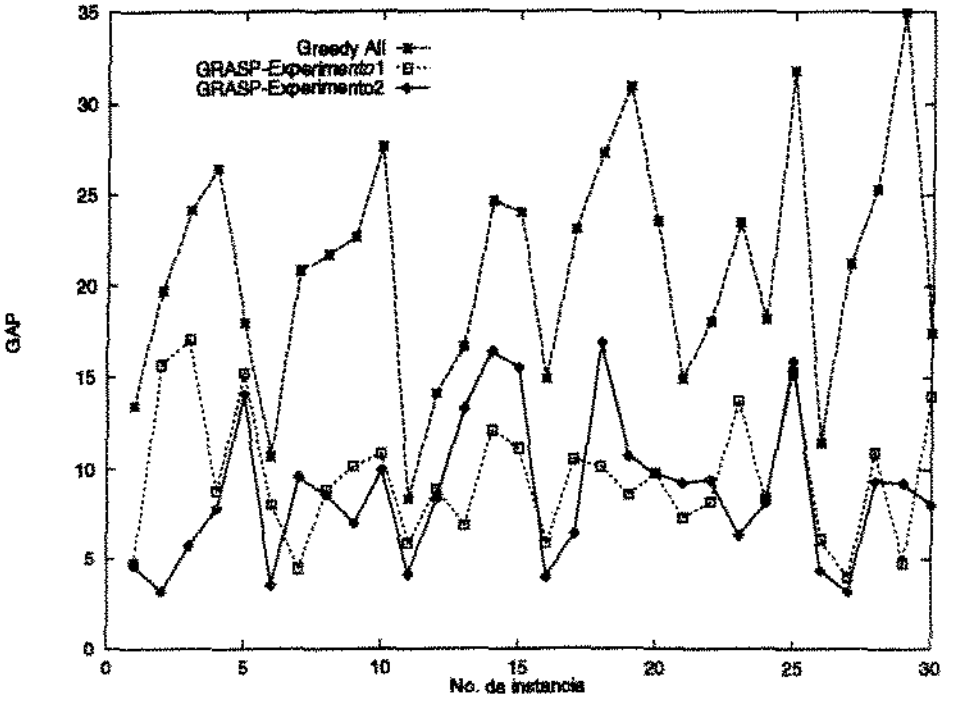


Figura 6.6: Comparação entre os métodos empregados para pesos com valores positivos e negativos.

Capítulo 7

Algoritmo Branch-and-Cut para o Problema da Clique Máxima com Peso nas Arestas

Neste capítulo descreveremos o algoritmo *branch-and-cut* implementado para o Problema da Clique Máxima com Peso nas Arestas. Esta técnica algorítmica é recente e vem sendo aplicada nos últimos anos na resolução de alguns problemas de otimização combinatória, como por exemplo, em problemas de escalonamento: Hoffman e Padberg [28] e Akker *et al.* [49]; em problemas de classificação: Barboza [2]; em problemas da Geometria Computacional: Nunes [37] e Meneses [12]; e em outros como: Jünger *et al.* [32], Padberg e Rinaldi [39], Jünger e Mutzel [31] e Ferreira *et al.* [19].

O "coração" de um algoritmo *branch-and-cut* está na geração de planos-de-corte específicos ao problema e na resolução de relaxações lineares (PLs) em cada nodo da árvore *branch-and-cut*. Esta característica traz como conseqüência alguns aspectos técnicos que tornam o projeto de um algoritmo *branch-and-cut* difícil. Alguns destes aspectos técnicos já foram mencionados no Capítulo 4 e serão descritos agora de acordo com a implementação realizada.

Iniciamos dando uma breve introdução sobre quais *frameworks* nosso algoritmo está baseado e levantamos alguns pontos que são relevantes no desempenho de qualquer algoritmo *branch-and-cut*. Apresentamos em seguida, a formulação inteira utilizada para calcular os limitantes superiores, descrevemos a estrutura *branch-and-bound* empregada, discutindo aspectos como: regras empregadas na seleção de um nodo, estratégias de *branching* e procedimento de *bounding*. Salientamos ainda, o emprego de heurísticas primais na tentativa de se melhorar o valor do limitante inferior, e o processo de fixação de variáveis a partir destes valores. Por último, relatamos diversas estratégias testadas para as rotinas de separação e os resultados computacionais obtidos.

7.1 Introdução

O algoritmo *branch-and-cut* implementado tem a sua estrutura baseada nos trabalhos de Padberg e Rinaldi [39] para o caixeiro viajente simétrico, e Jünger *et al.* [33] para problemas de otimização combinatória.

Ao contrário de Padberg e Rinaldi [39] que tiveram que desenvolver um um resolvedor de PLs, nós fazemos uso do pacote *CPLEX 3.0 Optimization* [11]. Este pacote oferece um conjunto de procedimentos disponíveis em uma biblioteca. Para que fosse possível o uso destes procedimentos se fez necessário o desenvolvimento de uma estrutura de dados que fosse compatível com aqueles. Foram desenvolvidas ainda estruturas de dados que representassem um grafo e uma árvore *branch-and-cut*.

Podemos mencionar quatro "gargalos", enumerados por Padberg e Rinaldi [39], que influenciam diretamente no desempenho de um algoritmo *branch-and-cut*. São eles:

1. o sucesso no processo de eliminação de nós na árvore *branch-and-cut* depende do valor da solução inicial encontrada por algum procedimento heurístico;
2. desenvolvimento de boas rotinas de separação;
3. muito do tempo computacional envolvido em algoritmo *branch-and-cut* se deve a resolução de um PL, pois a cada nova iteração possíveis novos planos-de-cortes são adicionados ao PL o que gradativamente aumenta o seu tamanho. Portanto, é conviniente estabelecer algumas estratégias que tornem eficiente o uso do resolvedor de PLs;
4. o comportamento do algoritmo *branch-and-cut* é sensível ao tipo de estratégia empregada na fase de *branching*.

Aliados a estes fatores podem-se enumerar alguns outros, como por exemplo: *tailing-off* e seleção de nós.

7.2 Formulação Inteira e Limitantes Superiores

Definimos dois tipos de limitantes superiores: um local e um global. O limitante superior local corresponde ao valor obtido com a resolução do conjunto de desigualdades - relaxação linear - representado em cada nó da árovre *branch-and-cut*. Por outro lado, o limitante superior global corresponde ao máximo dos limitantes superiores locais. Nesta seção, nos preocuparemos em descrever como obtemos o valor de um limitante superior local. Na seção 7.3, mostraremos a maneira de calcular o limitante superior global.

A escolha de uma formulação inteira que proporcionasse bons limitantes superiores locais foi uma das primeiras preocupações que tivemos na resolução do Problema da Clique Máxima com Peso nas Arestas de forma exata.

No Capítulo 5 foram apresentadas duas formulações inteiras para o PCMPA, ambas definidas no modelo estendido. A escolha desta ou daquela foi baseada em alguns critérios, previamente estipulados, que permitissem nos indicar qual formulação inteira seria a mais apropriada: se a de Faigle *et al.* [15] (PL2) ou se a de Park *et al.* [41] (PL3).

Os critérios empregados para a escolha foram o valor obtido no 1o. nó da árvore *branch-and-bound* e o tempo computacional envolvido. Com isso procuramos manter um compromisso entre a qualidade da solução obtida no 1o. nó através destas formulações e o tempo computacional empregado.

As Tabelas 7.1 e 7.2 apresentam um quadro comparativo entre as formulações. As quatro primeiras colunas caracterizam a instância, a quinta e a sexta colunas indicam o valor no 1o. nó e o tempo computacional obtidos na formulação PL2, e por último, a sétima e oitava colunas fazem referência aos mesmos critérios só que para a formulação PL3. As instâncias utilizadas como testes foram geradas como descrito no capítulo anterior e os tempos computacionais apresentadas, a partir dessa seção, estarão todos expressos em segundos e correspondem ao tempo de CPU.

n	b	k	w	PL2		PL3	
				1o. Nó	Tempo(seg)	1o. Nó	Tempo(seg)
40	20	1	2	138324.50	110.12	138324.50	109.18
40	20	3	2	93539.00	33.90	93511.50	124.00
44	22	1	2	167467.50	206.32	167467.50	240.85
44	22	2	2	132519.75	88.70	132519.00	174.12
48	24	1	2	204281.00	210.08	204281.00	247.95
48	24	4	2	117206.00	146.43	117206.00	472.80

Tabela 7.1: Comparação entre as formulação inteiras, PL2 e PL3, para instâncias com pesos positivos.

Apesar de ambas as formulações possuírem o mesmo número de variáveis e $O(n^2)$ restrições, a formulação de Faigle *et al.* consegue se sobressair em relação à formulação de Park *et al.*, pois aquela representa o melhor compromisso entre o tempo computacional e a qualidade do limitante superior local obtido.

n	b	k	w	PL2		PL3	
				1o. Nó	Tempo(seg)	1o. Nó	Tempo(seg)
40	20	1	2	130495.50	15.25	128389.25	68.00
40	20	3	2	61980.50	6.62	60309.50	47.42
44	22	1	2	158592.00	34.63	156527.75	213.62
44	22	2	2	101244.50	14.78	98878.00	201.15
48	24	1	2	198501.50	50.10	197028.75	192.88
48	24	4	2	68281.50	14.57	65443.63	129.60

Tabela 7.2: Comparação entre as formulação inteiras, PL2 e PL3, para instâncias com pesos positivos e negativos.

7.3 Estrutura *Branch-and-Bound*

A primeira parte implementada do algoritmo *branch-and-cut* foi o procedimento *branch-and-bound*. Como visto no Capítulo 4, os principais aspectos de qualquer procedimento *branch-and-bound* para um problema de otimização combinatória são: estratégias de *branching*, seleção de nós e procedimentos de *bounding*.

Em seguida, descreveremos cada um destes aspectos e o que nos levou a escolha de um ou outro fator para a nossa implementação.

7.3.1 Seleção de Nós

Em nossa árvore *branch-and-cut* predominam dois tipos de nós: ativos e inativos. Um nó é considerado estar *ativo* se o processo de exploração nele ainda não foi finalizado, ou seja, o nó não pode ser considerado como maduro. Por outro lado, um nó é dito ser *inativo* caso já tenha sido encerrado a sua exploração.

O processo de seleção é realizado sob os nós ativos. Caso não existam mais nós ativos a melhor solução viável conhecida até o momento será o ótimo inteiro. Do contrário, fazemos uso de uma estratégia de enumeração para selecionar um nó. Três são as estratégias bem conhecidas na literatura: busca em profundidade (*depth-first search*), busca em largura (*breadth-first search*) e busca do melhor limitante (*best-first search*). Em seguida mencionamos o motivo da escolha da estratégia do melhor limitante.

Ambas as estratégias, busca em profundidade e em largura, possuem uma desvantagem crucial em relação a estratégia do melhor limitante. Durante o processo de seleção podemos ter a busca direcionada a partes da árvore *branch-and-bound* com limitantes superiores locais ruins o que diretamente se refleteria no tempo computacional. Tal desvantagem foi comprovada por Jünger *et al.* [32] em experimentos para o caixeiro viajante

simétrico. Uma outra desvantagem é conseqüência da anterior: o processo de eliminação de nós seria prejudicado posto que temos valores considerados ruins para os limitantes superiores locais.

Devido a isto, a estratégia do melhor limitante foi a escolhida. Nesta estratégia, o nó a ser escolhido seria aquele considerado o mais promissor. Para o PCMPA isto corresponde a escolher o nó cujo valor do limitante superior local é o máximo dentre todos os nós ativos. Na busca são visitados todos os nós ativos e, durante este procedimento, elimina-se aqueles nós cujos limitantes superiores locais sejam inferiores ao valor da melhor solução viável disponível no momento.

7.3.2 Estratégias de *Branching*

Na avaliação de uma estratégia deve-se ter em mente que o objetivo é resolver o problema de otimização combinatória com o menor número possível de nós. Portanto, pode-se afirmar que uma "boa" estratégia de branching seria aquela que gerasse poucos sucessores em um nó da árvore.

Durante o processo de *branching* seleciona-se uma variável e dois novos nós são criados e adicionados a lista de nós ativos. No primeiro nó criado a variável escolhida receberá valor 1 enquanto que no segundo o seu valor será igual a 0. Antes de mencionarmos as estratégias por nós testadas, considere o lema abaixo.

Lema 7.1 *A solução (x, y) é uma solução inteira para o Problema da Clique Máxima com Peso nas Arestas, se somente se, x também for inteira.*

Prova:

Necessária. Trivial.

Suficiente. A prova será feita por contradição. Assuma que x é inteira, ou seja que $x_i = 0$ ou $x_i = 1$ para todo $i \in V_n$, e que $0 \leq y_{ij} \leq 1$. Considere que i e j sejam dois vértices quaisquer em V_n. Ao fazermos $x_i = x_j = 0$, ou $x_i = 1$ e $x_j = 0$ sem perdas de generalidades, obtemos $y_{ij} = 0$, pois $y_{ij} \leq x_i$ e $y_{ij} \leq x_j$. Por último, se considerarmos $x_i = x_j = 1$ temos $y_{ij} = 1$, pois $x_i + x_j - y_{ij} \leq 1$. Logo, para qualquer que seja a aresta $(i, j) \in E_n$ ela assumirá valores iguais a 0 ou 1 quando x for inteira.

□

Portanto, com base no Lema 7.1 acima o processo de *branching* pode ser realizado apenas sobre as variáveis definidas para os vértices com garantias de que sempre encontraremos uma solução inteira ao final do *branch-and-bound*. Apresentaremos em seguida as estratégias de *branching* testadas.

n	b	Estratégia 1			Estratégia 2		
		# Nós	# LPs	Tempo(seg)	# Nós	# LPs	Tempo(seg)
15	7	14	9	1.39	22	13	1.70
20	10	74	40	12.52	54	32	10.14
25	12	548	303	233.15	618	312	241.44
30	15	1284	685	1123.87	1326	849	1706.92
32	16	2208	1746	4136.45	3412	1782	4368.53
38	19	3990	3181	9338.21	5802	3033	8923.41

Tabela 7.3: Comparação entre as estratégias de *branching* para instâncias com pesos positivos.

n	b	Estratégia 1			Estratégia 2		
		# Nós	# LPs	Tempo(seg)	# Nós	# LPs	Tempo(seg)
15	7	18	14	1.76	30	18	3.04
20	10	74	56	15.41	94	69	17.00
25	12	736	410	268.97	526	372	297.80
30	15	1192	653	866.58	1466	752	1121.55
32	16	4712	3243	6492.84	5014	2832	5538.06
38	19	7276	6113	14436.89	9746	5725	13341.40

Tabela 7.4: Comparação entre as estratégias de *branching* para instâncias com pesos positivos e negativos.

Na primeira estratégia (Estratégia 1), a variável escolhida é aquela com valor mais próximo de 0.5. Na outra estratégia (Estratégia 2), escolhe-se a variável cujo valor é o mais próximo de 1.0.

As Tabelas 7.3 e 7.4 mostram a execução, até a otimalidade, do procedimento *branch-and-bound* com cada uma das estratégias acima. As duas primeiras colunas caracterizam a instância (geradas para $k = 1$ e $w = 2$). A terceira, quarta e quinta colunas indicam, respectivamente, o número de nós, PLs e o tempo computacional para a estratégia 1. Analogamente, a sexta, sétima e oitava colunas indicam estes mesmos resultados obtidos para a estratégia 2.

Fica claro que a melhor estratégia é a primeira, pois proporciona uma árvore com menos nós na maioria das instâncias testadas.

7.3.3 Procedimento de *Bounding*

O procedimento de *bouding* retorna um limitante superior global que corresponde ao maior valor dentre os limitantes superiores dos nós ativos.

Caso a árvore *branch-and-cut* tenha apenas um nó, o nó raiz, o valor do limitante superior global é dado pelo valor da relaxação linear corrente. Caso este valor seja igual ao limitante inferior fornecido por uma solução viável conhecida, o problema estará resolvido.

Por outro lado, se tivermos pelo menos dois nós ativos, o valor do limitante superior global corresponde ao maior valor da relaxação linear dentre todos os nós ativos.

7.4 Limitantes Inferiores

Como mencionado no capítulo anterior o cálculo do limitante inferior pode ser feito de duas formas. A primeira, que obtém o limitante inferior inicial, faz uso da metaheurística GRASP. A segunda, calcula o limitante inferior tendo por base informações fornecidas pela relaxação linear.

Na próxima seção nos detalharemos sobre esta segunda forma de calcular limitantes inferiores.

7.5 Heurísticas Primais

Até o final desta seção descreveremos as heurísticas primais implementadas para o PCMPA. Na seção 7.7, descreveremos o processo de fixação de variáveis a partir dos valores encontrados para os limitantes inferiores.

A primeira heurística primal implementada basea-se na escolha de b vértices de acordo com o valor das variáveis correspondentes na relaxação linear. A princípio são escolhidos

os b primeiros vértices cujo valor na relaxação seja maior ou igual a 0.5. Em seguida, calcula-se o peso da clique constituída por estes no limitante inferior.

Já a segunda heurística primal, seleciona os b primeiros vértices com maior valor na relaxação linear. Esta escolha é feita sem a necessidade de uma ordenação completa dos valores dos vértices. Para tanto fazemos mais uma vez uso do algoritmo Estatística de Ordem (por exemplo Cormen *et al.* [10]). Após a seleção dos vértices verifica-se se houve alguma melhoria no valor do limitante inferior.

n	b	Heurística primal 1			Heurística primal 2		
		# Nós	# LPs	Tempo(seg)	# Nós	# LPs	Tempo(seg)
15	7	12	9	1.48	10	9	1.47
20	10	74	40	12.98	74	40	11.55
25	12	548	303	276.25	544	303	246.39
30	15	1284	685	1123.29	1236	694	1073.35
32	16	2208	1746	3782.99	2208	1746	3991.39
38	19	3990	3181	10607.55	3990	3181	10071.86

Tabela 7.5: Comparação entre as heurísticas primais para instâncias com pesos positivos.

n	b	Heurística primal 1			Heurística primal 2		
		# Nós	# LPs	Tempo(seg)	# Nós	# LPs	Tempo(seg)
15	7	18	14	1.69	18	14	1.77
20	10	74	56	15.52	74	56	14.78
25	12	734	411	288.15	650	417	278.30
30	15	1192	653	879.02	1122	668	889.90
32	16	4712	3243	6056.67	4618	3280	6409.50
38	19	7276	6113	14281.99	7276	6113	14846.43

Tabela 7.6: Comparação entre as heurísticas primais para instâncias com pesos positivos e negativos.

As Tabelas 7.5 e 7.6 mostram o desempenho de cada heurística primal. As instâncias são executadas até a otimalidade, e pode-se verificar que a segunda heurística primal é melhor do que a primeira pelo fato de que ela consegue diminuir o número de nós da árvore, ou quando não pelo menos encontrar o mesmo valor.

7.6 *Tailing-off*

A parte de geração de planos-de-corte em um nó é finalizada se após *lnlp* PLs não tivermos obtido um ganho de pelo menos $p\%$ no valor da relaxação linear com o acréscimo dos planos-de-corte. O valor utilizado para *lnlp* é igual a 30 e o de $p\%$ de 10^{-1}.

Quando o fenômeno de *tailing-off* é detectado o processo de *branching* é chamado logo em seguida.

7.7 Fixação de Variáveis

Como já mencionado o processo de fixação de variáveis permite diminuir o tamanho da relaxação linear. Isto proporciona um ganho significativo no desempenho do algoritmo *branch-and-cut*.

Escolhemos o processo de fixar variáveis através dos custos reduzidos uma vez que os limitantes inferiores encontrados pela metaheurística *GRASP* e a heurística primal são bastante satisfatórios.

A rotina de fixação de variáveis por custo reduzido é chamada quando uma operação de *branching* está para ser realizada. Durante o processo computacional, o valor do limitante inferior aumenta, e portanto, em algum ponto na computação o critério de fixação poderá ser satisfeito.

7.8 Rotinas de Separação

O processo de geração de planos-de-corte é a parte mais importante de um algoritmo *branch-and-cut*. A escolha dos planos-de-corte a serem utilizados, a definição da seqüência em que os diferentes tipos de planos-de-corte são gerados, a forma de armazená-los e o método empregado na implementação das rotinas de separação são alguns dos mais diversos aspectos envolvidos nesta fase.

A princípio, faremos uma comparação entre os resultados obtidos com o emprego de algoritmos exatos ou heurísticas na implementação de rotinas de separação. Através destes resultados, definimos diversas seqüências de agruparmos os planos-de-corte.

7.8.1 Separação Heurística *versus* Separação Exata

Como visto anteriormente, várias são as famílias de facetas conhecidas na literatura e que podem ser empregadas como planos-de-corte ao PCMPA. Foram desenvolvidas algumas rotinas de separação para: desigualdades corte, clique e árvore. Realizamos ainda testes com todas as desigualdades encontradas neste trabalho.

As desigualdades que por ventura venham a ser geradas são armazenadas gradativamente em uma estrutura de dados, para logo em seguida serem adicionadas ao PL. Todas as desigualdades pertencentes ao PL atual são vistas globalmente por qualquer nó ativo da árvore *branch-and-cut*. Ou seja, não usamos uma política para distinguir entre aquelas desigualdades que estão ativas ou inativas no momento. Portanto, não dispomos de uma estrutura de dados auxiliar, no caso um *pool*, para a armazenagem das desigualdades tornaram-se inativas em alguma iteração do algoritmo.

Fica claro então, que se adotarmos esta política de considerar que todas as desigualdades estão ativas no PL, muito embora sabendo que isto não seja sempre verdade, caímos em um aspecto indesejável em nossa implementação: um tempo computacional elevado para resolver um PL.

Um dispositivo adicional utilizado para reduzir o tamanho do PL é identificar dentre aquelas desigualdades violadas as mais promissoras, ou seja adotar um critério de qualidade que limitaria o número de novas desigualdades a serem adicionadas. Uma medida razoável para garantir a qualidade de um corte seria baseada no valor obtido a partir da folga de uma desigualdade gerada. Quanto maior este valor, melhor será o plano-de-corte.

Não adotamos um critério específico para avaliar a qualidade de um plano-de-corte, apenas estabelecemos um número máximo de planos-de-corte que podem ser gerados por PL e uma seqüência na geração destes. Uma definição desta seqüência pode ser construída baseada na utilização de planos-de-corte de uma mesma família ou por planos-de-corte pertencentes a famílias diferentes. Porém, a definição desta seqüência e do número máximo de cortes por PL deve procurar obedecer ao compromisso entre resolver o PCMPA e o tempo computacional empregado na sua resolução.

Definidos quais os planos-de-corte que serão utilizados e a seqüência para gerá-los, resta definir o método empregado para a implementação das rotinas de separação. Dois são os métodos comumente utilizados: heurístico e exato. O primeiro, é empregado quando o problema de separação é NP-difícil, por exemplo no caso das desigualdades árvores. Já o segundo, é apropriado a problemas de separação que sejam polinomiais e que possuam complexidade baixa, por exemplo desigualdades corte-triangular e clique-triangular.

A seguir damos um exemplo de geração de planos-de-corte e uma forma de agrupá-los e de definir uma seqüência para a sua geração. A princípio estabelecemos que o número de planos-de-corte a serem adicionados era ilimitado. Fez-se uso das desigualdades cortes com $|S| = 1$ e $|T| = 2$ (desigualdades cortes-triangulares) e $|S| = 2$ e $|T| = 3$ (desigualdades cortes 2-3). Os gráficos representado nas Figuras 7.1 e 7.2 mostram o desempenho das heurísticas projetadas para estes planos-de-corte. As heurísticas propostas são ingênuas e gulosas podendo gerar várias vezes uma mesma desigualdade. A complexidade inerente a estas heurísticas é da $O(n^2)$ e $O(n^3)$ para as desigualdades cortes-triangulares e cortes 2-3, respectivamente.

Pode-se observar que a desigualdade corte-triangular gasta menos tempo do que a 2-3, sendo necessário menos planos-de-corte para resolver as instâncias. Tal fato se repete quando implementamos as rotinas de separação de forma exata. Com base nestes fatos estabelecemos a primeira estratégia:

Estratégia 1: *geram-se todos os possíveis planos-de-cortes do tipo corte-triangular e depois todos do tipo cortes 2-3.*

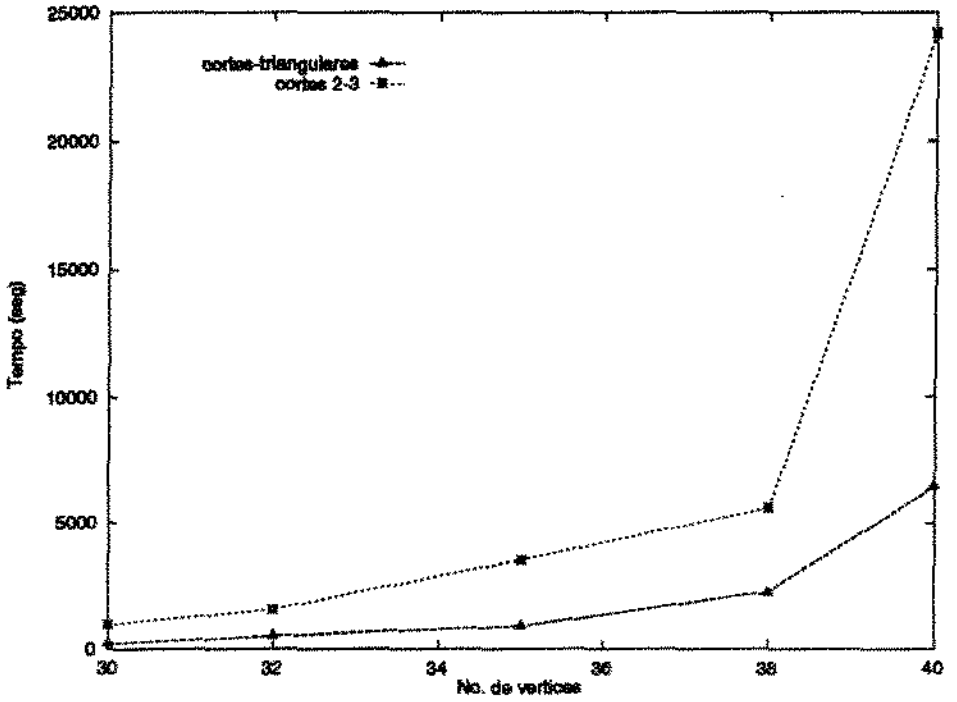


Figura 7.1: Desempenho dos métodos heurísticos projetados para os planos-de-corte corte-triangular e cortes 2-3 com relação ao tempo computacional.

A Tabela 7.7 mostra os resultados com o emprego desta estratégia para o método heurístico e a Tabela 7.8 para o método exato. É possível observar que através desta nova seqüência obtemos uma melhora significativa no tempo, no número de planos-de-corte gerados e no número de PLs, a exceção feita para a instância $n = 40$ e $b = 20$. Porém, como o número de planos-de-corte a serem adicionados é limitado e o método heurístico apresenta mais planos-de-corte do que o exato, pode-se supor que as heurísticas estejam gerando planos-de-corte repetidos por PL. Isto diretamente se refletiria no tempo computacional.

A segunda estratégia estabelecida procura diminuir o número de cortes por PL com o

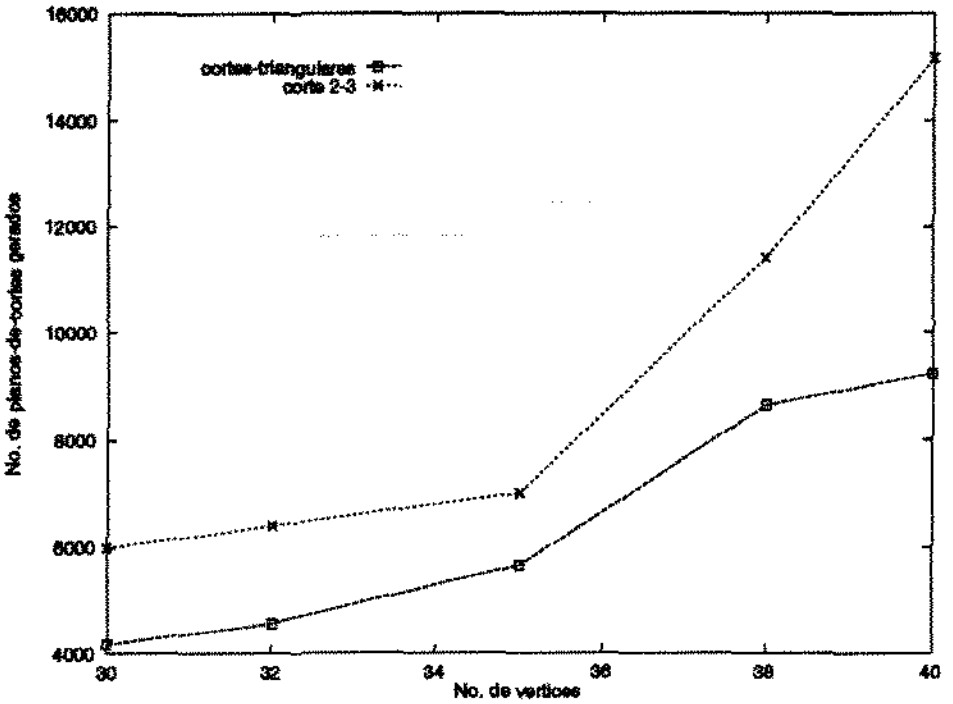


Figura 7.2: Desempenho dos métodos heurísticos projetados para os planos-de-corte corte-triangular e cortes 2-3 com relação ao número de planos-de-corte gerados.

				planos-de-corte				
n	b	# Nós	# PLs	# △	# 2-3	1o. Nó	Ótimo	Tempo(seg)
30	15	0	4	3642	858	64952	64952	266.64
32	16	0	4	4272	528	72504	72504	329.49
35	17	0	5	5604	1396	83482	83482	703.66
38	19	0	6	7396	2084	102259	102259	2063.86
40	20	0	8	9410	3416	109346	109346	5634.88

Tabela 7.7: Desempenho dos métodos heurísticos projetados para os planos-de-corte corte-triangular e cortes 2-3 segundo a estratégia 1.

				planos-de-corte				
n	b	# Nós	# PLs	# $\triangle$	# 2-3	1o. Nó	Ótimo	Tempo(seg)
30	15	0	2	1500	0	64952	64952	52.48
32	16	0	3	2296	904	72504	72504	304.48
35	17	0	3	3460	40	83482	83482	294.25
38	19	0	3	3734	66	102259	102259	378.08
40	20	0	9	4708	11292	109346	109346	10733.31

Tabela 7.8: Desempenho dos métodos exatos projetados para os planos-de-corte corte-triangular e cortes 2-3 segundo a estratégia 1.

intuito de que ocorram menos repetições quando trabalhamos com heurísticas:

Estratégia 2: *continua-se primeiro a chamar a rotina de separação para os cortes-triangulares e depois para os cortes 2-3, porém estabelece-se um número máximo de planos-de-corte a serem gerados por PL. Este número máximo, denotado por NUM_MAX, é expresso por* $\lceil 25 \times |V_n|/2 \rceil$.

Agora prevalece o método heurístico que, no geral, apresentou um número de PLs e um tempo computacional menor. Apesar desta vantagem da heurística, pode-se observar que os planos-de-corte encontrados pelo método exato são melhores uma vez que foi sempre possível atingir a otimalidade ainda no 1o. nó. As Tabelas 7.9 e 7.10 mostram estes fatos.

				planos-de-corte				
n	b	# Nós	# PLs	# $\triangle$	# 2-3	1o. Nó	Ótimo	Tempo(seg)
30	15	0	6	1875	1875	64952	64952	497.57
32	16	0	7	2400	2400	72504	72504	833.42
35	17	0	8	3066	3066	83482	83482	2066.51
38	19	0	10	4275	4275	102259	102259	3825.70
40	20	2	54	6610	3188	109434	109346	7321.67

Tabela 7.9: Desempenho dos métodos heurísticos projetados para os planos-de-corte corte-triangular e cortes 2-3 segundo a estratégia 2.

Sabendo-se então que o importante é gerar cortes bons e em um número não muito grande, adotamos a nova política para a geração de planos-de-corte fazendo uso apenas dos métodos exatos na implementação das rotinas de separação para os planos-de-corte

				planos-de-corte				
n	b	# Nós	# PLs	# $\triangle$	# 2-3	1o. Nó	Ótimo	Tempo(seg)
30	15	0	8	2564	2625	64952	64952	764.63
32	16	0	7	2400	2400	72504	72504	750.08
35	17	0	9	3498	3504	83482	83482	2081.30
38	19	0	10	4275	4275	102259	102259	2372.50
40	20	0	21	5084	10000	109346	109346	18246.12

Tabela 7.10: Desempenho dos métodos exatos projetados para os planos-de-corte corte-triangular e cortes 2-3 segundo a estratégia 2.

corte-triangular e cortes 2-3:

Estratégia 3: *primeiro chama-se a rotina de separação correspondente a desigualdade cortes-triangulares podendo-se gerar no máximo NUM_MAX. Caso o número de cortes-triangulares gerados seja inferior a NUM_MAX/2 chama-se então a rotina de separação para os cortes 2-3 podendo-se gerar até NUM_MAX planos-de-corte deste tipo.*

				planos-de-corte				
n	b	# Nós	# PLs	# $\triangle$	# 2-3	1o. Nó	Ótimo	Tempo(seg)
30	15	0	7	2250	0	64952	64952	225.24
32	16	0	8	2800	0	72504	72504	343.60
35	17	0	9	3385	0	83482	83482	644.89
38	19	0	10	4275	0	102259	102259	1008.54
40	20	0	50	5269	3900	109346	109346	8617.53

Tabela 7.11: Desempenho dos métodos exatos projetados para os planos-de-corte corte-triangular e cortes 2-3 segundo a estratégia 3.

Comparando-se que as Tabelas 7.9, 7.10 e 7.11 vê-se que a estratégia 3 ainda permite resolver as intâncias testadas no 1o. nó, a exemplo da estratégia 2. Além disso, observa-se que isto é atingido em um tempo computacional comparável e/ou mesmo inferior, aquele obtido com a estratégia 2 em ambos os experimentos. No entanto, a instância $n = 40$ e $b = 20$ continua com um tempo computacional alto, apesar de encontrar o ótimo inteiro ainda no 1o. nó.

Os resultados das últimas Tabelas, 7.9, 7.10 e 7.11, parecem indicar que a inclusão

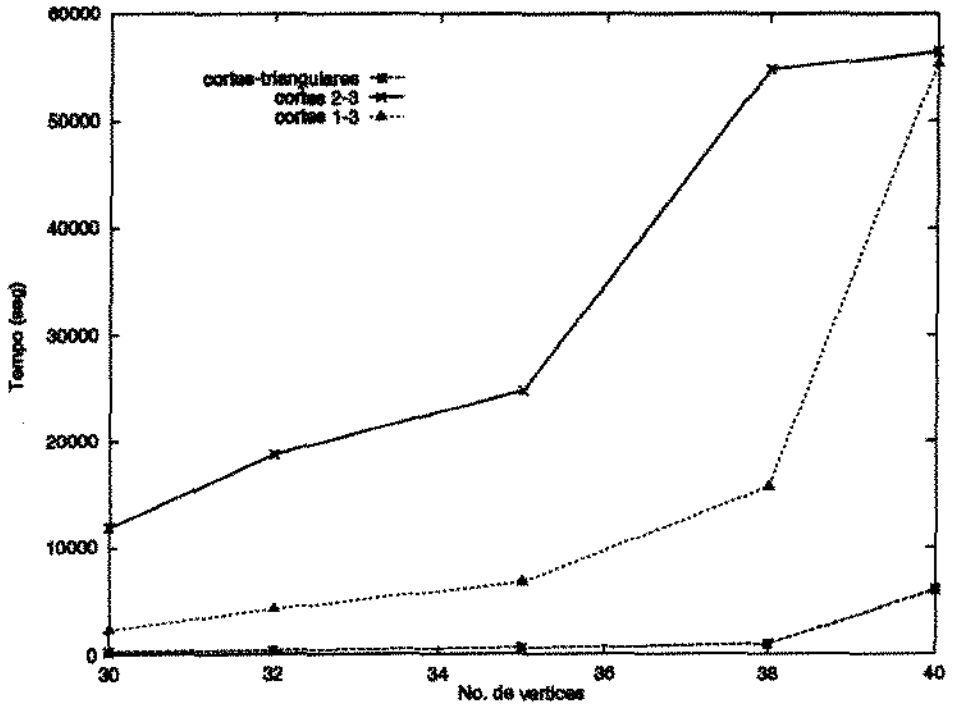


Figura 7.3: Desempenho dos métodos exatos projetados para os planos-de-corte corte-triangular, cortes 1-3 e cortes 2-3 com relação ao tempo computacional.

dos planos-de-corte 2-3 tornam os PLs demasiadamente pesados aumentando consideravelmente o tempo computacional. Optou-se então, por um novo tipo de plano-de-corte que apresentasse uma complexidade menor durante o seu processo de geração e uma densidade menor na matriz de desigualdades do PL. O escolhido foi o plano-de-corte cortes 1-3 que possui menos coeficientes não-nulos em comparação aos cortes 2-3 e uma rotina de separação que pode ser implementada através de um algoritmo exato cuja complexidade é da $O(n^4)$.

O gráfico da Figura 7.3 mostra um comparativo da execução dos planos-de-corte cortes 1-3 com os cortes-triangulares e cortes 2-3. Apesar de para a instância $n = 40$ e $b = 20$ não se ter atingido a otimalidade, pois se restringiu o tempo de computação para 54000 segundos, os cortes 1-3 produzidos se mostraram bem mais eficientes, nesta instância, do que os cortes 2-3. Com base nisto alteramos a estratégia 3 para:

Estratégia 4: *o número máximo de cortes a serem gerados pelas rotinas de separação das desigualdades cortes-triangulares e cortes 1-3 continua sendo igual a NUM_MAX , porém para os cortes 2-3 diminui para NUM_MAX/5. Alteramos também a seqüência de chamada das rotinas. Primeiro chama-se a rotina das desigualdades cortes-triangulares. Caso sejam gerados menos do que NUM_MAX/2 planos-de-corte então chama-se a rotina das desigualdades cortes 1-3. Caso, também, tenha-se gerado menos do que NUM_MAX/2*

planos-de-corte cortes 1-3 chamar-se-á a rotina de separação das desigualdades 2-3.

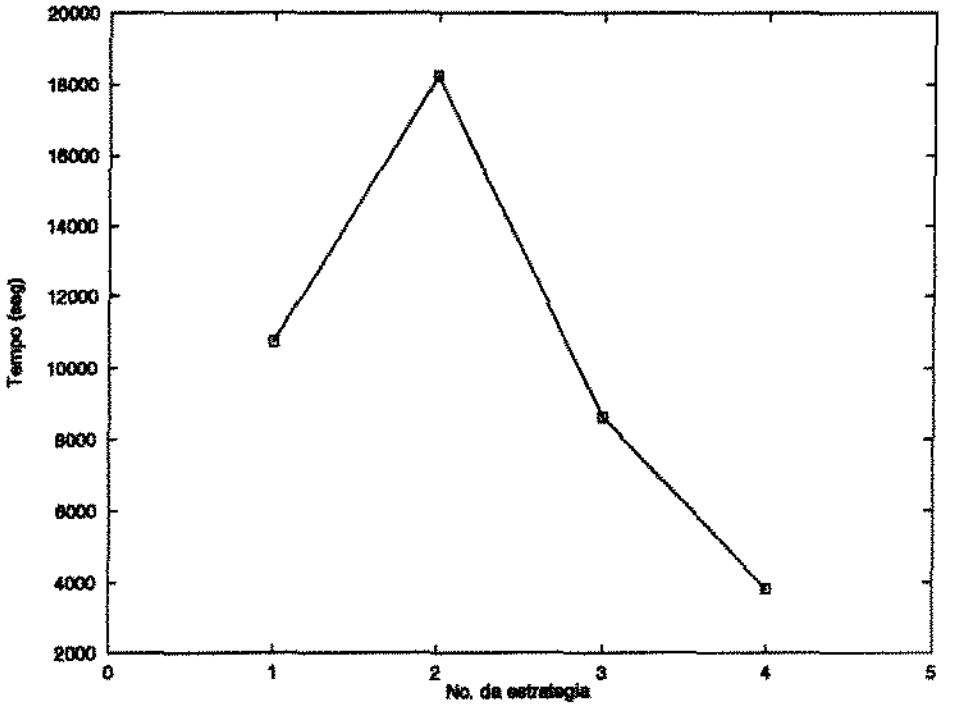


Figura 7.4: Desempenho de cada estratégia para a instância $n = 40$ e $b = 20$.

O gráfico da Figura 7.4 mostra um resumo das estratégias empregadas, em particular, para a instância $n = 40$ e $b = 20$. Cada ponto na curva representa o tempo computacional gasto na resolução desta instância segundo a estratégia adotada. Claramente, é possível observar que a estratégia 4 é a melhor.

Devido os resultados apresentados com o emprego da estratégia 4 optou-se por utilizar métodos exatos, quando possível na implementação das rotinas de separação, e os valores dos parâmetros testados para o número máximo de cortes.

7.8.2 Estratégias para Inclusão de Planos-de-Corte

Os resultados conhecidos na literatura sobre o PCMPA, Faigle e Dijkhuizen [14] e Park *et al.* [40], relatam instâncias resolvidas através de *branch-and-cut* ou algoritmos de planos-de-corte com no máximo 30 vértices. Testes realizados permitem afirmar que instâncias deste porte podem ser resolvidas em um tempo computacional razoável através do procedimento *branch-and-bound.*

Nesta seção, definiremos outras estratégias com o intuito de verificarmos o quão boas são as desigualdades a serem empregadas na resolução de instâncias de maior porte. Além

das desigualdades: corte-triangular ($\triangle$), cortes 1-3 (C1-3) e cortes 2-3 (C2-3); fez-se uso das desigualdades: árvore (ARV), clique-triangular ($C\triangle$), cortes 1-3 generalizado (C1-3g), cortes 1-4 generalizado (C1-4g), caminho clique-triangular ($cCl\triangle$), caminho corte-triangular ($cCr\triangle$) e [2:1]-flor-do-sul para $|S| = 1$ e $|T| = 4$ (FdS1-4).

As instâncias utilizadas nos testes são as mesmas descritas no capítulo anterior para ambos os casos: arestas com pesos positivos e com pesos positivos e negativos. Em seguida, descrevemos as novas estratégias.

Os nossos testes iniciam com a última estratégia da seção anterior: estratégia 4. A Tabela 7.12 apresenta os resultados obtidos com esta estratégia. Observe que os planos-de-corte continuam se mostrando bastante eficazes em ambos os casos, visto que o ótimo inteiro é atingido ainda no 1o. nó da árvore *branch-and-cut*.

Pesos positivos

					planos-de-corte					
n	b	k	# NÓs	#PLs	# $\triangle$	# C1-3	# C2-3	1o. Nó	Ótimo	Tempo(seg)
40	20	1	0	17	5289	3000	0	1099346	109346	3719.52
40	20	3	0	18	5075	2407	300	68759	68759	4063.29
40	20	4	0	27	5053	2523	1200	60782	60782	5419.51
44	22	1	0	15	6399	1650	0	136525	136525	3734.54
45	22	5	0	23	8090	3628	226	69563	69563	7873.28

Pesos positivos e negativos

					planos-de-corte					
n	b	k	# Nós	#PLs	# $\triangle$	# C1-3	# C2-3	1o. Nó	Ótimo	Tempo(seg)
40	20	1	0	55	4044	4197	3800	70348	70348	23805.20
40	20	5	0	10	1596	2127	0	27967	27967	522.46
42	21	5	0	8	2738	678	105	35460	35460	295.94
44	22	1	0	63	5604	6471	4510	90620	90620	47799.73
45	22	1	0	29	5987	6351	791	102295	102295	23572.92

Tabela 7.12: Resultados obtidos com o emprego dos planos-de-corte pertencentes a estratégia 4.

Nota-se ainda que para os pesos positivos são geradas menos desigualdades cortes 1-3 do que as desigualdades corte-triangular, o que comprova que estas últimas são bastante úteis quando lidamos com pesos positivos. Por outro lado, os cortes 1-3 se mostram mais importantes com pesos positivos e negativos visto que um número maior destes são encontrados se comparados aos corte-triangular.

A próxima estratégia faz uso das desigualdades: cortes-triangulares, cliques-triangulares e árvores. Para as duas primeiras desigualdades as rotinas de separação foram implemen-

tadas de forma exata. Por outro lado, para as desigualdades árvore utilizou-se uma heurística ingênua cuja complexidade é $O(n^3)$. A estratégia foi definida como:

estratégia 5: *o número máximo de planos-de-corte a serem gerados pelas rotinas de separação das desigualdades cortes-triangulares e cliques-triangulares é igual a NUM_MAX, e para os planos-de-corte árvore corresponderá a NUM_MAX/5. A seqüência de chamada das rotinas é: primeiro chama-se a rotina das desigualdades cortes-triangulares. Caso sejam gerados menos do que NUM_MAX/2 planos-de-corte então chama-se a rotina das desigualdades cliques-triangulares. Caso, também, tenha-se gerado menos do que NUM_MAX/2 planos-de-corte clique-triangulares chamar-se-á a rotina de separação das desigualdades árvore.*

A Tabela 7.13 mostra os resultados obtidos com a estratégia 5. Observar-se que houve uma redução no tempo computacional e no número total de planos-de-corte gerados tanto para os pesos positivos como para os positivos e negativos. No entanto, os planos-de-corte empregados nesta estratégia não atingiram o ótimo inteiro no 1o. nó da árvore *branch-and-cut*. Para uma das instâncias chegou-se a criar até 78 nós.

Pesos positivos

					planos-de-corte				
n	b	k	# Nós	#PLs	# Δ	# CΔ	# ARV	1o. Nó	Tempo(seg)
40	20	1	6	20	5216	802	0	110437.83	2851.44
40	20	3	8	27	4862	293	1	69901.67	2929.24
40	20	4	12	40	5033	360	3	61867.32	3917.87
44	22	1	0	14	6328	634	1	136525	2439.72
45	22	5	6	28	8047	356	0	70166.45	6702.60

Pesos positivos e negativos

					planos-de-corte				
n	b	k	# Nós	#PLs	# Δ	# CΔ	# ARV	1o. Nó	Tempo(seg)
40	20	1	64	107	4082	1407	74	76625.67	22335.44
40	20	5	0	7	1458	524	90	27967	162.77
42	21	5	0	8	2729	534	0	35460	285.00
44	22	1	78	122	5644	1554	74	97368.92	37371.73
45	22	1	10	30	5974	1254	44	105735.75	8717.98

Tabela 7.13: Resultados obtidos com o emprego dos planos-de-corte pertencentes a estratégia 5.

Na definição da estratégia 6 resolvemos utilizar desigualdades pertencentes a uma

mesma família. As desigualdades escolhidas foram: cortes-triangulares e cortes 1-3 e 1-4 generalizados. Para os três tipos de desigualdades implementou-se rotinas de separação de forma exata.

estratégia 6: *o mesmo número máximo de planos-de-corte adotado para cada uma das desigualdades da estratégia 5, ou seja NUM_MAX para cortes-triangulares e cortes 1-3 generalizado, e NUM_MAX/5 para os planos-de-corte 1-4 generalizado. A seqüência de chamada das rotinas de separação é cortes-triangulares, cortes 1-3 e 1-4 generalizados seguindo os mesmos critérios da estratégia 5 para a chamada.*

Comparando-se as Tabelas 7.13 e 7.14 pode-se dizer que as desigualdades da estratégia 6 descrevem melhor o PCE_b do que as desigualdades na estratégia 5. Isto pode ser constatado observando-se que o número de nós criados na estratégia 6 é sempre menor do que o número de nós na estratégia 5 ou comparando-se o valor da solução obtida no 1o. nó.

A exemplo do ocorrido na estratégia 4, os cortes-triangulares funcionam bem para ambas as atribuições de pesos. Já os cortes 1-3 generalizados são mais eficientes quando os pesos são positivos e negativos.

Com os resultados apresentados até o momento as desigualdades na estratégia 4 se mostram bastante adequadas para o compromisso entre o tempo computacional e o valor obtido no 1o. nó.

A estratégia definida para as desigualdades caminho corte-triangular e caminho clique-triangular seguiu os modelos das anteriores. As rotinas de separação implementadas para estas desigualdades basearam-se em heurísticas, no entanto não foram obtidos resultados satisfatórios com o emprego desta estratégia. A última estratégia testada utilizou as desigualdades: cortes-triangulares e [2:1]-flor-do-sul.

estratégia 7: *permiti-se gerar NUM_MAX planos-de-corte para ambas as desiguldades. A chamada das rotinas de separação é primeiro a rotina de separação para os cortes-triangulares, e em seguida a rotina dos planos-de-corte [2:1]-flor-do-sul caso tenham sido gerados menos do que NUM_MAX/2 cortes-triangulares.*

A Tabela 7.15 relata os resultados obtidos com estes planos-de-corte. Voltou-se a obter um ganho na qualidade da solução gerada nos PLs com o acréscimo destes planos-de-corte, pois agora todas as instâncias foram resolvidas no 1o. nó da árvore *branch-and-cut*. Porém, deve-se chamar a atenção para um aumento no tempo computacional gasto na resolução das instâncias. Provavelmente isto se deve ao número total de planos-de-corte gerados que é maior do que em algumas outras estratégias tornando mais lenta a resolução

Pesos positivos

					planos-de-corte				
n	b	k	# Nós	#PLs	# $\triangle$	# C1-3g	# C1-4g	1o. Nó	Tempo(seg)
40	20	1	0	17	5267	3000	0	109346	3777.23
40	20	3	0	16	4922	2451	0	68759	2881.20
40	20	4	2	22	5060	2597	155	60786.21	3154.60
44	22	1	0	15	6360	1650	0	136525	3942.37
45	22	5	2	27	8141	3687	164	69578.13	10775.66

Pesos positivos e negativos

					planos-de-corte				
n	b	k	# Nós	#PLs	# $\triangle$	# C1-3g	# C1-4g	1o. Nó	Tempo(seg)
40	20	1	8	40	4046	4639	433	73004.21	12832.67
40	20	5	0	10	1588	2127	200	27967	489.44
42	21	5	0	8	2739	678	105	35460	287.34
44	22	1	8	44	5543	5763	497	93037.97	35065.61
45	22	1	2	25	5948	4590	83	102994.05	12784.62

Tabela 7.14: Resultados obtidos com o emprego dos planos-de-corte pertencentes a estratégia 6.

do PLs.

Os resultados indicam que a estratégia 5 é a estratégia adequada se desejamos resolver o PCMPA em um tempo computacional razoável. Já os resultados das estratégias 4, 6 e 7 indicam que as desigualdades usadas nestas estratégias descrevem melhor o politopo na região onde se encontra a solução ótima, fornecendo melhores limites superiores, sem contudo provocar um aumento no tempo computacional em relação a estratégia 5.

Assim se fossemos estabelecer uma ordem de chamada que fosse flexível com o tipo de pesos que se estivesse trabalhando o ideal para os pesos positivos seria: cortes-triangulares, seguidos dos cortes C1-3, C1-3g e FdS1-4 em um mesmo nível, e C2-3 e C1-4g num nível abaixo. Já para os pesos positivos e negativos seria cortes-triangulares seguidos de FdS1-4, C1-3g e C1-3 no nível seguinte e depois C1-4g e C2-3. No entanto, resolveu-se utilizar a estratégia 4 para um grupo maior de instâncias. A escolha desta estratégia se deve ao fato dela apresentar um tempo computacional comparável aos das estratégias 6 e 7, e atingir o ótimo inteiro ainda no 1o. nó.

Pesos positivos

			planos-de-corte					
n	b	k	# Nós	#PLs	# $\triangle$	# FdS1-4	1o. Nó	Tempo(seg)
40	20	1	0	19	5251	4000	109346	5616.77
40	20	3	0	18	4849	3500	68759	3336.02
40	20	4	0	20	5027	4500	60782	6483.45
44	22	1	0	14	6313	1100	136525	3826.79
45	22	5	0	22	8051	3941	69563	9276.42

Pesos positivos e negativos

			planos-de-corte					
n	b	k	# Nós	#PLs	# $\triangle$	# FdS1-4	1o. Nó	Tempo(seg)
40	20	1	0	32	4034	11500	70348	23827.81
40	20	5	0	7	1546	1500	27967	214.15
42	21	5	0	8	2726	1050	35460	614.59
44	22	1	0	36	5537	13750	90620	37290.87
45	22	1	0	27	5962	9008	102295	23465.26

Tabela 7.15: Resultados obtidos com o emprego dos planos-de-corte pertencentes a estratégia 7.

7.9 Resultados Computacionais

As instâncias utilizadas são as mesmas descritas no capítulo anterior. Apresentamos os resultados obtidos com os procedimentos *branch-and-bound* e o algoritmo *branch-and-cut* implementado.

Para a execução do procedimento *branch-and-bound* utilizou-se o pacote CPLEX 3.0. Estabeleceu-se um critério de parada para os procedimentos *branch-and-bound*. O critério escolhido foi o número máximo de nós na árvore *branch-and-bound*. Estipulou-se que este número seria igual a 20000.

7.9.1 Pesos Positivos

Os resultados obtidos com os procedimentos *branch-and-bound*, apresentados na Tabela 7.16, mostram que em 10 das instâncias testadas é possível atingir o ótimo inteiro antes do limite máximo do número de nós da árvore *branch-and-bound*. Porém, na grande maioria das instâncias o limite máximo é alcançado antes do ótimo inteiro. Isto implicaria em mais tempo computacional na resolução destas instâncias caso se desejasse resolver o problema.

Já com o emprego do algoritmo *branch-and-cut* foi possível atingir o ótimo inteiro em

todas as instâncias ainda no 1o. nó. Vale ressaltar que o tempo computacional obtido foi bem abaixo do tempo empregado nos procedimentos *branch-and-bound* inclusive para a instância $n = 48$ e $k = 2$. A Tabela 7.17 apresenta os resultados do algoritmo *branch-and-cut*.

7.9.2 Pesos Positivos e Negativos

O número de instâncias resolvidas pelo procedimento *branch-and-bound* que atingíram a otimalidade aumentou para 13, porém a grande maioria das instâncias atinge o limite do número de nós antes da otimalidade. A Tabela 7.18 apresenta os resultados.

O algoritmo *branch-and-cut* voltou a superar os resultados obtidos com os procediemntos *branch-and-bound*. A exceção de quatro instâncias, em todas as outras foi possível atingir o ótimo inteiro ainda no 1o. nó. Mesmo não atingindo o ótimo inteiro no 1o. nó naquelas instâncias foi necessário apenas a criação de dois nós na árvore *branch-and-cut* com valores bastante expressivos quando comparados ao ótimo inteiro. A Tabela 7.19 mostra os resultados do algoritmo *branch-and-cut*.

É possível supor que diminuindo o número de cortes por PL e com isso permitindo mais *branchings* o tempo computacional empregado na estratégia 4 possa diminuir, ou ainda fazendo uso de uma combinação das desigualdades nas estratégias 4, 6 e 7, de acordo com o tipo dos pesos, possa-se alcançar o mesmo objetivo.

Grupo	k	w	# Nós	1o. Nó	Solução obtida	Ótimo	Tempo(seg)
Grafo 1 ($n = 40$)	1	2	20000(*)	138324.50	108945	109346	61057.12
	2	2	2680	103779.00	82451	82451	9187.35
	3	2	13631	93539	68759	68759	31695.57
	4	2	9473	82087.50	60782	60782	21606.12
	5	2	1064	72624.50	60513	60513	2932.97
Grafo 2 ($n = 42$)	1	2	20000(*)	152294.50	119268	120299	107819.05
	2	2	20000(*)	118639	86778	87810	107704.03
	3	2	20000(*)	102099.50	76554	76554	74483.65
	4	2	1949	89265.50	69482	69482	6149.32
	5	2	1920	85260.50	67383	67383	9823.88
Grafo 3 ($n = 44$)	1	2	20000(*)	167467.50	136525	136525	119831.10
	2	2	20000(*)	132519.75	93507	98186	98044.45
	3	2	11420	111656.75	84675	84675	44005.77
	4	2	16759	100877	75274	75274	69759.23
	5	2	2913	89575.50	69540	69540	11782.10
Grafo 4 ($n = 45$)	1	2	20000(*)	172833.42	138693	138694	122279.40
	2	2	20000(*)	134048.44	94201	98321	101369.13
	3	2	20000(*)	111690.75	82599	82743	84360.07
	4	2	20000(*)	102653.44	77500	77500	77999.10
	5	2	20000(*)	95011.64	69095	69563	92026.77
Grafo 5 ($n = 46$)	1	2	20000(*)	182588.75	133766	142985	131405.83
	2	2	20000(*)	146127	105876	108243	120581.58
	3	2	20000(*)	125576.25	93915	94859	108268.07
	4	2	20000(*)	109317	77348	78747	113628.40
	5	2	20000(*)	99558.25	70775	72431	96343.12
Grafo 6 ($n = 48$)	1	2	20000(*)	204281	151902	163397	165514.98
	2	2	20000(*)	159464.50	107034	115471	138695.73
	3	2	20000(*)	131673.50	94741	96666	154289.60
	4	2	18484	117206	88728	88728	97867.02
	5	2	10404	106879.50	82117	82117	57765.78

(*): número máximo de nós na árvore *branch-and-bound* atingindo.

Tabela 7.16: Resultados obtidos com o emprego do procedimento *branch-and-bound* em instâncias com pesos positivos.

Grupo	k	w	# Nós	#PLs	planos-de-corte # Δ	planos-de-corte # C1-3	planos-de-corte # C2-3	1o. Nó	Tempo(seg)
	1	2	0	17	5289	3000	0	109346	3719.52
	2	2	0	9	4000	0	0	82451	821.52
Grafo 1	3	2	0	18	5075	2407	300	68759	4063.29
($n = 40$)	4	2	0	27	5053	2523	1200	60782	5419.51
	5	2	0	8	3500	0	0	60513	817.41
	1	2	0	32	6149	4484	1260	120299	13679.36
	2	2	0	37	5419	4971	1785	87810	15919.71
Grafo 2	3	2	0	31	5470	2903	1575	76554	11298.25
($n = 42$)	4	2	0	11	5021	0	0	69482	1666.13
	5	2	0	9	4200	0	0	76383	823.63
	1	2	0	15	6399	1650	0	136525	3734.54
	2	2	0	31	6346	4628	1210	98186	15532.74
Grafo 3	3	2	0	14	6814	550	0	84675	2839.20
($n = 44$)	4	2	0	16	6591	1650	0	75274	3584.06
	5	2	0	12	5703	550	0	69540	2389.17
	1	2	0	60	7597	6065	4068	138694	39889.15
	2	2	0	42	7543	6086	2034	98321	30618.41
Grafo 4	3	2	0	43	6814	4103	2712	82743	25514.10
($n = 45$)	4	2	0	20	7367	3191	0	77500	6401.01
	5	2	0	23	8090	3628	226	69563	7873.28
	1	2	0	57	8645	7324	3450	142985	41205.22
	2	2	0	37	8712	5169	1495	108243	20712.69
Grafo 5	3	2	0	16	7575	1150	0	94859	2773.16
($n = 46$)	4	2	0	26	8750	3890	345	78747	13901.90
	5	2	0	29	8581	3708	805	72431	14504.89
	1	2	0	30	9665	7529	0	163397	25364.59
	2	2	0	105	9169	8783	9000	115471	103345.14
Grafo 6	3	2	0	64	8606	6435	4680	96666	51771.51
($n = 48$)	4	2	0	17	8492	1200	0	88728	3920.53
	5	2	0	16	9000	0	0	82117	3487.15

Tabela 7.17: Resultados obtidos com o emprego do algoritmo *branch-and-cut* em instâncias com pesos positivos.

Grupo	k	w	# Nós	1o. Nó	Solução obtida	Ótimo	Tempo(seg)
Grafo 1 ($n = 40$)	1	2	20000(*)	130495.50	63488	70348	41111.52
	2	2	3321	75922.50	45404	45404	4045.58
	3	2	6849	61980.50	34091	34091	5205.90
	4	2	2379	49639.50	27758	27758	1939.38
	5	2	500	44493	27967	27967	439.58
Grafo 2 ($n = 42$)	1	2	20000(*)	144528	70704	81633	44681.47
	2	2	20000(*)	87871	46828	46828	28045.73
	3	2	4256	65501.50	36689	36689	4801.35
	4	2	563	55223	35987	35987	762.02
	5	2	455	52775	35460	35460	585.23
Grafo 3 ($n = 44$)	1	2	20000(*)	158592	52442	90620	67374.03
	2	2	20000(*)	101244.50	53737	56960	35709.13
	3	2	6958	72423	40697	40697	9778.32
	4	2	7138	60668.50	32601	32601	6203.73
	5	2	5282	52705	29407	29407	5303.72
Grafo 4 ($n = 45$)	1	2	20000(*)	169461.35	90591	102295	86194.68
	2	2	20000(*)	103330.08	36095	55103	38568.02
	3	2	3318	73499	43914	43914	6678.07
	4	2	20000(*)	103330.09	36095	33990	39729.90
	5	2	11878	58428.50	30974	30974	14703.38
Grafo 5 ($n = 46$)	1	2	20000(*)	174863.25	88308	99550	69151.27
	2	2	20000(*)	110057.50	44202	58361	40722.13
	3	2	20000(*)	83123.50	43507	43915	30276.18
	4	2	20000(*)	64688.50	32951	32968	22002.88
	5	2	6505	57443	31000	31000	7225.50
Grafo 6 ($n = 48$)	1	2	20000(*)	198501.50	61279	113478	120495.85
	2	2	20000(*)	121501.50	41421	61768	54879.77
	3	2	20000(*)	83855.50	45941	45941	30545.90
	4	2	18228	68281.50	36903	36903	9174.18
	5	2	10965	61257.50	31351	31351	14420.33

(*): número máximo de nós na árvore *branch-and-bound* atingindo.

Tabela 7.18: Resultados obtidos com o emprego do procedimento *branch-and-bound* em instâncias com pesos positivos e negativos.

Grupo	k	w	# Nós	#PLs	planos-de-corte # $\triangle$	# C1-3	# C2-3	1o. Nó	Tempo (seg)
Grafo 1 ($n = 40$)	1	2	0	55	4044	4197	3800	70348	23805.20
	2	2	0	13	2233	4000	0	45404	2316.46
	3	2	0	11	2280	3000	0	34091	922.72
	4	2	2	34	2345	5212	1281	27772.52	4767.64
	5	2	0	10	1596	2127	0	27967	522.46
Grafo 2 ($n = 42$)	1	2	0	61	4799	5054	4410	81633	38285.58
	2	2	0	28	3540	4683	1260	46828	5873.03
	3	2	0	11	2664	2625	0	36689	1067.87
	4	2	0	5	1586	525	0	35987	54.99
	5	2	0	8	2739	678	105	35460	295.94
Grafo 3 ($n = 44$)	1	2	0	63	5604	6471	4510	90620	47799.73
	2	2	0	13	3439	3300	0	56960	4360.94
	3	2	0	12	3250	2750	0	40967	1255.87
	4	2	2	44	2415	5893	2698	32711.25	13740.63
	5	2	0	13	2507	4400	0	29407	1228.17
Grafo 4 ($n = 45$)	1	2	0	29	5987	6351	791	102295	23572.92
	2	2	0	30	4014	4805	1582	55103	9190.72
	3	2	0	8	2937	1100	0	43914	582.80
	4	2	0	27	2727	6374	1243	33990	6089.39
	5	2	0	32	3364	6564	1695	30974	10820.77
Grafo 5 ($n = 46$)	1	2	0	40	6093	7801	1840	99550	36453.30
	2	2	0	21	3873	4683	575	58361	6003.70
	3	2	0	27	3303	6588	1265	43915	9112.73
	4	2	2	52	3433	6851	3438	33054.06	28122.14
	5	2	0	13	3413	3450	0	31000	1400.60
Grafo 6 ($n = 48$)	1	2	0	99	7527	9368	8520	113478	124615.38
	2	2	0	62	5170	6197	5280	61768	45361.93
	3	2	0	17	4107	6000	0	45941	5214.58
	4	2	0	11	3160	2444	120	36903	1454.62
	5	2	2	33	3665	8387	1440	31404.64	11199.91

Tabela 7.19: Resultados obtidos com o emprego do algoritmo *branch-and-cut* em instâncias com pesos positivos e negativos.

Capítulo 8

Conclusões e Considerações Finais

Os estudos realizados por Faigle e Dijkhuizen [14] para o PCMPA no modelo natural não foram suficientes para se obter uma boa caracterização parcial do poliedro associado ao problema. Isto ficou evidente no esforço empregado na implementação de um algoritmo de planos-de-corte que obteve valores poucos expressivos para os limitantes, e só conseguiu resolver instâncias de pequeno porte do problema.

Os autores tentaram explicar este "fracasso" na abordagem empregada na resolução do PCMPA com duas suposições. A primeira foi de que o emprego de algoritmos de planos-de-corte depende muito do tipo de problema que se deseja resolver, e neste sentido o PCMPA podia ser considerado "intratável" através deste tipo de algoritmo, mesmo para instâncias de pequeno porte. A segunda explicação apontava para um estudo poliédrico mais aprofundado do PCMPA, para que fossem encontradas outras famílias de desigualdades que ao serem adicionadas à formulação, pudessem melhorar o desempenho do algoritmo.

Com o objetivo de melhorar os resultados obtidos em Faigle e Dijkhuizen [14], Faigle *et al.* [15] resolveram estudar o poliedro associado ao PCMPA usando um modelo estendido. Para isto os autores definiram formulações inteiras para o PCMPA neste modelo. A exemplo de Park *et al.* [40], os resultados obtidos com este novo modelo se mostraram bem mais satisfatórios.

Neste trabalho nós aprofundamos o estudo da estrutura facial do poliedro associado ao PCMPA tanto no modelo natural quanto no modelo estendido. O objetivo final deste estudo era a obtenção de planos-de-corte que permitissem resolver computacionalmente instâncias de maior porte do que aquelas encontradas na literatura. Nas seções seguintes nós enumeramos as principais contribuições deste trabalho e fazemos sugestões para direções futuras de pesquisa.

8.1 Contribuições

Nosso estudo da caracterização poliédrica do PCMPA foi iniciado ainda no modelo natural.

Foi possível encontrarmos, através de soluções fracionárias, desigualdades pertencentes a família 2-p-partição; e através de *liftings* realizados em desigualdades do modelo natural desigualdades pertencentes à família desigualdade corte.

Porém, todas estas desigualdades encontradas, de forma independente, já tinham sido propostas por Park *et al.* [40]. No entanto, dando continuidade ao estudo poliédrico do PCMPA, e mais uma vez através de soluções fracionárias e *liftings*, encontramos novas desigualdades. Tais desigualdades foram:

- desigualdades válidas
  - desigualdade flor-do-sul;
  - desigualdade árvore-clique;
  - desigualdade árvore-corte.
- desigualdades definidoras de facetas
  - desigualdade [s:t]-corte;
  - desigualdade flor-do-sul com $\alpha = 2$ e $\beta = 1$;
  - desigualdade flor-do-sul com $\alpha = 3$ e $\beta = 2$;
  - desigualdade caminho-clique;
  - desigualdade caminho-corte.

As maiores instâncias do PCMPA resolvidas de forma exata na literatura são para grafos completos com até 30 vértices. Para instâncias deste porte, os nossos resultados indicam que um simples procedimento de *branch-and-bound* já pode ser usado para resolver o PCMPA em tempo aceitável. No entanto, para instâncias com mais de 40 vértices o uso de um algoritmo *branch-and-cut* começa a ser realmente vantajoso.

Fazendo uso de um algoritmo *branch-and-cut*, implementado neste trabalho, foi possível verificar a qualidade das novas desigualdades quando resolvemos, de forma exata, instâncias para grafos completos com até 48 vértices.

Isto comprova a qualidade das desigualdades encontradas no modelo estendido uma vez que elas caracterizam melhor o politopo associado ao PCMPA, ou pelo menos a região onde está o ótimo inteiro.

8.2 Direções Futuras

Gostaríamos de deixar algumas direções futuras de pesquisa para a implementação do algoritmo *branch-and-cut* e para o estudo poliédrico do PCMPA. Do ponto de vista computacional poder-se-ia pensar no uso de um *pool* de desigualdades que diretamente apresenta duas vantagens com o seu emprego. A primeira que tornaria o PL mais enxuto, pois seriam retiradas do PL aquelas desigualdades que não estivessem ativas no momento. A segunda corresponderia a uma forma mais rápida de separação, pois as desigualdades inativas que são violadas pela solução fracionária atual podem ser recuperadas a partir do *pool* e reintegradas ao PL.

Não resta dúvidas de que a caracterização poliédrica do PCMPA é um dos fatores prepoderantes ao bom funcionamento de um algoritmo *branch-and-cut*. Alguns pontos que podem ser seguidos neste aspecto são:

- generalização das desigualdades árvore-clique e árvore-corte: encontrar condições necessárias para que ambas as desigualdades sejam definidoras de facetas para qualquer árvore, e não apenas no caso de caminhos;
- generalização da desigualdade flor-do-sul: mais uma vez procurar condições necessárias para que a desigualdade flor-do-sul defina facetas para quaisquer valores de α e β.

Além disso, o processamento paralelo desponta como uma das tendências em programação inteira (veja Johnson *et al.* [29] e Jünger e Störmer [34]) e seria interessante verificar o desempenho do algoritmo *branch-and-cut* em arquiteturas paralelas.

A última direção futura é proposta tanto na teoria quanto na prática. Ao invés de usar a técnica *branch-and-cut* sugere-se utilizar a técnica geração de colunas conjuntamente com planos-de-corte. Isto se deve a dois pontos: o primeiro porque no trabalhos de Johnson *et al.* [30] os autores afirmam que é possível resolver o subproblema de otimização do *min-cut clustering* com geração de colunas; e o segundo aspecto por causa da qualidade dos nossos planos-de-corte. Um trabalho semelhante, fazendo uso destas duas técnicas, foi desenvolvido por Akker *et al.* [50] para um problema de *scheduling* de uma única máquina.

Bibliografia

[1] E. Balas and W. Pulleyblank. The Perfectly matchable subgraph polytope of a bipartite graph. *Networks*, 13(4):495 a 516, 1983.

[2] Eduardo Uchoa Barboza. Problemas de classificação com restrições de conexidade flexibilizadas. Master's thesis, Universidade Estadual de Campinas, Instituto de Computação, Campinas, São Paulo, Março 1997.

[3] Béla Bollobás. *Graph Theory: an Introductory Course*. Springer-Verlag, 1979. Series Graduate Texts in Mathematics.

[4] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. Macmillan, London, 1976.

[5] E. Boros and P. L. Hammer. Cut-polytopes, boolean quadric polytopes and non negative quadratic pseudo-boolean functions. *Mathematics of Operations Research*, 18(1):245 a 253, February 1993.

[6] Sebastián Ceria. *Algoritmos para Programación Entera*. Tercera Escuela Latinoamericana de Verano de Investigación Operativa, 8 a 13 de Janeiro 1996.

[7] S. Chopra and M.R. Rao. The Partition problem. *Mathematical Programming*, 59:87 a 115, 1993.

[8] S. Chopra and M.R. Rao. The Partition problem I: formulations, dimensions and basic facets. Technical Report 89-31, New York University, April-1989.

[9] S. Chopra and M.R. Rao. The Partition problem II: valid inequalities and facets. Technical Report 89-26, New York University, April-1989.

[10] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[11] CPLEX Optimization, Inc. *Using the CPLEX Callable Library including Using the CPLEX Linear Optimizer with CPLEX Barrier and Mixed Integer Solvers*, 1989-1994. Versão 3.0.

[12] Cláudio Nogueira de Meneses. Resolução exata do problema da partição retangular de um retângulo com pontos no interior: Uma abordagem em combinatória poliédrica. Master's thesis, Universidade Estadual de Campinas, Instituto de Computação, Campinas, São Paulo, Junho 1997.

[13] Cid Carvalho de Souza. *The Graph Equipartition Problem: Optimal Solutions, Extensions and Applications.* PhD thesis, Université Catholique de Louvain, Faculté des Sciences Appliqués, Unité de Gestion Industrielle, Louvain-La-Neuve, Belgique, Novembro 1993.

[14] U. Faigle and G. Dijkhuizen. A Cutting-plane approach to the edge-weigthed maximal clique problem. *European Journal of Operational Research*, 69:121 a 130, 1993.

[15] U. Faigle, R. Garbe, K. Heerink, and B. Spieker. LP-Relaxations for the edge-weighted subclique problem. *Operations Research*, page 157 a 160, 1993.

[16] U. Faigle, R. Schrader, and R. Suletzki. A Cutting-plane algorithm for optimal graph partitioning. *Methods of Operations Research*, 57:109 a 116, 1986.

[17] Ulrich Faigle. The Clique polytope of a graph. Technical Report No. 87474, Institut für Operations Research, Universität Bonn, W. Germany, July-1987.

[18] Thomas A. Feo and Maurício G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:1 a 27, 1995.

[19] Carlos E. Ferreira, A. Martin, Cid C. de Souza, R. Weismantel, and L. A. Wolsey. The Node capacitated graph partitioning problem: a computational study. *para aparecer em Mathematical Programming*, B, 1996.

[20] Carlos E. Ferreira and Yoshiko Wakabayashi. *Combinatória Poliédrica e Planos-de-Corte Faciais.* Décima Escola de Computação, 8 a 13 de Julho 1996, Campinas-SP.

[21] M. R. Garey and S. J. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness.* Freeman,New York, 1979.

[22] R. E. Gomory. Solving linear programming problems in integer. page 211 a 215, 1960. Proceedings of the Symposium on Applied Mathematics.

[23] M. Grõstchel, M. Jünger, and G. Reinelt. A Cutting-plane algorithm for the linear ordering problem. *Operations Research*, 32:1195 a 1220, 1984.

[24] M. Grötschel, L. Lovász, and A. Schrijver. The Ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169 a 197, 1981.

[25] M. Grötschel, L. Lovász, and A. Schrijver. Corregendum to our paper "the Ellipsoid method and its consequences in combinatorial optimization". *Combinatorica*, 4:291 a 295, 1984.

[26] M. Grötschel and Y. Wakabayashi. A Cutting-plane algorithm for a clustering problem. *Mathematical Programming*, B-45:59 a 96, 1989.

[27] M. Grötschel and Y. Wakabayashi. Facets of the clique partitioning polytope. *Mathematical Programming*, 47:367 a 387, 1990.

[28] K. Hoffman and M. W. Padberg. Solving airline crew scheduling problem by branch-and-cut. *Management Science*, 39:657 a 682, 1993.

[29] E. L. Johnson, G. L. Nemhauser, and M. W. P. Savelsbergh. Progress in integer programming: an exposition. *Submetido a INFORMS Journal on Computing*, 1997.

[30] E. L. Jonhson, A. Mehrotra, and G. L. Nemhauser. Min-cut clustering. *Mathematical Programming*, 62:133 a 151, 1993.

[31] M. Jünger and P. Mutzel. Solving the maximum weight planar subgraph problem by branch-and-cut. page 479 a 492, 1993. Proceedings of the Third Conference on Integer Programming and Combinatorial Optimization.

[32] M. Jünger, G. Reinelt, and S. Thienel. Provably good solutions for the traveling salesman problem. Technical Report No. 92.114, Angewandte Mathematik und Informatik,Institut für Informatik, Universität zu Köln, 1992.

[33] M. Jünger, G. Reinelt, and S. Thienel. Pratical problem solving with cutting-plane algorithms in combinatorial optmization. Technical Report No. 94.156, Angewandte Mathematik und Informatik,Institut für Informatik, Universität zu Köln, 1994.

[34] Michael Jünger and Peter Störmer. Solving large-scale traveling salesman problems with paralell branch-and-cut. Technical Report No. 95191, Angewandte Mathematik und Informatik, Zentrum für Paralleles Rechen (ZPR), Institut für Informatik, Universität zu Köln, 1995.

[35] P. Miliotis. Integer programming approaches to the traveling salesman problem. *Mathematical Programming*, 15:177 a 188, 1976.

[36] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.

[37] Aminadab Pereira Nunes. Uma Abordagem de programação inteira para o problema da triangulação de custo mínimo. Master's thesis, Universidade Estadual de Campinas, Instituto de Computação, Campinas, São Paulo, em preparação.

[38] M. Padberg. The Boolean quadric polytope: some characteristics, facets and relatives. *Mathematical Programming*, B-45:139 a 172, 1989.

[39] M. W. Padberg and G. Rinaldi. A Branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33:60 a 100, 1991.

[40] Kyungchul Park, Kyungsik Lee, and Sung-Soo Park. An Extended formulation approach to the edge-weighted maximal clique problem. *European Journal of Operational Research*, 95:671 a 682, 1996.

[41] Kyungchul Park, Kyungsik Lee, and Sung-Soo Park. An Extended formulation approach to the edge-weighted maximal clique problem. Technical report, Korea Advanced Institute of Science and Technology, Departament of Industrial Engineering, December-1994.

[42] Kyungchul Park, Kyungsik Lee, Sungsoo Park, and Heesang Lee. Telecommunication node clustering with node compatibility and network survivability requeriments. Technical report, Korea Advanced Institute of Science and Technology, Departament of Industrial Engineering, February-1997. Submetido a Management Science.

[43] William R. Pulleyblank. Polyhedral combinatorics. In Springer-Verlag, editor, *Mathematical Programming: the State of the Art*, page 312 a 345. A. Bachem and Martin Grotschel and B. Korte, 1983.

[44] S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Heuristic and special case algorithms for dispersion problem. *Operations Research*, 42:299 a 310, 1994.

[45] Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, 1986.

[46] C. De Simone. The Cut polytope and the boolean quadric polytope. *Discrete Mathematics*, 79:71 a 75, 1989/90.

[47] Helmuth Späth. The Facility selection-location problem. *O. R. Spektrum*, 6:141 a 146, 1984.

[48] Helmuth Späth. Heuristically determining cliques of given cardinality and with minimal cost within weighted complete graphs. *Zeitschrift für Operations Research*, 29:125 a 131, 1985.

[49] J. M. van den Akker, C. A. J. Hurkens, and M. W. P. Savelsberg. A Time-indexed formulation for single-machine scheduling problems: branch-and-cut. Memorandum COSOR 95-24, Eindhoven University of Technology, Eindhoven, 1995. Submetido a Mathematical Programming.

[50] J. M. van den Akker, C. A. J. Hurkens, and M. W. P. Savelsberg. A Time-indexed formulation for single-machine scheduling problems: column generation. *Submetido a INFORMS Journal on Computing*, 1995.