PROBLEMAS DE COLORAÇÃO EM GRAFOS

Yuri Abitbol de Menezes Frota

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Nelson Maculan Filho, D.Sc.

Prof. Márcia Helena Costa Fampa, D.Sc.

Prof. Rafael Castro de Andrade, D.Sc.

Prof. Celso da Cruz Carneiro Ribeiro, D.Sc.

Prof. Claudio Thomas Bornstein, D.Sc.

Prof. Paulo Roberto Oliveira, D.Sc.

RIO DE JANEIRO – RJ, BRASIL

AGOSTO DE 2008

FROTA, YURI ABITBOL DE MENEZES

Problemas de Coloração em Grafos [Rio de Janeiro] 2008

XI, 132 p. 29, 7cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2008)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Branch-and-cut
2. Coloração em grafos
3. Programação inteira

I. COPPE/UFRJ II. Título (série)

Agradecimentos

Agradeço primeiro a Deus que me ilumina e está sempre comigo. Agradeço ao meu pai, à minha mãe e a minha família pelo que sou hoje. Agradeço aos meus professores Nelson Maculan e Márcia Fampa não apenas pelas orientações mas como exemplos de vida. Agradeço também a Thiago Noronha, ao Professor Celso da Cruz Carneiro Ribeiro e a Luidi Simonetti que participaram também da elaboração deste trabalho. Agradeço a Capes pelo apoio durante a elaboração desta tese. Agradeço a todas as pessoas maravilhosas que conheci neste doutorado, levo comigo um pedaço de todos vocês. E finalmente Agradeço também a *B*ttless and Happy* porque a verdadeira amizade não tem distância.

Make 'em Laugh, make 'em laugh...
Donald O'Connor †

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

PROBLEMAS DE COLORAÇÃO EM GRAFOS

Yuri Abitbol de Menezes Frota

Agosto/2008

Orientadores: Nelson Maculan Filho
Márcia Helena Costa Fampa
Programa: Engenharia de Sistemas e Computação

Neste trabalho apresentamos modelos e algoritmos para problemas relacionados ao problema de coloração em grafos. Três problemas foram estudados: O Problema de Coloração Particionada (PCP), o Problema de Coloração Equilibrada (PCE) e o Problema de Alocação de Freqüências (PAF). Na tese, introduzimos modelos de programação linear inteira e discutimos seu uso na solução de aplicações práticas. Apresentamos um estudo parcial do poliedro associado ao modelo PCP e introduzimos novas classes de desigualdades válidas para os demais modelos. Finalmente, descrevemos a implementação de um algoritmo branch and cut que utiliza os modelos apresentados, analisando suas vantagens e limitações.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

GRAPH COLORING PROBLEMS

Yuri Abitbol de Menezes Frota

August/2008

Advisors: Nelson Maculan Filho
Márcia Helena Costa Fampa
Department: Systems Engineering and Computer Science

In this work, we presented models and algorithms for problems related with the graph coloring problem. Three problems were studied: The Partition Coloring Problem (PCP), the Equitable Coloring Problem (ECP) and the Frequency Assignment Problem (FAP). In this thesys, we introduce linear programming formulations and discuss their use in the solution of practical applications. A study of the polytope associated with the PCP formulation is presented and new classes of valid inequalities are introduced. Finally, we describe the implementation of a branch and cut algorithm based on the proposed formulations, discussing its advantages and limitations.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Parte I

Um Algoritmo Branch and Cut para o Problema de Coloração Particionada

Capítulo 1

Introdução

Seja $G = (V, E)$ um grafo e $Q = \{Q_1, ...Q_q\}$ uma partição de seus vértices. O *Problema de coloração particionada de grafos* (PCP) consiste em encontrar um subconjunto de vértices V' que contenha exatamente um vértice de cada componente da partição e que o número cromático do grafo induzido por V' seja mínimo. Por ser uma generalização do problema de coloração em grafos, o PCP mostra-se um problema de difícil solução. Li and Simha [1] mostraram que o problema de decisão para o PCP é NP-completo.

Algoritmos para solucionar o PCP tem sido utilizados na literatura como ferramentas para algoritmos de *Problema de Roteamento e Atribuição de Comprimentos de Ondas* (PRACO) [2] em redes óticas. Nestas redes, uma conexão é definida por um par de vértices que devem comunicar-se entre si. As conexões entre os nós de uma rede totalmente ótica são estabelecidos por caminhos óticos (Figura 1.1). Cada caminho óptico usa um comprimento de onda único em todas as conexões da fonte ao destino. Tecnologias como *Wavelength Division Multiplexing* (WDM) permitem um uso mais eficiente da capacidade das fibras óticas, pois permitem que transmissões simultaneas sejam realizadas na mesma fibra utilizando comprimentos de onda diferentes. Entretanto, dois caminhos óticos que compartilham um nó na rede não podem ser atribuidos ao mesmo comprimento de onda.

O PRACO é definido como o problema de estabelecer rotas para um determinado conjunto de conexões pré-estabelecidas em uma rede, usando o menor número de comprimentos de ondas. Esta variante do problema é denominada min-PRACO *off line*. Outras variantes com diferentes características e critérios de otimização podem

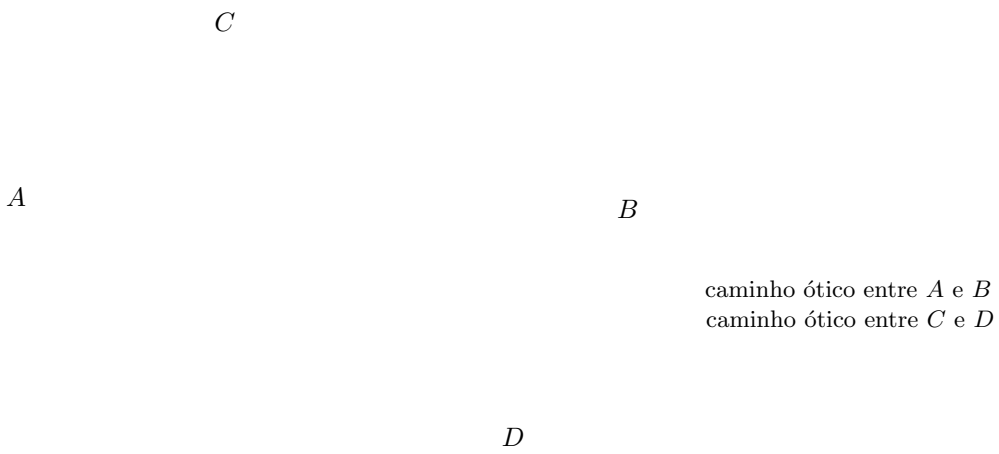


Figura 1.1: Rede ótica.

ser vistos em [2].

Li and Simha [1] propuseram uma estratégia de decomposição em duas fases para resolver o problema min-PRACO *off line*. Na primeira fase, determina-se um conjunto de k rotas possíveis para cada conexão. Em seguida, uma rota (entre aquelas previamente computadas) e um comprimento de onda são associados a cada conexão, de acordo com a solução de um problema de coloração particionada sobre um grafo $G = (V, E)$ e uma partição Q. No grafo G, os vértices correspondem às rotas pré-computadas da rede ótica e existe uma aresta entre quaisquer pares de vértices de G onde as rotas associadas a eles compartilham um elo na rede ótica. Todas as rotas associadas a uma mesma conexão na rede são colocadas na mesma componente da partição Q. Na Figura 1.2 temos um exemplo da geração de uma instância PCP a partir de uma rede ótica. Vemos que para cada uma das conexões $A \rightarrow B$ e $C \rightarrow D$ temos duas possíveis rotas (R^1_{AB}, R^2_{AB} e R^1_{CD}, R^2_{CD} respectivamente) que serão equivalentes a vértices no grafo particionado, e temos as arestas $R^1_{AB}R^2_{CD}$ e $R^2_{AB}R^1_{CD}$ que representam os elos compartilhados com as respectivas rotas.

Até a elaboração deste trabalho não é de nosso conhecimento a existência na literatura de um algoritmo exato para o problema PCP. Este fato serviu de motivação para o desenvolvimento deste trabalho que é um esforço compartilhado com a tese de doutorado de Noronha [3]. No texto a seguir apresentaremos um algoritmo *branch and cut* exato para o problema de coloração particionada baseada numa generalização de uma formulação para o problema de coloração em vértices [4] [5]. As

notações utilizadas durante o texto são apresentadas no Capítulo 2. O Capítulo 3 apresenta a formulação de programação inteira para o PCP. O Capítulo 4 descreve o algoritmo *branch and cut* proposto. Resultados computacionais são apresentados no Capítulo 5. Uma análise conclusiva é realizada no Capítulo 6.

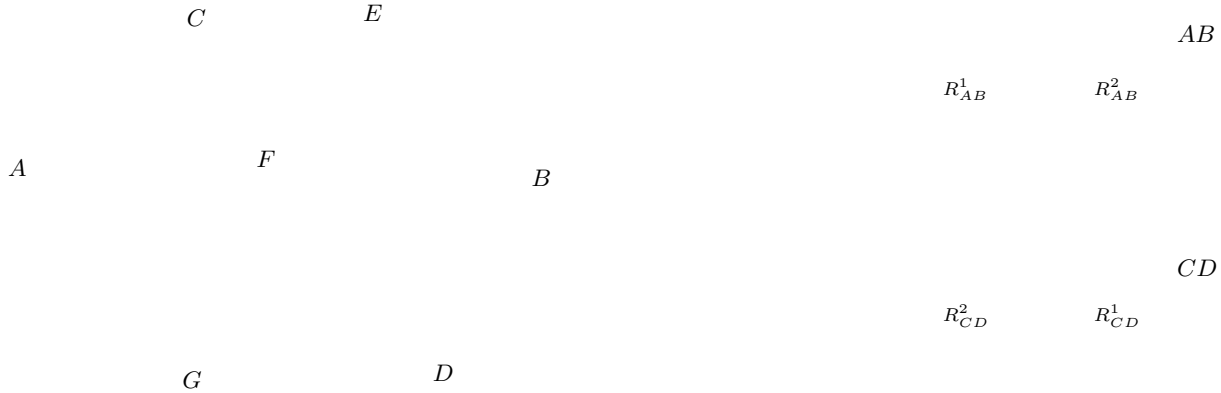


Figura 1.2: Instância de PCP gerada a partir da rede ótica.

Capítulo 2

Notações

Durante a leitura do texto, utilizamos uma notação consistente com o padrão geralmente aceito pelos campos de teoria dos grafos e programação inteira. Este capítulo não tem a intenção de ser um texto de referência, e sim de definição das notações e terminologias adotadas durante o texto. Na Seção 2.1 são apresentados os conceitos fundamentais em grafos. Na seção 2.2 são apresentados alguns conceitos introdutórios em álgebra linear. Na seção 2.3 são vistos conceitos de problema de otimização e algumas definições de programação linear e inteira. Finalizando o capítulo apresentamos alguns métodos para solucionar problemas de otimização na seção 2.4. Livros de referência em teoria dos grafos [6] [7], programação linear e inteira [8] [9] [10] [11] contêm as definições fundamentais apresentadas neste capítulo.

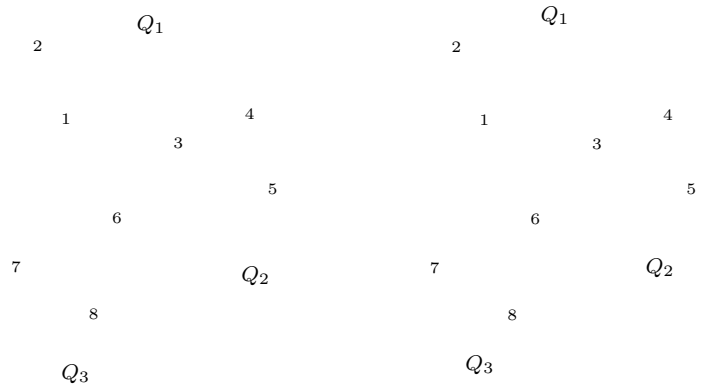
2.1 Teoria de Grafos

Um *grafo* $G = (V, E)$ é uma dupla ordenada de conjuntos, onde V é o *conjunto de vértices ou nós*, E é um conjunto de pares não ordenados de V chamados de *arestas* e $Q = \{Q_1, ...Q_q\}$ é uma partição de V em q subconjuntos, i.e., $Q_1 \cup \ldots \cup Q_q = V$ e $Q_i \cap Q_j = \emptyset$, para cada $i, j = 1, \ldots, q$ com $i \neq j$. Denotaremos $Q_1, \ldots, Q_q$ como as *componentes* da partição e $P[v]$ o índice da componente do vértice $v \in V$. i.e., $v \in Q_{P[v]}$. Alternativamente usaremos $V(G)$ e $E(G)$ para definir respectivamente o conjunto de vértices e o conjunto de arestas do grafo $G = (V, E)$. Denotaremos por n a cardinalidade do conjunto de vértices V, enquanto a cardinalidade do conjunto de arestas E será denotada por m. A aresta definida pelos vértices u e v é denotada

por uv, e os vértices u e v são chamados de *extremidades* da aresta uv. A menos que seja dito o contrário, todo grafo neste trabalho é *simples*, ou seja, não possui *arestas múltiplas* ou *laços*, onde duas arestas são múltiplas se elas coincidem em ambas as extremidades e onde um laço é uma aresta cujas extremidades são iguais.

Dois vértices u e v são *adjacentes* em G se $uv \in E$. Denotaremos por $N(u) = \{w \in V : (u, w) \in E, w \neq u\}$ o *conjunto de adjacências* do vértice u em G. Definimos $A(u) = \{w \in V : (u, w) \notin E, w \neq u\}$ como a *anti-vizinhança* do vértice u (i.e., o subconjunto de vértices que não são adjacentes a u) e $A_P(u) = \{v \in A(u) : P[u] \neq P[v]\}$ como a *anti-vizinhança particionada* do vértice $u \in V$ (i.e., o vértice na anti-vizinhança de u que está em outra componente da partição). Também definimos $A'(u) = A(u) \cup \{u\}$ e $A'_P(u) = A_P(u) \cup \{u\}$. Dado um subconjunto de vértices $V' \subseteq V$, denotaremos $E[V']$ o subconjunto de arestas induzido em $G = (V, E)$ por V'. Um vértice $v \in A_P(u)$ é dito *isolado* em $A_P(u)$ se $E[A_P(u)] = E[A_P(u) \setminus \{v\}]$ (i.e., o vértice v não tem vizinhos em $A_P(u)$). Definimos $\overline{G} = (V, \overline{E})$ o grafo *complementar* de $G = (V, E)$ se ambos têm o mesmo conjunto de vértices onde existe uma aresta vw em $\overline{G}$ se e somente se não existe tal aresta em G. Denotaremos de $\overline{m}$ a cardinalidade do conjunto $\overline{E}$ e $\overline{m}_c$ o número de arestas $(v, w) \in \overline{E}$ onde $P[v] \neq P[w]$.

Um *conjunto particionado* é um subconjunto de V onde todos os vértices pertecem a diferentes componentes, enquanto que um *conjunto independente particionado* é um conjunto particionado tal que não há dois vértices nesse conjunto adjacentes em G (Figura 2.1). O dual de um conjunto independente é uma *clique*, o qual consiste em um subconjunto de vértices dois a dois adjacentes. Uma clique C é denominada *clique particionada* se C também for um conjunto particionado. Um *buraco ímpar* em G é um subconjunto de vértices $B = \{v_0, v_1, \ldots, v_{2k+1}\}$ para $k \geq 2$ e cada vértice v_{i-1} é adjacente ao nó v_i para $i \in \{1, \ldots, 2k + 1\}$, onde $v_0 = v_{2k+1}$ e nenhum outro par de vértices é adjacente. Um *antiburaco ímpar* G é um buraco ímpar no seu grafo complementar. Extenderemos a notação de buraco ímpar/antiburaco ímpar para *buraco ímpar particionado/antiburaco ímpar particionado* quando este for também um conjunto particionado.



(a) **(b)**

Figura 2.1: (a) Um conjunto particionado (b) e um conjunto independente particionado definidos pela linha pontilhada.

2.2 Álgebra linear

Sejam $\mathbb{R}$, $\mathbb{Z}$ e $\mathbb{N}$ os conjuntos dos números reais, inteiros e naturais, respectivamente. Uma *matriz* é uma ordenação de elementos em linhas e colunas. Uma matriz tem *ordem* (m, n) quando ela tiver m linhas e n colunas. Um *vetor* é uma matriz de dimensão $(1, n)$ ou $(m, 1)$. O elemento na i-ésima linha e na j-ésima coluna de uma matriz A é denotado a_{ij}. Para n inteiro, denotaremos por $\mathbb{R}^n$, $\mathbb{Z}^n$ e $\mathbb{N}^n$ os conjuntos de todos os vetores com n componentes reais, inteiras e naturais, respectivamente. Similarmente denotaremos por $\mathbb{R}^{m\times n}$, $\mathbb{Z}^{m\times n}$ e $\mathbb{N}^{m\times n}$ os conjuntos das matrizes reais, inteiras e naturais com m linhas e n colunas.

Seja $A \in \mathbb{R}^{m\times n}$, a $j-$ésima coluna de A é um vetor com m elementos, denotado por a_{*j}.

$$a_{*j} = \begin{pmatrix} a_{1j} \\ \vdots \\ a_{mj} \end{pmatrix}$$

Analogamente, a $i-$ésima linha da matriz A é um vetor de dimensão n, denotado por a_{i*}.

$$a_{i*} = (a_{i1}, \ldots, a_{in})$$

Por convenção vamos utilizar letras maiúsculas quando nos referirmos a matrizes e letras minúsculas quando nos referirmos a vetores.

Uma matriz B é dita *submatriz* da matriz A se ela for obtida através da remoção de linhas ou colunas da matriz original A. Seja $A \in \mathbb{R}^{m \times n}$ e $B \in \mathbb{R}^{p \times q}$ matrizes.

Um conjunto de vetores $a_1, a_2, \ldots, a_k \in \mathbb{R}^k$ é *linearmente independente* se a seguinte condição é satisfeita:

$$\text{se } \sum_{j=1}^{k} \lambda_j a_j = 0 \text{ então } \lambda_j = 0 \text{ para todo } j = 1 \ldots k.$$

De forma similar, um conjunto de vetores $b_1, b_2, \ldots, b_k \in \mathbb{R}^k$ é *afim independente* se a seguinte condição é satisfeita:

$$\text{se } \sum_{j=1}^{k} \lambda_j b_j = 0 \text{ e } \sum_{j=1}^{k} \lambda_j = 0 \text{ então } \lambda_j = 0 \text{ para todo } j = 1 \ldots k.$$

Dada uma matriz $A \in \mathbb{R}^{m \times n}$, denominamos de *posto-linha* da matriz A o número máximo de linhas linearmente independentes de A. Analogamente, denominamos de *posto-coluna* da matriz A o número máximo de colunas linearmente independentes de A. Pode ser mostrado que o posto linha de uma matriz é sempre igual ao seu posto coluna, logo, por simplicidade vamos denotar o posto da matriz A por $posto(A)$ que corresponde ao posto linha ou posto coluna de A.

2.3 Problemas de Otimização

Em certos problemas que ocorrem em vários setores, como indústria, transportes, administração, etc., uma série de recursos precisam ser usados de forma a cumprir um objetivo específico. Todos estes problemas têm em comum o fato de possuir objetivos que estão sujeitos a restrições. Problemas com estas características são classificados como *problemas de otimização*. Como exemplos de problemas de otimização podemos citar o problema de encontrar a clique máxima, denominado *problema da clique máxima*.

Um problema de otimização Π é um par (X, f), onde X é o conjunto de *soluções viáveis* de Π e $f : X \to \mathbb{R}$ é a sua *função custo*. Alternativamente, a notação $X(\Pi)$ indicará o conjunto de pontos viáveis de Π. O objetivo de um problema de otimização Π é encontrar uma solução viável que minimize ou maximize a função custo. No caso de Π ser um problema de minimização, deseja-se encontrar um ponto $x^* \in X$ tal que

$$f(x^*) \leq f(y), \forall y \in X.$$

Por outro lado, no caso de um problema de maximização, o objetivo é encontrar $x^* \in X$ tal que

$$f(x^*) \geq f(y), \forall y \in X.$$

Em ambos os casos, o ponto x^* é chamado de *ótimo global* ou *solução ótima* do problema Π.

Uma função $f : \mathbb{R}^n \to \mathbb{R}$ é uma *função linear* se e somente se $f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$ para todos $x, y \in \mathbb{R}^n$ e $\alpha, \beta \in \mathbb{R}$. Para qualquer função linear $f(x)$ e qualquer número real b, a equação $f(x) = b$ e as inequações $f(x) \leq b$ e $f(x) \geq b$ são lineares. Um subconjunto $P \subseteq \mathbb{R}^n$ é chamado *poliedro* se

$$P = \{x \in \mathbb{R}^n \mid g_i(x) \leq b_i, i = 1 \ldots m\},$$

onde $g_i : \mathbb{R}^n \to \mathbb{R}$ são funções lineares e $b_i \in \mathbb{R}$. Uma função linear $f : \mathbb{R}^n \to \mathbb{R}$ pode ser escrita na forma $f(x) = c^T x$ onde $c \in \mathbb{R}^n$. Analogamente, um poliedro $P \subseteq \mathbb{R}^n$ é da forma $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$, onde $A \in \mathbb{R}^{m \times n}$ e $b \in \mathbb{R}^m$. Diz-se que um poliedro $P \subseteq \mathbb{R}^n$ é *limitado* se existe $u \in \mathbb{R}^n$, $\mid u \mid < \infty$ tal que $P = \{x \in \mathbb{R}^n \mid -u \leq x_i \leq u, i = 1 \ldots n\}$. Um poliedro limitado é chamado de *politopo*.

Seja $P \subseteq \mathbb{R}^n$ um poliedro, defini-se como *posto afim* de P a cardinalidade do maior subconjunto afim independente de P. A *dimensão* de P, denotada por $dim(P)$, é definida como o posto afim de P menos um. Dizemos que o poliedro $P \subseteq \mathbb{R}^n$ tem *dimensão cheia* se a dimensão de P for igual a n.

Seja $P \subseteq \mathbb{R}^n$ um poliedro, $a \in \mathbb{R}^n$ e $\alpha \in \mathbb{R}$. Uma inequação $a^T x \leq \alpha$ é *válida para* P se $P \subseteq \{x \in \mathbb{R}^n \mid a^T x \leq \alpha\}$. Quando o poliedro em questão estiver claro pelo contexto, dizemos simplesmente que a inequação é válida. F define uma *face* no poliedro P se $F = \{x \in P | a^T x = \alpha\}$ para alguma inequação válida $a^T x \leq \alpha$ em P. F é uma *faceta* de P se F é uma face de P e a dimensão de F for igual à dimensão de $P - 1$.

Um *problema de programação linear* (PPL) é um problema de otimização (P, f) onde P é um poliedro e f é uma função linear. Um PPL pode sempre ser expresso sob a forma:

$$\begin{array}{lllll} \text{(PPL)} & \text{Minimizar} & c^T x & & \\ & \text{Sujeito a} & Ax & = & b \\ & & x & \geq & 0 \end{array}$$

onde $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ e $b \in \mathbb{R}^m$. Vamos representar este PPL por (A, b, c). O vetor c é denominado *vetor de coeficientes de custo* e o vetor x é o *vetor de variáveis* a serem determinadas no problema. As equações do tipo $Ax = b$ são chamadas *restrições do problema*. As restrições do tipo $x \geq 0$ são denominadas de *restrições de não negatividade*. A matriz A é denominada *matriz de restrições*.

Um importante tipo de problema de otimização ocorre quando as variáveis do problema são restritas a valores inteiros. Um problema deste tipo envolve uma quantidade inteira finita de objetos. Um *problema de programação linear inteira* (PPI) é um problema de programação linear em que as variáveis devem ser números inteiros. Em termos matemáticos temos

$$\begin{array}{lllll} \text{(PPI)} & \text{Minimizar} & c^T x & & \\ & \text{Sujeito a} & Ax & = & b \\ & & x & \geq & 0 \\ & & x & \in & \mathbb{Z}^n \end{array}$$

As restrições do tipo $x \in \mathbb{Z}^n$ são denominadas de *restrições de integralidade*. Um PPI onde todas as variáveis devem ser binárias (0 ou 1) é denominado de *problema de programação linear inteira 0-1*. Seja um problema de programação linear inteira $Z = \text{Min}\{c^T x \mid x \in P \cap \mathbb{Z}^n\}$ onde $P = \{x \mid Ax = b, x \geq 0\}$, definimos sua *relaxação linear* como o problema $Z^r = \text{Min}\{c^T x \mid x \in P\}$

Problemas de programação inteira, similarmente aos problemas de programação linear, possuem algoritmos para solucioná-los em um número finito de passos. Mas a semelhança pára neste ponto: O problema de programação linear inteira pertencente à classe $\mathcal{N}P-$Difícil.

2.4 Métodos de Solução para (PPI)

- *Branch and Bound*

A maioria dos métodos exatos usa propriedades estruturais do espaço de solução para guiar a busca pela solução ótima do problema em um esquema *branch-and-bound* [11] [12].

O método denominado *branch-and-bound* é um algoritmo de árvore de busca que usa um método de enumeração implícita do espaço de solução para problemas

de otimização combinatória. Seu princípio baseia-se em sucessivas decomposições do problema original em problemas menores com o objetivo de encontrar soluções viáveis e de provar sua otimalidade. Esta enumeração é implícita pois nem todos os subproblemas são analisados. Para isto, utiliza-se a combinação de duas estratégias principais: a primeira consiste em particionar o domínio do problema em uma série de subespaços disjuntos (*branch*) e a segunda estratégia consiste em tentar provar que todas as soluções em algumas destas partições são garantidamente sub-ótimas (*bound*) e não precisam ser analisadas.

Para exemplificar, seja $\overline{z}$ um limite superior para o (PPI) a ser resolvido. A cada iteração do método, tomamos um nó i da árvore de enumeração e encontramos um limite inferior z_i para o problema através da solução da relaxação linear do (PPI). O nó i é analisado e caso $z_i \geq \overline{z}$ o nó é *podado* pois ele não conduzirá a uma solução melhor que $\overline{z}$. Caso contrário, se a solução z_i for inteira faremos $\overline{z} = z_i$, senão ramificamos o nó i gerando novos subproblemas.

- *Planos de Corte*

Outro método bastante utilizado para resolução de (PPI) é o algoritmo de planos de corte. Neste método, a cada iteração i considera-se uma relaxação linear $(\text{PPI})^r$ de (PPI). Resolvido o problema linear $(\text{PPI})^r$, seja x^i a solução ótima encontrada. Se x^i satisfaz às restrições de integralidade, então x^i é também solução ótima de (PPI). Caso contrário, busca-se desigualdades $a^T x \leq \alpha$ que sejam válidas para (PPI) e que sejam violadas por x^i, isto é, $a^T x^i > \alpha$. As novas desigualdades encontradas são adicionadas a $(\text{PPI})^r$ e uma nova iteração é realizada. Caso não seja encontrada nenhuma desigualdade violada por x^i, o método é encerrado sem encontrar a solução ótima de (PPI). O problema de identificar as equações violadas por x^i é denominado *problema de separação* de x^i sobre (PPI).

O algoritmo de planos de corte pode ser incorporado a um método *branch and bound* e utilizado para resolver o problema inteiro em cada nó da árvore de enumeração. A este método denominamos de *branch and cut*.

Capítulo 3

Problema de Coloração Particionada de Grafos

Descrever soluções ótimas para problemas do tipo coloração de grafos é um dos problemas mais desafiadores em otimização combinatória. Uma possível abordagem é formular o problema como um problema de programação linear inteira 0-1. Esta abordagem tem sido bastante utilizada nos últimos anos. O objetivo deste capítulo é descrever o problema de coloração particionada de grafos e o seu modelo matemático.

3.1 Problema de Coloração de Grafos

Uma coloração de vértices de um grafo não direcionado é uma atribuição de cores para cada vértice do grafo onde quaisquer dois vértices adjacentes devem possuir cores distintas. Um grafo G é *k-colorível* se existir uma coloração com k cores para G. Chamamos de *número cromático* de G, denotado por $\chi(G)$, o menor número de cores necessárias para fazer uma coloração de G. O problema de coloração de um grafo pode ser definido como o problema de achar o número cromático do grafo.

Muitos problemas de interesse prático como escalonamento [13], alocação de freqüências [14], alocação de registros [15] [16] e método de elementos finitos [17] podem ser modelados como um problema de coloração. A forma geral destas aplicações envolve a modelagem de um grafo com vértices representando itens de interesse onde uma aresta conecta dois itens incompatíveis. O problema de coloração mínima tem como objetivo atribuir uma cor a cada item tal que cada par que seja incompatível

receba cores diferentes.

O problema de coloração é $\mathcal{N}P-$Difícil para grafos genéricos [18] [19], isto significa que é pouco provável existir algoritmos em tempo polinomial para solucionar estes problemas. De fato, baseado no resultado de Yannakaki e Lund [20], é também pouco provável encontrar rapidamente uma solução aproximada: Assumindo $\mathcal{P} \neq \mathcal{N}P$, não existe algoritmo polinomial aproximativo que possa encontrar soluções que sejam garantidas a estar a uma certa razão da solução ótima.

Apesar das evidentes dificuldades do problema, vários algoritmos para encontrar a coloração ótima do grafo baseados em enumerações implícitas tem sido propostos no decorrer dos anos. O primeiro a utilizar esta técnica foi Brown em 1972 [21], que apresentou uma enumeração implícita de cores seguindo uma ordem de vértices. Seguindo o modelo de Brown, vários outros algoritmos foram criados na tentativa de melhorar o desempenho destas enumerações [22] [23] [24] [25].

Além dos esquemas de enumeração implícita, alguns métodos baseados em programação inteira foram propostos no decorrer dos anos. Mehrotra e Trick [26] propuseram um método de geração de colunas para solucionar o modelo dos conjuntos independentes para coloração. Esta abordagem requer a solução de difíceis subproblemas a cada iteração porém consegue solucionar instâncias de pequeno e médio porte rapidamente. Em [27], Diaz e Zabala apresentaram um método *branch and cut* baseado numa formulação clássica do problema de coloração com a adição de novas famílias de restrições para evitar simetria do modelo. Figueiredo et al. [28] apresentaram também um novo método *branch and cut* baseado no estudo das estruturas das facetas de um politopo 0/1 associado às orientações acíclicas de um grafo. Mais recente Campêlo et al. [4] [5] apresentaram um estudo dos modelos de programação inteira para o problema de coloração de um grafo assim como um novo algoritmo *branch-and-cut* para o problema baseado no modelo de representantes.

3.1.1 Formulação dos Representantes

A formulação dos representantes apresentada por Campêlo et al. [4] procura construir um modelo compacto para o problema de coloração evitando um número de variáveis exponencial do modelo de Mehrotra e Trick [26] porêm mantendo sua robustez. O modelo é apresentado a seguir.

Uma coloração de G pode ser vista como uma família $S_1, ..., S_k$ de $k \geq \chi(G)$ conjuntos independentes de G, com cada conjunto independente nesta família associada a uma cor. Suponha que, para cada cor i, escolhamos um vértice para ser o representante da correspondente classe de cor S_i. Então, cada vértice pode estar em um destes dois estados: ou ele representa uma determinada classe de cor ou existe um outro vértice que representa sua cor. Para descrever esta situação, definimos as variáveis binárias x_{uv}, para todo $u \in V$ e $v \in A[u]$ com a seguinte interpretação: $x_{uv} = 1$ se e somente se u representa a cor de v. Um vetor x formado por todas estas variáveis binárias é um vetor incidente a uma $\chi(G)-$coloração de G se existe uma solução ótima do seguinte modelo de programação inteira.

$$\min \sum_{u \in V} x_{uu} \tag{3.1}$$

sujeito a:

$$\sum_{v \in A'(u)} x_{vu} \geq 1 \quad \forall u \in V \tag{3.2}$$

$$x_{uv} + x_{uw} \leq x_{uu} \quad \forall u \in V, \quad \forall (v,w) \in E \text{ com } v, w \in A(u) \tag{3.3}$$

$$x_{uv} \leq x_{uu} \quad \forall u \in V, \quad \forall v \in A(u) \text{ tal que } v \text{ é isolado em } A(u) \tag{3.4}$$

$$x_{uv} \in \{0,1\} \quad \forall u \in V, \quad \forall v \in A(u). \tag{3.5}$$

O primeiro grupo de restrições indica que cada vértice $u \in V$ precisa ser representado ou por si mesmo ou por outro vértice em sua antivizinhança. Isto significa que esta restrição força que pelo menos uma cor seja atribuída para u. Como os vértices extremos de cada aresta precisam receber cores diferentes, as restrições 3.3 asseguram que eles tenham representantes diferentes. As restrições 3.3 e 3.4 garantem que um vértice v apenas pode ser representado por um vértice u que seja representante.

Esta formulação é mais simples e mais compacta do que as outras formulações da literatura, em particular, o modelo pode ser visto como uma variação da formulação de Mehrotra e Trick [26] que utiliza um número restrito de variáveis binárias. Entretanto, ao mesmo tempo que esta formulação é mais compacta, ela apresenta alguma simetria pois existem $|S|$ maneiras diferentes de representar um conjunto independente S. Em [5] o modelo é revisado e é apresentado uma versão assimétrica da formulação dos representantes que evita a simetria do modelo através da utilização de uma ordenação dos vértices do grafo.

A tabela 3.1 ilustra uma comparação entre os modelos de programação inteira para o problema de coloração dos vértices de um grafo. Para cada modelo, é apresentado o número de variáveis binárias empregadas, o número de restrições lineares e a dimensão do correspondente poliedro.

Formulação	Variáveis	Restrições	Dimensão
Zabala e Méndez [27]	$O(n^2)$	$O(nm)$	$n^2 - \chi(G) - 1$
Mehrotra e Trick [26]	Exponencial	$O(n)$	-
Figueiredo et al. [28]	$O(n+m)$	Exponencial	$2n + 4m$
Representantes [4] [5]	$O(n+\overline{m})$	$O(nm)$	$n + 2\overline{m}$

Tabela 3.1: Modelos de Programação Inteira para Problema de Coloração em Grafos

3.2 Problema de Coloração Particionada de Grafos

O Problema de coloração particionada (PCP) para um grafo $G = (V, E)$ e uma partição $Q = \{Q_1, ...Q_q\}$ de V, consiste em encontrar um subconjunto de vértices $V' \subseteq V$ tal que $|V' \cap Q_i| = 1$, para cada $i = 1, \ldots, q$ (i.e., V' contém um vértice de cada componente Q_i), e o número cromático do grafo induzido em G por V' é mínimo. O PCP é claramente uma generalização do problema de coloração em grafos quando o número de vértices por componente é exatamente igual a 1. Li and Simha [1] mostraram que o problema de decisão para o PCP é NP-completo.

Teorema 1 *PCP é fortemente $NP-$Difícil.*

Prova: O problema de se decidir se um grafo possui uma $k-$coloração, isto é, uma coloração que utilize k cores é $NP-$Completo para qualquer $k \geq 3$ [20]. Seja I_C uma determinada instância do problema de $k-$coloração, uma instância I_{CP} de PCP pode ser construida a partir de I_C simplesmente atribuindo uma componente diferente para cada vértice do grafo. Vemos que como uma solução ótima de I_{CP} pode ser usada para decidir o problema de $k-$coloração e esta transformação é realizada em tempo polinomial sobre o número de vértices do grafo temos que PCP é $NP-$Difícil. O resultado de que PCP é fortemente $NP-$Difícil segue-se pelo fato de que o problema é $NP-$Difícil mesmo quando $k \leq \mid V \mid$. □

A Figura 3.1 ilustra uma instância de PCP. O grafo possui 7 vértices e 3 componentes. A solução ótima utiliza 2 cores: a primeira cor é atribuida para os vértices 2 e 5, enquanto que a segunda cor colore o vértice 3.

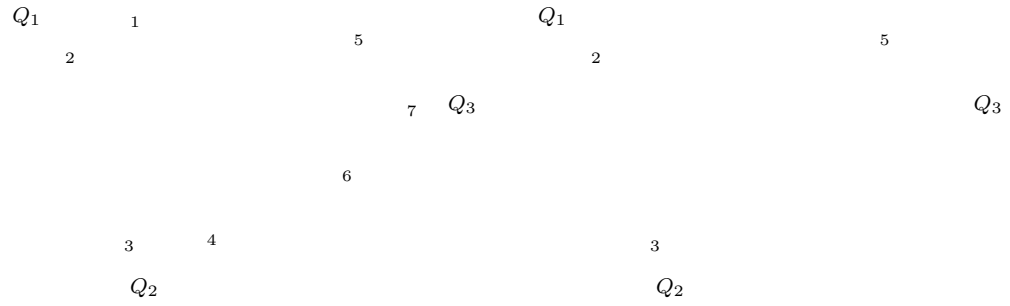


Figura 3.1: (a) Instância de PCP e (b) solução ótima com duas cores.

Li and Simha [1] foram os primeiros a abordar o problema PCP através de um grupo de heurísticas construtivas baseadas em heurísticas gulosas de coloração. Em [29] Noronha e Ribeiro propuseram uma busca local e uma heurística do tipo tabu para o problema que aprimora sensivelmente a qualidade das soluções obtidas pelas heurísticas construtivas. O fato de não existir algoritmos exatos para solucionar o problema PCP na literatura incentivou o desenvolvimento deste trabalho. A seguir apresentaremos uma generalização do modelo de representantes para o PCP.

3.2.1 Formulação dos Representantes Simétrica

A formulação dos representantes para PCP é baseada na seguinte idéia. Um vértice v é escolhido para ser o representante de todos os vértices que tenham a mesma cor de v. Então cada vértice pode estar em um desses 3 estados: (i) ele esta colorido e representa os vértices de uma determinada classe de cor (ii) ele esta colorido e existe um outro vértice que representa sua cor, ou (iii) ele não esta colorido e, consequentemente, não existe nenhum vértice representando sua cor. Definimos as variáveis binárias x_{uv} para todo $u \in V$ e para todo $v \in A'_P(u)$, tal que $x_{uv} = 1$ se e somente se o vértice u representa a cor do vértice v; caso contrário $x_{uv} = 0$. O modelo de representantes para o PCP possui $\mid V \mid + \overline{m}_c$ variáveis e pode ser formulado como o seguinte problema de programação inteira:

$$\min \sum_{u \in V} x_{uu} \tag{3.6}$$

sujeito a:

$$\sum_{u \in Q_p} \sum_{v \in A'_P(u)} x_{vu} \geq 1 \quad \forall p = 1, \ldots, q \tag{3.7}$$

$$x_{uv} + x_{uw} \leq x_{uu} \quad \forall u \in V, \quad \forall (v, w) \in E \text{ com } v, w \in A_P(u) \text{ e } P[v] \neq P[w] \tag{3.8}$$

$$x_{uv} \leq x_{uu} \quad \forall u \in V, \quad \forall v \in A_P(u) \text{ tal que } v \text{ é isolado em } A_P(u) \tag{3.9}$$

$$x_{uv} \in \{0, 1\} \quad \forall u \in V, \quad \forall v \in A'_P(u). \tag{3.10}$$

Este modelo é denominado *formulação de representantes* para PCP. A função objetivo (3.6) conta o número de vértices representantes, i.e., o número total de cores utilizadas na solução. As restrições do tipo (3.7) garantem que cada componente $Q_p, p = 1, \ldots, q$, tem pelo menos um vértice $u \in Q_p$ representado por si mesmo ($x_{uu} = 1$) ou por outro vértice v ($x_{vu} = 1$, com $v \neq u$ e $v \in A_P(u)$), i.e., um vértice de outra componente que não possui uma aresta com u. As restrições do tipo (3.8) obrigam que vértices adjacentes tenham representantes distintos. As restrições (3.8) junto com as restrições (3.9) garantem que um vértice u apenas pode representar algum vértice v caso u seja representante. Notemos que uma solução viável para as restrições (3.7) - (3.10) pode atribuir múltiplos representantes para o mesmo vértice. Entretanto, qualquer um destes vértices representantes leva a uma solução viável.

A formulação de representantes para PCP mantém a estrutura compacta do modelo original, porém ela também apresenta a mesma simetria do modelo para coloração. Com o objetivo de eliminar a simetria deste novo modelo, utilizamos a mesma técnica apresentada em [5] para criar uma formulação assimétrica, generalizando assim, o modelo assimétrico dos representantes.

3.2.2 Formulação dos Representantes Assimétrica

A formulação assimétrica de representantes [5] é generalizada com o objetivo de quebrar a simetria do modelo simétrico para PCP. É estabelecido que um vértice u só pode representar a cor de um vértice v se $P[u] < P[v]$. Logo, vemos que em uma coloração do grafo, o vértice representante de uma cor será aquele de menor índice de componente. Para refletir esta nova situação precisaremos de mais algumas definições.

Definimos $A_{P>}(u) = \{v \in A_P(u) : P[u] < P[v]\}$ como a *anti-vizinhança maior* do vértice $u \in V$ (i.e. os vértices que não podem representar o vértice u) e $A_{P<}(u) =$

$\{v \in A_P(u) : P[v] < P[u]\}$ como a *anti-vizinhança menor* do vértice u (i.e. os vértices que podem representar o vértice u). Definimos também $A'_{P>}(u) = A_{P>}(u) \cup \{u\}$ e $A'_{P<}(u) = A_{P<}(u) \cup \{u\}$. Baseado nestas definições, as inequações (3.7) - (3.10) podem ser reescritas como as inequações (3.12) - (3.15), respectivamente.

Um vértice que é único em uma componente é chamado de *vértice elementar*. Definimos $V^e \subseteq V$ como o conjunto de todos os vértices elementares em G (i.e. o conjunto de vértices que com certeza serão coloridos) e $V^0 = \{u \in V^e : A_{P<}(u) = \emptyset\}$ como o conjunto de vértices elementares que não tem anti-vizinhança menor em G (i.e. vértices que serão sempre representantes), assim como Q^e e Q^0 como o conjunto de componentes que contém vértices em V^e e V^0 respectivamente. Como os vértices $v \in V^0$ são sempre representantes, $x_{vv} = 1$ em qualquer solução de PCP. Logo, estas variáveis são removidas da formulação assimétrica, e o número de variáveis no modelo é reduzido para $\overline{m}_c + |V \backslash V^0|$. A função objetivo (3.6) é reescrita como (3.11).

$$\min \sum_{v \in V \backslash V^0} x_{vv} + \mid V^0 \mid \tag{3.11}$$

sujeito a:

$$\sum_{u \in Q_p} \sum_{v \in A'_{P<}(u)} x_{vu} \geq 1 \quad \forall Q_p \in Q \backslash Q^0 \tag{3.12}$$

$$x_{uv} + x_{uw} \leq \beta_u \quad \forall u \in V, \quad \forall (v,w) \in E \text{ com } v, w \in A_{P>}(u) \text{ e} P[v] \neq P[w] \tag{3.13}$$

$$x_{uv} \leq x_{uu} \quad \forall u \in V, \quad \forall v \in A_{P>}(u) \text{ tal que } v \text{ é isolado em } A_{P>}(u) \tag{3.14}$$

$$x_{uv} \in \{0,1\} \quad \forall u \in V, \quad \forall v \in A'_{P>}(u), \tag{3.15}$$

onde $\beta_u = 1$ se $u \in V^0$; caso contrário $\beta_u = x_{uu}$.

3.2.3 Facetas e Politopo Associado

A relaxação linear da formulação dos representantes para o problema de coloração é fraca. Logo, não foi surpresa averiguar que a generalização do modelo para o PCP também era fraca. Com o objetivo de fortificar a formulação foi generalizado também as duas famílias de facetas descritas em [4, 5]. Primeiro apresentaremos o resultado referente a dimensão do politopo, depois, a generalização das facetas para o modelo PCP.

Para isso, considere $P_c(G) = \{x \in \mathbb{R}^{\overline{m}_c+|V\backslash V^0|} \mid x \text{ satisfaz (3.12) - (3.15) }\}$ como sendo o politopo da formulação assimétrica para PCP. A cada solução viável de $P_c(G)$, associamos um vetor $\mathbb{X} \in \{0,1\}^{\overline{m}_c+|V\backslash V^0|}$, denominado de *vetor de incidência a coloração* em $P_c(G)$ definido da seguinte forma: $\mathbb{X}_{uv} = 1$ se $x_{uv} = 1$ e $\mathbb{X}_{uv} = 0$ caso contrário, para todo $u \in V$ e $v \in A'_{P>}(u)$. Para todo $u \in V$ e $v \in A'_{P>}(u)$, definimos também o *vetor incidente da variável* x_{uv} como $e^{uv} \in \{0,1\}^{\overline{m}_c+|V\backslash V^0|}$ da seguinte forma: $e^{uv}_{ij} = 1$ quando $i = u$ e $j = v$, caso contrário $e^{uv}_{ij} = 0$, i.e., o vetor incidente e^{uv} será nulo com excessão da posição relativa a variável x_{uv}.

Teorema 2 *$P_c(G)$ tem dimensão cheia, i.e., $dim(P_c(G)) = \overline{m}_c + |V\backslash V^0|$.*

Prova: Para provar que $P_c(G)$ tem dimensão cheia, mostraremos que existem $\overline{m}_c + |V\backslash V^0| + 1$ vetores afim independentes incidentes a soluções em $P_c(G)$ [11].

Vamos considerar os seguintes $\overline{m}_c+ \mid V\backslash V^0 \mid +1$ vetores incidentes a coloração em $P_c(G)$:

$$\mathbb{A} = \sum_{u \in V\backslash V^0} e^{uu}$$

$$\mathbb{B}^u = \mathbb{A} - e^{uu}, \quad \forall u \in V\backslash V^e$$

$$\mathbb{C}^u = \mathbb{A} - e^{uu} + e^{ru}, \quad \forall u \in V^e\backslash V^0, \text{ para algum } r \in A_{P<}(u)$$

$$\mathbb{D}^{uv} = \mathbb{A} + e^{uv}, \quad \forall u \in V, \ \forall v \in A_{P>}(u)$$

e os multiplicadores a (associados à $\mathbb{A}$), b^u (associados à $\mathbb{B}^u$, para qualquer $u \in V\backslash V^e$), c^u (associados à $\mathbb{C}^u$, para qualquer $u \in V^e\backslash V^0$), e d^{uv} (associados à $\mathbb{D}^{uv}$, para qualquer $u \in V$ e qualquer $v \in A_{P>}(u)$).

Os vetores incidentes acima são afim independentes caso as equações (3.16) e (3.17) ocorram e todos os seus respectivos multiplicadores forem iguais a zero.

$$\Delta = a + \sum_{u \in V\backslash V^e} b^u + \sum_{u \in V^e\backslash V^0} c^u + \sum_{u \in V, v \in A_{P>}(u)} d^{uv} = 0 \tag{3.16}$$

$$\Omega = a \cdot \mathbb{A} + \sum_{u \in V\backslash V^e} b^u \cdot \mathbb{B}^u + \sum_{u \in V^e\backslash V^0} c^u \cdot \mathbb{C}^u + \sum_{u \in V, v \in A_{P>}(u)} d^{uv} \cdot \mathbb{D}^{uv} = 0 \tag{3.17}$$

Notamos que (3.17) é um sistema de equações, então para cada $w \in V$ e $z \in A'_{P>}(w)$:

$$\Omega_{wz} = a \cdot \mathbb{A}_{wz} + \sum_{u \in V \setminus V^e} b^u \cdot \mathbb{B}^u_{wz} + \sum_{u \in V^e \setminus V^0} c^u \cdot \mathbb{C}^u_{wz} + \sum_{u \in V, v \in A_{P>}(u)} d^{uv} \cdot \mathbb{D}^{uv}_{wz} = 0$$

Primeiro, na equação (3.18), consideramos cada entrada Ω_{ww} da equação (3.17), para cada $w \in V \setminus V^e$. A partir das equações (3.18) e (3.16), temos que $\Delta - b^w = 0$ e $\Delta = 0$, respectivamente. Consequentemente, $b^w = 0$, para todo $w \in V \setminus V^e$.

$$\Omega_{ww} = a + \left(\sum_{u \in V \setminus V^e} b^u - b^w \right) + \sum_{u \in V^e \setminus V^0} c^u + \sum_{u \in V, v \in A_{P>}(u)} d^{uv} = 0 \quad \forall w \in V \setminus V^e \tag{3.18}$$

Agora, na equação (3.19), consideramos cada entrada Ω_{ww} da equação (3.17), para cada $w \in V^e \setminus V^0$. A partir de (3.19) e (3.16), temos que $\Delta - c^w = 0$ e $\Delta = 0$, respectivamente. Consequentimente, $c^w = 0$, para todo $w \in V^e \setminus V^0$.

$$\Omega_{ww} = a + \sum_{u \in V \setminus V^e} b^u + \left(\sum_{u \in V^e \setminus V^0} c^u - c^w \right) + \sum_{u \in V, v \in A_{P>}(u)} d^{uv} = 0 \quad \forall w \in V^e \setminus V^0 \tag{3.19}$$

Na equação (3.20), consideramos agora as entradas Ω_{zw} da equação (3.17), para cada $z \in V, w \in A_{P>}(z)$, onde definimos $\phi^w = c^w$, para todo $w \in V^e \setminus V^0$, caso contrário $\phi^w = 0$. A partir das equações (3.20) e do resultado anterior para c^w, vemos que $d^{zw} = 0$, para todo $z \in V, w \in A_{P>}(z)$.

$$\Omega_{zw} = \phi^w + d^{zw} = 0 \quad \forall z \in V, \quad \forall w \in A_{P>}(z) \tag{3.20}$$

Finalmente, a partir dos resultados anteriores, temos que $\sum_{u \in V \setminus V^e} b^u = 0$, $\sum_{u \in V^e \setminus V^0} c^u = 0$, e
$\sum_{u \in V, v \in A_{P>}(u)} d^{uv} = 0$. Estes resultados juntos com a equação (3.16) implica que $a = 0$.

□

A seguir, verificamos dentre as desigualdades presentes no modelo de representantes assimétrico para PCP quais definem facetas para o politopo $P_c(G)$.

Teorema 3 *As inequações* $x_{ww} \leq 1$ *definem facetas em* $P_c(G)$ $\forall w \in V \backslash V^0$.

Prova: O método de prova para este teorema se baseia na definição de uma faceta como uma face de dimensão igual a $dim(P_c(G)) - 1$. Para isto, iremos exiber $dim(P_c(G))$ soluções afim independente do problema que satisfazem a inequação na igualdade. Considere os $\overline{m}_c + \mid V \backslash V^0 \mid +1$ vetores afim independentes definidos anteriormente $\mathbb{A}, \mathbb{B}^u, \mathbb{C}^u$ e $\mathbb{D}^{uv}$ incidentes a coloração de $P_c(G)$. Caso $w \in V \backslash V^e$ vemos que todos os vetores satisfazem a inequação $x_{ww} \leq 1$ na igualdade com excessão de $\mathbb{B}^u$ quando $u = w$. Caso contrário teremos que $w \in V^e \backslash V^0$ e neste caso vemos que todos os vetores irão satisfazer a inequação na igualdade com excessão de $\mathbb{C}^u$ quando $u = w$. □

Teorema 4 *As inequações* $x_{wz} \geq 0$ *definem facetas em* $P_c(G)$ $\forall w \in V$, $z \in A_{P>}(w)$ $e \mid A_{P<}(z) \mid \geq 2$.

Prova: Similarmente a prova anterior, iremos mostrar novamente $\overline{m}_c + \mid V \backslash V^0 \mid$ soluções afim independentes em $P_c(G)$. Seja $y \in A_{P<}(z)$ tal que $y \neq w$. Notemos que os vetores afim independentes $\mathbb{A}, \mathbb{B}^u, \mathbb{C}^u$ satisfazem a inequação $x_{wz} \geq 0$ na igualdade (caso $z \in V^e$ faremos $\mathbb{C}^z = \mathbb{A} - e^{zz} + e^{yu}$) e junto com os vetores $\mathbb{D}^{uv}$ para $u \neq w$ e $v \neq z$ teremos $\overline{m}_c + \mid V \backslash V^0 \mid$ soluções afim indenpendentes que satisfazem a inequação na igualdade. □

Teorema 5 *As inequações do tipo* $\sum_{u \in Q_p} \sum_{v \in A'_{P<}(u)} x_{vu} \geq 1$ *definem facetas em* $P_c(G)$ $\forall Q_p \in Q \backslash Q^0$

Prova: Considere os seguintes vetores incidentes a coloração em $P_c(G)$:

$$\mathbb{Q}_i^t = \sum_{\substack{u \in V \backslash V^0 \\ u \notin Q_i}} e^{uu} + e^{tt} \text{ para todo } Q_i \in Q \backslash Q^0$$

onde $t \in Q_i$ é o vértice escolhido para ser colorido na componente Q_i.

Primeiro, mostraremos que $F' = \{x \in P_c(G) \mid \sum_{u \in Q_p} \sum_{v \in A'_{P<}(u)} x_{vu} = 1\}$ é uma face de $P_c(G)$. Para isso, basta observarmos que o vetor incidente $\mathbb{Q}_p^t$ a $P_c(G)$ satisfaz F' na igualdade para algum $t \in Q_p$, logo dizemos que $\mathbb{Q}_p^t \in F'$.

Agora, provaremos que F' é uma faceta de $P_c(G)$. Se F' define uma faceta em $P_c(G)$ então não deve existir uma outra faceta $F = \{x \in P_c(G) \mid \lambda x = \sigma\}$ que estritamente contenha F', onde σ é um escalar e λ é um vetor de coeficientes indexado por i, j, para todo $i \in V$ e $j \in A'_{P>}(i)$. Então, provaremos que se $F \supseteq F'$ existe, a inequação $\lambda x \geq \sigma$ é uma combinação linear positiva de $\sum_{u \in Q_p} \sum_{v \in A'_{P<}(u)} x_{vu} \geq 1$ [30].

Iniciaremos mostrando os valores nulos de λ.

Primeiro, considere as λ_{wz} entradas de λ, onde $z \in V \backslash Q_p$ e $w \in A_{P<}(z) \backslash Q_p$. Seja $(\mathbb{Q}_p^t + e^{wz}) \in F'$, temos que $\lambda \mathbb{Q}_p^t = \sigma = \lambda(\mathbb{Q}_p^t + e^{wz})$. Logo,

$$\lambda_{wz} = 0 \quad z \in V \backslash Q_p \text{ e } w \in A_{P<}(z) \backslash Q_p \tag{3.21}$$

Agora considere as λ_{wz} entradas de λ, onde $w \in Q_p$ e $z \in A_{P>}(w)$. Novamente vemos que $(\mathbb{Q}_p^w + e^{wz}) \in F'$. Então, $\lambda \mathbb{Q}_p^w = \sigma = \lambda(\mathbb{Q}_p^w + e^{wz})$. Logo, temos que

$$\lambda_{wz} = 0 \qquad w \in Q_p \text{ e } z \in A_{P>}(w) \tag{3.22}$$

Considerando agora as λ_{zz} entradas de λ, onde $z \in V \backslash Q_p$. Caso $z \notin V^e$ temos que $(\mathbb{Q}_p^t - e^{zz}) \in F'$ nos leva a $\lambda_{zz} = 0$. Caso $z \in V^e \backslash V^0$ vemos que $(\mathbb{Q}_p^t - e^{zz} + e^{wz}) \in F'$ para algum $w \in A_{P<}(z)$. Logo temos que $\lambda \mathbb{Q}_p^w - e^{zz} + e^{wz} = \sigma = \lambda \mathbb{Q}_p^w$ o que leva a $\lambda_{zz} = \lambda_{wz} = 0$.

Agora abordaremos as entradas não nulas de λ. Considere primeiro as λ_{zz} entradas de λ, onde $z \in Q_p$. Caso $Q_p \notin Q^e$, podemos ver que $\mathbb{Q}_p^z - e^{zz} + e^{ww} \in F'$ para algum $w \in Q_p$ onde $w \neq z$. Logo temos que

$$\begin{aligned}
&\lambda \mathbb{Q}_p^z = \sigma = \lambda(\mathbb{Q}_p^z - e^{zz} + e^{ww}) \quad \Rightarrow \\
&\lambda\Big(\sum_{\substack{u \in V \backslash V^0 \\ u \notin Q_p}} e^{uu} + e^{zz}\Big) = \lambda\Big(\sum_{\substack{u \in V \backslash V^0 \\ u \notin Q_p}} e^{uu} + e^{zz}\Big) - e^{zz} + e^{ww} \quad \Rightarrow \\
&\lambda(e^{zz}) = \lambda(e^{ww}) \quad \Rightarrow \\
&\lambda_{zz} = \lambda_{ww}
\end{aligned}$$

Para finalizar mostraremos a igualdade entre as entradas λ_{wz} e λ_{zz} de λ, onde $z \in Q_p$ e $w \in A_{P<}(z)$ e $A_{P<}(z) \neq \emptyset$. O resultado é alcançado através dos vetores $\mathbb{Q}_p^z$ e $(\mathbb{Q}_p^z - e^{zz} + e^{wz}) \in F'$ que nos leva a $\lambda(\mathbb{Q}_p^z) = \lambda(\mathbb{Q}_p^z - e^{zz} + e^{wz}) \in F'$ resultando

em $\lambda_{wz} = \lambda_{zz}$. $\square$

Existem duas famílias de cortes para o modelo de representantes para coloração denominadas de família de *cortes externos* e família de *cortes internos* [5]. A família de cortes externos é baseada na idéia de determinar quantos vértices de um subconjunto $K \subseteq A_{P>}(u)$ podem ser representados por um vértice $u \in V$. Estes cortes são versões fortificadas das inequações (3.13).

Para apresentarmos a família de *cortes externos* alguns termos especiais relacionados a K são utilizados. Para qualquer vértice $v \in K$, definimos α_v como o tamanho máximo de um conjunto independente de $G[K]$ induzido em G por K que contenha v, e $\alpha_K = \max_{v \in K}\{\alpha_v\}$ é o tamanho do conjunto independente máximo de $G[K]$. Definimos *grafo seguro* $G_s = (K, E_s)$ como o grafo composto apenas por *arestas seguras* em K. Uma aresta $vw \in E[K]$ é dita *segura* se existem dois conjuntos independentes W_v e W_w de $G[K]$ e um dado escalar μ onde $v \in W_v$ e $w \in W_w$, tal que $W_v \backslash W_w = \{v\}$, $W_w \backslash W_v = \{w\}$ e $\alpha_z = \mu$, $\forall z \in \{W_v \cup W_w\}$. Logo, se v e w estão na mesma componente conexa de G_s então $\alpha_v = \alpha_w$ [5]. A generalização dos cortes externos para PCP é apresentada a seguir:

Teorema 6 *Seja $u \in V$ e $K \subseteq A_{P>}(u)$ um conjunto particionado não vazio. A inequação (3.23) define uma faceta em $P_c(G)$, se as seguintes duas propriedades forem válidas.*

$$\sum_{v \in K} \frac{x_{uv}}{\alpha_v} \leq \beta_u. \tag{3.23}$$

Propriedade 1 *$G[K]$ é α_K-maximal em $G[A_{P>}(u)]$, i.e., se existe um conjunto particionado K' tal que $K \subset K' \subseteq A_{P>}(u)$ então $\alpha_{K'} > \alpha_K$.*

Propriedade 2 *Para cada $v \in K$, existe um conjunto particionado W_v, que satisfaz três condições: (i) $v \in W_v$, (ii) $\alpha_v = |W_v|$, e (iii) todos os vértices em W_v estão na mesma componente conexa do grafo seguro de K.*

Prova:

Primeiro iremos provar que $F' = \{x \in P_c(G) \mid \sum_{v \in K} \frac{x_{uv}}{\alpha_v} = \beta_u\}$ é uma face de $P_c(G)$. Seja $W \subseteq K$ o conjunto independente particionado máximo de $G[K]$ (i.e.

$\alpha_v = |W|$ para todo $v \in W$). Dado o vetor incidente $\mathbb{E} = \sum_{v \in V \setminus V^0} e^{vv} + \sum_{v \in W} e^{uv}$ de uma solução viável, onde $x_{uv} = 1$, para todo $v \in W \cup \{u\}$, e $x_{uv} = 0$ caso contrário, temos que

$$\sum_{v \in K} \frac{x_{uv}}{\alpha_v} = \sum_{v \in W} \frac{x_{uv}}{\alpha_v} = \sum_{v \in W} \frac{1}{\alpha_v} = \sum_{v \in W} \frac{1}{|W|} = 1$$

Logo, $\mathbb{E} \in F'$ e F' é uma face de $P_c(G)$.

Agora, provaremos que F' é uma faceta de $P_c(G)$ novamente supondo que se existe uma outra faceta $F = \{x \in P_c(G) \mid \lambda x = \sigma\}$ que contenha F', então ela deve ser uma combinação linear positiva de F'. Novamente, iniciaremos mostrando os valores nulos de λ.

Primeiro, considere as λ_{wz} entradas de λ, onde $w \in V \setminus \{u\}$ e $z \in A_{P>}(w)$. Seja $(\mathbb{E} + e^{wz}) \in F'$, temos que $\lambda\mathbb{E} = \sigma = \lambda(\mathbb{E} + e^{wz})$. Logo,

$$\lambda_{wz} = 0 \quad \forall w \in V \setminus \{u\}, \quad \forall z \in A_{P>}(w) \tag{3.24}$$

Agora considere as λ_{uz} entradas de λ, onde $z \in A_{P>}(u) \setminus K$. O resultado de que $(\mathbb{E} + e^{uz}) \in F'$ vem do fato de que a Propriedade 1 garente que o conjunto $W + \{z\}$ é um conjunto independente. Então, $\lambda\mathbb{E} = \sigma = \lambda(\mathbb{E} + e^{uz})$. Logo, temos que

$$\lambda_{uz} = 0 \quad \forall z \in A_{P>}(u) \setminus K \tag{3.25}$$

Considerando agora as λ_{zz} entradas de λ, onde $z \in (V \setminus V^e) \setminus (K \cup \{u\})$. Dado que $(\mathbb{E} - e^{zz}) \in F'$, temos que $\lambda\mathbb{E} = \sigma = \lambda(\mathbb{E} - e^{zz})$. Então,

$$\lambda_{zz} = 0 \quad \forall z \in (V \setminus V^e) \setminus (K \cup \{u\}) \tag{3.26}$$

A seguir, consideramos as λ_{zz} entradas de λ, onde $z \in (V^e \setminus V^0) \setminus (K \cup \{u\})$. Dado que $w \in A_{P<}(z)$ e $(\mathbb{E} - e^{zz} + e^{wz}) \in F'$, temos que $\lambda\mathbb{E} = \sigma = \lambda(\mathbb{E} - e^{zz} + e^{wz})$. Logo,

$$\lambda_{zz} = 0 \quad \forall z \in (V^e \setminus V^0) \setminus (K \cup \{u\}) \tag{3.27}$$

Para completar as entradas nulas de λ, vamos considerar as entradas λ_{zz} onde $z \in K$. Seja W_z um conjunto independente particionado que contém z e que satisfaz as três condições da Propriedade 2. Dessa forma podemos afirmar que $\mathbb{F} = \sum_{v \in V \setminus V^0} e^{vv} + \sum_{v \in W_z} e^{uv} \in F'$ e $(\mathbb{F} - e^{zz}) \in F'$ pois utilizando o conjunto W_z iremos satisfazer F' na igualdade. Logo temos que $\lambda\mathbb{F} = \sigma = \lambda(\mathbb{F} - e^{zz})$. Então,

$$\lambda_{zz} = 0 \quad \forall z \in K \tag{3.28}$$

Agora abordaremos as entradas não nulas de λ. Considere primeiro as λ_{uw} e λ_{uz} entradas de λ, onde $z, w \in K$ e $vw \in E_s$. Seja W_z e W_w dois conjuntos independentes particionados que satisfaçam a condição de aresta segura. Podemos ver que $\mathbb{G} = (\sum_{v \in V \setminus V^0} e^{vv} + \sum_{v \in W_z} e^{uv}) \in F'$ e $\mathbb{H} = (\sum_{v \in V \setminus V^0} e^{vv} + \sum_{v \in W_w} e^{uv}) \in F'$, logo temos que $\lambda\mathbb{G} = \sigma = \lambda\mathbb{H}$. Então,

$$\lambda_{uz} = \lambda_{uw} \quad \forall z, w \in K \mid vw \in E_s \tag{3.29}$$

Agora consideremos as λ_{uu} entradas de λ, para $u \in V^e \setminus V^0$. Seja $C \subseteq G_s$ uma componente conexa do grafo seguro e $D \subseteq C$ um conjunto independente particionado em $G[K]$ que satisfaz as três condições da Propriedade 2. Dado que $r \in A_{P<}(u)$, $\mathbb{I} = (\sum_{v \in V \setminus V^0} e^{vv} + \sum_{v \in D} e^{uv} + e^{ru}) \in F'$, e $\mathbb{J} = (\sum_{v \in V \setminus V^0} e^{vv} - e^{uu} + e^{ru}) \in F'$, temos que

$$\begin{aligned}
&\lambda\mathbb{I} = \sigma = \lambda\mathbb{J} \quad \Rightarrow \\
&\lambda(\sum_{v \in V \setminus V^0} e^{vv} + \sum_{v \in D} e^{uv} + e^{ru}) = \lambda(\sum_{v \in V \setminus V^0} e^{vv} - e^{uu} + e^{ru}) \quad \Rightarrow \\
&\lambda(\sum_{v \in D} e^{uv}) = \lambda(-e^{uu}) \quad \Rightarrow \\
&\sum_{v \in D} \lambda_{uv} = -\lambda_{uu}
\end{aligned}$$

A equação (3.29) implica que $\lambda_{uv} = \lambda_{uz}$, para todo $z, v \in D$. Então

$$\begin{aligned}
&\lambda_{uz} \sum_{v \in D} 1 = -\lambda_{uu} \quad \Rightarrow \\
&\lambda_{uz}|D| = -\lambda_{uu} \quad \Rightarrow \\
&\lambda_{uz} = -\frac{\lambda_{uu}}{\alpha_z}
\end{aligned}$$

O que nos leva a equação (3.30).

$$\lambda_{uu} = -\alpha_z \lambda_{uz} \quad \forall u \in V^e \setminus V^0 \quad \forall z \in D \tag{3.30}$$

Similarmente, considere as λ_{uu} entradas de λ, para $u \in V \setminus V^e$. Dado que $\mathbb{B}^u \in F'$,

e $\mathbb{K} = (\sum_{v\in V\setminus V^0} e^{vv} + \sum_{v\in D} e^{uv}) \in F'$, temos que

$$\begin{aligned}
&\lambda\mathbb{K} = \sigma = \lambda\mathbb{B}^u \quad \Rightarrow \\
&\lambda(\sum_{v\in V\setminus V^0} e^{vv} + \sum_{v\in D} e^{uv}) = \lambda(\sum_{v\in V\setminus V^0} e^{vv} - e^{uu}) \quad \Rightarrow \\
&\lambda(\sum_{v\in D} e^{uv}) = \lambda(-e^{uu}) \quad \Rightarrow \\
&\sum_{v\in D} \lambda_{uv} = -\lambda_{uu}
\end{aligned}$$

A equação (3.29) garante que $\lambda_{uv} = \lambda_{uz}$, para todo $z, v \in D$. Então

$$\begin{aligned}
&\lambda_{uz}\sum_{v\in D} 1 = -\lambda_{uu} \quad \Rightarrow \\
&\lambda_{uz}|D| = -\lambda_{uu} \quad \Rightarrow \\
&\lambda_{uz} = -\frac{\lambda_{uu}}{\alpha_z}
\end{aligned}$$

O que nos leva a equação (3.31).

$$\lambda_{uu} = -\alpha_z\lambda_{uz} \quad \forall u \in u \in V\setminus V^e \quad \forall z \in D \tag{3.31}$$

Por último considere as λ_{uu} entradas de λ, para $u \in V^0$. Dado que $z \in D$ e $\mathbb{K} \in F'$, nós temos que

$$\lambda\mathbb{K} = \lambda\left(\sum_{v\in V\setminus V^0} e^{vv} + \sum_{v\in D} e^{uv}\right)$$

As equações (3.26)-(3.28) implicam que $\sum_{v\in V\setminus V^0} e^{vv} = 0$. Então

$$\lambda\mathbb{K} = \lambda\left(\sum_{v\in D} e^{uv}\right) = \sum_{v\in D} \lambda_{uv}$$

A equação (3.29) nos garante que $\lambda_{uz} = \lambda_{uv}$, para todo $z, v \in D$. Então

$$\lambda\mathbb{K} = \lambda_{uz}\sum_{v\in D} 1 = \lambda_{uz}|D| = \lambda_{uz}\alpha_z = \sigma,$$

O que nos leva a equação (3.32) o que finaliza a prova.

$$\lambda_{uz} = \frac{\sigma}{\alpha_z} \quad \forall u \in V^0, \quad \forall z \in D \tag{3.32}$$

□

Seja K uma clique particionada do grafo G. O Corolário 1 é consequência do fato de que $\alpha_v = 1$, para qualquer $v \in K$

Corolário 1 *Para qualquer $u \in V$ e qualquer clique maximal particionada $K \subseteq A_{P>}(u)$,*

$$\sum_{v \in K} x_{uv} \leq \beta_u \tag{3.33}$$

define uma faceta em $P_c(G)$.

Além das cliques, outras estruturas no grafo podem ser utilizadas para compor os cortes externos. O corolário a seguir tenta explorar o teorema forte de grafos perfeitos que afirma que um grafo G de número cromático $\chi(G)$ deve possuir uma clique de tamanho $\chi(G)$ ou um buraco ou um antiburaco como subgrafos induzidos [31].

Corolário 2 *Para cada $u \in V$ e um buraco/anti-buraco ímpar particionado $H \subseteq A_{P>}(u)$ que satisfaz a Propriedade 1, temos que*

$$\sum_{v \in H} x_{uv} \leq \alpha_H \beta_u \tag{3.34}$$

define uma faceta em $P_c(G)$.

Os cortes internos dos representantes são baseados na idéia de quantas cores seriam necessárias para colorir um buraco/anti-buraco ímpar particionado do grafo. O problema deste tipo de corte é que ele parte da suposição que todos os vértices do grafo serão coloridos, o que não ocorre para o problema PCP. Os únicos vértices do grafo que receberão cores garantidamente seram os vértices elementares, que são únicos em sua componente. Os cortes internos são definidos pela inequação (3.35), onde $H \subseteq V^e$ induz um buraco/anti-buraco ímpar particionado em G e $\chi(G[H])$ é o número cromático do subgrafo $G[H]$ induzido em G por H:

$$\left(\sum_{v \in H \setminus V^0} x_{vv} + \mid H \cap V^0 \mid \right) + \sum_{v \in H \setminus V^0, w \in A_{P<}(v) \setminus H} x_{wv} \geq \chi(G[H]). \tag{3.35}$$

Teorema 7 *Se $H \subseteq V^e$ induz um buraco ou um anti buraco em G, então (3.35) é uma inequação válida para o modelo assimétrico dos representantes para PCP.*

Prova: A primeira componente do lado esquerdo contabiliza os vértices em H que são representantes. A segunda componente do lado esquerdo conta o número de vértices em H que são representados por algum vértice que não pertence a H. Em qualquer coloração viável de G, cada cor presente em H contabiliza uma unidade ou no primeiro ou no segundo termo do lado esquerdo da inequação. Como precisamos de pelo menos $\chi(G[H])$ cores para colorir $G[H]$, a inequação (3.35) é válida. □

Vemos que os cortes internos para PCP se reduzem a cortes de coloração clássica pelo fato de serem aplicados apenas sobre vértices elementares do grafo particionado. Por esse motivo a prova de que os cortes internos para PCP definem facetas é essencialmente igual a encontrada em [5].

Capítulo 4

Um Método Branch and Cut para PCP

Neste capítulo descrevemos os principais componentes do algoritmo de *branch and cut* para o problema de coloração particionada[32] baseado na formulação assimétrica de representantes apresentada no capítulo anterior. Primeiro na seção 4.1 explicamos regras de pré-processamento que ajudaram a diminuir o tamanho das instâncias de PCP. A seguir na seção 4.2 apresentamos uma estratégia de ramificação para o problema. Na seção 4.3 mostramos um método de obtenção de soluções viáveis para cada subproblema da árvore de *branch and cut*. Finalizando, na seção 4.4 são apresentados métodos heurísticos para os problemas de identificação dos cortes violados do PCP.

4.1 Pré-processamento

Antes de se aplicar algum método para a solução do PCP, o grafo pode ser pré-processado com o objetivo de simplificá-lo. Vemos que mesmo grafos de tamanhos moderados podem possuir um grande número de variáveis e restrições, logo é importante o uso de pré-processamento para tentar eliminar o maior número destas e deixar o modelo com um tamanho razoável.

O pré-processamento pode ser dividido em três partes principais. Primeiro todas as arestas $(u, v) \in E$ onde $P[u] = P[v]$ são removidas do grafo pois apenas um dos dois vértices será colorido em qualquer solução viável do problema.

A seguir, é realizada uma busca por vértices elementares universais no grafo (i.e., vértices que são únicos em suas respectivas componentes e que são adjacentes a todos os demais vértices do grafo). Caso seja encontrado algum, este vértice é removido do grafo. O número de vértices elementares universais removidos é adicionado ao número cromático obtido no final do método de solução.

Depois destes dois passos, o pré-processamento finaliza seu procedimento procurando eliminar vértices de baixo grau do grafo. Dado um limite inferior inicial para o problema, os vértices de grau abaixo deste limitante inferior são retirados do grafo. Isto é possível pois estes vértices sempre poderão ser coloridos com qualquer cor da solução que não estiver sendo utilizada por seus vizinhos.

4.2 Estratégia de Ramificação

O procedimento de ramificação é responsável pela decomposição do problema original em subproblemas menores com o objetivo de percorrer o espaço de solução do problema buscando integralidade sem aumentar a complexidade dos subproblemas. Para o problema de coloração particionada de grafos este objetivo é alcançado através de uma estratégia de ramificação similar a desenvolvida por Mehrotra e Trick [26] para o problema de coloração que assegura que os subproblemas gerados em cada ramo da árvore de *branch and bound* são também problemas de coloração particionada. Além disto, a solução inteira ótima do problema repousa em apenas um dos ramos. O procedimento de ramificação é apresentado a seguir.

Definimos duas operações para o PCP, para qualquer $i, j = 1, \ldots, q$ tal que $i \neq j$:

- IGUAL(i, j) assegura que uma mesma cor será utilizada para colorir as componentes Q_i e Q_j; e
- DIFERENTE(i, j) garante que cores diferentes serão utilizadas para colorir as componentes Q_i e Q_j.

A primeira operação pode ser implementada unificando Q_i e Q_j em apenas uma única componente: esta nova componente será formada pela identificação de cada par de vértices $w \in Q_i$ e $z \in A_P(w) \cap Q_j$ tal que $(w, z) \notin E$ em um novo vértice y, onde os vizinhos de y serão definidos como a união da vizinhança de w e z. Isto

nos garante que qualquer cor atribuida ao vértice y em uma solução, poderá ser utilizada pelos vértices originais w e z.

A segunda operação DIFERENTE(i, j) pode ser facilmente implementada apenas inserindo arestas entre todos os pares de vértices não adjacentes entre as componentes Q_i e Q_j. Este processo irá garantir que os vértices coloridos destas componentes terão cores diferentes.

Esta estratégia de ramificação tem como vantagem gerar grafos mais densos sem aumentar a complexidade do subproblema. A Figura 4.1 ilustra a decomposição do problema definido pelo grafo G em dois subproblemas disjuntos definidos pelos grafos G_1 e G_2, onde G_1 é gerado por IGUAL$(1, 2)$ e G_2 é gerado por DIFERENTE$(1, 2)$.

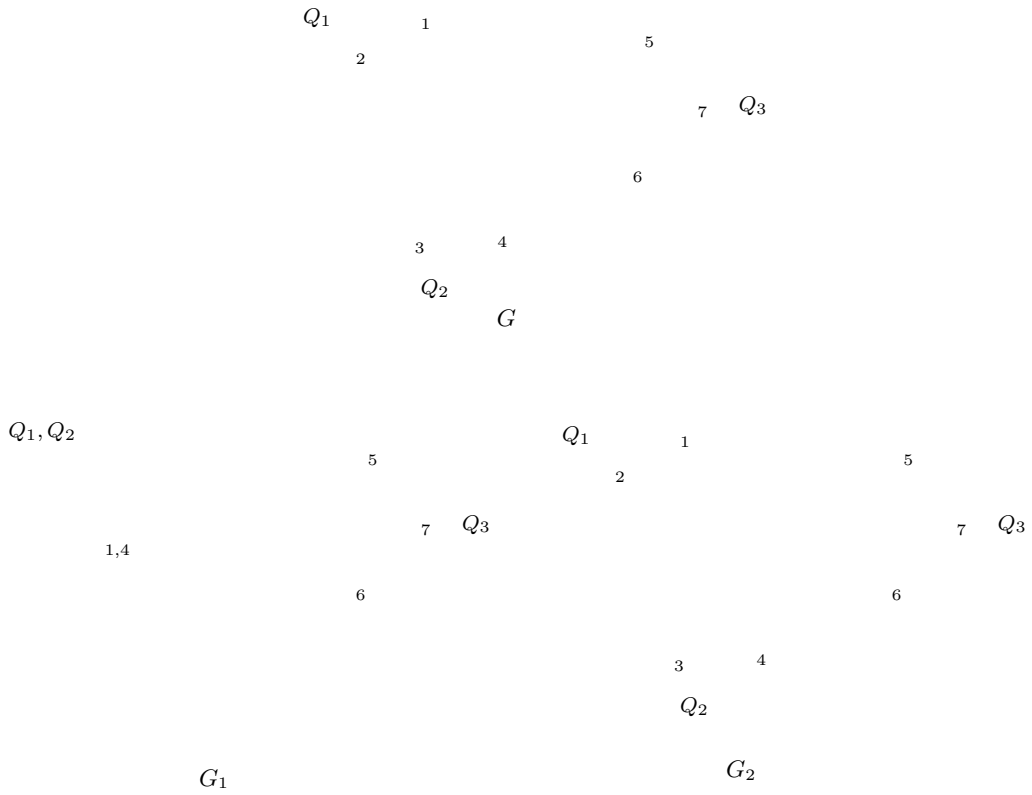


Figura 4.1: Decomposição do grafo G em dois subproblemas disjuntos G_1 e G_2.

4.3 Métodos Primais

Li e Simha [1] propuseram dois grupos de heurísticas construtivas denominadas de algoritmos *one-step* e algoritmos *two-step*. As heurísticas *one-step* são extensões de tradicionais métodos para o problema de coloração como: *Largest–First* [33],

Smallest–Last [34], e *Color–Degree* [35]. A cada iteração destas heurísticas, um vértice é selecionado em uma componente ainda não colorida da partição, de acordo com o mesmo critério guloso usado na respectiva heurística para coloração em grafos. A seguir, o vértice selecionado é colorido com a menor cor ainda disponível (i.e., que ainda não foi atribuida a um de seus vizinhos), e todos os demais vértices desta componente são descartados. Estes passos são repetidos até que toda componente tenha um vértice colorido. Os algoritmos da família *two-step* possuem duas fases. Primeiro, um vértice é selecionado em cada componente porém não é colorido, a seguir é aplicado uma heurística gulosa para o problema de coloração sobre o grafo induzido pelos vértices selecionados. Dentre estas duas classes de algorítmos, Li e Simha obtiveram os melhores resultados com a heurística `one-stepCD` (i.e., o algoritmo *one-step* com critério guloso de *Color–Degree*).

Recentemente Noronha e Ribeiro [29] propuseram um algoritmo de busca tabu [36] para o problema PCP. O algoritmo `one-stepCD` é utilizado para criar uma solução viável inicial $\mathbb{X}$ para PCP com `max`C cores. O procedimento de busca tabu `TS-PCP` descrito no Algoritmo 1 busca encontrar soluções viáveis com o menor número de cores. Denotamos por *conflito de cores* quando um par de vértices adjacentes em diferentes componentes do grafo estão coloridos com a mesma cor. Denotamos tambem por `conflito`($\mathbb{X}$) o número de conflitos de cores na solução $\mathbb{X}$. O algoritmo inicia seu procedimento na linha 2 construindo uma nova solução $\mathbb{X}'$, possivelmente inviável, usando `max`$C-1$. O contador de iteração e a lista tabu são inicializadas na linha 3. O conjunto Q_c de componentes envolvidas em conflitos de cores é iniciado na linha 4. O laço nas linhas 6-15 investiga a vizinhança da solução corrente e enquanto for possível, diminui o número de conflitos de cores. Cada vizinho é obtido por duas operações: (a) recolorir um vértice envolvido em um conflito de cores com uma cor diferente ou (b) trocar o vértice que está colorido na componente envolvido no conflito de cores por outro vértice da mesma componente e colorí-lo. Uma componente $Q' \in Q_c$ envolvida num conflito de cores é aleatoriamente selecionada na linha 6. A variável *reducao* é utilizada para indicar se uma nova solução com um menor número de conflitos foi encontrada enquanto que a variável `maxConflitos` guarda o número de conflitos de cores do vizinho da solução corrente com o menor número de conflitos. O laço nas linhas 9-15 investiga a vizinhança correspondente

a componente selecionada Q'. Uma nova possível solução $\mathbb{X}''$ é construida na linha 11 colorindo o vértice $i \in Q'$ com uma cor $\ell = 1, \ldots, \texttt{max}C - 1$. O melhor vizinho $\bar{\mathbb{X}}$ da solução corrente é atualizado nas linhas 12-13. Se o número de conflitos de cores de $\mathbb{X}''$ for menor que o número de conflitos de $\mathbb{X}'$, então $\mathbb{X}'$ é atualizado na linha 15, junto com o conjunto de componentes envolvidas nos conflitos de cores e a variável *reducao*. A iteração também é incrementada.

Se não existirem conflitos de cores na nova solução $\mathbb{X}'$, a solução corrente $\mathbb{X}$ é atualizada nas linhas 17 e 18, e uma nova tentativa de encontrar uma nova solução com menos cores é iniciada na linha 20. Caso contrário o vértice $\bar{i}$ colorido com a cor $\bar{\ell}$ na solução corrente $\mathbb{X}'$ é identificado e o par $(\bar{i}, \bar{\ell})$ é inserido na lista tabu. Na linha 24, a solução corrente é atualizada para o melhor vizinho encontrado $\bar{\mathbb{X}}$. Se o critério de parada não for satisfeito, uma nova iteração é iniciada na linha 26. O número máximo de iterações é definido por $\texttt{maxIter} = q \cdot (\texttt{max}C - 1) \cdot F_{end}$, onde q é o número de componentes do grafo e F_{end} é um parâmetro a ser calibrado.

Este método de busca tabu descrito por Noronha e Ribeiro [29] é utilizado para gerar limites primais para cada nó da árvore de busca do *branch and cut*. Entretanto, utilizamos uma estratégia diferente para gerar as soluções inicias para o método do `TS-PCP`.

A estratégia de ramificação descrita na seção 4.2 nos garante que o grafo associado a um nó pai na árvore de ramificação difere de no máximo 2 componentes em relação aos grafos associados aos nós filhos. Esta propriedade nos permite construir uma solução inicial para o `TS-PCP` em qualquer nó da árvore de busca usando a solução prévia do nó pai. O procedimento de construção inicia seu processo com uma solução primal parcial igual a solução do nó pai, exceto pelas duas componentes envolvidas no procedimento de ramificação. Esta solução parcial, que possui no máximo duas componentes não coloridas, é extendida utilizando o algoritmo guloso `one-stepCD` [1]. Logo, o método `TS-PCP` inicia seu procedimento com uma solução inicial que possui no máximo duas cores adicionais em relação ao nó pai da árvore de busca.

1. **Algoritmo:** `TS-PCP`$(G, \mathbb{X}, \texttt{max}C)$
2. Constroi nova solução $\mathbb{X}'$ recolorindo aleatoriamente com uma das $1, \ldots, \texttt{max}C - 1$ cores cada vértice originalmente colorido com a cor $\texttt{max}C$;
3. Inicialize lista tabu e faça $iter \leftarrow 0$;
4. Seja Q_c o conjunto de componentes envolvidas em conflitos de cores em $\mathbb{X}'$;
5.
6. **Enquanto** $Q_c \neq \emptyset$ **Faça**
7. Selecione aleatoriamente $Q' \in Q_c$ e atualize $Q_c \leftarrow Q_c \setminus \{Q'\}$
8. Faça `maxConflitos` $\leftarrow \infty$
9. **Para** $i \in Q'$ **e** $\ell = 1, \ldots, \texttt{max}C - 1$ **Enquanto** .`NOT`.*reducao* **Faça**
10. **Se** o par i, ℓ não estiver na lista tabu **ou** satisfizer o critério de aspiração **Então**
11. Constroi a solução $\mathbb{X}''$ recolorindo o vértice i com a cor ℓ
12. **Se** `conflito`$(\mathbb{X}'') <$ `maxConflitos` **Então**
13. `maxConflitos` $\leftarrow$ `conflito`$(\mathbb{X}'')$, $\bar{\mathbb{X}} \leftarrow \mathbb{X}''$, e $\bar{Q} \leftarrow Q'$;
14. **Se** `conflitos`$(\mathbb{X}'') <$ `conflitos`$(\mathbb{X}')$ **Então**
15. $\mathbb{X}' \leftarrow \mathbb{X}''$, *reducao* $\leftarrow$.`TRUE`., $iter \leftarrow iter + 1$, e atualiza Q_c;
16.
17. **Se** `conflitos`$(\mathbb{X}') = 0$ **Então**
18. $\mathbb{X} \leftarrow \mathbb{X}'$
19. $\texttt{max}C \leftarrow \texttt{max}C - 1$;
20. **Retorne para** linha 2
21. **Senão**
22. Seja $\bar{i}$ o vértice colorido com a cor $\bar{\ell}$ na componente $\bar{Q}$ de $\mathbb{X}'$;
23. Inserir o par $\bar{i}, \bar{\ell}$ na lista tabu;
24. $\mathbb{X}' \leftarrow \bar{\mathbb{X}}$;
25. $iter \leftarrow iter + 1$;
26. **Se** $iter <$ `maxIter` **Então Retorne para** linha 3
27.
28. **Retorne** $\mathbb{X}$

Algoritmo 1: Busca tabu para PCP

4.4 Identificação de Cortes

Um algoritmo de planos de cortes foi desenvolvido baseado na identificação das inequações (3.33) até (3.35). O problema de separação destas desigualdades consiste na identificação de alguma sub-estrutura como cliques, buracos e anti-buracos particionados nos grafos associados com cada nó da árvore de busca. Na seção 4.4.1 apresentaremos uma heurística GRASP para encontrar cortes externos cliques. Uma modificação do algoritmo de Hoffman e Padberg [37] para encontrar cortes baseados em buracos e anti-buracos particionados será apresentada nas seções 4.4.2 e 4.4.3.

4.4.1 Cortes Externos Clique

O GRASP [38, 39, 40, 41, 42] é um método multi-partida onde cada iteração consiste na construção de uma solução míope aleatória seguida de uma busca local usando a solução construída como ponto inicial da busca. Este procedimento se repete várias vezes e a melhor solução encontrada dentre todas as iterações do GRASP é retornada.

Definimos $G^u = (V^u, E^u)$ o subgrafo induzido no grafo G_{no} assossiado a um nó da árvore de busca pela anti-vizinhança maior $V^u = A_{P>}(u)$ do vértice $u \in V$. Definimos também $\bar{x}_{uv}$ como sendo o valor ótimo da variável x_{uv} na relaxação linear do modelo assimétrico para PCP, para todo $u \in V$ e $v \in A'_P(u)$. A heurística GRASP tenta encontrar uma clique C com maior peso $\sum_{v \in K} \bar{x}_{uv}$ para cada vértice $u \in V$ tal que $\bar{x}_{uu} > 0$. A sua fase construtiva inicia seu procedimento com uma clique C vazia e constroi uma clique particionada inserindo vértices em C até que C se torne maximal.

Seja $C \subseteq V^u$ uma clique particionada de G_{no} e $\delta(C) = \{w \in V^u \backslash C \mid C \cup \{w\}$ é também uma clique particionada de $G_{no}\}$. A seleção do próximo vértice que irá compor C é realizada através de um conceito similar ao de *roleta russa* em algoritmos genéticos [43]. Este método consiste em associar a cada vértice v uma fatia da roleta igual ao seu valor $\bar{x}_{uv}$, dada por:

$$Prob_v = \bar{x}_{uv} / (\sum_{z \in \delta(C)} \bar{x}_{uz})$$

De acordo com essa expressão, a probabilidade de um vértice ser escolhido é proporcional ao seu peso, assim nós de peso maior possuem maior probabilidade de serem

escolhidos. A cada iteração um vértice é selecionado e o conjunto $\delta(C)$ é atualizado. O procedimento se repete até $\delta(C) = \emptyset$.

Como em qualquer heurística de construção, a solução inicial gerada pelo método não possui nenhuma garantia de ser localmente ótima com respeito a uma vizinhança de soluções qualquer. Então é quase sempre interessante aplicar um método de busca local para tentar melhorar esta solução inicial. Um algoritmo de busca local funciona iterativamente substituindo a solução corrente por uma possível solução melhor presente na vizinhança desta. O método termina quando não existe tal solução.

Definimos a vizinhança $\gamma(C)$ como o conjunto de todas as cliques particionadas que podem ser obtidas através da troca de um vértice $v \in C$ por outro vértice $u \in \delta(C \backslash \{v\})$. A fase de busca local inicia seu processo com a solução gerada pela fase construtiva. O método iterativamente troca a solução corrente por uma de maior peso presente em sua vizinhança. A fase de busca local finaliza seu processo quando não exister nenhuma solução de peso maior na vizinhança da solução corrente.

O método GRASP pára após a realização de $10 \cdot |V^u|$ iterações sem melhoria da melhor solução. As $|V^u|$ cliques particionadas de maior peso encontradas servirão de base para os cortes cliques que serão adicionados no modelo assimétrico para PCP.

4.4.2 Cortes Externos de Buracos e Anti-buracos

Para cada $u \in V$ tal que $\bar{x}_{uu} > 0$, o problema de separação dos cortes externos de buracos ímpares consiste em encontrar um buraco ímpar ou um anti-buraco ímpar $H \in V^u$ tal que $\sum_{v \in H} \bar{x}_{uv} > \alpha_H \beta_u$.

Hoffman e Padberg [37] desenvolveram uma heurística para identificação de restrições violadas baseadas em estruturas buraco. Estas inequações foram utilizadas num método de enumeração implícita para o problema de escalonamento de tripulação em companhias aérias. Desenvolvemos uma generalização do algoritmo de Hoffman para encontrar inequações de buraco ímpar em $G^u = (V^u, E^u)$. O mesmo algoritmo é utilizado no grafo complementar para encontrar inequações de anti-buraco que estejam violadas.

A heurística de Hoffman e Padberg utiliza a noção de *grafo de níveis*. Um grafo de nível é construído a partir de um vértice raiz $r \in V^u$ escolhido aleatoriamente.

A seguir, um rótulo h_w é atribuido para cada vértice $w \in V^u$. O valor h_w é definido como a distância mínima em arestas entre o vértice w e o vértice raiz r. Por exemplo, o rótulo do vértice raiz r é igual a 0, o rótulo dos vértices vizinhos de r é 1, e assim por diante. A Figura 4.2 ilustra um grafo de níveis, construído a partir do vértice r. Para qualquer dois vértices adjacentes w e z com níveis $h_w = h_z \geq 2$, se existir dois caminhos disjuntos em vértices p_w (de w para r) e p_z (de z para r), então existe um ciclo ímpar que contém w, z, e r.

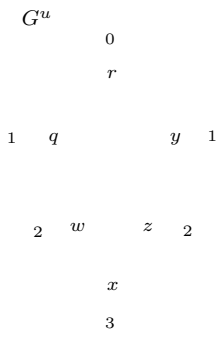


Figura 4.2: Grafo de níveis.

O algoritmo então atribui pesos $t_{wz} = 2 - \bar{x}_{uw} - \bar{x}_{uz}$ para cada aresta $(w, z) \in E^u$. A seguir, para cada aresta $(w, z) \in E^u$ com $h_w = h_z \geq 2$ e $P[w] \neq P[z] \neq P[r]$, o algoritmo busca dois caminhos mínimos particionados disjuntos em vértices (i.e., caminhos mínimos compostos apenas por vértices de diferentes componentes), um de z para r e outro de w para r. Se ambos caminhos existirem, então um buraco ímpar particionado é encontrado e pode ser utilizado para gerar um corte externo possivelmente violado. Caso contrário, o algoritmo passa para a próxima aresta. O algoritmo é executado $0.4 \cdot |V^u|$ vezes, começando sempre de diferentes vértices raízes.

4.4.3 Cortes Internos de Buracos e Anti-buracos

Definimos $G^e = (V^e, E^e)$ o subgrafo induzido no grafo G_{no} assossiado a um nó da árvore de busca pelos vértices elementares $V^e \subseteq V$. O problema de separação para cortes internos baseados em buracos e anti-buracos ímpares consiste em encontrar um buraco ou anti-buraco particionado $H \in V^e$ tal que

$$\sum_{v \in H \setminus V^0} \bar{x}_{vv} + |H \cap V^0| + \sum_{v \in H \setminus V^0, w \in A_{P<}(v) \setminus H} \bar{x}_{wv} < \chi(G[H]).$$

O método aplicado para resolver este problema de separação é similar ao método aplicado para encontrar cortes externos baseados em buracos. Porém desta vez não buscamos estruturas contidas na anti-vizinhança particionada de um vértice, logo todos os vértices elementares do grafo são considerados nesta busca. O mesmo método é aplicado no grafo complementar com a finalidade de encontrar inequações baseadas em anti-buracos que estejam violadas.

Similarmente ao algorítmo anterior, o método constroi um grafo de níveis a partir de um vértice raiz $r \in V^e$ aleatoriamente selecionado. A seguir, o algorítmo atribui pesos

$$t_{wz} = \sum_{u \in A'_{P>}(w)} \bar{x}_{uw} + \sum_{u \in A'_{P>}(z)} \bar{x}_{uz}$$

para cada aresta $(w, z) \in E^e$. Notemos agora que os pesos atribuidos às arestas refletem positivamente o valor das variáveis do problema pois buscamos minimizar o valor destas relacionadas ao buraco, o que implicará numa maior possibilidade de encontrar um corte interno violado.

A seguir, para cada aresta $(w, z) \in E^e$ com $h_w = h_z \geq 2$, o algoritmo calcula um caminho mínimo particionado p_w em G^e do vértice w até o vértice raiz r. Caso p_w exista, o método tenta encontrar um novo caminho mínimo particionado p_z também em G^e, disjunto em vértices do caminho p_w, entre o vértice z e o vértice raiz r onde os pesos das arestas do caminho p_z são definidos por

$$t_{zz'} = \sum_{u \in A'_{P>}(z) \setminus p_w} \bar{x}_{uz} + \sum_{u \in A'_{P>}(z') \setminus p_w} \bar{x}_{uz'}$$

para cada aresta $(z, z') \in p_z$. Caso p_z exista, um buraco ímpar particionado que pode possivelmente ser usado para gerar um corte interno é encontrado. Caso contrário o algoritmo prossegue para a próxima aresta. O algoritmo é executado $0.4 \cdot |V|$ vezes, começando sempre de diferentes vértices raízes.

Capítulo 5

Resultados Computacionais

Neste capítulo apresentamos os experimentos computacionais executados sobre o método de *branch-and-cut* descrito no capítulo anterior. Na seção 5.1 são apresentados alguns detalhes de implementação referentes ao método de solução. Na seção 5.2 apresentamos os resultados e análises do método *branch-and-cut* para o PCP.

5.1 Detalhes de Implementação

A estrutura do algoritmo *branch-and-cut* foi implementada na linguagem de programação $C++$ e compilada com a versão v3.14 do compilador Linux/GNU. A relaxação linear do modelo de programação inteira foi resolvida utilizando $XPRESS-MP$ versão 2005-a sobre uma estação AMD-Atlon 1.8 Ghz de velocidade e um Gbyte de memória RAM.

Ordenação das Componentes

A ordem em que as componentes de vértices se encontram possui uma grande influencia no desempenho da formulação dos representantes. Este resultado é consequência do fato de que a ordem das componentes irá determinar a cardinalidade do conjunto V^0. Durante a implementação foi considerada uma ordem onde as componentes que contém vértices da maior clique maximal particionada C_{ini} encontrada são dispostos nas primeiras posições da ordem. Apesar da determinação da clique inicial C_{ini} ter um profundo impacto no desempenho do modelo, testes realizados indicaram que a ordem das demais componentes pouco interferem no resultado. Para a determinação da clique inicial C_{ini} foi utilizado a heurísta GRASP descrita na

Seção 4.4.1.

Limitantes Primais

Em cada nó da árvore de busca do método de *branch-and-cut* é executada a heurística primal `TS-PCP` descrita na Seção 4.3 na tentativa de aprimoramento da melhor solução viável para o problema. A heurísticaTS-PCP obteve os melhores resultados com o parâmetro $F_{end} = 50$.

Metodologia de Ramificação

Em nossa implementação, foi utilizada uma estratégia de *best-first* para a escolha do nó da árvore de busca a ser averiguado. Os nós da árvore foram ponderados pela média entre o seu limite inferior e seu limite superior.

Para implementar a escolha de quais componentes iriam ser ramificadas foi utilizado o critério de escolha de componentes que continham vértices que induziam variáveis com valores fracionários, sendo escolhidos preferencialmente as componentes que compunham a clique maximal inicial C_{ini} do grafo. Adotando esta estratégia foi visto que o número de vértices da clique inicial tendia a se expandir, fortalecendo o modelo dos representantes. Foi experimentado também uma escolha aleatória de componentes que induziam variáveis fracionárias, porém este critério tende a aumentar o esforço computacional.

Planos de Corte

Em um algoritmo de planos de corte, algumas decisões importantes devem ser tomadas afim de parametrizar o funcionamento do método. Questões como quantas iterações o algoritmo irá realizar e quantos cortes devem ser gerados por iteração podem determinar o desempenho do método. É importante encontrar um equilíbrio entre qualidade da solução e tempo computacional gasto. Logo, com o objetivo de balancear o algoritmo de planos de corte, os seguintes parâmetros são utilizados:

- A geração do número de cortes do algoritmo esta limitada por um parâmetro que finaliza o método de planos de corte sempre que a função objetivo do modelo não alcança um incremento Inc num determinado número P de passos do algoritmo. Um balanceamento proveitoso entre o número de cortes gerados e seu ganho foi considerado na parametrização destes fatores. Após testes realizados, foram utilizados utilizado os valores $Inc = 0.1$ e $P = n/20$ durante o decorrer do trabalho.

- Foi estipulado um tempo limite de 10 minutos para o processo de planos de corte que engloba o tempo gasto pelos métodos de separação junto com o tempo de solução dos problemas lineares. Após este tempo limite o algoritmo de planos de cortes é finalizado. Esta estratégia tende a balancear o tempo gasto entre a estratégia de ramificação e o método de planos de corte no algoritmo *branch and cut*.

- Um corte é inserido no modelo dos representantes somente se a restrição estiver sendo violada por um parâmetro mínimo. Este parâmetro é iniciado com valor igual a 0.05 e cortes que estejam sendo violados por este valor serão inseridos no modelo. Quando não houver mais cortes violados, este parâmetro é decrescido iterativamente até o valor de 0.02. Estes valores foram alcançados empiricamente e seu objetivo é diminuir o número total de cortes do modelo, inserindo primeiramente cortes teoricamente mais fortes.

5.2 Sumário de Resultados

Primeiro analisamos o desempenho do método *branch and cut* junto a um grupo de grafos aleatoriamente gerados com 20 a 120 vértices. Cada componente destes grafos possui exatamente 2 vértices e a probabilidade de ocorrer uma aresta entre quaisquer dois vértices é de 0.5. Cinco instâncias foram geradas para cada possibilidade de número de vértices do conjunto $\{20, 40, 60, ..., 120\}$. O método *branch and cut* foi executado três vezes para cada instância com diferentes sementes de entrada para o gerador de números pseudo-aleatórios. O método finaliza seu processo quando a solução ótima é encontrada ou após o tempo limite de 2 horas. As médias dos limites inferiores e superiores em relação ao número de cores para as 15 execuções de cada classe de grafo estão ilustradas na Figura 5.1. Notamos que o método *branch and cut* foi capaz de encontrar a solução ótima de todas as instâncias até 80 vértices. Para as instâncias maiores, a complexidade do problema cresce e o método não consegue chegar a otimalidade no período limite de 2 horas, porém a distância entre os limites superior e inferior é de exatamente uma cor apenas.

Num segundo experimento, aplicamos o método *branch and cut* sobre um conjunto de grafos de 90 vértices distribuidos sobre 45 componentes de dois vértices

Figura 5.1: Média dos limites inferiores e superiores para o número de cores ao variar o número de vértices dos grafos.

cada. Neste conjunto de instâncias a densidade de arestas varia entre 0.1 e 0.9. Novamente cinco instâncias são geradas para cada possibilidade de densidade de arestas. O método *branch and cut* é executado três vezes para cada instância com diferentes sementes de entrada para o gerador de números pseudo-aleatórios. Como no primeiro experimento, as médias dos limites inferiores e superiores em relação ao número de cores para as 15 execuções de cada classe de grafo estão ilustradas na Figura 5.2. Notamos que a distância entre as médias dos limitantes nunca ultrapassa de uma cor. Esta distância permanece estável até os grafos de densidade 0.5 quando começa a diminuir até quase ser nula nas instâncias de densidade 0.6.

O número de instâncias em que a solução foi encontrada para cada densidade de aresta é apresentada na Tabela 5.1. Para densidades de arestas de valores pequenos, o método consegue solucionar alguns grafos devido a pequena complexidade destas instâncias. Neste ponto, a formulação encontra dificuldades devido a ausência de cliques no grafo. As instâncias mais difíceis são aquelas com densidade de aresta entre 0.3 e 0.5, isto se deve ao fato de que para problemas de coloração em grafo, as instâncias de densidade de aresta mediana tendem a conter subestruturas que dificultam a solução do problema. Vemos que esta característica é herdada pelo PCP. O número de instâncias resolvidas cresce a partir da densidade de aresta 0.6

Figura 5.2: Média dos limites inferiores e superiores para o número de cores ao variar o número de vértices dos grafos.

pois estes grafos tendem a possuir muitas cliques de tamanho grande que fortificam o valor da relaxação linear da formulação matemática.

Densidade de Aresta	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Instâncias Solucionadas	5	3	—	1	—	12	15	15	15

Tabela 5.1: Número de instâncias solucionadas dentro do tempo limite para diferentes densidades de arestas.

No próximo grupo de experimentos, foram gerados grafos de 90 vértices e densidade de aresta igual a 0.5. Desta vez, o número de vértices em cada componente é o fator variante entre 1,2,4,5 e 10 vértices por componente. Novamente vemos que as médias dos limites inferiores e superiores em relação ao número de cores estão ilustradas na Figura 5.3.

Podemos observar que quanto maior é o tamanho da componente no grafo, mais fácil a instância se torna de ser solucionada. Esta característica vem do fato de que a dificuldade do problema aumenta diretamente em relação ao número de componentes do grafo e não em relação ao número de vértices. Percebemos que instâncias com 1 ou 2 vértices por componente se tornam difíceis de resolver devido a sua proximidade com o problema clássico de coloração, enquanto que instâncias onde o número de

Figura 5.3: Média dos limites inferiores e superiores para o número de cores ao variar o tamanho de cada componente.

vértices por componente é grande se tornam mais fáceis.

Uma maneira de mensurar a qualidade de um algoritmo de planos de cortes é observar o aumento do limitante inferior produzido pela relaxação linear do modelo quando adicionamos cortes a ele. Uma relaxação linear forte geralmente esta relacionada a um conjunto de restrições que definem cortes profundos no politopo relaxado. A Tabela 5.2 apresenta a contribuição dos cortes externos e dos cortes internos no valor da relaxação linear do modelo de representantes para PCP nos grupos de instâncias testadas nos dois primeiros experimentos onde variamos o número de vértices e a densidade de arestas. As primeiras três colunas fornecem os valores do número de vértices, número de arestas e do número de componentes de cada grupo das cinco instâncias geradas aleatoriamente. A média da relaxação linear é apresentada na quarta coluna. As três últimas colunas apresentam o valor da média da melhora relativa da relaxação linear do modelo de representantes adicionados (i) os cortes externos, (ii) os cortes internos e (iii) os cortes externos e internos, respectivamente. A tabela mostra que a boa performance do algoritmo de planos de cortes tem como causa principal os cortes externos. Observamos que os cortes externos melhoraram o valor da relaxação linear em 32.92% na média, enquanto que ao incluirmos também os cortes internos vemos que a melhoria tem um acréscimo

Vértices	Arestas	Comps.	Relaxação	Melhora Relativa (%)		
			Linear	Cortes Externos	Cortes Internos	Cortes Ext. e Int.
20	0.5	2	2.33	1.87	0.00	1.87
40	0.5	2	2.83	18.62	0.00	18.40
60	0.5	2	3.21	31.97	0.00	32.37
70	0.5	2	3.41	35.47	0.00	35.50
80	0.5	2	3.56	40.94	0.30	40.96
90	0.5	2	3.68	44.51	0.00	44.54
100	0.5	2	3.76	46.80	0.00	46.80
120	0.5	2	3.94	52.96	0.00	52.97
90	0.1	2	1.28	27.95	0.00	27.97
90	0.2	2	1.67	35.81	0.00	35.89
90	0.3	2	2.18	43.08	0.50	43.08
90	0.4	2	2.80	45.28	0.00	45.32
90	0.5	2	3.68	44.51	0.00	44.54
90	0.6	2	4.92	40.83	0.00	40.84
90	0.7	2	6.66	32.78	0.00	32.79
90	0.8	2	9.65	15.68	2.00	15.72
90	0.9	2	14.77	0.78	0.00	0.78
			Média	32.93	0.155	32.96

Tabela 5.2: Desempenho do algoritmo de planos de cortes.

de apenas 0.03%. Visto que os cortes externos representam a força principal do algoritmo de planos de corte, executamos a mesma bateria de testes comparando os valores e os ganhos relativos da relaxação linear adicionada a (i) os cortes cliques externos, (ii) os buracos externos e (iii) os cortes cliques e buracos externos. Os resultados apresentados na Tabela 5.3 deixam claro que a família de cortes cliques externos representam o núcleo do algoritmo de planos de cortes.

Agora, no próximo experimento, instâncias PCP são geradas a partir de instâncias clássicas do problema de coloração através do particionamento do grafo colocando cada vértice em uma componente diferente. Neste caso, o problema PCP se reduz ao problema clássico de coloração e o método é comparado com o algoritmo *branch and*

Vértices	Arestas	Comps.	Relaxação	Melhora Relativa (%)		
			Linear	Cliques Ext.	Buracos Ext.	Cliques e Buracos Ext.
20	0.5	2	2.33	1.87	0.00	1.87
40	0.5	2	2.83	13.09	9.54	18.62
60	0.5	2	3.21	28.97	15.62	31.97
70	0.5	2	3.41	35.43	12.90	35.47
80	0.5	2	3.56	40.89	17.13	40.94
90	0.5	2	3.68	44.51	16.57	44.51
100	0.5	2	3.76	46.80	16.75	46.80
120	0.5	2	3.94	52.95	16.24	52.96
90	0.1	2	1.28	25.02	18.11	27.95
90	0.2	2	1.67	35.6	17.46	35.81
90	0.3	2	2.18	43.6	18.34	43.08
90	0.4	2	2.80	45.29	17.14	45.28
90	0.5	2	3.68	44.51	16.57	44.51
90	0.6	2	4.92	40.71	15.27	40.83
90	0.7	2	6.66	32.65	10.67	32.78
90	0.8	2	9.65	14.18	2.79	15.68
90	0.9	2	14.77	0.78	0.00	0.78
			Média	30.35	13.01	32.93

Tabela 5.3: Desempenho do algoritmo de planos de cortes.

cut proposto por Méndez e Zabala em [44]. Os resultados obtidos para o conjunto das instâncias mais difíceis apresentadas em [44] são apresentadas na Tabela 5.4 (uma descrição das instâncias pode ser visto na Seção 10.1) . Para cada instância, as primeiras três colunas ilustram o nome, o número de vértices e o número de arestas do grafo. As três colunas seguintes apresentam o limite superior, o limite inferior e o intervalo relativo entre os limitantes do método. As três últimas colunas apresentam a mesma informação para o algoritmo de Zabala, após um tempo limite de 2 horas de processamento em uma estação Sun Ultra.

Podemos verificar que o método para PCP obteve resultados melhores que o algoritmo descrito em [44] nas instâncias que continham cliques de grande cardinalidade (instâncias DSJC) e obteve resultados piores nas instâncias em que a clique maximal possui uma cardinalidade baixa (instâncias Mycielsky e algumas Insertions). O motivo deste comportamento vem do fato que os cortes externos cliques são responsáveis diretos pela força do método de planos de cortes do nosso modelo.

Em nossos 2 últimos experimentos, apresentaremos os resultados computacionais para instâncias PCP que foram geradas a partir de um problema de roteamento e atribuição de comprimentos de ondas (PRACO) [2]. Primeiro, iremos considerar topologias de anéis citadas em [45]. Três topologias de anéis com 10, 15 e 20 nós são analisadas onde todas as conexões nesta rede são bi-direcionais e os caminhos óticos não são simétricos [29], i.e., o caminho ótico estabelecido do nó i para o nó j pode ser diferente do caminho ótico estabelecido do nó j para o nó i. Inicialmente foram geradas instâncias de PRACO a partir das redes de anéis onde o caminho ótico entre o nó i para o nó j deve ser estabelecido com uma probabilidade variando entre os valores 0.1, 0.2, ... 1.0. Cinco instâncias PRACO foram geradas para cada probabilidade. Cada instância PRACO gerada foi transformada em uma instância PCP $G = (V, E)$. Neste processo foi gerado um vértice $v \in V$ para cada possível rota de um caminho ótico e uma aresta $e \in E$ entre cada par de vértices onde as rotas associadas a estes compartilham um elo na rede de anéis. Vemos que no caso de redes de anéis, cada caminho ótico possui apenas duas alternativas de rotas (sentido horário e anti-horário). Por fim todos os vértices associados com o mesmo caminho ótico são colocados na mesma componente da partição. Esta transformação garante que a solução ótima para o problema PCP corresponde a solução ótima para

			Branch and cut para PCP			*Branch and cut* em [44]		
Grafo	Vértices	Arestas	L_S	L_I	Intervalo(%)	L_S	L_I	Intervalo(%)
DSJC125_5	125	3891	17	14	21.4	20	13	53.8
DSJC125_9	125	6961	43	43	0.0	47	42	11.9
DSJC250_1	250	3218	9	5	80.0	9	5	80.0
DSJC250_5	250	15668	29	15	93.3	36	13	176.9
DSJC250_9	250	27897	72	71	1.4	88	47	87.2
DSJR500_1c	500	121275	85	85	0.0	88	47	87.2
queen.9_9	81	2112	10	9	11.1	10	9	11.1
myciel6	95	755	7	4	75.0	7	5	40.0
myciel7	191	2360	8	4	100.0	8	5	60.0
1-Insertions-5	202	1227	6	3	100.0	6	4	50.0
1-Insertions-6	607	6337	7	3	133.3	6	4	50.0
2-Insertions-4	149	541	5	3	66.7	5	4	25.0
2-Insertions-5	597	3936	6	3	100.0	6	3	100.0
3-Insertions-4	281	1046	5	3	66.7	5	3	66.7
4-Insertions-3	79	156	4	3	33.3	4	3	33.3
4-Insertions-4	475	1795	5	3	66.7	5	3	66.7
1-FullIns-5	282	3247	6	4	50.0	6	4	50.0
2-FullIns-4	212	1621	6	5	20.0	6	5	20.0
2-FullIns-5	852	12201	7	5	40.0	7	5	40.0
3-FullIns-4	405	3524	7	6	16.7	7	6	16.7
3-FullIns-5	2030	33751	8	6	33.3	8	6	33.3
4-FullIns-4	690	6650	8	7	14.3	8	7	14.3
		Média			51.06			53.37

Tabela 5.4: Instâncias clássicas de coloração

o respectivo problema PRACO.

O método *branch and cut* é então aplicado para cada uma das 150 instâncias do PCP com um tempo limite de duas horas. Os resultados computacionais são apresentados na Tabela 5.5. As primeiras duas colunas apresentam o número de nós na rede de anéis e a probabilidade de conexão de um caminho ótico entre os nós. As três colunas seguintes ilustram a média do número de vértices, arestas e componentes das cinco instâncias PCP correspondentes. As colunas seguintes mostram o número médio e máximo de nós analisados na árvore de busca do *branch and cut*, o intervalo absoluto médio e máximo entre os limitantes superiores e inferiores ($\lceil LS - LI \rceil$, onde LS e LI representam o melhor limite superior e inferior respectivamente), o intervalo relativo médio e máximo dos limitantes ($(UB - LB)/LB$) e o número de instâncias resolvidas. O termo "-" indica que a instância não foi resolvida no tempo limite.

Podemos ver pela Tabela 5.5 que todas as instâncias de 10 e 15 nós e 40 das 50 instâncias baseadas em anéis de tamanho 20 foram resolvidas. Além disto, o intervalo entre os limitantes inferiores e superiores das instâncias não solucionadas dentro do tempo limite não é maior do que 6%.

No último experimento, analisamos a topologia de uma rede ótica real chamada NSFnet que contém 14 nós e 21 conexões, amplamente utilizada em experimentos na literatura. Esta rede é ilustrada pela Figura 5.4. Assim como nas redes em anéis, todas as conexões nesta rede são bi-direcionais e os caminhos óticos não são simétricos. Logo vemos que existem até $14 \cdot 13 = 182$ caminhos óticos a serem estabelecidos.

Novamente foram geradas instâncias de PRACO a partir da instância NSFnet onde o caminho ótico entre o nó i para o nó j deve ser estabelecido com uma probabilidade variando entre os valores 0.1, 0.2, ... 1.0. Cinco instâncias PRACO foram geradas para cada probabilidade. Cada instância PRACO foi transformada em uma instância PCP através do procedimento `2-EDR` proposto em [29]. Este procedimento primeiro calcula duas possíveis rotas para cada caminho ótico. Então, uma instância PCP é construida com um vértice para cada possível rota e uma aresta entre cada par de vértices onde as rotas associadas a estes compartilham um elo na rede ótica. Todos os vértices associados com o mesmo caminho ótico são colocados na mesma

Anel Nós	Prob.	Vértices média	Arestas média	Comps. média	B&C nodes média (max.)	Intervalo média (max.)	% Intervalo avg. (max.)	Resol.
10	0.1	17.6	63.6	8.8	1(1)	–(–)	–(–)	5
10	0.2	35.6	242.8	17.8	1.2(2)	–(–)	–(–)	5
10	0.3	49.6	477.2	24.8	1(1)	–(–)	–(–)	5
10	0.4	67.6	886.2	33.8	1(1)	–(–)	–(–)	5
10	0.5	83.2	1342.6	41.6	1(1)	–(–)	–(–)	5
10	0.6	102.4	2024	51.2	1(1)	–(–)	–(–)	5
10	0.7	127.2	3154	63.6	1(1)	–(–)	–(–)	5
10	0.8	145.2	4119.2	72.6	1(1)	–(–)	–(–)	5
10	0.9	164.4	5297.4	82.2	1(1)	–(–)	–(–)	5
10	1.0	180	6360	90	1(1)	–(–)	–(–)	5
15	0.1	47.2	457.8	23.6	1(1)	–(–)	–(–)	5
15	0.2	92.8	1723.4	46.4	1(1)	–(–)	–(–)	5
15	0.3	136.4	3729.4	68.2	1(1)	–(–)	–(–)	5
15	0.4	178	6358.2	89	1.2(2)	–(–)	–(–)	5
15	0.5	215.6	9316.8	107.8	1(1)	–(–)	–(–)	5
15	0.6	255.6	13076.8	127.8	1.4(3)	–(–)	–(–)	5
15	0.7	303.2	18492.6	151.6	1(1)	–(–)	–(–)	5
15	0.8	339.6	23151.4	169.8	1(1)	–(–)	–(–)	5
15	0.9	381.6	29285.2	190.8	1.2(2)	–(–)	–(–)	5
15	1.0	420	35490	210	1(1)	–(–)	–(–)	5
20	0.1	78.4	1251	39.2	2.2(7)	–(–)	–(–)	5
20	0.2	155.2	4893.2	77.6	1.4(3)	–(–)	–(–)	5
20	0.3	232	10925.6	116	8(20)	0.4(1)	2(6)	3
20	0.4	308.4	19319.4	154.2	2.4(8)	0.2(1)	1(4)	4
20	0.5	383.2	29848.8	191.6	1(1)	–(–)	–(–)	5
20	0.6	454	41860.8	227	1.2(2)	–(–)	–(–)	5
20	0.7	540.8	59507.2	270.4	1(1)	–(–)	–(–)	5
20	0.8	612.8	76398	306.4	1(1)	–(–)	–(–)	5
20	0.9	689.2	96702.8	344.6	1(1)	0.4(1)	1(2)	3
20	1.0	760	117420	380	1(1)	1.4(2)	3(4)	0

Tabela 5.5: Resultados Computacionais para as instâncias PCP associadas com redes em anéis.

Prob.	Vértices média	Arestas média	Comps. média	B&C nós média (max.)	Intervalo média (max.)	% Intervalo média (max.)	Resol.
0.1	26.4	36.0	22.2	1 (1)	– (–)	– (–)	5
0.2	52.6	149.0	39.0	1 (1)	– (–)	– (–)	5
0.3	79.0	359.6	58.6	1 (1)	– (–)	– (–)	5
0.4	103.0	634.4	75.4	1 (1)	– (–)	– (–)	5
0.5	125.2	947.6	92.0	1 (1)	– (–)	– (–)	5
0.6	151.2	1410.4	109.6	24.2 (75)	0.4 (1)	4 (11)	3
0.7	181.0	2093.8	130.4	4.0 (16)	0.2 (1)	2 (10)	4
0.8	204.6	2638.2	146.6	8.6 (18)	0.6 (1)	5 (9)	2
0.9	224.8	3118.2	165.6	4.6 (8)	0.6 (1)	5 (8)	2
1.0	248.8	3885.6	182.0	2.2 (4)	0.4 (1)	3 (8)	3

Tabela 5.6: Resultados Computacionais para as instâncias PCP associadas com a rede NSFnet.

componente da partição. Ressaltamos que neste caso a solução ótima do problema PCP não é garantidamente ótima para o problema PRACO correspondente pois o método 2-`EDR` não gera todas as possíveis rotas de cada caminho ótico.

O método *branch and cut* é então aplicado para cada uma das 50 instâncias do PCP com um tempo limite de duas horas. Os resultados computacionais são apresentados na Tabela 5.6. A primeira coluna ilustra a probabilidade de existência de cada caminho ótico. As três próximas colunas informam a média do número de vértices, arestas e componentes das cinco instâncias de PCP. As colunas seguintes mostram o número médio e máximo de nós analisados na árvore de busca do *branch and cut*, o intervalo absoluto médio e máximo entre os limitantes superiores e inferiores ($\lceil LS - LI \rceil$, onde LS e LI representam o melhor limite superior e inferior respectivamente), o intervalo relativo médio e máximo dos limitantes ($(UB - LB)/LB$) e o número de instâncias resolvidas.

Vemos pela Tabela 5.6 que todas as instâncias de PCP com probabilidade de caminhos óticos até 0.5 foram solucionadas no nó raiz da árvore de busca. O fato de que a relaxação inicial do modelo assimétrico para PCP junto com a geração dos cortes descritos no trabalho alcançar o valor ótimo para o problema expressa a

força desta formulação de representantes. Para as instâncias de maior complexidade, onde a probabilidade de caminhos óticos é maior do que 0.6, vemos que o método *branch and cut* não conseguiu resolver algumas instâncias dentro do tempo limite de duas horas. Porém, ao observar os valores de intervalo absoluto máximo entre os limitantes destas instâncias, vemos que seu valor nunca passa de uma unidade (cor) e que seu intervalo relativo nunca é superior a 5%.

Figura 5.4: Topologia da NSFnet.

Capítulo 6

Conclusão

Apresentamos neste trabalho um método *branch and cut* para o problema de coloração particionada em grafos. Até o momento da divulgação deste trabalho [46] [47], este é o único algoritmo exato para PCP presente na literatura. No trabalho propomos uma formulação de programação inteira 0-1 baseada no modelo de representantes para o problema PCP; um estudo poliédrico do modelo com a descrição de algumas facetas do politopo; novas heurísticas de separação para o algoritmo de planos de cortes; uma nova abordagem de uma heurística primal para o problema PCP; e uma nova estratégia de ramificação para o problema.

Testes computacionais foram exaustivamente realizados sobre vários grafos aleatórios e sobre instâncias PCP originárias do problema de roteamento e atribuição de comprimentos de ondas (PRACO). Os resultados demonstraram a eficiência e força da abordagem. O método *branch and cut* foi capaz de resolver instâncias de grande porte e quando o método não obteve êxito em solucionar o problema dentro do tempo limite, a diferença entre os limitantes nunca ultrapassaram 6%.

Uma possível extensão deste trabalho seria investigar como outras formulações matemáticas para o problema de coloração se comportariam se fossem extendidas para o problema de PCP. Além disto, a criação de um método exato para PCP abre caminhos para futuras aplicações do problema em outras áreas de problemas de redes óticas ou fora dela.

Parte II

Um Algoritmo Branch and Cut para o Problema de Coloração Equilibrada

Capítulo 7

Introdução

Seja $G = (V, E)$ um grafo não direcionado onde V é definido como o conjunto de vértices e E o conjunto de arestas. Uma *coloração equilibrada* de G é uma partição do conjunto de vértices V em uma família de conjuntos independentes disjuntos onde a diferença entre a cardinalidade de dois cojuntos independentes é de no máximo 1. Cada conjunto independente é associado com uma cor diferente. Denotamos por *classe de cor* o conjunto de vértices do grafo coloridos com uma mesma cor em uma solução viável do problema de coloração equilibrada. O *problema de coloração equilibrada* (PCE) consiste em encontrar uma coloração equilibrada de G com um número mínimo de cores. Este número é denominado $\chi_{eq}(G)$.

O fato do PCE ser um problema de coloração com restricões adicionais torna o problema difícil de ser resolvido. Grafos que possuem colorações equilibradas com k cores podem não possuir uma solução viável com $(k + 1)$ cores. Por exemplo, um grafo bipartite $K_{3,3}$ possui uma coloração equilibrada com 2 cores porém é inviável colorir este grafo com 3 cores de forma equilibrada (Figura 7.1).

Figura 7.1: Grafo $K_{3,3}$ colorido equilibradamente com 2 cores.

O problema de coloração equilibrada (PCE) foi primeiro apresentado em [48]. A motivação para seu estudo veio de um problema de coleta de lixo municipal

definido em [49]. Neste problema, a cidade é mapeada em um grafo onde os vértices representam rotas de coletas de lixo, e existe uma aresta entre dois vértices se as rotas de lixo correspondentes possuem intersecção nas áreas da cidade atendida (Figura 7.2). O problema é decidir quais rotas devem ser executadas a cada dia da semana de maneira que uma área da cidade não seja atendida mais de uma vez por dia e que o número de rotas executadas por dia seja aproximadamente igual. Este problema se reduz a um problema de uma 6-coloração equilibrada.

Figura 7.2: Mapeamento das rotas de lixo em um grafo

Outra área de aplicação para o PCE é a construção de planilhas de alocação escolares (*school timetable*) onde se procura maximizar o uso de salas de aula. Este outro problema pode ser formulado como um PCE de maneira que as cores correspondem a horários de aula, os vértices do grafo correspondem às disciplinas a serem ministradas e existe uma aresta entre dois vértices se as disciplinas correspondentes não podem ser lecionadas no mesmo horário. A solução deste problema irá fornecer uma partição dos vértices em conjuntos de cardinalidade semelhante com o objetivo de maximizar o espaço de salas de aula.

Um outro exemplo para uma aplicação mais recente do uso de coloração equilibrada é o problema de distribuição de dados em sistemas de memória paralela [50]. O problema ocorre quando sistemas que possuem multiprocessadores acessam bancos de memória de forma paralela. Nestes sistemas é necessário um esquema de distribuição de dados eficiente que minimize os conflitos de memória e garanta uma distribuição balanceada de dados. Um conflito de memória ocorre quando dois pro-

cessadores tentam acessar um mesmo módulo de memória simultaneamente. Uma distribuição de dados balanceada garante que cada módulo de memória armazena quantidades aproximadamente iguais de dados. O problema de distribuição de dados pode ser modelado como um PCE de forma que os dados de uma determinada aplicação são representados por um grafo onde dois vértices são adjacentes se os respectivos dados não podem ser acessados simultaneamente. Neste problema cada cor representa um módulo de memória e o objetivo é encontrar uma partição onde não exista conflitos de memória. Além disso, uma partição é dita balanceada se os vértices do grafo estão distribuidos de forma balanceada entre os módulos de memória, e dita perfeitamente balanceada se a carga de quaisquer dois módulos difere de apenas uma unidade.

O problema clássico de coloração em grafos pode ser reduzido a um problema de PCE. Logo, o problema de coloração equilibrada é um problema da classe *NP-Difícil* [51]. Na literatura, algoritmos polinomiais são conhecidos apenas para classes específicas de grafos como árvores [52] e grafos *split*s [53]. Porém vimos que muitas aplicações que podem ser modeladas como um PCE (problemas de *scheduling* [49] [54] e problemas de distribuição de dados [50]) nescessitam de soluções para o problema geral. Estas aplicações estimulam a busca por soluções eficientes para o problema.

A seguir apresentamos um algoritmo *branch and cut* exato para o problema de coloração equilibrada baseada numa especificação da formulação dos representantes [4] [5]. No Capítulo 8 apresentamos formulações de programação inteira para o PCE assim como alguns resultados teóricos. O Capítulo 9 descreve as principais partes do algoritmo *branch and cut* proposto. Uma descrição dos resultados computacionais é apresentada no Capítulo 10 enquanto que o Capítulo 11 realiza uma análise conclusiva do trabalho.

Capítulo 8

Coloração Equilibrada em Grafos

No Capítulo 3, nos dedicamos ao estudo do modelo para o problema de coloração particionada, assim como um estudo parcial de seu poliedro. No presente capítulo, descreveremos o problema de coloração equilibrada e alguns resultados teóricos iniciais em relação a propriedades estruturais do problema na Seção 8.1. Nas Seções 8.2, 8.3 e 8.4 abordaremos modelos de programação inteira para o problema baseados em formulações existentes para o problema de coloração em vértices. Para o modelo baseado em representantes, serão apresentadas famílias de desigualdades válidas para a formulação.

8.1 Problema de Coloração Equilibrada em Grafos

Dada uma coloração C para um grafo G, definimos por $c(i)$ o número de vértices coloridos com a cor i. Uma coloração de G é dita equilibrada se para quaisquer duas cores i e j temos que $|c(i) - c(j)| \leq 1$. O PCE pode ser agora formalmente definido como o problema de encontrar uma coloração equilibrada de G com $\chi_{eq}(G)$ cores.

Um dos principais resultados para coloração equilibrada foi enunciado por Hajnal-Szemerédi [55]. Este teorema estabelece um limite superior para o valor de $\chi_{eq}(G)$.

Teorema 8 (Hajnal-Szemerédi [55]) *Seja* $\Delta_G = MAX_{v \in V}(\mid N(v) \mid)$ *o* grau máximo *do grafo* G. *Todo grafo* G *possui uma coloração equilibrada com* $(\Delta_G + 1)$ *cores.*

Como visto anteriormente, o Teorema 8 estabelece um limite superior para o valor de $\chi_{eq}(G)$. Além disto, podemos facilmente notar que o valor 2 é um limitante

inferior para $\chi_{eq}(G)$ em qualquer grafo conexo G. Estes dois fatos nos levam ao seguinte resultado:

Corolário 3 *Seja $G = (V, E)$ um grafo conexo. A Equação 8.1 é válida para G.*

$$2 \leq \chi_{eq}(G) \leq \Delta_G + 1. \tag{8.1}$$

O lema a seguir foi desenvolvido com o objetivo de estabelecer limites para a cardinalidade máxima e mínima das classes de cores em uma coloração equilibrada do grafo G.

Lema 1 *Se C é uma coloração equilibrada do grafo $G = (V, E)$ com k cores, então a cardinalidade máxima de uma classe de cor em C é $\overline{z} = \lceil \frac{n}{k} \rceil$, e a cardinalidade mínima de uma classe de cor é $\underline{z} = \lfloor \frac{n}{k} \rfloor$, onde $n =\mid V \mid$.*

Prova:

Seja $\overline{\alpha}$ o número de classes de cores que tenham cardinalidade $\overline{z}$. Como C é uma coloração equilibrada que utiliza k cores, as cardinalidades das classes de cores em C diferem de no máximo uma unidade. Logo a equação a seguir é válida.

$$n = \overline{\alpha} \cdot \overline{z} + (k - \overline{\alpha}) \cdot \underline{z} \tag{8.2}$$

Caso k divida n, todas as classes de cores na coloração C terão a mesma cardinalidade $\overline{z} = \underline{z}$. Analisando a Equação 8.2, temos que

$$\begin{aligned} n &= \overline{\alpha} \cdot \overline{z} + (k - \overline{\alpha}) \cdot \overline{z} \\ n &= \overline{\alpha} \cdot \overline{z} + k \cdot \overline{z} - \overline{\alpha} \cdot \overline{z} \\ n &= k \cdot \overline{z} \end{aligned}$$

este fato é válido se $\overline{z} = \lceil \frac{n}{k} \rceil = \lfloor \frac{n}{k} \rfloor = \underline{z}$. Caso contrário, primeiro mostraremos que $\overline{z} = \lceil \frac{n}{k} \rceil$ e depois que $\underline{z} = \lfloor \frac{n}{k} \rfloor$.

Como C é uma coloração equilibrada e k não divide n, a seguinte equação é válida.

$$\overline{z} - \underline{z} = 1 \tag{8.3}$$

Vamos assumir que $\overline{z} \neq \lceil \frac{n}{k} \rceil$. Então temos dois casos a considerar: (i) $\overline{z} > \lceil \frac{n}{k} \rceil$ ou (ii) $\overline{z} < \lceil \frac{n}{k} \rceil$. No primeiro caso temos $\overline{\alpha}$ classes de cores de cardinalidade $\overline{z} > \lceil \frac{n}{k} \rceil$ e

$k - \overline{\alpha}$ classes de cores de cardinalidade $\underline{z} = \overline{z} - 1 \geq \lceil \frac{n}{k} \rceil$. Pela Equação 8.2 temos que

$$n = \overline{\alpha} \cdot \overline{z} + (k - \overline{\alpha}) \cdot \underline{z} > \overline{\alpha} \cdot \left\lceil \frac{n}{k} \right\rceil + (k - \overline{\alpha}) \cdot \left\lceil \frac{n}{k} \right\rceil = k \cdot \left\lceil \frac{n}{k} \right\rceil > n.$$

Por contradição temos que, $\overline{z} \not> \lceil \frac{n}{k} \rceil$. No segundo caso temos $\overline{\alpha}$ classes de cores com cardinalidade $\overline{z} < \lceil \frac{n}{k} \rceil$ e $k - \overline{\alpha}$ classes de cores com cardinalidade $\underline{z} = \overline{z} - 1 < \lfloor \frac{n}{k} \rfloor$. Novamente, a partir da Equação 8.2 temos que

$$n = \overline{\alpha} \cdot \overline{z} + (k - \overline{\alpha}) \cdot \underline{z} < \overline{\alpha} \cdot \left\lfloor \frac{n}{k} \right\rfloor + (k - \overline{\alpha}) \cdot \left\lfloor \frac{n}{k} \right\rfloor = k \cdot \left\lfloor \frac{n}{k} \right\rfloor < n.$$

Por contradição verificamos que $\overline{z} \not< \lceil \frac{n}{k} \rceil$. Logo,

$$\overline{z} = \left\lceil \frac{n}{k} \right\rceil.$$

A partir deste resultado e da Equação 8.3 temos que

$$\underline{z} = \overline{z} - 1 = \left\lceil \frac{n}{k} \right\rceil - 1 = \left\lfloor \frac{n}{k} \right\rfloor,$$

o que finaliza a prova. □

Com o resultado do lema anterior podemos agora estabelecer limites para a cardinalidade das classes de cores de uma coloração equilibrada.

Teorema 9 *Seja C uma coloração equilibrada do grafo $G = (V, E)$ com k cores e seja LS_k e LI_k um limite superior e inferior para k, respectivamente. Temos que $\left\lceil \frac{n}{LS_k} \right\rceil$ é um limite inferior e $\left\lceil \frac{n}{LI_k} \right\rceil$ é um limite superior para a cardinalidade máxima de uma classe de cor em C, respectivamente.*

Prova:

Seja $\overline{z}$ a cardinalidade máxima de uma classe de cor em C. A partir do Lema 1 temos que

$$\overline{z} = \left\lceil \frac{n}{k} \right\rceil.$$

Como

$$\left\lceil \frac{n}{LS_k} \right\rceil \leq \left\lceil \frac{n}{k} \right\rceil \leq \left\lceil \frac{n}{LI_k} \right\rceil,$$

então a Equação 8.4 é válida.

$$\left\lceil \frac{n}{LS_k} \right\rceil \leq \overline{z} \leq \left\lceil \frac{n}{LI_k} \right\rceil. \tag{8.4}$$

□

O objetivo do teorema anterior é analisar a relação que existe entre os limitantes superiores e inferiores de uma coloração equilibrada de um grafo G em relação a cardinalidade de suas classes de cores. No corolário a seguir, o Teorema 9 é especificado em termos de uma coloração mínima.

Corolário 4 *Uma coloração equilibrada do grafo $G = (V, E)$ com $\chi_{eq}(G)$ cores satisfaz a Equação 8.5.*

$$\left\lceil \frac{n}{LS_{\chi_{eq}(G)}} \right\rceil \leq \overline{z} \leq \left\lceil \frac{n}{LI_{\chi_{eq}(G)}} \right\rceil, \tag{8.5}$$

onde $\overline{z}$ é a cardinalidade máxima de uma classe de cor numa coloração equilibrada com $\chi_{eq}(G)$ cores.

8.2 Formulação de Atribuição de Cores PCE

A primeira formulação de programação inteira para o PCE foi desenvolvida recentemente por Bahiense et al. [56]. Nesta formulação, existem três tipos de variáveis. O primeiro tipo são as varivéis binárias x_{vc} para todo $v \in V$ e $c \in \{1, ..., \Delta_G + 1\}$ com a seguinte interpretação.

$$x_{vc} = \begin{cases} 1 & \text{caso a cor } c \text{ for atribuida ao vértice } v \\ 0 & \text{caso contrário} \end{cases}$$

Sabemos pelo Teorema 8 que $\Delta_G + 1$ cores são suficientes para colorir equilibradamente um grafo G. Logo, com o objetivo de minimizar o número de cores usadas nesta formulação, o segundo tipo de variáveis é definido como as variáveis binárias w_c, para $c \in \{1, ..., \Delta_G + 1\}$ onde:

$$w_c = \begin{cases} 1 & \text{caso } x_{vc} = 1 \text{ para algum } v \in V \\ 0 & \text{caso contrário} \end{cases}$$

Uma coloração equilibrada pode ser definida como uma coloração de vértices com uma restrição de equilíbrio entre as classes de cores. Com o objetivo de formular esta condição, uma classe de variáveis se faz necessária para refletir a cardinalidade

de cada classe de cor da coloração equilibrada. Logo, o terceiro tipo de variáveis denominadas y_c é definido a seguir:

$$y_c = \sum_{v \in V} x_{vc}, \text{ para todo } c \in \{1, ..., \Delta_G + 1\}$$

A formulação de atribuição de cores para o PCE é apresentada a seguir:

$$FAT = \min \sum_{c=1}^{\Delta_G+1} w_c \tag{8.6}$$

sujeito a:

$$\sum_{c=1}^{\Delta_G+1} x_{vc} = 1 \quad \forall u \in V \tag{8.7}$$

$$x_{vc} + x_{uc} \leq w_c \quad \forall (v,u) \in E, \text{ e } c \in \{1, ..., \Delta_G + 1\} \tag{8.8}$$

$$y_c = \sum_{v \in V} x_{vc} \quad \forall c \in \{1, ..., \Delta_G + 1\} \tag{8.9}$$

$$w_c - w_{c+1} \geq 0 \quad \forall c \in \{1, ..., \Delta_G\} \tag{8.10}$$

$$| \, y_c - y_l \, | \leq 1 \quad \forall c, l \in \{1, ..., \Delta_G + 1\} \tag{8.11}$$

$$x_{vc} \in \{0,1\} \quad \forall v \in V, \quad \forall c \in \{1, ..., \Delta_G + 1\} \text{ e } w_c \in \{0,1\} \quad \forall c \in \{1, ..., \Delta_G + 1\} \tag{8.12}$$

$$y_c \in \mathbb{Z}^+ \quad \forall c \in \{1, ..., \Delta_G + 1\}. \tag{8.13}$$

As restrições 8.7 garantem que cada vértice do grafo receberá uma única cor enquanto que as restrições 8.8 asseguram que os pares adjacentes de vértices recebem cores distintas. As variáveis do tipo y_c indicam a cardinalidade de cada classe de cor c através das inequações 8.9. As inequações 8.10 tem como objetivo eliminar em parte a simetria do modelo induzindo que uma cor $c + 1$ só possa ser atribuida a algum vértice se a cor c já tiver sido atribuida. Desta maneira, se tivermos uma coloração equilibrada que utiliza k cores, o modelo irá considerar permutações entre as primeiras k cores em seu conjunto de soluções viáveis, porém eliminará permutações com cores de índices maiores que k. Finalizando, as restrições 8.11 garantem que a cardinalidade de duas classes de cores diferem de no máximo uma unidade.

Notemos que o modelo FAT é não linear pela presença do módulo nas inequações 8.11. Com o objetivo de linearizar o modelo, as inequações 8.11 são substituidas pelas restrições a seguir:

$$y_c - y_l \leq 1 + M(2 - w_c - w_l) \quad \forall c, l \in \{1, ..., \Delta_G + 1\} \text{ onde } c < l,$$

$$y_c - y_l \geq -1 - M(2 - w_c - w_l) \quad \forall c, l \in \{1, ..., \Delta_G + 1\} \text{ onde } c < l,$$

onde M é grande o suficiente para forçar $y_c - y_l \leq 1$ somente quando w_c e w_l forem diferentes de zero.

O modelo de programação obtido pela substituição das restrições 8.11 pelas restrições lineares citadas, denominado de FAT_l, possui $(\mid V \mid +2)(\Delta_G+1)$ variáveis e $\mid V \mid +(\mid E \mid +4)(\Delta_G + 1)$ restrições e é definido a seguir.

$$FAT_l = \min \sum_{c=1}^{\Delta_G+1} w_c \tag{8.14}$$

sujeito a:

Inequações (8.7)-(8.10)

$$y_c - y_l \leq 1 + M(2 - w_c - w_l) \quad \forall c, l \in \{1, ..., \Delta_G + 1\} \text{ onde } c < l, \tag{8.15}$$

$$y_c - y_l \geq -1 - M(2 - w_c - w_l) \quad \forall c, l \in \{1, ..., \Delta_G + 1\} \text{ onde } c < l, \tag{8.16}$$

$$x_{vc} \in \{0, 1\} \quad \forall v \in V, \quad \forall c \in \{1, ..., \Delta_G + 1\} \text{ e } w_c \in \{0, 1\} \quad \forall c \in \{1, ..., \Delta_G + 1\} \tag{8.17}$$

$$y_c \in \mathbb{Z}^+ \quad \forall c \in \{1, ..., \Delta_G + 1\}. \tag{8.18}$$

O fato de este modelo para PCE ser baseado no modelo clássico para coloração em vértices de um grafo incentivou o desenvolvimento de uma formulação baseada no modelo de representantes [4, 5]. Na próxima seção apresentaremos a versão simétrica da formulação e logo a seguir duas versões assimétricas para o problema.

8.3 Formulação dos Representantes Simétrica para PCE

A formulação de representantes para PCE é baseada na seguinte idéia: Ao invéz de associar vértices a cores, escolhemos um vértice para representar todos os vértices de uma mesma cor. Logo, cada vértice do grafo pode ter um destes dois estados: (i) ou ele representa sua própria cor (ii) ou existe algum outro vértice que o está

representando. A formulação por representantes tem sido utilizada com sucesso em outros tipos de problemas de coloração como [4, 5, 47].

Definimos $A(u) = \{w \in V : (u, w) \notin E, w \neq u\}$ como a *anti-vizinhança* do vértice u (i.e., o subconjunto de vértices que não são adjacentes a u). Definimos também $A'(u) = A(u) \cup \{u\}$. Dado um subconjunto de vértices $V' \subseteq V$, denotaremos por $E[V']$ o subconjunto de arestas induzidas em $G = (V, E)$ por V'. Um vértice $v \in A(u)$ é dito *isolado* em $A(u)$ se $E[A(u)] = E[A(u) \setminus \{v\}]$ (i.e., o vértice v não possui vizinhos em $A(u)$). Definimos as variáveis binárias x_{uv} para cada $u \in V$ e para cada $v \in A'(u)$, tal que $x_{uv} = 1$ se e somente se o vértice u representa a cor do vértice v; caso contrário $x_{uv} = 0$. Definimos também a variável de equilíbrio $Z \in \mathbb{R}$, onde Z indica a cardinalidade da maior classe de cor na coloração equilibrada, i.e., a cardinalidade de cada classe de cor é Z ou $Z - 1$. Definimos também Z_{inf} e Z_{sup} como os limitantes inferior e superior para o valor de Z, respectivamente.

O modelo para PCE pode ser formulado como um problema de programação matemática definido a seguir:

$$FSR = \min \sum_{u \in V} x_{uu} \tag{8.19}$$

sujeito a:

$$\sum_{v \in A'(u)} x_{vu} = 1 \quad \forall u \in V \tag{8.20}$$

$$x_{uv} + x_{uw} \leq x_{uu} \quad \forall u \in V, \quad \forall (v, w) \in E \text{ com } v, w \in A(u) \tag{8.21}$$

$$x_{uv} \leq x_{uu} \quad \forall u \in V, \quad \forall v \in A(u) \text{ tal que } v \text{ é isolado em } A(u) \tag{8.22}$$

$$x_{uu} + \sum_{v \in A(u)} x_{uv} \leq Z x_{uu}, \quad \forall u \in V \tag{8.23}$$

$$x_{uu} + \sum_{v \in A(u)} x_{uv} \geq (Z - 1) x_{uu}, \quad \forall u \in V \tag{8.24}$$

$$x_{uv} \in \{0, 1\} \quad \forall u \in V, \quad \forall v \in A'_P(u) \text{ e } Z \in [Z_{inf} ... Z_{sup}]. \tag{8.25}$$

O modelo acima é denominado de *formulação por representantes para coloração equilibrada*. A função objetivo (8.19) contabiliza o número de vértices representantes, i.e., o número de cores. As restrições (8.20) garantem que cada vértice $u \in V$ precisa ser representado por si mesmo ou por outro vértice v contido na sua anti-vizinhança. As inequações (8.21) forçam que vértices adjacentes tenham representantes diferentes. As restrições (8.21) junto com as restrições (8.22) garantem

que um vértice apenas pode ser representado por um vértice representante. As inequações (8.23) e (8.24), denominadas *restrições de equilíbrio*, garantem o equilíbrio entre as classes de cores.

Novamente iremos aplicar a técnica descrita em [5] com a finalidade de quebrar a simetria do modelo e construir uma formulação assimétrica dos representantes para o problema PCE.

8.4 Formulação dos Representantes Assimétrica para PCE

Com o objetivo de quebrar a simetria da formulação FSR, adaptamos o modelo de PCE para a formulação assimétrica de representantes descrita em [5]. Estabelecemos uma ordem $\prec$ entre os vértices e definimos que um vértice v apenas pode ser representado por um vértice u se $u \prec v$ (i.e. se u for anterior a v na ordem $\prec$). Logo, o vértice representante de uma classe de cor será o vértice de menor índice. Com a utilização desta ordem de representação entre os vértices conseguimos eliminar soluções simétricas em que uma mesma classe de cor poderia possuir diferentes representantes.

Definimos $A_{\succ}(u) = \{v \in A(u) : u \prec v\}$ como a *anti-vizinhança maior* do vértice $u \in V$ (i.e. o conjunto de vértices que u pode representar) e $A_{\prec}(u) = \{v \in A(u) : v \prec u\}$ como a *anti-vizinhança menor* do vértice u (i.e. os vértices que podem representar o vértice u). Definimos também $A'_{\succ}(u) = A_{\succ}(u) \cup \{u\}$ e $A'_{\prec}(u) = A_{\prec}(u) \cup \{u\}$. Definimos $V^s = \{u \in V : A_{\prec}(u) = \emptyset\}$ como o conjunto de *vértices fontes* que não possuem vizinhança menor em G (i.e. o conjunto de vértices que sempre serão representantes). Como os vértices $v \in V^s$ são sempre representantes, $x_{vv} = 1$ em qualquer solução viável de PCE. Logo, estas variáveis são removidas da formulação assimétrica e a restrição (8.19) pode ser reescrita como a restrição (8.26).

Baseado nestas propriedades, as restrições (8.20)-(8.25) podem ser reescritas também como as restrições (8.27)-(8.30), respectivamente.

$$FAR = \min \sum_{v \in V \backslash V^s} x_{vv} + \mid V^s \mid \tag{8.26}$$

sujeito a:

$$\sum_{v \in A'_{\prec}(u)} x_{vu} = 1 \quad \forall u \in V \backslash V^s \tag{8.27}$$

$$x_{uv} + x_{uw} \leq \beta_u \quad \forall u \in V, \quad \forall (v,w) \in E \text{ com } v, w \in A_{\succ}(u) \tag{8.28}$$

$$x_{uv} \leq x_{uu} \quad \forall u \in V \backslash V^s, \quad \forall v \in A_{\succ}(u) \text{ tal que } v \text{ é isolado em } A_{\succ}(u) \tag{8.29}$$

$$\beta_u + \sum_{v \in A_{\succ}(u)} x_{uv} \leq Z\beta_u, \quad \forall u \in V \tag{8.30}$$

$$\beta_u + \sum_{v \in A_{\succ}(u)} x_{uv} \geq (Z-1)\beta_u, \quad \forall u \in V \tag{8.31}$$

$$x_{uv} \in \{0,1\} \quad \forall u \in V, \quad \forall v \in A'_{\succ}(u), \text{ e } Z \in [Z_{inf} ... Z_{sup}] \tag{8.32}$$

onde $\beta_u = 1$ se $u \in V^s$; caso contrário $\beta_u = x_{uu}$.

Podemos perceber que o modelo FAR é não linear por causa das restrições de equilíbrio (8.30) e (8.31). Com o objetivo de linearizar esta formulação, as inequações de equilíbrio são reescritas como as inequações (8.34) e (8.35). Uma possível formulação linear para o problema PCE é apresentada a seguir.

$$FAR_1 = \min \sum_{v \in V \backslash V^s} x_{vv} + \mid V^s \mid \tag{8.33}$$

sujeito a:

Inequações (8.27)-(8.29)

$$\beta_u + \sum_{v \in A_{\succ}(u)} x_{uv} \leq Z - Z_{inf}(1 - \beta_u), \quad \forall u \in V \tag{8.34}$$

$$\beta_u + \sum_{v \in A_{\succ}(u)} x_{uv} \geq (Z-1) - (Z_{sup} - 1)(1 - \beta_u), \quad \forall u \in V \tag{8.35}$$

$$x_{uv} \in \{0,1\} \quad \forall u \in V, \quad \forall v \in A'_{\succ}(u), \text{ e } Z \in [Z_{inf} ... Z_{sup}], \tag{8.36}$$

onde $\beta_u = 1$ se $u \in V^s$; caso contrário $\beta_u = x_{uu}$.

No modelo FAR_1 utilizamos os limitantes da variável Z para substituir a multiplicação por variáveis por uma aproximação linear. O número de variáveis deste modelo é de $\mid V \backslash V^s \mid + \overline{m} + 1$.

Uma outra maneira de linearizar as restrições (8.30) e (8.31) é através da criação de novas variáveis y_i, para todo $i \in \{Z_{inf}, ..., Z_{sup}\}$, onde $y_i = 1$ se a maior cardinalidade de uma classe de cor for igual a i, e $y_i = 0$ caso contrário. Logo, podemos substituir a variável Z pelo somatório a seguir.

$$Z = \sum_{i=Z_{inf}...Z_{sup}} i.y_i \tag{8.37}$$

Além disto, precisamos garantir que apenas uma das variáveis y tenha o valor igual a 1. Isto é alcançado com a seguinte equação.

$$\sum_{i=Z_{inf}...Z_{sup}} y_i = 1 \tag{8.38}$$

Apenas a substituição da variável Z pelas variáveis y não torna o problema linear, porém notemos agora que as multiplicações são entre variáveis binárias do tipo y e x. Com o objetivo de linearizar esta multiplicação, apresentamos as variáveis $z_{ui} = \beta_{uu} \cdot y_i$, para todo $u \in V$ e $i \in \{Z_{inf}...Z_{sup}\}$. Esta condição das variáveis z_{ui} pode ser formulada pelas seguintes inequações lineares.

$$z_{ui} \leq y_i, \quad \forall u \in V \text{ e } i = Z_{inf}...Z_{sup} \tag{8.39}$$

$$z_{ui} \leq \beta_u, \quad \forall u \in V \text{ e } i = Z_{inf}...Z_{sup} \tag{8.40}$$

$$z_{ui} \geq y_i + \beta_u - 1, \quad \forall u \in V \text{ e } i = Z_{inf}...Z_{sup} \tag{8.41}$$

Agora, reescrevemos as Inequações (8.30) como:

$$\begin{aligned}
\beta_u + \sum_{v \in A_\succ(u)} x_{uv} &\leq Z\beta_u, \quad \forall u \in V \\
\beta_u + \sum_{v \in A_\succ(u)} x_{uv} &\leq \sum_{i=Z_{inf}...Z_{sup}} i.y_i.\beta_u, \quad \forall u \in V \\
\beta_u + \sum_{v \in A_\succ(u)} x_{uv} &\leq \sum_{i=Z_{inf}...Z_{sup}} i.z_{u,i}, \quad \forall u \in V
\end{aligned}$$

Aplicamos agora a mesma substituição na Inequação (8.31):

$$
\begin{aligned}
\beta_u + \sum_{v \in A_\succ(u)} x_{uv} &\geq (Z-1)\beta_u, \quad \forall u \in V \\
\beta_u + \sum_{v \in A_\succ(u)} x_{uv} &\geq Z\beta_u - \beta_u, \quad \forall u \in V \\
\beta_u + \sum_{v \in A_\succ(u)} x_{uv} &\geq \sum_{i=Z_{inf}...Z_{sup}} i.y_i.\beta_u - \beta_u, \quad \forall u \in V \\
2\beta_u + \sum_{v \in A_\succ(u)} x_{uv} &\geq \sum_{i=Z_{inf}...Z_{sup}} i.y_i.\beta_u, \quad \forall u \in V \\
2\beta_u + \sum_{v \in A_\succ(u)} x_{uv} &\geq \sum_{i=Z_{inf}...Z_{sup}} i.z_{ui}, \quad \forall u \in V
\end{aligned}
$$

Aplicando estas alterações no modelo FAR, o problema de coloração equilibrada pode ser reformulado como:

$$
FAR_2 = \min \sum_{v \in V \setminus V^s} x_{vv} + \mid V^s \mid \tag{8.42}
$$

sujeito a:

Inequações (8.27)-(8.29)

$$
\beta_u + \sum_{v \in A_\succ(u)} x_{uv} \leq \sum_{i=Z_{inf}...Z_{sup}} i.z_{ui}, \quad \forall u \in V \tag{8.43}
$$

$$
2\beta_u + \sum_{v \in A_\succ(u)} x_{uv} \geq \sum_{i=Z_{inf}...Z_{sup}} i.z_{ui}, \quad \forall u \in V \tag{8.44}
$$

Inequações (8.38)-(8.41)

$$
x_{uv} \in \{0,1\} \quad \forall u \in V, \quad \forall v \in A'_\succ(u) \tag{8.45}
$$

$$
y_i \in [0,1], \text{ e } z_{ui} \in \mathbb{R} \quad \forall u \in V, \text{ com } i = Z_{inf}...Z_{sup} \tag{8.46}
$$

onde $\beta_u = 1$ se $u \in V^s$; caso contrário $\beta_u = x_{uu}$.

No modelo FAR_2 os valores possíveis que Z pode assumir são discretizados através da inclusão das variáveis y e de um novo conjunto de restrições. O número inicial de variáveis do modelo FAR_2 é de $|V \setminus V^s| + \overline{m} + \Omega(1 + |V|)$, onde $\Omega = Z_{sup} - Z_{inf}$. Porém vemos que $z_{ui} = y_i$ para todo $u \in V^s$ e ao eliminarmos estas variáveis alcançamos o número de $|V \setminus V^s| + \overline{m} + \Omega(1 + |V \setminus V^s|)$ variáveis do modelo FAR_2.

8.4.1 Inequações Válidas

Novamente foi percebido que os limitantes das relaxações lineares do modelo de representantes para o PCE é fraco. Logo, utilizamos os cortes desenvolvidos por Campêlo et al. [4, 5] para fortalecer a relaxação linear da formulação. O problema de coloração equilibrada é um caso particular do problema de coloração em grafos e conseqüentemente o politopo formado pelo conjunto de soluções viáveis para o PCE está contido dentro do politopo de coloração. Logo, podemos averiguar que as inequações descritas em [5, 4] são também válidas para as formulações FAR_1 e FAR_2.

A família de cortes internos previamente apresentada pelo Teorema 7 é reescrita na inequação 8.47 para os modelos FAR_1 e FAR_2.

$$\sum_{v \in H \setminus V^s} x_{vv} + \mid H \cap V^s \mid + \sum_{v \in H \setminus V^s, w \in A_{\prec}(v) \setminus H} x_{wv} \geq \chi(G[H]). \tag{8.47}$$

Teorema 10 (Campêlo et al. [5]) *Se $H \subseteq V$ induz um buraco ou um anti-buraco ímpar em G, então a inequação (8.47) é válida para LF_1 e LF_2.*

A família de cortes externos baseados em estruturas clique está representada no modelo para PCE através do Teorema 11.

Teorema 11 (Campêlo et al. [5]) *Para cada $u \in V$ e para cada clique $K \subseteq A_{\succ}(u)$,*

$$\sum_{v \in K} x_{uv} \leq \beta_u \tag{8.48}$$

é uma inequação válida para LF_1 e LF_2.

Utilizamos também uma segunda família de cortes externos baseados em estruturas buraco e anti-buraco ímpar apresentados pelo Teorema 12.

Teorema 12 (Campêlo et al. [5]) *Para cada $u \in V$ e um buraco ou anti-buraco ímpar $H \subseteq A_{\succ}(u)$,*

$$\sum_{v \in H} x_{uv} \leq \alpha_H \beta_u \tag{8.49}$$

é uma inequação válida para LF_1 e LF_2, onde α_H corresponde à cardinalidade de um conjunto independente máximo de H.

Capítulo 9

Um Método Branch and Cut para o Problema de Coloração Equilibrada

Como descrito na Seção 2.4, os modelos introduzidos no capítulo anterior podem ser usados dentro de um algoritmo *branch and cut* junto com as equações válidas apresentadas. Neste capítulo apresentamos os principais componentes do algoritmo de *branch and cut* para o problema de coloração equilibrada. Na Seção 9.1 apresentamos uma estratégia de ramificação para o problema. A seguir, a Seção 9.2 mostra uma breve nota sobre as rotinas de separação utilizadas. Finalizando, na Seção 9.3 exibimos algumas estratégias para a obtenção de limites primais para o problema de coloração equilibrada.

9.1 Estratégia de Ramificação

A estratégia de ramificação é de fundamental importância para o desempenho de um algoritmo *branch and cut*. Infelizmente as propriedades do problema de coloração equilibrada dificultam ramificações baseadas em estruturas de grafos como a descrita na Seção 4.2. Além disso, realizar a ramificação sobre as variáveis do tipo x_{uv}, para $u \in V \backslash V^s$ e $v \in A'_{\succ}(u)$, não é eficiente pois a maioria delas assume valores nulos em soluções inteiras. Logo, nossa estratégia de ramificação prioriza as variáveis de equilíbrio Z no modelo LF_1 e y_i no modelo LF_2. A ramificação sobre estas variáveis

tem a propriedade de gerar subproblemas onde o intervalo em que Z varia é menor, e quanto menor for este intervalo, menor o espaço de solução do problema. A estratégia de ramificação é descrita a seguir.

Caso Z tenha valor fracional na relaxação linear de LF_1, dois novos subproblemas serão gerados: No primeiro, adicionamos a restrição $Z \leq \lfloor Z^* \rfloor$, enquanto que no segundo adicionamos a restrição $Z \geq \lceil Z^* \rceil$, onde Z^* é o valor da variável Z na relaxação linear de LF_1. Caso exista alguma variável y_i fracional em LF_2, a ramificação será realizada sobre a variável com valor mais próximo a 0.5 na relaxação linear. Novamente dois subproblemas serão gerados: no primeiro a seguinte restrição será adicionada

$$\sum_{j=Z_{inf}}^{i-1} y_j = 0$$

e no segundo adicionamos a restrição

$$\sum_{j=i}^{Z_{sup}} y_j = 0.$$

Caso não exista variáveis de equilíbrio com valor fracional na relaxação linear dos modelos, a ramificação será realizada primeiramente sobre as variáveis x_{uu}, onde $u \in V \backslash V^s$, com valor da relaxação mais próximo de 0.5. Mais uma vez dois subproblemas serão gerados: no primeiro fixamos $x_{uu} = 0$ e no segundo $x_{uu} = 1$. Finalizando, caso não exista variáveis x_{uu} fracionais, ramificamos sobre as variáveis x_{uv}, onde $u \in V \backslash V^s$ and $v \in A_{\succ}(u)$ com o mesmo critério de seleção e ramificação descrito para as variáveis x_{uu}. Realizamos a ramificação sobre as variáveis x_{uu} primeiro pois a fixação destas variáveis tem a propriedade de fixar um maior número de variáveis x_{uv} nos modelos.

9.2 Identificação de Cortes

Vemos que os cortes para coloração equilibrada descritos nos Teoremos 10, 11 e 12 que utilizamos no algoritmo *branch and cut* se baseiam em estruturas clique, buracos e anti-buracos. Logo o problema de separação destas desigualdades consiste na identificação destas estruturas que estejam sendo violadas nas relaxações lineares associadas com cada nó da árvore de busca. Para solucionar estes problemas de separação utilizamos os algoritmos descritos na Seção 4.4. Basicamente os algoritmos

descritos anteriormente são generalizados para buscar estruturas clique, buracos e anti-buracos genéricos e não apenas particionados.

9.3 Métodos Primais

A presença de bons limites primais podem reduzir o tamanho da árvore de busca significantemente. Além disto, é possível que o algoritmo de *branch and cut* não finalize seu procedimento dentro de um tempo viável, por isso é importante que o método seja capaz de dar boas soluções no decorrer do processo. Outro fator importante e expresso pelo Corolário 4 é que bons limites primais para o problema de coloração equilibrada nos levam a limites inferiores da variável Z mais fortes, o que consequentemente nos levam a uma diminuição do espaço de solução a ser percorrido. A seguir apresentaremos alguns métodos de obtenção de soluções viáveis para PCE.

9.3.1 Um método Tabu para o problema PCE

Recentemente Touhami [57] apresentou um método Tabu adaptativo para o problema de alocação de freqüência em redes de celulares. Os bons resultados obtidos por Touhami [57] neste problema de coloração com restrições adicionais nos motivaram a desenvolver uma adaptação de seu método Tabu adaptativo para o problema PCE. A seguir apresentaremos as principais componentes do método.

9.3.1.1 Função objetivo

Dado um grafo $G = (V, E)$, definimos uma solução $(\hat{x}, \overline{z})$ para o problema PCE como o conjunto das variáveis binárias x_{uv} do modelo FAR, onde:

$$\hat{x} = \{x_{uv}, \forall u \in V \text{ e } v \in A'(u)\}$$

e $\overline{z}$ a cardinalidade máxima de uma classe de cor na solução viável.

Definimos também as funções $\Theta_{e1}(\hat{x}, \overline{z})$ e $\Theta_{e2}(\hat{x}, \overline{z})$ para descrever o grau de viabilidade da solução $\hat{x}$ com respeito às restrições de equilíbrio como:

$$\Theta_{e1}(\hat{x}, \overline{z}) = \sum_{u \in V} \quad \max([(\overline{z} - 1) - \sum_{v \in A'(u)} x_{uv}], \quad 0), \text{ e}$$

$$\Theta_{e2}(\hat{x}, \overline{z}) = \sum_{u \in V} \max([\sum_{v \in A'(u)} x_{uv} - \overline{z}], \quad 0).$$

Notemos que quando a solução não viola as restrições de equilíbrio temos que $\Theta_{e1}(\hat{x}, \overline{z}) = \Theta_{e2}(\hat{x}, \overline{z}) = 0$. Além disto, definimos uma outra função $\Theta_c(\hat{x}, \overline{z})$ para contabilizar o número de restrições de conflitos em arestas violadas pela solução $\hat{x}$, onde:

$$\Theta_c(\hat{x}, \overline{z}) = \sum_{u \in V} \sum_{\substack{(v,w) \in E \\ v,w \in A_{\succ}(u)}} \max([x_{uv} + x_{uw} - 1], \quad 0)$$

De forma similar, caso a solução $\hat{x}$ não viole as restrições de arestas temos que $\Theta_c(\hat{x}, \overline{z}) = 0$. Finalizando, definimos a função objetivo global $\Theta_g(\hat{x}, \overline{z})$ do método como:

$$\Theta_g(\hat{x}, \overline{z}) = \overline{z} + \mu_e.(\Theta_{e1}(\hat{x}, \overline{z}) + \Theta_{e2}(\hat{x}, \overline{z})) + \mu_c.\Theta_c(\hat{x}, \overline{z}) \tag{9.1}$$

onde μ_e e μ_c representam coeficientes de penalidade associados à violação das restrições de equilíbrio e conflito em arestas respectivamente.

Vemos que o primeiro termo da função 9.1 representa o objetivo de minimizar o número de cores total utilizadas na solução $\hat{x}$ através da minimização de $\overline{z}$ enquanto que o segundo e terceiro termos descrevem a viabilidade da solução. A presença de fatores multiplicativos nas funções relativas à viabilidade possibilitarão ao método visitar soluções inviáveis durante a busca pela minimização da função objetivo $\Theta_g(\hat{x}, \overline{z})$.

9.3.1.2 Operador de busca

O espaço de solução para o problema PCE é muito vasto, especialmente quando consideramos soluções em que a viabilidade não é requerida, logo um operador de busca tem como objetivo identificar soluções interessantes para o método.

O espaço de busca é explorado através de alterações na representação dos vértices realizadas por movimentos a cada iteração. Um movimento consiste em trocar o representante de um determinado vértice $v \in V$ de u para w onde u e w pertencem à $A'(v)$. Denotaremos este movimento por $Troca(v : u \rightarrow w)$. Aplicando este movimento a uma solução $(\hat{x}, \overline{z})$, obteremos uma nova solução $(\hat{x}', \overline{z}')$ definido por $(\hat{x}', \overline{z}') = (\hat{x}, \overline{z}) + Troca(v : u \rightarrow w)$.

O operador de busca aplicado neste método procura identificar o melhor movimento para um determinado vértice $v \in V$ dentre um subconjunto de possíveis representantes. Definimos o operador $Rep(v)^p$ da seguinte forma: considere o vértice $v \in V$ que possui o representante $u \in A'(v)$ numa solução $(\hat{x}, \overline{z})$. O operador $Rep(v)^p$ identifica o movimento de troca de representante de v, dentre um subconjunto de $H \subseteq A'(v)$, que produz o menor valor da função objetivo $\Theta_g(\hat{x}, \overline{z})$. O subconjunto de representantes é definido através da análise de uma fração p das possíveis mudanças de representantes em H. Com esta parametrização, conseguimos manipular a intensidade do operador $Rep(v)^p$. Logo vemos que:

$$\Theta_g((\hat{x}, \overline{z}) + Rep(v)^p) = \Theta_g((\hat{x}, \overline{z}) + Troca(v : u \rightarrow w^*)) = \min_{\substack{w \in H \\ Rand(w) < p}} \Theta_g((\hat{x}, \overline{z}) + Troca(v : u \rightarrow w))$$

onde $Rand(w)$ é um número aleatoriamente gerado no intervalo $[0, 1]$.

O operador $Rep(v)^p$ definido acima é utilizado para a composição do principal operador de busca denominado MOV_q^p. Dada uma solução $(\hat{x}, \overline{z})$, este operador identifica o movimento $Rep(v)^p$, onde v pertence a um subconjunto de V, que novamente produz o menor valor da função objetivo $\Theta_g(\hat{x}, \overline{z})$. Similarmente ao operador $Rep(v)^p$, o subconjunto de vértices analisados será definido por uma probabilidade de busca q. Logo temos que:

$$\Theta_g((\hat{x}, \overline{z}) + MOV_q^p) = \Theta_g((\hat{x}, \overline{z}) + Rep(v^*)^p) = \min_{\substack{v \in V \\ Rand(v) < q}} \Theta_g((\hat{x}, \overline{z}) + Rep(v)^p)$$

onde $Rand(v)$ é um número aleatoriamente gerado no intervalo $[0, 1]$.

Denotamos por $N_q^p(\hat{x}, \overline{z})$ a vizinhança da solução $(\hat{x}, \overline{z})$ perante o operador MOV_q^p. Este operador seleciona a melhor troca $Troca(v : u \rightarrow w)$ na vizinhança da solução $N_q^p(\hat{x}, \overline{z})$ que não seja um movimento proibido (Tabu) pelo método.

9.3.1.3 Algoritmo do método Tabu adaptativo

O método Tabu adaptativo para PCE utiliza o operador MOV_q^p para caminhar pelo espaço de solução do problema, restrito a uma lista de movimentos proibitivos atualizados a cada iteração. A busca é lentamente intensificada no decorrer do procedimento através do aumento da vizinhança analisada (i.e., incremento dos parâmetros p e q).

Com o objetivo de alcançar uma convergência mais rápida, o método alterna entre dois modos: uma busca Tabu normal e uma busca local não proibitiva. Enquanto que o modo busca Tabu consegue diversificação nas soluções visitadas, o modo de busca local é uma fase de intensificação que percorre o espaço de solução através do operador MOV_q^p movendo-se apenas para soluções em que haja melhoria na função objetivo.

O procedimento da busca Tabu adaptativa $\texttt{TS}_{adap}\texttt{-PCE}$ apresentado pelo Algoritmo 2 segue a estrutura de um método de busca Tabu com a diferença da alternância de modos da busca Tabu normal para uma busca local. O algoritmo inicia seu procedimento na linha 2 construindo uma nova solução gulosa $(\hat{x}, \overline{z})$ usando uma atribuição de representantes (cores) diante uma ordem aleatória para os vértices do grafo. A melhor solução encontrada $(\hat{x}^*, \overline{z}^*)$ é inicializada na linha 3, assim como os contadores de iteração t, t^T e t^L que representam o número de iterações do método, do modo busca Tabu e do modo busca local respectivamente. O modo inicial de execução Tabu é estabelecido na linha 4. Este modo continua até que uma das duas seguintes condições seja satisfeita: (i) Uma nova solução viável é encontrada na linha 17 e o algoritmo alterna para o modo busca local com o objetivo de convergir rapidamente para um ótimo local ou (ii) foram realizadas T_{MAX}^T iterações consecutivas no modo Tabu sem nenhuma melhoria na melhor solução $(\hat{x}^*, \overline{z}^*)$, neste caso o algoritmo alterna para o modo busca local na linha 21. O valor de T_{MAX}^T é estabelicido de modo que a fase Tabu seja utilizada para escapar de um ótimo local alcançado durante a útima fase de busca local. Com esse objetivo o valor de T_{MAX}^T é igual a dez vezes o número de iterações da última fase de busca local. Ao entrar em modo de busca local o algoritmo inicia uma fase de intensificação da busca no espaço de solução e permanece neste estado até alcançar um número máximo de iterações sem melhoria T_{MAX}^L alternando novamente para o modo Tabu na linha 24. O número máximo de iterações sem melhoria no modo busca local é de $\mid V \mid$.

O método da busca Tabu adaptativa altera dinamicamente o tamanho da vizinhança analisada $N_q^p(\hat{x}, \overline{z})$ através da alteração dos parâmetros p e q. No início do método utilizamos os valores $p = 0.5$ e $q = 0.7$ para a fase do Tabu. Durante o decorrer do processo de busca, estes valores são lentamente ampliados, para intensificar a busca, até atingir os valores de $p = q = 1$. Sempre que o método alterna

1. **Algoritmo:** $\texttt{TS}_{adap}\texttt{-PCE}(G)$
2. Constroi solução inicial $(\hat{x}, \overline{z})$
3. $(\hat{x}^*, \overline{z}^*) \leftarrow (\hat{x}, \overline{z})$, $t \leftarrow 0$, $t^T \leftarrow 0$ e $t^L \leftarrow 0$;
4. Inicialize lista tabu e faça $modo \leftarrow Tabu$;
5.
6. **Enquanto** $t < T_{MAX}$ **Faça**
7. $(\hat{x}', \overline{z}') = (\hat{x}, \overline{z}) + MOV_q^p$;
8. Atualiza lista Tabu;
9.
10. **Se** $modo = Tabu$ **Então**
11. $(\hat{x}, \overline{z}) \leftarrow (\hat{x}', \overline{z}')$ e $t^T \leftarrow t^T + 1$;
12. **Senão**
13. $t^L \leftarrow t^L + 1$;
14. **Se** $\Theta_g(\hat{x}', \overline{z}') < \Theta_g(\hat{x}, \overline{z})$ **Então**
15. $(\hat{x}, \overline{z}) \leftarrow (\hat{x}', \overline{z}')$;
16.
17. **Se** $\Theta_{e1}(\hat{x}, \overline{z}) = \Theta_{e2}(\hat{x}, \overline{z}) = 0$ e $\Theta_c(\hat{x}, \overline{z}) = 0$ e $\Theta_g(\hat{x}, \overline{z}) < \Theta_g(\hat{x}^*, \overline{z}^*)$ **Então**
18. $(\hat{x}^*, \overline{z}^*) \leftarrow (\hat{x}, \overline{z})$;
19. $modo = Local$ e $t^L \leftarrow 0$;
20.
21. **Se** $modo = Tabu$ e $t^T > T_{MAX}^T$ **Então**
22. $modo = Local$ e $t^L \leftarrow 0$;
23.
24. **Se** $modo = Local$ e $t^L > T_{MAX}^L$ **Então**
25. $modo = Tabu$ e $t^T \leftarrow 0$;
26.
27. $t \leftarrow t + 1$;

Algoritmo 2: Busca Tabu adaptativa para PCE

para a fase de intensificação de busca local, os valores de p e q são atualizados para 1 com o objetivo de alcançar soluções viáveis mais facilmente.

Além dos parâmetros de vizinhança, os parâmetros da função objetivo são atualizados a cada iteração do método. O algoritmo atualiza os coeficientes de multiplicação μ_e e μ_c, assossiados as restrições de equilibrio e arestas respectivamente, de maneira que permita ao método buscar a viabilidade das soluções caminhando por espaços inviáveis. Uma vez atingida a viabilidade, o método relaxa as penalizações das restrições e busca minimizar o objetivo principal. No início do processo temos que $\mu_e = \mu_c = 2$ e a cada iteração, onde a solução corrente é definida por $(\hat{x}', \overline{z}')$, os multiplicadores são atualizados da seguinte forma:

- $\mu_e = \begin{cases} \mu_e(1.05) & \text{caso } \Theta_{e1}(\hat{x}', \overline{z}') > 0 \text{ ou } \Theta_{e2}(\hat{x}', \overline{z}') > 0 \\ \mu_e(0.97) & \text{caso contrário} \end{cases}$
- $\mu_c = \begin{cases} \mu_c(1.03) & \text{caso } \Theta_c(\hat{x}', \overline{z}') > 0 \\ \mu_c(0.97) & \text{caso contrário} \end{cases}$

Vemos que os parâmetros multiplicativos variam de acordo com a viabilidade das restrições. Além disto, a penalização das restrições não é simétrica. Podemos observar que a penalização para soluções que violam as restrições de equilíbrio aumenta rapidamente em relação a de conflito de cores. Desta forma podemos direcionar o método para soluções mais equilibradas e depois tentar focar na viabilidade das colorações.

Com respeito aos parâmetros Tabu, sempre que um movimento $Troca(v : u \rightarrow w)$ é realizado pelo método, a representação do vértice v pelo vértice u fica proibida (Tabu) no modo de busca Tabu por determinado número de iterações. Este número é dinâmico e proporcional a ocorrência do vértice u na representação do vértice v. Seu valor é determinado pela expressão:

$$\eta \frac{\psi(u, v)}{\psi(v)}$$

onde $\psi(u, v)$ é o número de vezes que o vértice u representou o vértice v, $\psi(v)$ é o número de vezes que o vértice v trocou de representante e η é uma constante que indica o tamanho da lista Tabu. Com esta estratégia tentamos priorizar os representantes que o método considera interessantes em detrimento daqueles pouco utilizados.

9.3.2 RINS

Vimos na Seção 9.3.1 como um método baseado em busca local pode ser aplicado ao problema de coloração equilibrada em grafos. Um outro algoritmo para calcular valores primais do problema PCE que utiliza conceitos de busca local e a estrutura do método *branch and bound* é descrito nesta seção.

Técnicas de busca local têm sido aplicadas com grande sucesso em uma grande quantidade de problemas de otimização combinatória. Dado uma solução viável para um problema de programação inteira, é natural pensar em como melhorar esta solução através de alguma técnica de busca local. Infelizmente a composição destes métodos necessita de noções de vizinhança, e estas noções são dependentes da estrutura particular do problema a ser solucionado sendo difícil extrair tal informação de um modelo de programação inteira. Entretanto Danna e Rothberg [58] apresentaram um método denominado RINS (*Relaxation Induced Neighborhood Search*) que pode ser visto como um método de busca em vizinhanças que utiliza as relaxações do modelo de programação inteira para fixar variáveis e assim definir suas vizinhanças. Esta idéia de unir busca local e programação inteira foi primeiramente utilizada numa heurística anterior denominada *Local Branching* desenvolvida por Fischetti e Lodi [59] porém com resultados inferiores. A seguir será descrito o funcionamento do método RINS.

Ao se explorar a árvore de busca do *branch-and-bound*, duas soluções estão sempre ao nosso alcance. A primeira é a melhor solução primal que é viável com respeito as restrições de integralidade mas que não temos garantia de otimalidade até que a última solução inteira seja encontrada. A segunda é a relaxação linear que é constantemente calculada a cada nó da árvore. Estas soluções geralmente não são inteiras mas seus valores objetivos são sempre melhores (menores) que os da melhor solução viável.

Vemos que cada uma destas soluções alcança um dos objetivos conflitantes do problema: integralidade e otimização da função objetivo. Enquanto algumas variáveis claramente assumem valores diferentes na melhor solução viável e na relaxação, é importante notar que algumas delas assumem os mesmos valores. O método RINS se baseia na intuição de que estas variáveis que possuem os mesmos valores formem uma solução parcial do problema que pode ser expandida para uma

solução completa que possa alcançar os objetivos de integralidade e de uma "boa" função objetivo. Então o método pode focar sua atenção nas variáveis que diferem seus valores entre as duas soluções que intuitivamente são aquelas que merecem mais atenção.

O Algoritmo 3 ilustra o método RINS que pode ser aplicado em qualquer nó da árvore de busca do *branch and bound*.

```
1.  Algoritmo: RINS
2.  Entrada: Problema de programação inteira PPI, a melhor solução primal x̂ e
3.           o valor da melhor solução primal U
4.  Saida: Solução primal x̂′ com valor objetivo melhor que U se existir
5.
6.    Fixa_Variaveis(PPI, x̂);
7.    Melhor_Limite(PPI, U);
8.    Soluciona(PPI);
9.    Se (Solucao(PPI) melhor U) Então
10.     Retorne Solucao(PPI)
11.   Senão
12.     Retorne Nulo
```

Algoritmo 3: Heurística RINS

O algoritmo inicialmente fixa as variáveis do modelo de programação inteira que possuem o mesmo valor que a melhor solução primal $\hat{x}$ através da função Fixa_Variaveis. A seguir a função Melhor_Limite na linha 7 atribui ao PPI o valor de poda U. Por fim, é acionado um método de otimização exato sobre o agora restrito problema de programação inteira na linha 8. Caso uma melhor solução primal seja encontrada, ela será retornada pela função.

É possível identificar no método RINS os conceitos que caracterizam uma heurística baseada no paradigma de busca local. Primeiro como já foi dito, o conceito de vizinhança de uma solução é alcançado através da fixação de variáveis em relação à melhor solução primal encontrada. Segundo, vemos que as relaxações lineares se alteram de um nó da árvore de busca para outro, assim é alcançado automaticamente uma diversificação na vizinhança das soluções. Por fim, uma intensificação

do método é conseguida através da solução do problema restrito.

O método RINS pode ser visto como um algoritmo híbrido [60] pois combina conceitos de programação inteira, árvores de busca e busca local. Entretanto, diferente de outros algoritmos híbridos, ele integra todos estes conceitos em uma única estrutura de otimização.

Capítulo 10

Resultados Computacionais

Neste capítulo discutimos os resultados obtidos pelos algoritmos *branch and cut* implementados a partir dos modelos FAR_1 e FAR_2. Na Seção 10.1 são descritas as instâncias do problema de coloração que utilizamos para testar os métodos. Na Seção 10.2 são apresentados os detalhes de implementação referentes ao método de solução. Finalizando, na Seção 10.3 apresentamos os resultados e as análises do método *branch and cut* para PCE.

10.1 Descrição das Instâncias

Para testar a robustez do método *branch and cut*, vários testes foram realizados sobre uma grande bateria de instâncias. Grande parte destas instâncias foram extraídas do conjunto de problemas de coloração de grafos de DIMACS [61]. A seguir, iremos descrever rapidamente algumas classes de grafos que foram utilizadas.

Grafos Aleatórios:

Grafos Geométricos Aleatórios: Os grafos **DSJCx_y** e **DSJRx_y** são formados por x vértices aleatoriamente colocados no espaço geométrico $[0,1]x[0,1]$, então são inseridas arestas entre quaisquer dois vértices que estejam a uma distância de $0.y$. Estes tipos de grafos podem ser usados para modelar aplicações de atribuição de freqüência para telefonia celular [62]. Outro tipo de grafo geométrico são os grafos **milesx**, porém estes grafos não são aleatórios. Os vértices representam cidades americanas e suas respectivas distâncias em território americano.

Grafos de Registros: Uma aplicação muito utilizada para coloração em grafos é o problema de alocação de registros compartilhados por uma compilação dado um código sequencial. Os vértices nestes grafos representam variáveis e dois vértices são conectados por uma aresta se e somente se as duas variáveis correspondentes a estes vértices utilizam o mesmo registro ao mesmo tempo em um fragmento de código. O número cromático da coloração clássica destes grafos indicam o número mínimo de registros necessários naquele fragmento de código. **mulsol.i.x** e **zeroin.i.x** são grafos de registros.

Grafos Rainha: Dado um tabuleiro de xadrez q por r, um grafo rainha **queen.q_-r** é um grafo com qr vértices cada um correspondendo a um quadrado do tabuleiro, e uma aresta existe se somente se conecta dois quadrados na mesma linha, coluna ou diagonal. Esta classe de grafos representa o problema de decidir a questão: Existe uma maneira de se colocar q conjuntos de r rainhas num tabuleiro q por r onde rainhas de um mesmo conjunto não possam se atacar? A resposta é positiva se somente se o grafo correspondente ao problema tiver número cromático $\chi(G)$ igual a r.

Grafos Jogos: Grafos que representam jogos da temporada de futebol americano universitário. Os vértices representam os times de futebol de cada universidade e dois vértices são conectados por uma aresta se os respectivos times se enfrentaram durante a temporada. Os grafos **gamesx** são grafos de jogos.

Grafos de Mycielski: Dado um grafo $G = (V, E)$ onde $n = \mid V \mid$, definimos a transformação de Mycielski [63] como o processo de obtenção do grafo MY_G de G onde

$$V(MY_G) = \{x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n, z\}.$$

O conjunto de arestas $E(MY_G)$ é formado pelas arestas $x_i x_j$ para todo $v_i v_j \in E(G)$, $x_i y_j$ para todo $v_i v_j \in E(G)$ e $y_i z$ para todo $i = \{1...n\}$. Sabemos que esta transformação não afeta o tamanho da maior clique do grafo e incrementa o número cromático $\chi(G)$ em uma unidade [64]. Definimos grafos de Mycielski, denotados por **mycielx**, como o conjunto de **x** transformações de Mycielski sucessivas, a partir de um grafo com dois vértices e uma aresta.

Esta classe de grafos parece ser difícil de se resolver pois não possui triângulos (a clique máxima é 2) mas possui número cromático igual a **x**+1. Os grafos **x-Insertion-y** e **x-FullIns-y** são generalizações dos grafos de Mycielski com o incremento de mais vértices para aumentar o tamanho do grafo mas não sua densidade.

Grafos Livro: Dado um determinado livro de literatura, um grafo é criado a partir das letras presentes no livro. Dois vértices são adjacentes no grafo se as letras aparecerem juntas em alguma parte do livro. Em nossos testes utilizaremos quatro destas instâncias baseadas em clássicos: Anna Karenina (**anna**), David Copperfield (**david**), a Ilíada de Homero (**homer**) e (**jean**).

Grafos Kneser: Dado um conjunto $C = \{1, 2, ..., q\}$, o grafo de Kneser $\mathbf{K}_{p,q}$ é o grafo com conjunto de vértices formado por todos os subconjuntos de C com p elementos e tal qual dois vértices são adjacentes quando os respectivos subconjuntos forem disjuntos.

Nas tabelas 10.1 e 10.2 são listados o número de vértices e arestas de cada instância DIMACS.

10.2 Experimentos Computacionais

Assim como os experimentos descritos na Seção 5.1, a estrutura do algoritmo *branch and cut* foi implementada na linguagem de programação *C ANSI* utilizando *XPRESS−MP* versão 2005-a para resolver os problemas de programação linear sobre uma estação AMD-Atlon 1.8 Ghz de velocidade e um Gbyte de memória RAM.

Ordenação de vértices

Como já visto nos modelos baseados em representantes, a ordem em que os vértices se encontram possui um grande impacto no desempenho do método. Similarmente ao problema de coloração particionada, consideramos a ordem onde os vértices de uma clique maximal C_{ini} são dispostos nas primeiras posições da ordem. No caso da coloração equilibrada, o tamanho da maior clique é um limitante inferior para o valor do número cromático equilibrado. Novamente para a determinação da clique inicial C_{ini} foi utilizada a heurísta GRASP descrita na Seção 4.4.1. As tabelas

Grafo	Tamanho		Grafo	Tamanho	
	n	m		n	m
mulsol.i.1	197	3925	**myciel3**	11	20
mulsol.i.2	188	3885	**myciel4**	23	71
mulsol.i.3	184	3916	**myciel5**	47	236
mulsol.i.4	185	3946	**myciel6**	95	755
mulsol.i.5	186	3973	**DSJC125_1**	125	736
miles750	128	2113	**DSJC125_5**	125	3891
miles1000	128	3216	**DSJC125_9**	125	6961
miles1500	128	5198	**DSJC250_1**	250	3218
zeroin.i.1	211	4100	**DSJC250_5**	250	15668
zeroin.i.2	211	3541	**DSJC250_9**	250	27897
zeroin.i.3	206	3540	**DSJR500_1**	500	3555
queen.6_6	36	290	**jean**	80	508
queen.7_7	49	476	**anna**	138	493
queen.8_8	64	728	**david**	87	406
queen.9_9	81	2112	**homer**	561	1629
queen.8_12	96	1368	**games120**	120	1276
queen.10_10	100	2940	$\mathbf{K}_{5,2}$	10	15
queen.11_11	121	3960	$\mathbf{K}_{7,2}$	21	105
queen.12_12	144	5192	$\mathbf{K}_{7,3}$	35	70
queen.13_13	169	6656	$\mathbf{K}_{9,4}$	126	315

Tabela 10.1: Instâncias de grafos

Grafo	Tamanho		Grafo	Tamanho	
	n	m		n	m
1-Insertions-4	67	232	**1-FullIns-3**	30	100
1-Insertions-5	202	1227	**1-FullIns-4**	93	593
2-Insertions-3	37	72	**2-FullIns-3**	52	201
2-Insertions-4	149	541	**2-FullIns-4**	212	1621
3-Insertions-3	56	110	**3-FullIns-3**	80	346
3-Insertions-4	281	1046	**4-FullIns-3**	114	541
4-Insertions-3	79	156	**5-FullIns-3**	154	792

Tabela 10.2: Instâncias de grafos

10.3 e 10.4 ilustram o valor da clique maximal encontrada assim como o seu tempo de processamento em segundos.

Grafo	clique	tempo	Grafo	clique	tempo
mulsol.i.1	49	1	**myciel3**	2	0
mulsol.i.2	31	0	**myciel4**	2	0
mulsol.i.3	31	1	**myciel5**	2	0
mulsol.i.4	31	1	**myciel6**	2	0
mulsol.i.5	31	1	**DSJC125_1**	4	0
miles750	29	0	**DSJC125_5**	9	0
miles1000	41	0	**DSJC125_9**	32	0
miles1500	73	2	**DSJC250_1**	4	0
zeroin.i.1	49	1	**DSJC250_5**	10	0
zeroin.i.2	30	1	**DSJC250_9**	34	2
zeroin.i.3	30	0	**DSJR500_1**	12	0
queen.6_6	6	0	**jean**	10	0
queen.7_7	7	0	**anna**	11	0
queen.8_8	8	0	**david**	11	0
queen.9_9	8	0	**homer**	13	0
queen.8_12	12	0	**games120**	9	0
queen.10_10	9	0	$\mathbf{K}_{5,2}$	2	0
queen.11_11	10	0	$\mathbf{K}_{7,2}$	3	0
queen.12_12	11	0	$\mathbf{K}_{7,3}$	2	0
queen.13_13	12	0	$\mathbf{K}_{9,4}$	2	0

Tabela 10.3: Clique maximal inicial

Limitantes Primais

No método Tabu adaptativo utilizamos os parâmetros $T_{MAX} = 500.\mid V \mid$ e $\eta = 100$. Estes valores foram ajustados empiricamente com o objetivo de alcançar rapidamente um bom limite superior para o PCE. Além disso, executamos também a heurística Tabu adaptativa nos nós da árvore de busca do método de *branch and cut* em que pelo menos 40% das variáveis x_{uv} do modelo tenham o valor da relaxação linear maior que 0.5. Neste caso, estas variáveis são utilizadas para a formação de uma solução parcial que servirá como solução inicial do método. Nestas demais

Grafo	clique	tempo	Grafo	clique	tempo
1-Insertions-4	2	0	**1-FullIns-3**	3	0
1-Insertions-5	2	0	**1-FullIns-4**	3	0
2-Insertions-3	2	0	**2-FullIns-3**	4	0
2-Insertions-4	2	0	**2-FullIns-4**	4	0
3-Insertions-3	2	0	**3-FullIns-3**	4	0
3-Insertions-4	2	0	**4-FullIns-3**	6	0
4-Insertions-3	2	0	**5-FullIns-3**	7	0

Tabela 10.4: Clique maximal inicial

execuções do método Tabu adaptativo utilizamos o parâmetro $T_{MAX} = 100. \mid V \mid$. As Tabelas 10.5 e 10.6 ilustram a comparação do limite superior $\Delta_G + 1$ fornecido pelo Teorema 8 e a execução inicial do método Tabu adaptativo, assim como o seu tempo de processamento em segundos.

Embora o método RINS possa ser evocado em cada nó da árvore de busca, a vizinhança de soluções induzida pelas relaxações lineares de nós consecutivos são bastante similares, logo parametrizamos sua aplicação a cada 100 nós do método.

Metodologia de Ramificação

Em nossa implementação do método para PCE, utilizamos novamente a estratégia de *best-search* para a escolha do nó da árvore de busca a ser verificado. Ponderamos os nós da árvore de busca pelo limite inferior fornecido pela relaxação linear dos modelos.

Planos de Corte

- A geração do número de cortes do algoritmo é novamente limitada por um conjunto de parâmetros que finaliza o método de planos de corte sempre que a função objetivo do modelo não alcança um incremento Inc num determinado número P de passos ou se atingir um tempo limite estipulado. Utilizamos os valores $Inc = 0.1$ e $P = n/20$ com o tempo limite de 10 minutos de processamento.

- Durante a realização do trabalho optamos por uma estratégia de geração de cortes onde primeiro geramos todos os cortes baseados em estruturas clique que estavam sendo violados por um valor mínimo de 0.05. Após este processo

Grafo	$\Delta_G + 1$	TS_{adap}	tempo	Grafo	$\Delta_G + 1$	TS_{adap}	tempo
mulsol.i.1	122	77	20	**myciel3**	6	4	0
mulsol.i.2	157	109	19	**myciel4**	12	5	0
mulsol.i.3	158	111	20	**myciel5**	24	7	1
mulsol.i.4	159	114	21	**myciel6**	48	10	4
mulsol.i.5	160	115	19	**DSJC125_1**	24	12	5
miles750	65	34	13	**DSJC125_5**	76	26	13
miles1000	87	52	13	**DSJC125_9**	121	72	13
miles1500	107	81	13	**DSJC250_1**	39	16	26
zeroin.i.1	112	76	22	**DSJC250_5**	148	140	26
zeroin.i.2	141	95	21	**DSJC250_9**	235	229	27
zeroin.i.3	141	97	23	**DSJR500_1**	26	12	51
queen.6_6	20	7	1	**jean**	37	10	6
queen.7_7	25	7	2	**anna**	72	13	14
queen.8_8	28	10	7	**david**	83	30	9
queen.9_9	33	11	9	**homer**	100	71	57
queen.8_12	33	12	10	**games120**	14	11	6
queen.10_10	36	14	11	$\mathbf{K}_{5,2}$	4	4	0
queen.11_11	41	14	13	$\mathbf{K}_{7,2}$	6	6	0
queen.12_12	44	17	15	$\mathbf{K}_{7,3}$	5	5	1
queen.13_13	49	17	17	$\mathbf{K}_{9,4}$	6	6	3

Tabela 10.5: Limite Superior Inicial

Grafo	$\Delta_G + 1$	TS_{adap}	tempo	Grafo	$\Delta_G + 1$	TS_{adap}	tempo
1-Insertions-4	23	9	1	**1-FullIns-3**	12	6	0
1-Insertions-5	68	15	21	**1-FullIns-4**	33	10	4
2-Insertions-3	10	6	0	**2-FullIns-3**	16	8	0
2-Insertions-4	38	12	8	**2-FullIns-4**	56	16	22
3-Insertions-3	12	8	0	**3-FullIns-3**	20	9	1
3-Insertions-4	57	17	29	**4-FullIns-3**	24	11	5
4-Insertions-3	14	9	1	**5-FullIns-3**	28	13	8

Tabela 10.6: Limite Superior Inicial

iniciamos a geração dos demais cortes, incluindo os de estrutura clique, que estivessem sendo violados por um valor de no mínimo 0.01. Com esta estratégia conseguimos diminuir o tempo de execução do algoritmo de planos de corte. Além disso, optamos por gerar os cortes internos baseados em buracos e anti-buracos apenas no nó raiz da árvore de busca. Isto se deve porque os limites inferiores das relaxações lineares dos modelos se mostraram pouco sensíveis aos cortes internos inseridos posteriormente.

- A cada iteração do método de planos de corte, o valor de Z_{sup} é atualizado sempre que existe um aumento do limite inferior de $\chi_{eq}(G)$ fornecido pela relaxação linear dos modelos. Com esta atualização dinâmica conseguimos uma convergência mais rápida do método.

10.3 Resultados Finais

Após as considerações sobre os detalhes de implementação, investigaremos agora a robusteza do código sobre as instâncias de DIMACS. Primeiro apresentamos nas Tabelas 10.7 e 10.8 uma comparação entre os valores das relaxações lineares dos 3 modelos de programação inteira apresentados no Capítulo 8. As seguintes informações são apresentadas a seguir:

- Grafo: nome da instância analisada.
- FAT_l: valor da relaxação linear do modelo de atribuição de cores.
- FAR_1 e FAR_1^c: valor da relaxação linear do modelo FAR_1 e valor da relaxação linear com a geração dos planos de corte respectivamente.
- FAR_2 e FAR_2^c: valor da relaxação linear do modelo FAR_2 e valor da relaxação linear com a geração dos planos de corte respectivamente.

Vemos que o modelo de atribuição de cores FAT_l possui uma relaxação linear com valor igual 2.00 para todas as instâncias testadas. Mesmo para instâncias mais densas, o modelo FAT_l não consegue vencer esta barreira. Os modelos FAR_1 e FAR_2 conseguiram valores de relaxação linear sempre iguais ou superiores ao modelo FAT_l e esta distância é ainda maior ao considerarmos os valores das relaxações

Grafo	FAT_l	FAR_1	FAR_1^c	FAR_2	FAR_2^c
mulsol.i.1	2.00	49.00	49.00	49.00	49.00
mulsol.i.2	2.00	33.39	34.00	33.39	34.00
mulsol.i.3	2.00	34.00	34.00	34.00	34.00
mulsol.i.4	2.00	34.00	34.00	34.00	34.00
mulsol.i.5	2.00	33.83	33.83	33.83	33.83
miles750	2.00	31.00	31.00	31.00	31.00
miles1000	2.00	42.00	42.00	42.00	42.00
miles1500	2.00	73.00	73.00	73.00	73.00
zeroin.i.1	2.00	49.00	49.00	49.00	49.00
zeroin.i.2	2.00	32.36	32.70	32.36	32.70
zeroin.i.3	2.00	33.32	33.32	33.32	33.32
queen.6_6	2.00	6.00	6.20	6.00	6.21
queen.7_7	2.00	7.00	7.00	7.00	7.00
queen.8_8	2.00	8.00	8.00	8.00	8.00
queen.9_9	2.00	9.00	9.00	9.00	9.00
queen.8_12	2.00	12.00	12.00	12.00	12.00
queen.10_10	2.00	10.00	10.00	10.00	10.00
queen.11_11	2.00	11.00	11.00	11.00	11.00
queen.12_12	2.00	12.00	12.00	12.00	12.00
queen.13_13	2.00	13.00	13.00	13.00	13.00
myciel3	2.00	2.75	3.00	2.75	3.00
myciel4	2.00	2.85	3.83	2.85	3.83
myciel5	2.00	2.86	3.92	2.86	3.92
myciel6	2.00	2.82	3.96	2.82	3.96
DSJC125_1	2.00	4.00	5.00	4.00	5.00
DSJC125_5	2.00	10.05	13.89	10.05	13.89
DSJC125_9	2.00	41.73	43.00	41.68	43.00
DSJC250_1	2.00	4.00	5.00	4.00	5.00
DSJC250_5	2.00	12.00	15.24	12.00	15.21
DSJC250_9	2.00	51.43	70.09	51.42	70.16
DSJR500_1	2.00	12.00	12.00	12.00	12.00

Tabela 10.7: Valor da relaxação inicial dos modelos

Grafo	FAT_l	FAR_1	FAR_1^c	FAR_2	FAR_2^c
jean	2.00	10.00	10.00	10.00	10.00
anna	2.00	11.00	11.00	11.00	11.00
david	2.00	29.33	29.33	29.33	29.33
homer	2.00	13.00	13.00	21.33	21.33
games120	2.00	9.00	9.00	9.00	9.00
$\mathbf{K}_{5,2}$	2.00	2.50	2.50	2.50	2.50
$\mathbf{K}_{7,2}$	2.00	3.50	3.50	3.50	3.50
$\mathbf{K}_{7,3}$	2.00	2.00	2.92	2.00	2.92
$\mathbf{K}_{9,4}$	2.00	2.00	3.00	2.00	3.00
1-Insertions-4	2.00	2.33	2.91	2.33	2.91
1-Insertions-5	2.00	2.33	2.97	2.33	2.97
2-Insertions-3	2.00	2.15	2.85	2.15	2.85
2-Insertions-4	2.00	2.03	2.98	2.03	2.98
3-Insertions-3	2.00	2.07	2.95	2.07	2.95
3-Insertions-4	2.00	2.00	2.99	2.00	2.99
4-Insertions-3	2.00	2.00	2.93	2.00	2.93
1-FullIns-3	2.00	3.20	3.75	3.20	3.75
1-FullIns-4	2.00	3.32	3.88	3.32	3.88
2-FullIns-3	2.00	4.00	4.73	4.00	4.73
2-FullIns-4	2.00	4.00	4.93	4.00	4.93
3-FullIns-3	2.00	5.00	5.71	5.00	5.71
4-FullIns-3	2.00	6.00	6.71	6.00	6.71
5-FullIns-3	2.00	7.00	7.70	7.00	7.70

Tabela 10.8: Valor da relaxação inicial dos modelos

lineares após a aplicação do método de planos de cortes. Outro fator importante a se considerar é a igualdade entre os valores das relaxações lineares dos modelos FAR_1 e FAR_2. Este fato deve-se a similaridade entre os dois modelos, que diferem entre si apenas em relação ao tratamento das restrições de equilíbrio. Esta diferença não foi notada nos valores das relaxações lineares nem nos valores das relaxações com geração de planos de cortes já que todos os cortes utilizados foram realizados sobre as restrições de coloração.

Podemos observar também que os modelos baseados em representantes conseguem uma grande melhoria do valor da relaxação linear através do método de planos de cortes sobre os problemas do tipo **DSJCx_y**. Os problemas desta classe possuem estruturas clique muito fortes que servem de base para a principal classe de cortes dos representantes expressos pelo Teorema 11. Por outro lado, instâncias do tipo **mycielx** e **x-Insertion-y** apresentam dificuldades em obter limitantes inferiores fortes para o problema. Isto ocorre devido ao fato de que estas instâncias possuem clique máxima de baixa cardinalidade, penalizando assim a família de cortes cliques do modelo de representantes.

Nas Tabelas 10.9 e 10.10 apresentamos os resultados da aplicação do método *branch-and-cut* sobre a bateria de instâncias descritas anteriormente. Foi utilizado um tempo limite de 2 horas para o processamento do algoritmo que envolve métodos primais, planos de cortes e ramificação. Nestas tabelas ilustramos uma comparação entre os modelos FAR_1 e FAR_2 apresentados originalmente neste trabalho. O modelo anterior de atribuição de cores FAT_l se mostrou inviável computacionalmente em resultados preliminares por não possuir uma família de cortes para fortificar suas relaxações lineares. As seguintes informações são apresentadas nas Tabelas 10.9 e 10.10:

- Grafo: nome da instância analisada.
- L_I: valor do limite inferior encontrado no final do processamento.
- L_S: valor do limite superior encontrado no final do processamento.
- B&C nós: número de nós analisado durante a execução do método de *branch-and-cut*.

- Tempo: tempo de processamento do método em segundos. O símbolo ″−″ nesta coluna indica que o algoritmo foi encerrado no tempo limite de 2 horas de processamento.

Analisando primeiramento os resultados dos limites superiores vemos que a nova heurística baseada no tabu adaptativo foi capaz de se aproximar da solução ótima do problema em diversas instâncias, porém vemos que em outras instâncias de grande porte como as do tipo **DSJCx_y** e algumas do tipo **x-Insertion-y** a solução primal encontrada está distante do limite inferior do problema. Duas características intrínsecas do PCE dificultam a criação de heurísticas para o problema: (i) o problema pode ter uma solução viável com k cores e pode não possuir uma solução com $k + 1$ cores e (ii) uma solução parcial para um subgrafo do problema pode não ser expansível para uma solução inteira do problema. Estas duas características específicas do PCE tendem a dificultar a abordagem heurística do problema.

Alguns valores de limites primais foram melhorados através de soluções inteiras encontradas no processo de busca ou pela aplicação do método RINS. Em ambos os casos isto ocorreu principalmente com o modelo FAR_2 que conseguiu resultados primais superiores ao modelo FAR_1 nas instâncias **zeroin.i.2**, **zeroin.i.3** e **DSJR500_1**. Apesar do modelo FAR_2 ter um número de varíaveis muito superior ao modelo FAR_1, a estratégia de ramificação adotada sobre as variáveis de equilíbrio favoreceu a uma maior integralização das variáveis do modelo FAR_2.

Por ser um modelo mais compacto, a formulação FAR_1 consegue solucionar a relaxação linear do problema em um tempo menor do que a formulação FAR_2. Este fato pode ser constatado na grande parte das instâncias em que ambos os modelos conseguem solucionar o problema dentro do tempo limite. Porém a medida que a estratégia de ramificação age sobre as instâncias, foi observado que o modelo FAR_2 tende a encontrar a solução mais rapidamente, seja por chegar mais rápido em limitantes inferiores fortes como na instância **queen.8_8** ou por alcançar limitantes superiores mais rapidamente como nas instâncias **x-FullIns-y**. Além disso, vemos que o modelo FAR_2 alcançou limites inferiores melhores que o modelo FAR_1 nas instâncias **zeroin.i.2**, **zeroin.i.3** e **homer** dentro do limite de tempo estipulado.

Apesar do método ter encontrado o valor ótimo do problema de várias instâncias, muitas outras permanecem em aberto. Algumas instâncias que podem ser facilmente

Grafo	FAR_1				FAR_2			
	L_I	L_S	B&C nós	Tempo	L_I	L_S	B&C nós	Tempo
mulsol.i.1	49	49	38	1143	49	49	62	1727
mulsol.i.2	34	56	120	-	34	57	118	-
mulsol.i.3	34	59	88	-	34	58	122	-
mulsol.i.4	34	63	123	-	34	60	121	-
mulsol.i.5	34	63	122	-	34	63	122	-
miles750	31	31	5	124	31	31	6	171
miles1000	42	42	10	214	42	42	13	267
miles1500	73	73	1	13	73	73	1	13
zeroin.i.1	49	49	3	79	49	49	1	50
zeroin.i.2	35	61	97	-	36	36	23	510
zeroin.i.3	35	61	101	-	36	36	28	491
queen.6_6	7	7	1	2	7	7	1	1
queen.7_7	7	7	1	3	7	7	1	0
queen.8_8	9	9	525	572	9	9	297	441
queen.9_9	9	11	634	-	9	10	5027	-
queen.8_12	12	12	1	10	12	12	1	11
queen.10_10	10	12	417	-	10	12	412	-
queen.11_11	11	13	260	-	11	14	78	-
queen.12_12	12	15	121	-	12	16	129	-
queen.13_13	13	17	53	-	13	16	55	-
myciel3	4	4	5	0	4	4	7	0
myciel4	5	5	255	4	5	5	237	5
myciel5	5	6	1407	-	5	6	982	-
myciel6	4	10	1422	-	4	10	26	-
DSJC125_1	5	12	216	-	5	12	184	-
DSJC125_5	14	22	141	-	14	22	113	-
DSJC125_9	44	45	301	-	44	45	294	-
DSJC250_1	5	16	13	-	5	16	7	-
DSJC250_5	16	139	5	-	16	138	4	-
DSJC250_9	71	221	74	-	71	221	75	-
DSJR500_1	12	23	1	-	12	12	1	7213

Tabela 10.9: Resultados do *branch-and-cut*

Grafo	FAR_1				FAR_2			
	L_I	L_S	B&C nós	Tempo	L_I	L_S	B&C nós	Tempo
jean	10	10	1	3	10	10	1	4
anna	11	11	1	18	11	11	2	26
david	30	30	1	11	30	30	1	13
homer	13	71	1	-	22	71	1	-
games120	9	9	1	35	9	9	1	30
$\mathbf{K}_{5,2}$	3	3	1	0	3	3	1	0
$\mathbf{K}_{7,2}$	6	6	407	4	6	6	357	6
$\mathbf{K}_{7,3}$	3	3	4	1	3	3	4	2
$\mathbf{K}_{9,4}$	3	3	7	461	3	3	4	809
1-Insertions-4	4	9	706	-	4	5	514	-
1-Insertions-5	3	15	11	-	3	15	9	-
2-Insertions-3	4	4	3013	163	4	4	3065	176
2-Insertions-4	3	13	17	-	3	13	4	-
3-Insertions-3	3	4	12287	-	3	4	4110	-
3-Insertions-4	3	17	8	-	3	17	5	-
4-Insertions-3	3	9	689	-	3	9	467	
1-FullIns-3	4	4	10	1	4	4	34	2
1-FullIns-4	4	6	955	-	4	10	933	-
2-FullIns-3	5	5	5	4	5	5	84	25
2-FullIns-4	5	15	13	-	5	15	10	-
3-FullIns-3	6	6	121	146	6	6	38	85
4-FullIns-3	7	7	8	98	7	7	3	72
5-FullIns-3	8	8	77	1649	8	8	5	268

Tabela 10.10: Resultados do *branch-and-cut*

solucionadas para o problema clássico de coloração, como as do tipo **mulsol.i.x**, se revelaram de difícil abordagem. Este fato mostra a diferença entre os dois tipos de problema que a primeira vista podem ser semelhantes mas as características de equilíbrio do PCE tornam o problema difícil de ser resolvido.

Capítulo 11

Conclusão

Neste trabalho apresentamos dois métodos de *branch-and-cut* baseados em dois modelos de representantes para o problema PCE. Os novos métodos foram testados frente a uma bateria de instâncias de DIMACS e se mostraram superiores ao modelo anterior baseado numa formulação de coloração de atribuição de cores [56]. No trabalho propomos duas novas modelagens de programação inteira 0-1 baseadas no modelo de representantes para o problema PCE; definição de limites teóricos para a cardinalidade dos conjuntos independentes em uma coloração equilibrada; um novo método primal baseado numa heurística tabu adaptativa; e uma nova estratégia de remificação para os modelos propostos.

Nos testes realizados os métodos mostraram força na solução de diversas instâncias de coloração. Além disso, os modelos FAR_1 e FAR_2 mostraram uma particular eficiência nas instâncias dos grafos geométricos **DSJCx_y**. Foi constatado também uma pequena superioridade do modelo FAR_2 que foi o único a alcançar a solução ótima das instâncias **zeroin.i.2**, **zeroin.i.3** e **DSJR500_1**.

Uma possível extensão deste trabalho seria estudar as famílias de cortes existentes para o modelo de atribuição de cores e adaptar para o problema PCE. Além disso, a busca por novos cortes para as restrições de equilíbrio poderia ajudar a fortificar estes modelos propostos. Uma outra abordagem seria pesquisar novas heurísticas que possivelmente poderiam se adaptar melhor com as características de equilíbrio do problema.

Parte III

Um Algoritmo Branch and Cut para o Problema de Min-Span em Alocação de Freqüências

Capítulo 12

Introdução

Os sistemas de telefonia móvel atuais utilizam uma estrutura de rede sem fio onde diferentes freqüências de banda eletromagnética são utilizadas na comunicação. Dependendo da distância geográfica dos pontos de conexão e da distância entre as freqüências atribuídas, estas conexões podem gerar interferência entre si. Com o objetivo de evitar estas interferências devemos atribuir freqüências distantes a pontos de conexão fisicamente próximos. Por outro lado, enquanto que o número de freqüências é limitado e caro, o número de conexões é vasto, o que torna interessante buscar por soluções que utilizem um número mínimo de freqüências.

A aquisição de freqüências é regulamentada por orgãos governamentais e organizações internacionais que autorizam companhias telefônicas a utilizarem uma faixa de freqüência $[f_{min}, f_{max}]$ em determinadas regiões do país. Por ter um custo muito alto, existe um grande interesse em minimizar esta largura de freqüências em que as comunicações são realizadas. O *problema de alocação de freqüência* (PAF) pode ser definido como o problema de atribuir freqüências a um conjunto de transmissores de uma rede de forma que os requisitos de interferência sejam respeitados. O *problema de alocação de freqüência Min-Span* (PAF_{Span}) é definido como um PAF onde temos como objetivo a minimização da maior freqüência atribuída a um vértice. Muitos objetivos podem ser considerados no PAF (como por exemplo PAF_{MO} em que o objetivo é minimizar o número de freqüências utilizadas) porém ao abordarmos o problema PAF_{Span} estamos diretamente visando a diminuição da largura de banda $[f_{min}, f_{max}]$ utilizada e conseqüentemente, seu custo.

Mesmo o problema PAF_{Span} não sendo um problema de coloração em grafos,

este tipo de problema se enquadra nesta categoria devido a sua semelhança estrutural com o problema de coloração. Por ser um problema NP-Difícil [65] de grande relevância prática e financeira [66], muitos trabalhos foram realizados na busca por eficientes heurísticas para encontrar soluções viáveis para o problema. Em particular podemos citar técnicas como Tabu [67] [68], *Simulated annealing* [67] e Algoritmos genéticos [69]. Uma outra linha de estudos consiste em desenvolver limitantes inferiores para o problema PAF_{Span} com a finalidade de validar a qualidade destas soluções heurísticas. Limitantes combinatoriais e poliédricos podem ser vistos nos trabalhos [14], [70], [71] ,[72] ,[73] e [74]. Porém vemos que poucos trabalhos abordam o problema diretamente com o objetivo de alcançar uma solução exata, geralmente estes métodos tendem a falhar quando o tamanho da rede ou a demanda por freqüências é muito grande. Apesar da dificuldade, métodos de busca *branch and cut*, *branch and cut and price* e enumeração ímplicida podem ser vistos em [75],[76] e [77] respectivamente.

A grande relevância do problema motivou o desenvolvimento de um método *branch and cut* apresentado neste trabalho baseado na formulação apresentada por Aardal et al. [75]. No Capítulo 13 apresentamos a formulação de programação inteira para o PAF_{Span} assim como algumas famílias de cortes válidos para o modelo. O Capítulo 14 é dedicado às principais partes do algoritmo *branch and cut* proposto. Apresentamos no Capítulo 15 alguns resultados computacionais enquanto que no Capítulo 16 analisamos o trabalho realizado.

Capítulo 13

Problema de Alocação de Freqüências de Min-Span

O PAF_{Span} pode ser representado como o grafo de alocação de freqüências $G = (V, D, E)$ onde o conjunto de vértices V corresponde a um conjunto de transmissores na rede. Para cada vértice $v \in V$ definimos o *domínio* D_v como o conjunto de todas as freqüências disponíveis que podem ser atribuídas ao vértice v. A união de todos os domínios é denotada por D. Um par de vértices $u, v \in V$ é conectado por uma aresta $uv \in E$ se existe algum requerimento de distância físico d_{uv} entre os transmissores u e v (Figura 13.1). Uma *atribuição de freqüências* é definida como a atribuição de uma freqüência $f \in D_v$ para cada vértice $v \in V$. Uma atribuição de freqüências é dita *viável* se as freqüências f_v e f_u atribuídas aos vértices $u, v \in V$ para $uv \in E$ diferem de pelo menos d_{uv} (i.e., $\mid f_v - f_u \mid \geq d_{uv}$).

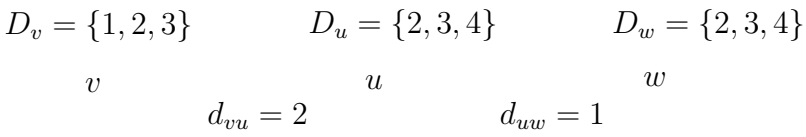


Figura 13.1: Grafo de alocação de freqüências.

13.0.1 Formulação para PAF Min-Span

Devido a grande dificuldade do problema e do pobre desempenho, poucas formulações matemáticas foram concebidas para abordar diretamente o problema PAF_{Span}. A formulação a seguir, apresentada originalmente por Aardal et al. [75] para o prob-

lema PAF_{MO} e adaptada para PAF_{Span} em [66], é baseada na formulação clássica de atribuição de cores desenvolvida por Christofides [78].

Definimos a variável binária x_{vf} para todo $v \in V$ e para todo $f \in D_v$ com a seguinte interpretação: $x_{vf} = 1$ se e somente se a freqüência f for atribuída ao vértice v, caso contrário $x_{vf} = 0$. Definimos também a variável binária y_f para todo $f \in D$ para indicar se uma determinada freqüência f é utilizada ou não (i.e., $y_f = 1$ caso exista algum vértice v tal que $x_{vf} = 1$, e $y_f = 0$ caso contrário). Além das variáveis binárias definimos uma variável inteira $z_{max} \in \mathbb{Z}^+$ que irá representar a maior freqüência utilizada na solução do problema. Esta formulação para PAF_{Span} possui no máximo $(\mid V \mid . \mid D \mid)+ \mid D \mid +1$ variáveis e pode ser representada pelo seguinte modelo de programação inteira

$$PAF_1 = \min z_{max} \tag{13.1}$$

sujeito a:

$$\sum_{f \in D_v} x_{vf} = 1 \quad \forall v \in V \tag{13.2}$$

$$x_{vf} + x_{ug} \leq 1 \quad \forall vu \in E, f \in D_v, g \in D_u : \quad \mid f - g \mid < d_{uv} \tag{13.3}$$

$$z_{max} \geq f y_f \quad \forall f \in D \tag{13.4}$$

$$x_{vf} \leq y_f \quad \forall v \in V, f \in D_v \tag{13.5}$$

$$x_{vf} \in \{0,1\} \quad \forall v \in V, f \in D_v \text{ e } y_f \in \{0,1\} \quad \forall f \in D \tag{13.6}$$

$$z_{max} \in \mathbb{Z}^+ \tag{13.7}$$

A função objetivo 13.1 busca minimizar a maior freqüência utilizada na solução do problema, e conseqüentemente, seu *span*. As restrições 13.2, denominadas de restrições de unicidade, garantem que cada vértice do grafo receberá uma única freqüência enquanto que as restrições de interferência 13.3 asseguram que pares de vértices adjacentes não receberão freqüências que provoquem interferência. As inequações 13.4 garantem que a variável z_{max} irá receber o valor da maior freqüência utilizada. As variáveis do tipo y_f indicam se uma determinada freqüência f é utilizada através das inequações 13.5.

13.0.2 Inequações válidas para PAF Min-Span

A relaxação linear do modelo PAF_1 é fraca [75], logo é interessante fortalecer a formulação com um conjunto de inequações válidas fortes. Em [75] Aardal apresenta uma família de cortes para o problema PAF_{MO}, que podem ser utilizadas em PAF_{Span}, baseadas em estruturas clique de um *grafo de conflito das variáveis*.

Definimos o grafo de conflito $G_c = (V_c, E_c)$ onde o conjunto V_c é composto pelos vértices x_{vf} para cada $v \in V$ e para cada $f \in D_v$. Uma aresta entre os vértices x_{vf} e x_{ug} corresponde a: (i) uma violação das restrições de interferência 13.3 entre os vértices v e u do grafo de alocação de freqüências (i.e., $\mid f - g \mid < d_{vu}$) ou (ii) uma violação das restrições de unicidade 13.2 quando v é igual a u (Figura 13.2). Notemos que uma solução para o problema PAF_{Span} corresponde a um conjunto independente $S \subseteq V_c$ no grafo G_c onde temos exatamente um vértice x_{vf} para cada $v \in V$.

$v, 1$	$u, 2$	$w, 1$
$v, 2$	$u, 3$	$w, 2$
$v, 3$	$u, 4$	$w, 3$

Figura 13.2: Grafo de conflito de variáveis construido a partir do grafo de alocação de freqüências.

Seja $C \subseteq V_c$ uma clique em G_c. Notemos que sempre que duas variáveis em C tiverem o valor igual a 1 teremos uma inconsistência na solução, gerada ou por uma interferência ou por uma violação na unicidade de um vértice. Logo temos que a restrição 13.8 é válida para o problema PAF_{Span}.

$$\sum_{x_{vf} \in C} x_{vf} \leq 1 \tag{13.8}$$

Uma maneira de fortalecer a restrição 13.8 é considerarmos o caso em que todos os vértices de uma clique $C_f \subseteq V_c$ possuam uma mesma freqüência f, desta forma é possível reduzir o lado direito da inequação para o valor da variável y_f. Logo temos que a restrição 13.9 é também válida para o problema PAF_{Span}.

$$\sum_{x_{vf} \in C_f} x_{vf} \leq y_f \tag{13.9}$$

13.0.3 Novas inequações para PAF Min-Span

Devido ao fraco desempenho de formulações matemáticas para o problema PAF_{Span}, muitos estudos foram realizados sobre diversos limitantes inferiores para o problema. Limitantes calculados isoladamente baseados em cliques [14] [70], caminhos hamiltonianos [71] [72] [73] e outras estruturas [74] foram utilizados com sucesso na literatura. Baseado no conceito de limites inferiores para o problema PAF_{Span} foram desenvolvidos dois grupos de equações válidas visando o fortalecimento do modelo PAF_1.

O primeiro grupo de restrições visa o fortalecimento das variáveis x e y através da utilização de limitantes inferiores sobre subproblemas do problema original. Seja $G = (V, E)$ o grafo de alocação de freqüências e $Z^*_{Spam}(G)$ o valor ótimo para o problema PAF_{Span} em G. Notemos que para todo subgrafo $H = (V_H, E_H)$ de G onde $V_H \subseteq V$ e $E_H = E[V_H]$, temos que $LI_{Spam}(H) \leq Z^*_{Spam}(H) \leq Z^*_{Spam}(G)$, onde $LI_{Spam}(H)$ um limitante inferior para o problema PAF_{Span} no subgrafo H. Por este fato, as variáveis x relacionadas com o subgrafo H podem ser fortalecidas através da restrição a seguir:

$$\sum_{\substack{v \in H \\ f \geq LI_{Spam}(H)}} x_{vf} \geq 1, \tag{13.10}$$

de forma semelhante buscamos o fortalecimento das variáveis y através da seguinte restrição:

$$\sum_{f \geq LI_{Spam}(H)} y_f \geq 1. \tag{13.11}$$

A utilização das inequações 13.10 e 13.11 tem como objetivo refletir a força de limitantes inferiores de sub-estruturas sobre as variáveis do modelo. Dessa forma, limitantes que eram inviáveis de serem computados para o grafo inteiro tornam-se úteis na composição da solução do problema.

O segundo grupo de restrições visa fortalecer diretamente a variável da função objetivo z_{max} através novamente de limitantes inferiores do grafo. O teorema a seguir utiliza a força de limitantes inferiores junto com a noção de cliques no grafo de conflitos G_c para fortalecer a variável z_{max}.

Teorema 13 *Seja G um grafo de alocação de freqüências e $C \subseteq V_c$ uma clique no seu grafo de conflito G_c. A inequação 13.12 é válida para PAF_1.*

$$LI_{Spam}(G) + \sum_{\substack{x_{vf} \in C \\ f > LI_{Spam}(G)}} (f - LI_{Spam}(G))x_{vf} \leq z_{max} \tag{13.12}$$

Prova: Vemos que para qualquer solução viável de PAF_1 temos que

$$LI_{Spam}(G) + (f - LI_{Spam}(G))x_{vf} \leq z_{max} \quad \text{, para todo } v \in V \text{ e } f > LI_{Spam}(G) \tag{13.13}$$

Vemos também que para uma clique $C \subseteq V_c$ de G_c temos que

$$1 \geq \sum_{x_{vf} \in C} x_{vf} \geq \sum_{\substack{x_{vf} \in C \\ f > LI_{Spam}(G)}} x_{vf} \tag{13.14}$$

Logo vemos que por 13.14 apenas uma das parcelas do somatório do lado esquerdo da equação 13.12 estará ativo em uma solução viável de PAF_1, resultando na inequação 13.13. □

De forma semelhante atingimos o fortalecimento da variável z_{max} através das variáveis y pela restrição a seguir:

$$LI_{Spam}(G) + (f - LI_{Spam}(G))y_f \leq z_{max} \quad \text{, para todo } f > LI_{Spam}(G). \tag{13.15}$$

Capítulo 14

Um Método Branch and Cut para PAF Min-Spam

Neste capítulo descrevemos os componentes principais do método de *branch and cut* para o problema PAF_{Span} baseado no modelo de Aardal et al. [66] com a adição dos cortes descritos nas Seções 13.0.2 e 13.0.3. Na Seção 14.1 são explicadas regras de pré-processamento para diminuir o tamanho das instâncias de PAF_{Span}. Na Seção 14.2 apresentamos uma estratégia de ramificação para o modelo do problema enquanto que nas Seções 14.3 e 14.4 mostramos métodos primais para obtenção de soluções viáveis. Finalizando, a Seção 14.5 explica uma estratégia de solução para acelerar a execução do método.

14.1 Pré-processamento

O uso de pré-processamento para diminuir o tamanho das instâncias de problemas de otimização é bastante comum. Estas técnicas são utilizadas para diminuir o esforço computacional e são cruciais para encontrar a solução de instâncias grandes. Pré-processamentos para o problema PAF_{Span} são descritas em [75], [66], [77] e [79] e neste trabalho usamos duas das principais técnicas descritas a seguir.

Eliminação de Freqüências

Este tipo de pré-processamento permite a redução do número de freqüências disponíveis nos domínios dos vértices. Seja $v, u \in V$ onde $vu \in E$ e seja D_v e D_u seus respectivos domínios de freqüência. Seja também $f_< = MIN_{f \in D_u} f$ e seja

$f_> = MAX_{f \in D_u} f$. Dado uma freqüência $g \in D_v$ dizemos que a freqüência g domina D_u se e somente se $(g - d_{vu}) < f_<$ e $(g + d_{vu}) > f_>$. Vemos que caso g domine D_u o problema se torna inviável pois não teriamos nenhuma freqüência livre para atribuirmos para u. Logo todos as freqüências dominantes são retiradas do grafo.

Eliminação de Vértices

Uma maneira de eliminar vértices no problema PAF_{Span} é identificar vértices do grafo que possam receber a mesma freqüência. Seja $v, u \in V$ tal que $vu \notin E$, dizemos que o vértice u domina v se somente se $D_v \subseteq D_u$ e para todo $w \in (N(v) \cap N(u))$ temos que $d_{wv} \geq d_{wu}$. Vemos que em qualquer atribuição de freqüências viável para o grafo $G \backslash \{v\}$, onde u domina v, sempre será possível atribuir a mesma freqüência de u para v no grafo G. Logo todos os vértices dominados do grafo são retirados do problema.

14.2 Estratégia de Ramificação

No método *branch and cut* para o problema PAF_{Span} é utilizada uma tradicional ramificação do problema em que uma variável binária fracionária é selecionada e dois novos problemas são gerados através da fixação dos valores da variável em 0 ou 1. Logo dada uma solução fracionária da relaxação linear do modelo PAF_1, caso exista alguma variável y_f fracionária, a ramificação será realizada na variável com o valor mais próximo de 1. Esta estratégia visa fixar as freqüências em que a relaxação linear considera mais apropriadas. Caso não existam variáveis y_f fracionárias, a ramificação será realizada sobre a variável x_{vf} com o valor mais próximo de 0.5.

14.3 Um método Tabu para o problema PAF_{Span}

Devido à dificuldade do problema PAF_{Span} diversos trabalhos tem sido realizados com abordagens meta-heurísticas. Técnicas como métodos Tabu [67] [68], *Simulated annealing* [67] e Algoritmos genéticos [69] foram realizadas com sucesso. Mais recentemente Touhami [57] apresentou um método Tabu adaptativo para o problema de alocação de freqüência em redes de celulares onde busca-se minimizar interferências na solução. Na Seção 9.3.1 apresentamos uma adaptação deste método para o

problema de coloração equilibrada. A seguir, o método é revisto e suas principais componentes são adaptadas para o problema PAF_{Span}.

14.3.1 Função objetivo

Dado um grafo de alocação de freqüências $G = (V, D, E)$, definimos uma solução $\hat{x}$ para o problema PAF_{Span} como o conjunto das variaveis binárias x_{vf} do modelo PAF_1, onde:

$$\hat{x} = \{x_{vf}, \forall v \in V \text{ e } f \in D_v\}.$$

Definimos também a função $\Theta_u(\hat{x})$ para descrever o grau de viabilidade da solução $\hat{x}$ com respeito às restrições de unicidade como:

$$\Theta_u(\hat{x}) = \sum_{v \in V}(1 - \sum_{f \in D_v} x_{vf})$$

Notemos que quando a solução não viola as restrições de unicidade temos que $\Theta_u(\hat{x}) = 0$. Além disto, definimos uma outra função $\Theta_i(\hat{x})$ para contabilizar o número de restrições de interferência violadas pela solução $\hat{x}$, onde:

$$\Theta_i(\hat{x}) = \sum_{vu \in E} \sum_{\substack{f \in D_v, g \in D_u \\ |f-g| < d_{vu}}} x_{vf}x_{ug}$$

De forma similar, caso a solução $\hat{x}$ não viole as restrições de interferência temos que $\Theta_i(\hat{x}) = 0$. Finalizando, definimos a função objetivo global $\Theta_g(\hat{x})$ do método como:

$$\Theta_g(\hat{x}) = \min(f.x_{vf}) + \mu_u.\Theta_u(\hat{x}) + \mu_i.\Theta_i(\hat{x}) \tag{14.1}$$

onde μ_u e μ_i representam coeficientes de penalidade associados à violação das restrições de unicidade e interferência respectivamente.

Vemos que o primeiro termo da função 14.1 representa o objetivo de minimizar a maior freqüência utilizada na solução $\hat{x}$ enquanto que o segundo e terceiro termos descrevem a viabilidade da solução. Novamente a presença de fatores multiplicativos nas funções relativas a viabilidade possibilitarão o método visitar soluções inviáveis durante a busca pela minimização da função objetivo $\Theta_g(\hat{x})$.

14.3.2 Operador de busca

O espaço de busca é explorado através de alterações na atribuição das freqüências realizadas por movimentos a cada iteração. Para o problema PAF_{Span} um movimento consiste em trocar a freqüência de um determinado vértice $v \in V$ de f para g onde $f, g \in D_v \cup \{\emptyset\}$. Denotaremos este movimento por $Troca(v : f \rightarrow g)$. Aplicando este movimento a uma solução $\hat{x}$, obteremos uma nova solução $\hat{x}'$ definido por $\hat{x}' = \hat{x} + Troca(v : f \rightarrow g)$.

O operador de busca aplicado neste método procura identificar o melhor movimento para um determinado vértice $v \in V$ dentre um subconjunto de freqüências. Seja $D'_v = D_v \cup \{\emptyset\}$ para todo $v \in V$, definimos o operador $Freq(v)^p$ da seguinte forma: considere o vértice $v \in V$ que possui a freqüência $f \in D'_v$ numa solução $\hat{x}$. O operador $Freq(v)^p$ identifica o movimento de troca de freqüências de v, dentre um subconjunto de D'_v, que produz o menor valor da função objetivo $\Theta_g(\hat{x})$. O subconjunto de freqüências é definido através da análise de uma fração p das possíveis mudanças de freqüências em D'_v. Com esta parametrização, conseguimos manipular a intensidade do operador $Freq(v)^p$. Logo vemos que:

$$\Theta_g(\hat{x}+Freq(v)^p) = \Theta_g(\hat{x}+Troca(v : f \rightarrow g^*)) = \min_{\substack{g \in D'_v \\ Rand(g) < p}} \Theta_g(\hat{x}+Troca(v : f \rightarrow g))$$

onde $Rand(g)$ é um número aleatoriamente gerado no intervalo $[0, 1]$.

O operador $Freq(v)^p$ definido acima é utilizado para a composição do principal operador de busca denominado OPE_q^p. Dado uma solução $\hat{x}$, este operador identifica o movimento $Freq(v)^p$, onde v pertence a um subconjunto de V, que novamente produz o menor valor da função objetivo $\Theta_g(\hat{x})$. Similarmente ao operador $Freq(v)^p$, o subconjunto de vértices analisados será definido por uma probabilidade de busca q. Logo temos que:

$$\Theta_g(\hat{x} + OPE_q^p) = \Theta_g(\hat{x} + Freq(v^*)^p) = \min_{\substack{v \in V \\ Rand(v) < q}} \Theta_g(\hat{x} + Freq(v)^p)$$

onde $Rand(v)$ é um número aleatoriamente gerado no intervalo $[0, 1]$.

14.3.3 Algoritmo do método Tabu adaptativo

O algoritmo do método Tabu adaptativo para PAF_{Span}, denominado $\texttt{TS}_{adap}\texttt{-PAF}$, é o mesmo algoritmo apresentado na Seção 9.3.1, porém uma parametrização que se adapte melhor às propriedades do problema PAF_{Span} é utilizada. No início do método utilizamos os valores $p = 0.3$ e $q = 0.5$ para a fase do Tabu. Durante o decorrer do processo de busca, estes valores são lentamente ampliados, para intensificar a busca, até atingir os valores de $p = q = 1$. Sempre que o método alterna para a fase de intensificação de busca local, os valores de p e q são atualizados para 1 com o objetivo de alcançar soluções viáveis mais facilmente.

Além disso, os parâmetros da função objetivo são atualizados a cada iteração do método da seguinte forma.

- $\mu_u = \begin{cases} \mu_u(1.03) & \text{caso } \Theta_u(\hat{x}) > 0 \\ \mu_u(0.97) & \text{caso contrário} \end{cases}$
- $\mu_i = \begin{cases} \mu_i(1.1) & \text{caso } \Theta_i(\hat{x}) > 0 \\ \mu_i(0.97) & \text{caso contrário} \end{cases}$

Desta forma induzimos o método a caminhar mais facilmente por soluções que gerem interferência, e através desta liberdade encontrar soluções viáveis mais rapidamente.

Com respeito aos parâmetros Tabu, a proibição do movimento $Troca(v : f \rightarrow g)$ é realizada pelo método de forma similar ao método para coloração equilibrada. A atribuição da freqüência f ao vértice v fica proibida (Tabu) no modo de busca Tabu pelo seguinte número de passos:

$$\eta \frac{\psi(v, f)}{\psi(v)}$$

onde $\psi(v, f)$ é o número de vezes que a freqüência f foi atribuíada ao vértice v, $\psi(v)$ é o número de vezes que o vértice v trocou de freqüência e η é uma constante que indica o tamanho da lista Tabu.

14.4 Um algoritmo enumerativo para PAF_{Span}

Uma outra forma de alcançar soluções viáveis para o problema foi desenvolvida por Mannino e Sassano através de um simples porém eficiente método enumerativo para o problema PAF_{Span} [77]. Este método se mostrou bastante eficaz para instâncias

em que o número de freqüências necessárias para uma atribuição viável do grafo é pequeno.

O procedimento recursivo deste método enumerativo $\texttt{Enum}_{PAF}$ apresentado pelo Algoritmo 4 recebe como entrada o grafo de alocação de freqüências $G = (V, D, E)$, um subconjunto de vértices $W \subseteq V$ que já possuem uma freqüência atribuída, e uma solução parcial corrente $\hat{x}$. O método retorna a variável $Viavel$ que indica se uma solução viável foi encontrada e a solução corrente $\hat{x}$. O algoritmo inicia seu procedimento na linha 5 onde a presença de uma solução viável é verificada, isto é, se todos os vértices possuem freqüências. Caso uma solução não tenha sido encontrada, um vértice $v \in V \backslash W$ é selecionado na linha 10 pelo procedimento $Escolhe_v(G, W)$ de acordo com o seguinte critério:

$$v = \max_{i \in V \backslash W} \sum_{\substack{j \in N(i) \\ j \in V \backslash W}} d_{ij}.$$

Após esta seleção, atribuimos ao vértice v uma freqüência $f \in D_v$ na linha 12 e chamamos recursivamente o método $\texttt{Enum}_{PAF}$ para esta nova solução parcial na linha 14. O procedimento $Remove_f(G, v, f)$ atualiza os conjuntos de freqüências viáveis D_u onde $u \in N(v)$ retirando as freqüências que se tornaram inviáveis aos vizinhos de v devido a atribuição da freqüência f, isto é, $D_u = \{g \in D_u$ onde $\mid f - g \mid \geq d_{vu}\}$. Caso uma solução tenha sido encontrada, o algoritmo retorna com uma solução viável na linha 15. Caso contrário, a atribuição anterior é desfeita na linha 17, os conjuntos de freqüências reestabelecidos na linha 18 pelo procedimento $Retorna_f(G, v, f)$ e uma nova iteração do laço é realizada.

O método de Mannino por ser um procedimento enumerativo pode se tornar uma alternativa bastante ineficaz em termos computacionais para instâncias de grande porte. Durante o método *branch and cut* proposto, optamos por utilizar uma estratégia híbrida para o cálculo do valor primal inicial. Primeiro uma solução viável é adquirida através da execução do método Tabu adaptativo descrito na Seção 14.3.3. Em seguida, executamos o método de Mannino considerando apenas as freqüências menores que o valor primal alcançado pelo método $\texttt{TS}_{adap}\texttt{-PAF}$. Desta forma, conseguimos diminuir o espaço de solução que o método enumerativo deveria percorrer na busca por melhores soluções primais.

```
1.  Algoritmo Enum_PAF
2.  Entrada: G, W, x̂
3.  Saida: x̂, Viavel
4.
5.  Se W = V Então
6.     Viavel =verdade;
7.     Retorne;
8.
9.  Viavel =falso;
10. v = Escolhe_v(G, W);
11. Para f ∈ D_v Faça
12.    x_vf = 1; W = W ∪ {i};
13.    Remove_f(G, v, f);
14.    Enum_PAF(G, W, x̂);
15.    Se Viavel =verdade Então Retorne;
16.    Senão
17.       x_vf = 0; W = W\{i};
18.       Retorna_f(G, v, f);
```

Algoritmo 4: Algoritmo enumerativo para PAF_{Span}

14.5 Acelerando o método *branch and cut*

Durante o desenvolvimento do método de *branch and cut* para PAF_{Span} foi utilizada uma técnica para acelerar o processo de solução do procedimento. Esta técnica geralmente utilizada em problemas de grande porte [76] [77] [80] consiste em solucionar uma sequência de subproblemas $G_1, G_2, ..., G_p$ gerados a partir do problema original G onde cada subproblema está estritamente contido no subproblema posterior. Logo, uma vez solucionado o subproblema inicial $G_1 = (V_1, D_1, E_1)$, onde $V_1 \subseteq V$, $D_1 \subseteq D$ e $E_1 \subseteq E$ temos que (i) o valor da solução de G_1 é um limitante inferior para o problema G e (ii) a solução de G_1 fornece uma solução parcial de G que possivelmente pode ser extendida para o problema inteiro. Caso a solução de G_1 não possa ser extendida para G ou o limitante inferior fornecido por G_1 não prove a otimalidade do problema G, um novo subproblema $G_2 = (V_2, D_2, E_2)$, onde $V_1 \subset V_2$, $D_1 \subseteq D_2$ e $E_1 \subset E_2$ é solucionado e o processo é repetido até que o último subproblema $G_p = G$ seja alcançado ou a otimalidade do problema G seja provada.

Com o objetivo de encontrar um subproblema pequeno o suficiente para poder ser solucionado dentro de um tempo aceitável porém grande o suficiente para poder representar a dificuldade do problema, Mannino e Sassano [77] elaboraram uma regra de geração de subproblemas para PAF ponderada pelas distâncias estabelecidas entre os vértices do grafo. O Algoritmo 5 ilustra o procedimento de criação do subproblema $G' = (V', D', E')$ onde iterativamente são inseridos vértices v em V' de acordo com o valor de $\xi(v, H)$, até o número de vértices tam desejado ser alcançado, onde $\xi(v, H) = \sum_{\substack{u \in N(v) \\ u \in H}} d_{vu}$. A regra de seleção da linha 8 seleciona como primeiro vértice do subproblema o vértice que tenha a maior soma de pesos das arestas conectadas a ele. Após a seleção do vértice inicial, a regra de seleção da linha 10 procura identificar vértices que gerem subproblemas fortemente conectados onde a soma das arestas tenha o maior peso possível.

Em nosso algoritmo utilizamos o parâmetro $tam = \min(30, \frac{|V|}{10})$ como tamanho do subproblema inicial G_1 e aplicamos o método *branch and cut*. Caso a solução ótima de G_1 não prove a otimalidade do problema G, o valor tam é iterativamente atualizado para $tam = tam+10$ com o objetivo de gerar os subproblemas posteriores. Além disso, sempre que um subproblema $G_i = (V_i, D_i, E_i)$ é solucionado, tentamos

```
1.  Algoritmo Encontra_Sub
2.  Entrada: G = (V, D, E), tam
3.  Saida: G' = (V', D', E')
4.
5.  V' = ∅;
6.  Enquanto | V' | < tam Faça
7.    Se | V' | = 0 Então
8.      v = max_{u∈V} ξ(u, V);
9.    Senão
10.     v = max_{u∈V\V'} ξ(u, V');
11.   V' = V' ∪ {v};
12.
13. E' = E[V'];
14. D' = D[V'];
```

Algoritmo 5: Procedimento para encontrar subproblema PAF de Mannino

expandir sua solução para o problema G através da aplicação do método $\texttt{Enum}_{PAF}$ no grafo formado pelos vértices restantes $V \backslash V_i$ que ainda não foram atribuídos freqüências.

Capítulo 15

Resultados Computacionais

Neste capítulo são discutidos os resultados obtidos pelo método *branch and cut* para PAF_{Span} descrito nos capítulos anteriores. Na Seção 15.1 apresentamos uma breve descrição das instâncias do problema PAF_{Span} que foram utilizadas para testar o método. Na Seção 15.2 discutimos alguns detalhes de implementação e finalizando apresentamos os resultados e uma posterior análise do método na Seção 15.3.

15.1 Descrição das Instâncias

Com o objetivo de testar o método proposto, escolhemos duas das principais classes de problemas para PAF_{Span}. Estas instâncias utilizadas em nossa bateria de testes e descritas a seguir podem ser encontradas em [81].

Radio e Televisão: Estas são instâncias fornecidas por uma grande companhia italiana de Radio e Televisão. Assim como na telefonia móvel, seu objetivo é atribuir freqüências a transmissores de sinal de radio e televisão de maneira que os requerimentos de distância sejam satisfeitos e que busque minimizar a maior freqüência utilizada. Denominamos por `r_i` e `t_i` as instâncias de radio e televisão respectivamente.

Philadelphia: As instâncias Philadelphia são caracterizadas por 21 hexágonos representando uma rede celular de telefonia ao redor de Philadelphia na Pennsylvania (Figura 15.1). Cada hexágono demanda um grande número de freqüências e a distância necessária para não haver interferência é proporcional à distância

entre os hexágonos. As instâncias Philadelphia são uma classe de problemas de difícil solução pois o número de freqüências necessárias para sua solução é elevado. Logo abordaremos em nosso trabalho a subclasse de problemas **P_i-x** baseadas nas instâncias Philadelphia porém com valor de demandas de freqüência dividivas por um fator **x** [82].

Figura 15.1: Hexágonos das instâncias Philadelphia.

15.2 Detalhes de Implementação

Seguindo o mesmo padrão dos experimentos anteriores, o método *branch and cut* foi implementado utilizando a linguagem de programação $C\ ANSI$ com a biblioteca $XPRESS - MP$ versão 2005-a para resolver os problemas de programação linear em uma estação AMD-Atlon 1.8 Ghz de velocidade e um Gbyte de memória RAM.

Limitantes Primais

O limitante primal inicial é fornecido pelo método híbrido descrito nas Seções 14.3.3 14.4. Para o método Tabu adaptativo utilizamos os parâmetros $T_{MAX} = 20. \mid V \mid$ e $\eta = 30$ enquanto que ao método híbrido total foi imposto um tempo limite de $\mid V \mid /20$ segundos. Além deste limitante inicial, executamos também o método enumerativo nos nós da árvore de busca do método de *branch and cut* em que pelo menos 60% das variáveis x_{vf} tenham o valor da relaxação linear igual a 1. Neste caso, tentamos encontrar atribuições de freqüências para os demais vértices pelo método Mannino com um tempo limite de 5 segundos. O mesmo procedimento enumerativo é executado sempre que um subproblema G_i é solucionado na tentativa de expandir a atribuição encontrada para o problema G. A Tabela 15.1 ilustra uma comparação dos limites superiores iniciais obtidos pelo método enumerativo puro e

em sua forma híbrida para o grupo de instâncias Philadelphia. Vemos que dentro do tempo limite estipulado o método híbrido foi capaz de alcançar limites mais fortes em 3 instâncias (**p_9-12**, **p_10-12** e **p_9-10**).

Grafo	$\texttt{Enum}_{PAF}$	$\texttt{TS}_{adap}\texttt{-PAF} + \texttt{Enum}_{PAF}$
p_1-12	12	12
p_2-12	19	19
p_3-12	32	32
p_4-12	73	73
p_5-12	158	158
p_6-12	11	11
p_7-12	19	19
p_8-12	32	32
p_9-12	43	39
p_10-12	34	32
p_1-10	25	25
p_2-10	27	27
p_3-10	40	40
p_4-10	94	94
p_6-10	22	22
p_7-10	25	25
p_8-10	40	40
p_9-10	52	48
p_10-10	39	39

Tabela 15.1: Resultados do limite superior inicial para PAF_{Span}

Metodologia de Ramificação

Para o método *branch and cut* utilizamos a estratégia de *best-search* para a escolha do próximo nó da árvore de busca a ser verificado. Ao contrário dos problemas dos capítulos anteriores, ponderamos os nós da árvore de busca pela média entre limite inferior fornecido pela relaxação linear do modelo e o limite superior dos problemas. Esta abordagem levou a resultados ligeiramente melhores do que a ponderação apenas pelo limite inferior.

Planos de Corte

tailing-off O número de cortes gerados em instâncias grandes do problema PAF_{Span} podem impossibilitar a resolução até da relaxação inicial. Por isso, limitamos o número de cortes gerados por iteração para 300 e o tempo limite de geração total de cortes para 10 minutos. Além disso, se a a função objetivo do modelo não alcançar um incremento de 0.1 em 5 iterações das famílias de cortes, a geração será finalizada.

Cortes O problema de separação dos cortes baseados em estruturas cliques das inequações 13.8 e 13.9 é resolvido heuristicamente por uma generalização do algoritmo descrito na Seção 4.4.1. Este algoritmo generalizado é aplicado sobre o grafo de conflito com a finalidade de identificar cortes clique violados pela inequação 13.8 e sobre o grafo de alocação de freqüências para identificar cortes clique violados pela inequação 13.9. Devido ao tamanho do grafo de conflito para certas instâncias, apenas uma parte de seus cortes clique são gerados. A busca por cliques violadas no grafo de conflito é realizada em subgrafos de sua estrutura que correspondem às variáveis com valor não nulo na relaxação linear do problema. Esta estratégia é adotada pois a geração explícitada do grafo de conflitos se mostrou impraticável.

Novos Cortes Os novos cortes expressos pelas inequações 13.10, 13.11, 13.12 e 13.15 são baseados na identificação de estruturas cliques violadas no grafo de conflito e em limites inferiores de subestruturas do grafo de alocação de freqüências. No primeiro caso adotamos a mesma estratégia para identificar estruturas clique no grafo de conflito para os cortes da inequação 13.8. No segundo caso utilizamos uma outra generalização do algoritmo descrito na Seção 4.4.1 para encontrar cliques no grafo de alocação de freqüências que maximizem o limitante inferior de clique descrito por Gamst [14]: em uma clique de tamanho k e com uma distância mínima de aresta d temos que $(k-1)d$ é um limitante inferior para PAF_{Span}. Além disso, utilizamos uma versão do algoritmo enumerativo descrito na Seção 14.4 sobre subgrafos H de tamanho até 15 vértices para encontrar as soluções exatas destes subproblemas. Estes valores servirão de limitantes inferiores para o problema. Por ser uma técnica custosa, o processo enumerativo é realizado apenas na relaxação inicial do problema sobre um conjunto de subgrafos que irão cobrir todos os vértices de G.

15.3 Resultados computacionais

Nesta seção, apresentaremos os resultados da execução do algoritmo *branch and cut* descrito para o problema PAF_{Span} analisando suas vantagens e limitações quando aplicado sobre o grupo de instâncias da literatura [82] [81] apresentado neste capítulo.

Em nosso primeiro experimento analisamos o desempenho do método *branch and cut* junto às instâncias de radio e televisão descritas na seção anterior. O objetivo deste teste é mensurar a qualidade dos novos cortes baseados em limites inferiores para o problema PAF_{Span}. A tabela 15.2 apresenta uma comparação dos resultados obtidos pelo método com e sem os novos cortes em um processamento com tempo limite de 2 horas. Para cada instância, as primeiras três colunas ilustram o nome, o número de vértices e o número de arestas do grafo. A quarta coluna ilustra o valor da melhor solução primal encontrada. As três colunas seguintes apresentam o valor da relaxação linear do modelo para o subgrafo inicial G_1, o valor do limitante inferior final e o tempo de processamento do método *branch and cut* sob o modelo PAF_1 sem os novos cortes. As três últimas colunas da tabela apresentam a mesma informação para o método *branch and cut* sob o modelo PAF_1 com os novos cortes de limitantes inferiores. O símbolo $''-''$ na coluna Tempo indica que o algoritmo foi encerrado no tempo limite de 2 horas de processamento.

Podemos verificar que o método para PAF_{Span} sem os novos cortes (PAF_1) não foi capaz de solucionar nenhuma instância de rádio e televisão dentro do tempo limite de 2 horas. Vemos que seus limitantes baseados apenas nos cortes cliques tradicionais manteve um grande intervalo em relação a melhor solução primal encontrada. Por outro lado o método *branch and cut* com os novos cortes (PAF_1^c) foi capaz de encontrar a solução ótima do problema de todas as instâncias de rádio e algumas de televisão. O motivo deste comportamento vem dos fortes limitantes inferiores fornecidos pelo algoritmo enumerativo que obteve bons resultados em subgrafos de até 15 vértices nestas instâncias e gerou cortes que fortificaram a relaxação linear do problema. O bom desempenho do algoritmo enumerativo vem do fato de que estas instâncias utilizam poucas freqüências nas suas soluções ótimas e da existência de arestas com valor de distância elevado que diminuem o número de soluções viáveis.

A Tabela 15.3 apresenta as mesmas informações da tabela anterior abordando agora as instâncias do tipo Philadelphia. Nesta classe de problemas o número total

Grafo	n	m	L_S	PAF_1			PAF_1^c		
				G_1^r	L_I	Tempo	G_1^r	L_I	Tempo
r_1	20	134	18	4.25	13	-	18.00	18.00	2.58
r_2	30	228	18	4.25	12	-	18.00	18.00	6.46
r_3	45	336	18	4.25	12	-	18.00	18.00	4.81
r_4	100	697	18	4.25	12	-	18.00	18.00	29.97
r_5	200	1363	18	4.25	12	-	18.00	18.00	50.00
r_6	350	2405	18	4.25	12	-	18.00	18.00	115.33
r_7	857	4023	18	4.25	10	-	18.00	18.00	410.71
t_1	20	110	14	3.53	11	-	13.33	14.00	0.95
t_2	30	180	14	3.53	10	-	13.07	14.00	1.38
t_3	45	310	16	4.62	11	-	15.25	16.00	11.31
t_4	100	710	18	4.25	11	-	16.00	16.00	-
t_5	200	1425	16	4.25	11	-	14.00	14.00	-
t_6	350	2419	18	4.25	11	-	16.00	16.00	-
t_7	857	4023	18	4.25	10	-	16.00	16.00	-

Tabela 15.2: Resultados do *branch-and-cut* para PAF_{Span}

de freqüências utilizadas nas soluções ótimas é superior às instâncias de rádio e televisão. Por este motivo, os novos cortes mais fortes fornecidos para estas instâncias foram originados a partir dos limitantes inferiores de clique descrito por Gamst [14]. O algoritmo enumerativo teve dificuldades em encontrar limitantes inferiores fortes em instâncias onde o espaço de freqüências viáveis é amplo. Novamente podemos observar que o desempenho do método com os novos cortes é superior a sua versão original, principalmente em instâncias que não possuem estruturas cliques muito fortes como as instâncias **p_3**, **p_8** e **p_10**. Este fato penalizou os cortes cliques das inequações 13.8 e 13.9, porém os limitantes de clique conseguiram fortalecer a relaxação através dos novos cortes das equações 13.10, 13.11, 13.12 e 13.15.

Grafo	n	m	L_S	PAF_1			PAF_1^c		
				G_1^r	L_I	Tempo	G_1^r	L_I	Tempo
p_1-12	21	58	12	12.00	12.00	0.62	12.00	12.00	0.26
p_2-12	31	340	19	13.60	18.00	-	19.00	19.00	4.54
p_3-12	31	407	32	13.66	21.00	-	26.00	26.00	-
p_4-12	72	2192	73	10.28	56.00	-	56.00	56.00	-
p_5-12	153	9960	158	27.61	93.00	-	111.00	111.00	-
p_6-12	21	118	11	4.27	10.00	-	11.00	11.00	0.52
p_7-12	31	268	19	7.33	16.00	-	19.00	19.00	5.84
p_8-12	31	339	32	10.62	17.00	-	27.00	27.00	-
p_9-12	31	407	39	11.12	35.00	-	35.00	35.00	-
p_10-12	31	407	32	13.21	21.00	-	27.00	27.00	-
p_1-10	42	653	25	18.70	24.00	-	24.00	24.00	-
p_2-10	41	599	26	15.70	23.00	-	23.00	23.00	-
p_3-10	36	551	54	14.61	29.00	-	35.00	35.00	-
p_4-10	88	3279	94	7.89	41.00	-	43.00	43.00	-
p_6-10	42	493	22	8.22	17.00	-	22.00	22.00	9.61
p_7-10	41	479	25	10.15	19.00	-	23.00	23.00	-
p_8-10	36	462	39	11.92	17.00		33.00	33.00	-
p_9-10	36	551	48	12.74	43.00	-	43.00	43.00	-
p_10-10	38	560	39	14.97	21.00	-	29.00	29.00	-

Tabela 15.3: Resultados do *branch-and-cut* para PAF_{Span}

Queremos deixar claro que estes resultados apresentados deste algoritmo exato não são competitivos com outros métodos já apresentados para o problema. Para o leitor com este interesse sugerimos o trabalho de Avenali et al. [76] que apresenta um *branch and cut and price* baseado numa formulação de caminhos hamiltonianos no grafo para resolver o problema PAF_{Span}.

Apesar dos resultados superiores de Avenali et al. [76], principalmente para o grupo de instâncias Philadelphia, seu modelo baseado em caminhos hamiltoneanos é uma abordagem indireta ao problema PAF_{Span}, o que torna inviável sua adaptação para outras classes de problemas PAF (PAF_{MO}, PAF_{INTER}, etc). Por outro lado, é possível adaptar o modelo clássico apresentado por Aardal et al. [66] para as outras classes de problemas. Mesmo com as novas classes de desigualdades apresentadas neste trabalho, o algoritmo *branch and cut* não se mostrou competitivo com o método de Avenali. Possivelmente o conhecimento de outras classes de desigualdades válidas e definidoras de facetas são ainda necessárias para definirmos uma boa aproximação do politopo associado ao modelo. Este trabalho representa o passo inicial na tarefa de obter um modelo forte e competitivo para a classe de problemas PAF.

Capítulo 16

Conclusão

Neste trabalho apresentamos um novo método *branch and cut* para o problema de PAF_{Span} baseado na formulação de Aardal et al. [66]. Além disto apresentamos uma rápida nova abordagem heurística para o problema e novas classes de cortes para o modelo baseadas em limites inferiores fortes existentes na literatura. Os testes computacionais demonstraram a força dos novos cortes para o modelo de Aardal e apesar dos resultados obtidos não estarem competitivos com a melhor solução da literatura, sabemos que o modelo abordado pode ser adaptado para outras classes de problemas PAF, o que não ocorre com o modelo de Avenali et al. [76]. Logo este trabalho representa o primeiro passo para a criação de uma formulação forte para a classe de problemas PAF.

A extensão natural deste trabalho seria continuar a estudar novas classes de desigualdades válidas para o modelo de PAF_{Span} que sejam capazes de tornar o modelo competitivo com os métodos existentes na literatura. Além disso, adaptar os novos cortes para as outras versões de problemas PAF para verificar sua eficiência.

Referências Bibliográficas

[1] LI, G., SIMHA, R., "The partition coloring problem and its application to wavelength routing and assignment". In: *Proceedings of the First Workshop on Optical Networks*, Dallas, 2000.

[2] CHOI, J., GOLMIE, N., LAPEYRERE, F., et al., "A functional classification of routing and wavelength assignment schemes in DWDM networks: Static case". In: *Proceedings of the 7th International Conference on Optical Communication and Networks*, pp. 1109–1115, Paris, 2000.

[3] NORONHA, T., "Algoritmos para Problemas de Otimização Aplicados a Roteamento e Atribuição de Comprimentos de Onda". In: *Tese de doutorado*, 2008.

[4] CORRÊA, R. C., CAMPELÔ, M., FROTA, Y. A. M., "Cliques, holes and the vertex coloring polytope", *Information Processing Letters*, v. 89, pp. 159–164, 2004.

[5] CAMPÊLO, M., CAMPOS, V., CORRÊA, R., "On the asymmetric representatives formulation for the vertex coloring problem". In: *Proceedings of the 2th Brazilian Symposium on Graphs, Algorithms and Combinatorics*, v. 19, *Electronic Notes in Discrete Mathematics*, pp. 337–343, Angra dos Reis, 2005.

[6] BONDY, J. A., MURTY, U. S. R., *Graphy theory with applications*. American Elsevier Publishing, 1976.

[7] HARARY, F., *Graphy Theory*. Addison-Wesley Publishing, 1969.

[8] BAZARAA, M. S., JARVIS, J. J., SHERALI, H. D., *Linear Programming and Network Flows*. John Wiley & Sons, 1977.

[9] WINSTON, W. L., *Operations Research: Applications and Algorithms*. International Thomson Publishing, 1994.

[10] PAPADIMITRIOU, C. H., STEIGLITZ, K., *Combinatorial Optimization*. Dover Publications, 1998.

[11] WOLSEY, L. A., *Integer Programming*. Wiley-Interscience Publication, 1998.

[12] LAND, A. H., DOIG, A. G., "An Automatic Method for Solving Discrete Programming Problems". In: *Econometrica 28, 497-520*, 1960.

[13] DE WERRA, D., "An introduction to timetabling". In: *Europian Journal of Operations Research, Vol 19*, pp. 151–162, 1985.

[14] GAMST, A., "Some lower bounds for a class of frequency assignment problems". In: *IEEE Transactions of Vehicular Technology, Vol 35, n. 1*, pp. 8–14, 1986.

[15] CHOW, F., HENNESSY, J., "Register allocation by priority-based coloring". In: *Proceedings of ACM SIGPLAN 84 Symposium on Compiler Construction, New York, NY, USA*, pp. 222–232, 1984.

[16] CHOW, F., HENNESSY, J., "The priority-based coloring approach to register allocation". In: *ACM Transactions on Programming Languages and Systems, Vol 12, n. 4*, pp. 501–536, 1990.

[17] SAAD, Y., *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company: Boston, USA, 1996.

[18] KARP, R. M., "Reducibility among combinatorial problems". In: *Complexity of Computations, Advances in Computing Research*, pp. 85–103, 1972.

[19] GAREY, M. R., JOHNSON, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[20] LUND, C., YANNAKAKIS, M., "On the Hardness of Approximation Minimization Problems". In: *Proceedings of 25th Annual ACM Symposium on Theory of Computing*, 1993.

[21] BROWN, R., "Chromatic scheduling and the chromatic number problem", *Management Science*, v. 19, pp. 451–463, 1972.

[22] BRÉLAZ, D., "New Methods to Color the Vertices of a Graph". In: *Communications of the ACM, Vol. 22, n. 4*, 1979.

[23] KORMAN, S., "The Graph-Coloring Problem", In: CHRISTOPHIDES, N., TOTH, P., SANDI, C. (eds), *Combinatorial Optimization*, pp. 211–235, Wiley: New York, 1979.

[24] KUBALE, M., JACKOWSKI, B., "A Generalized Implicit Enumeration Algorithm for Graph Coloring". In: *Communications of the ACM 28/4*, pp. 400–412, 1985.

[25] BLUM, A., "New Approximation Algorithms for Graph Coloring". In: *Journal of Association for Computing Machinery*, pp. 470–516, May 1994.

[26] MEHROTRA, A., TRICK, M. A., "A Column Generation Approch for Graph Coloring". In: *INFORMS Journal on Computing, Vol. 8, pp. 344-354*, 1996.

[27] DÍAS, I. M., ZABALA, P., "A Branch-and-Cut Algorithm for Graph Coloring". In: *Optimization On-Line*, 2003.

[28] FIGUEIREDO, R., BARBOSA, V., MACULAN, N., et al., "New 0-1 integer formulations of the graph coloring problem". In: *Proceedings of the XI Congreso Latino Iberoamericano de Investigación de Operaciones*, Concepción, 2002.

[29] NORONHA, T., RIBEIRO, C., "Routing and wavelength assingn by partition coloring", *European Journal of Operational*, v. 171, pp. 797–810, 2006.

[30] NEMHAUSER, G., WOLSEY, L., *Integer Combinatorial Optimization*. Wiley-Interscience Publication, 1988.

[31] GOLUMBIC, M. C., *Algorithmic, Graph Teory and Perfect Graphs*. Academic Press, 1980.

[32] CAPRARA, A., FISCHETTI, M., "Branch-and-Cut Algorithms: An annotated bibliography". In: *Annotated Bibliographies in Combinatorial Optimization*, pp. 45–63, Wiley, 1997.

[33] WELSH, D., POWELL, M., "An upper bound to the chromatic number of a graph and its application to time-table problems", *Computer Journal*, v. 10, pp. 85–86, 1967.

[34] MATULA, D., MARBLE, G., ISAACSON, J., "Graph Coloring Algorithms", In: READ, R. (ed), *Graph Theory and Computing*, pp. 109–122, Academic Press, New York, 1972.

[35] BRÉLAZ, D., "New methods to color the vertices of a graph", *Communications of the ACM*, v. 22, pp. 251–256, 1979.

[36] GLOVER, F., LAGUNA, M., *Tabu Search*. Kluwer, 1997.

[37] HOFFMAN, K. L., PADBERG, M., "Solving Airline Crew Scheduling Problems". In: *Management Science, volume 39, number 6, 657-682*, 1993.

[38] FEO, T., RESENDE, M., "A probabilistic heuristic for a computationally difficult set covering problem", *Operations Research Letters*, v. 8, pp. 67–71, 1989.

[39] FEO, T., RESENDE, M., SMITH, S., "A greedy randomized adaptive search procedure for maximum independent set", *Operations Research*, v. 42, pp. 860–878, 1994.

[40] RESENDE, M., RIBEIRO, C., "A GRASP with path-relinking for private virtual circuit routing", *Networks*, v. 41, pp. 104–114, 2003.

[41] RESENDE, M., RIBEIRO, C., "Greedy randomized adaptive search procedures", In: GLOVER, F., KOCHENBERGER, G. (eds), *Handbook of Metaheuristics*, pp. 219–249, Kluwer, 2003.

[42] RESENDE, M., RIBEIRO, C., "GRASP with path-relinking: Recent advances and applications", In: IBARAKI, T., NONOBE, K., YAGIURA, M. (eds), *Metaheuristics: Progress as real problem solvers*, pp. 29–63, Springer, 2005.

[43] GOLDBERG, D., *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Publishing, 1989.

[44] DÍAZ, I., ZABALA, P., "A Branch-and-Cut Algorithm for Graph Coloring", *Discrete Applied Mathematics*, v. 154, pp. 826–847, 2006.

[45] JAUMARD, B., "Network and traffic data sets for optical network optimization", online publication on http://users.encs.concordia.ca/~bjaumard, last visited on January 3th, 2008.

[46] FROTA, Y., MACULAN, N., NORONHA, T. F., et al., "A Branch and Cut Algorithm for the Partition Coloring Problem". In: *Proceedings of the International Network Optimization Conference*, Spa, 2007.

[47] FROTA, Y., MACULAN, N., NORONHA, T., et al., "A Branch-and-Cut Algorithm for Partition Coloring", *Networks*, v. to appear, 2007.

[48] MEYER, W., "Equitable coloring", *American Mathematical Monthly*, v. 80, pp. 143–149, 1973.

[49] TUCKER, A., "Perfect graphs and an application to optimizing municipal services", *SIAM Review*, v. 15, pp. 585–590, 1973.

[50] DAS, S. K., FINOCCHI, I., PETRESCHI, R., "Conflict-free star-access in parallel memory systems", *J. Parallel Distrib. Comput.*, v. 66, n. 11, pp. 1431–1441, 2006.

[51] LIH, K., "The Equitable Coloring of Graphs". In: *Handbook of Combinatorial Optimization*, pp. 543–566, Kluwer Acad. Publ.: Boston, MA, 1998.

[52] CHEN, B., LIH, K., "Equitable Coloring of Trees", *Journal of Combinatorial Theory – Series B*, v. 61, 1994.

[53] CHEN, B., KO, M., LIH, K., "Equitable and m-Bounded Coloring of Split Graphs". In: *Combinatorics and Computer Science – Brest 1995*, v. 1120, *Lecture Notes in Computer Science*, pp. 1–5, Springer – Berlin, 1996.

[54] BAKER, B., COFFMAN, E., "Mutual exclusion scheduling", *Theoret. Comput. Sci.*, v. 162, pp. 225–243, 1996.

[55] HAJNAL, A., SZEMERÉDI, E., "Proof of a conjecture of P. Erdos Combinatorial. Theory and its Applications II". In: *(Ed. P.Erdos, A. Renyi e VT Sos) Colloq. Soc. J. Bolyay 4*, pp. 601–623, Atlanta, 1970.

[56] BAHIENSE, L., FRIEDMAN, C., JURKIEWICZ, S., et al., "An Integer Programming Approach To Equitable Coloring Problems". In: *Technical Report RT-EP 001/07*, COPPE-Produção/UFRJ, 2007.

[57] TOUHAMI, S., "Optimization Problems in cellular Networks". In: *Ph.D. dissertation*, 2004.

[58] DANNA, E., ROTHBERG, E., PAPE, C. L., "Exploring relaxation induced neighborhoods to improve MIP solutions". In: *Mathematical Programming, 102, 71-90*, 2005.

[59] FISCHETTI, M., LODI, A., "Local Branching". In: *Mathematical Programming 98, 23-47*, 2003.

[60] DANNA, E., PAPE, C. L., "Two Generic Schemes for Efficient and Robust Cooperative Algorithms. Constraint and Integer Programming". In: *Michela Milano (ed.), Kluwer Academic Publishers, 33-57*, 2003.

[61] JOHNSON, D., TRICK, M., "Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge". In: *Vol 26 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society*, 1996, Também encontrado em *http://mat.gsia.cmu.edu/COLOR/instances.html*.

[62] JONHSON, D. S., ARAGON, C. R., MCGEOCH, L. A., et al., "Optimization by Simulated Annelaing: An Experimental Evaluation; Part II, Graph

Coloring and Number Partitioning". In: *Operations Research, 31, 378-406*, 1991.

[63] MYCIELSKI, J., "Sur le coloriage des graphes". In: *Colloquium Mathematiques, Vol. 3, pp. 161-162*, 1955.

[64] LARSEN, M., PROPP, J., ULLMAN, D., "The fractional chromatic number of a graph and a construction of Mycielski". In: *preprint*, 1994.

[65] HALE, W. K., "Frequency Assigment: Theory and Application". In: *Proceedings of IEEE*, v. 68(12), pp. 1497–1513, 1980.

[66] AARDAL, K. I., VAN HOESEL, S. P. M., KOSTER, A. M. C. A., et al., "Models and solution techniques for frequency assignment problems". In: *Annals of Operations Research*, v. 153, pp. 79–129, Springer Netherlands, 2007.

[67] COSTA, D., "On the use of some known methods for t-colourings of graph". In: *Annals of Operations Research*, v. 41, pp. 343–358, 1993.

[68] HAO, J. K., PERRIER, L., "Tabu search for frequency assignment problem in celular radio networks". In: *Jornal of Heuristics*, v. 4, pp. 47–62, 1998.

[69] VALENZUELA, C., HURLEY, S., SMITH, D. H., "A permutation based genetic algorithm for minimum span frequency assigment". In: *Lecture Notes in Computer Science*, v. 1498, pp. 907–916, 1998.

[70] RAYCHAUDHURI, A., "Further Results on T-Coloring and Frequency Assignment Problems", *SIAM J. Discret. Math.*, v. 7, n. 4, pp. 605–613, 1994.

[71] ALLEN, S. M., SMITH, D. H., HURLEY, S., "Lower bounding techniques for frequency assignment". In: *Discrete Mathematics*, v. 197/198, pp. 41–52, 1999.

[72] JANSSEN, J. C. M., KILAKOS, K., "An optimal solution to the Philadelphia channel assignment problem". In: *IEEE Transactions on Vehicular Technology*, v. 48, pp. 1012–1014, 1999.

[73] TCHA, D., CHUNG, Y., CHOI, T., "A new lower bound for frequency assignment problem". In: *IEEE/ACM Transactions on Networking*, v. 5, pp. 34–39, 1997.

[74] JANSSEN, J. C. M., WENTZELL, T., "Lower bounds from tile covers for the channel assignment problem". In: *Tech-Report G-2000-09, GERAD, HEC*, Montreal, Canada, 2000, available at http://www.mscs.dal.ca/ janssen/cv/research.html.

[75] AARDAL, K., HIPOLITO, A., HOESEL, C., et al., *A branch-and-cut algorithm for the frequency assignment problem*, Tech. rep., Maastricht : METEOR, Maastricht Research School of Economics of Technology and Organization, 1998, available at http://ideas.repec.org/p/dgr/umamet/1996007.html.

[76] AVENALI, A., MANNINO, C., SASSANO, A., "Minimizing the span of d-walks to compute optimum frequency assignments". In: *Mathematical Programming*, v. 91, pp. 357–374, 2002.

[77] MANNINO, C., SASSANO, A., "An enumerative algorithm for the frequency assignment problem". In: *Discrete Applied Mathematics*, v. 129, pp. 155–169, 2003.

[78] CHRISTOFIDES, N., "Graph Theory: An algorithmic approach". In: *Academic Press*, London, 1975.

[79] EISENBLATTER, A., "Frequency assignment in GSM networks: models, heuristics, and lower bounds". In: *Ph.D. dissertation*, 2001.

[80] SMITH, D. H., HURLEY, S., THIEL, S. U., "Improving heuristics for the frequency assigment problem". In: *Europian Journal of Operational Research*, v. 107, pp. 76–86, 1998.

[81] EISENBLATTER, A., KOSTER, A., "FAP web - A website about Frequency Assignment Problems". 2000, *http://www.zib.de/fap/*.

[82] MANIEZZO, V., MONTEMANNI, R., "An exact algorithm for the min-interference frequency assigment problem". In: *Technical Report CSR99-02, Scienze dell'Informazione, Universitá di Bologna*, 2000.