UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

JOSÉ FAGNER RODRIGUES MEDEIROS

ESTRATÉGIAS DE RESOLUÇÃO EXATA PARA O PROBLEMA DO CORTE GLOBAL ROTULADO MÍNIMO

João Pessoa

2020

JOSÉ FAGNER RODRIGUES MEDEIROS

ESTRATÉGIAS DE RESOLUÇÃO EXATA PARA O PROBLEMA DO CORTE GLOBAL ROTULADO MÍNIMO

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Centro de Informática da Universidade Federal da Paraíba, como requisito parcial para obtenção do Grau de Mestre em Informática.

Orientador: Prof. Dr. Gilberto Farias de Sousa Filho

Coorientador: Prof. Dr. Thiago Gouveia da Silva

João Pessoa

2020

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de José Fagner Rodrigues Medeiros, candidato ao título de Mestre em Informática na Área de Sistemas de Computação, realizada em 06 de julho de 2020.

Aos seis dias do mês de julho do ano de dois mil e vinte, às quatorze horas, por meio de videoconferência, reuniram-se os membros da Banca Examinadora constituída para julgar o José Fagner Rodrigues Medeiros, vinculado a esta Universidade sob a matrícula nº 20181000920, candidato ao grau de Mestre em Informática, na área de "Sistemas de Computação", na linha de pesquisa "Computação Distribuída", do Programa de Pós-Graduação em Informática, da Universidade Federal da Paraíba. A comissão examinadora foi composta pelos professores: Gilberto Farias de Sousa Filho (PPGI-UFPB) Orientador e Presidente da Banca, Lucídio dos Anjos Formiga Cabral (PPGI-UFPB), Examinador Interno, Teobaldo Leite Bulhões Junior (UFPB), Examinador Externo ao Programa, Thiago Gouveia da Silva (IFPB), Examinador Externo à Instituição. Dando início aos trabalhos, o Presidente da Banca cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse a exposição oral do trabalho de dissertação intitulado: "Estratégias de Resolução Exata para o Problema do Corte Global Rotulado Mínimo". Concluída a exposição, o candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: "***aprovado***". Do ocorrido, eu, Ruy Alberto Pisani Altafim, Coordenador do Programa de Pós-Graduação em Informática, lavrei a presente ata que vai assinada por mim e pelos membros da banca examinadora. João Pessoa, 06 de julho de 2020.

Prof. Dr. Ruy Alberto Pisani Altafim

Prof. Gilberto Farias de Sousa Filho
Orientador (PPGI-UFPB)

Prof. Lucídio dos Anjos Formiga Cabra
Examinador Interno (PPGI-UFPB)

Prof. Teobaldo Leite Bulhões Junior
Examinador Externo ao Programa (UFPB)

Prof. Thiago Gouveia da Silva
Examinador Externo à Instituição (IFPB)

Catalogação na publicação
Seção de Catalogação e Classificação

M488e Medeiros, José Fagner Rodrigues.
Estratégias de resolução exata para o problema do corte global rotulado mínimo / José Fagner Rodrigues Medeiros. - João Pessoa, 2020.
56 f. : il.

Orientação: Gilberto Farias de Sousa Filho Sousa Filho.
Coorientação: Thiago Gouveia da Silva Silva.
Dissertação (Mestrado) - UFPB/CI.

1. Grafos com arestas rotuladas. 2. Conectividade. 3. Programação matemática. I. Sousa Filho, Gilberto Farias de Sousa Filho. II. Silva, Thiago Gouveia da Silva. III. Título.

UFPB/BC

RESUMO

Neste trabalho, abordamos o Problema do Corte Global Rotulado Mínimo (PCGRM), que é um problema de análise combinatória e pode ser definido formalmente como: seja $G = (V, E, L)$ um grafo com arestas rotuladas, no qual V é o conjunto de vértices de G, E é o conjunto de arestas, L é o conjunto de rótulos (cores) sobre E e cada aresta $e \in E$ possui um rótulo $L(e)$ associado; o PCGRM tem como objetivo encontrar um subconjunto de rótulos $L' \subseteq L$ de modo que o grafo $G = (V, E', L \backslash L')$ seja desconexo e $|L'|$ seja minimizado. Então, com o objetivo de solucionar este problema, desenvolvemos algumas estratégias de resolução exata, estendemos e adaptamos os conceitos de fecho cromático, e desenvolvemos uma nova família de formulações matemáticas chamada $\mathcal{MF}_d$. Para construção do $\mathcal{MF}_d$, tivemos como base um modelo presente na literatura chamado $PART_2$, que é definido em Silva et al (2016). Os experimentos computacionais demonstraram que o modelo proposto neste trabalho obteve uma grande melhoria de tempo em relação ao modelo $PART_2$.

Palavras-chave: grafos com arestas rotuladas, *branch-and-cut*, conectividade.

ABSTRACT

In this work, we approach the Minimum Labeling Global Cut Problem (MLGCP), which is a combinatorial analysis problem and can be formally defined as: Let $G = (V, E, L)$ be an edge-labeled graph in which V is the set of vertices of G, E is the set of edges, L is the set of labels (colors) on E and each edge $e \in E$ has a label $L(e)$ associated; The goal of MLGCP is to find a subset $L' \subseteq L$ of labels such that $G = (V, E', L \backslash L')$ is not connected and $|L'|$ is minimized. So, in order to solve this problem, we developed some strategies for exact resolution, extended and adapted the concept of chromatic closure, and developed a new family of mathematical formulations called $\mathcal{MF}_d$. For the construction of the $\mathcal{MF}_d$, we were based on a model present in literature called $PART_2$, defined in Silva et al (2016). The computational experiments demonstrated that the model proposed in this work obtained a great improvement of time compared to the $PART_2$ model.

Keywords: edge-labeled graphs, branch-and-cut, connectivity.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

GAR	Grafos com Arestas Rotuladas
PCGRM	Problema do Corte Global Rotulado Mínimo
PAGRM	Problema da Árvore Geradora com Rotulação Mínima
PERM	Problema do Emparelhamento Rotulado Máximo
PCVC	Problema do Caixeiro Viajante Colorido
PCCM	Problema do Corte Colorido Mínimo
PCGM	Problema do Corte Global Máximo
PART2	Modelo Baseado em Particionamento
P3E	Modelo Baseado em Agrupamento de Vértices
EAC	Modelo Baseado em Eliminação de Árvores
B&B	*Branch-and-Bound*
B&C	*Branch-and-Cut*
VND	*Variable Neighborhood Descent*
VNS	*Variable Neighborhood Search*
MVCA	*Maximum Vertex Covering Algorithm*
PLIM	Programação Linear Inteira Mista
PLI	Programação Linear Inteira
PL	Programação Linear

1 INTRODUÇÃO

Problemas definidos sobre Grafos com Arestas Rotuladas (coloridas) (GAR) têm atraído bastante atenção nos últimos anos. Nestes grafos, em vez de peso, cada aresta possui um rótulo (ou cor) associado. Um problema clássico e certamente o mais estudado dentre os problemas definidos sobre GAR é o Problema da Árvore Geradora com Rotulação Mínima (PAGRM), proposto por Chang e Leu (1997). Esse problema consiste em, dado um GAR, encontrar uma árvore geradora que utilize o menor número de rótulos.

Vários trabalhos despertaram o interesse em pesquisas envolvendo GAR. Estes são encontrados na literatura, como o Problema do Emparelhamento Rotulado Máximo (PERM) (Carrabs et al, 2009), Problema do Clique Rotulado Máximo (Carrabs et al, 2014), Problema de Cobertura do Ciclo do Arco-íris (Silvestri et al, 2016) e Problema do Caixeiro Viajante Colorido (PCVC) (Xiong et al, 2007). Informações sobre outros problemas definidos sobre grafos rotulados podem ser encontrados na revisão realizada por Granata et al (2013) e na Tese de Silva (2019).

Neste trabalho apresentamos algumas definições e conceitos de fecho cromático, que é um nova maneira de representar GARs. Este conceito foi proposto na tese de Silva (2019) sendo estendido e adptado para o objetivo desta pesquisa. Dado o conceito de fecho cromático, propomos uma nova família de formulações matemáticas chamada $\mathcal{MF}_d$, que é um modelo de Programação Linear Inteira Mista (PLIM), para resolver o Problema do Corte Global Rotulado Mínimo (PCGRM). Ainda como contribuições, desenvolvemos um algoritmo *branch-and-cut* (B&C) para resolver o $\mathcal{MF}_d$ e algumas estratégias de *branching*.

Por fim, os experimentos computacionais realizados demonstram que o modelo $\mathcal{MF}_*$ obteve uma grande melhoria de tempo em relação ao modelo $PART_2$, tornando-se, com o melhor do nosso conhecimento, o modelo exato com melhores resultados para o problema.

1.1 Definição do Tema

Nesta seção introduzimos algumas definições e conceitos básicos do PCGRM.

O PCGRM é um problema de Otimização Combinatória e tem aplicações em diferentes áreas. Este problema tem como entrada um GAR conexo não orientado e o objetivo é encontrar o menor número de rótulos no grafo tal que a retirada das arestas associadas a estes rótulos resulte em um grafo desconexo. A Figura 1.1 ilustra o problema: na Figura 1.1(a) temos o grafo de entrada, a Figura 1.1(b) exibe um corte composto pelos rótulos E e F e a Figura 1.1(c) mostra que o grafo sem os rótulos do corte é desconexo.

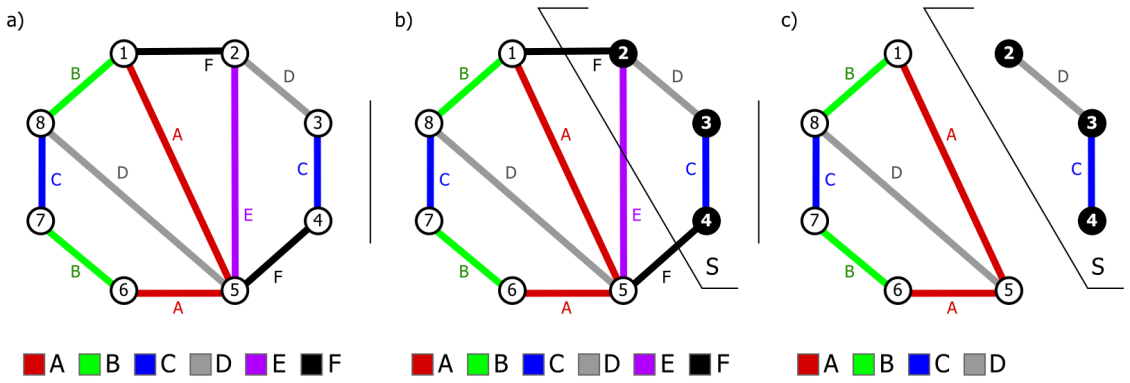

Figura 1.1: Instância exemplo para o PCGRM.

Formalmente, o PCGRM pode ser descrito como: dado um grafo não orientado $G = (V, E, L)$, sendo V o conjunto de vértices de G, E o conjunto de arestas e L o conjunto de rótulos (cores) sobre E, onde cada aresta $e \in E$ possui um rótulo $L(e)$ associado; o objetivo é encontrar um conjunto de rótulos $L' \subseteq L$ tal que o grafo $G = (V, E, L \backslash L')$ seja desconexo e $|L'|$ seja minimizado.

Zhang (2014) demonstra que o PCGRM pode ser resolvido em tempo polinomial em alguns casos especiais, isto é, quando o parâmetro f_{max} é limitado, onde o autor define f_{max} como sendo o número máximo de aparências de um rótulo no grafo de entrada, quando a largura em árvore é limitada e o grafo de entrada é planar. A complexidade computacional deste problema ainda continua como uma questão teórica em aberto (Xu e Faragó, 2019).

1.2 Justificativa

Nos útimos anos, com a disseminação de redes multicamadas MPLS e IP/WDM, vários cenários de falhas se tornaram inevitáveis (Coudert et al, 2007). Então, dado este cenário, desenvolver redes com um nível baixo de vulnerabilidade, se tornou um aspecto

de suma importância quando falamos em projetar redes. Faragó (2006) define como quantificar a vulnerabilidade de uma rede: seja uma topologia de rede modelada por um grafo conexo não orientado, e considerarmos falhas de *link*, a medição da vulnerabilidade desta rede é o tamanho de um corte mínimo, isto é, remover o número mínimo de arestas (*links*) para desconectar o grafo. Essa medição é chamada de conectividade do grafo. Então, podemos observar, que quanto menos falhas de *links* puderem desconectar a rede, mais vulnerável a rede será.

Um fator importante em relação a falhas de *link* é que, em muitos cenários, vários conjuntos de *links* são propensos a falhas simultâneas. Por exemplo, em uma rede *Wireless*, todos os enlaces em uma certa frequência podem ser derrubados por um adversário adicionando um forte sinal de ruído. Faragó (2006) lista outros exemplos de falhas simutâneas de *links*. Citamos alguns destes:

- Se, em uma rede sem fio, um adversário transmite um sinal de interferência em uma determinada área, todos os *links* próximos o suficiente provavelmente falham juntos;
- Se um nó for destruído, todos os *links* adjacentes falharão;
- Se os *links* funcionarem em diferentes bandas de frequência em uma rede sem fio, o bloqueio de uma banda de frequência em uma determinada área poderá inibir todos os *links* na área que usam a determinada faixa de frequência, enquanto outros podem permanecer intactos;
- Em uma rede com fio, vários cabos podem compartilhar seu caminho físico no mesmo duto. Nesse caso, se o duto estiver danificado, é provável que todos os cabos nele falhem juntos;
- Se existirem redes lógicas de sobreposição em uma rede física, vários *links* lógicos poderão usar o mesmo *link* físico. Então, a falha de um único *link* físico pode resultar na falha de um conjunto (possivelmente grande) de *links* lógicos, que naturalmente falham juntos.

Portanto, o PCGRM está intimamente relacionado à vulnerabilidade de redes multicamadas e tem como objetivo medir a conectividade da rede quando seus links possuem riscos compartilhados de falha (Coudert et al, 2007, 2016).

Além de vulnerabilidade de redes, este problema pode ser aplicado em sistemas de planejamento de transporte público de ônibus, onde as regiões servidas por ônibus são

os vértices, as linhas que ligam as regiões são as arestas e as empresas prestadoras do serviço são os rótulos. Seguindo este escopo, a solução do problema é fornecer um número mínimo de empresas que ao pararem de trabalhar desconectam duas regiões, tornando-as inacessíveis entre si.

A Figura 1.2 ilustra a aplicação do PCGRM em sistemas de planejamento de transporte. O conjunto dos rótulos $L = \{A, B, C, D, E, F\}$ representa as empresas prestadoras do serviço público e o conjunto dos vértices $V = \{1, 2, 3, 4, 5, 6, 7\}$ representa as regiões do mapa.

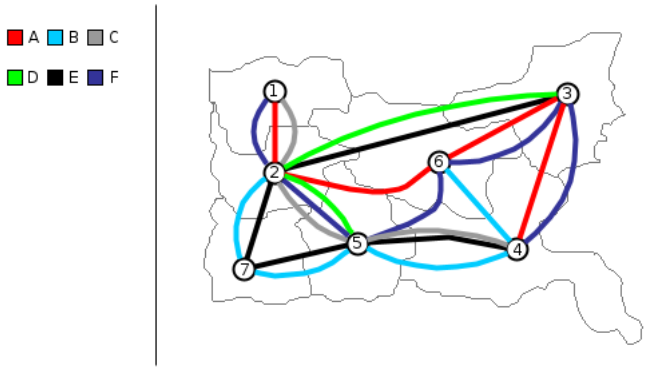

Figura 1.2: Exemplo do PCGRM aplicado ao sistema de planejamento de transporte.

1.3 Objetivos

Nesta seção, apresentamos o objetivo geral e os objetivos específicos do trabalho.

1.3.1 Objetivo Geral

Propor estratégias de resolução exata para o Problema do Corte Global Rotulado Mínimo.

1.3.2 Objetivos Específicos

- Propor novas formulações matemáticas para o problema.
- Propor novas desigualdades válidas para as formulações.
- Desenvolver algoritmos para construção de fechos cromáticos.

- Criação de novos algoritmos de separação das novas desigualdades propostas a serem adicionadas à estratégia de *branch-and-cut* adotada.
- Desenvolver uma estratégia *pre-branching*.
- Comparar os modelos propostos.

1.4 Estrutura do Trabalho

O restante do trabalho está organizado da seguinte forma:

- No Capítulo 2, são apresentados alguns conceitos de Teoria dos Grafos, o método *branch-and-bound* para resolução de PLIM e uma breve revisão da literatura.
- No Capítulo 3, são descritos alguns métodos para solucionar o PCGRM.
- No Capítulo 4, é apresentada toda formulação matemática do $\mathcal{MF}_d$, os conceitos de fecho cromático e também os algoritmos *branch-and-cut* e *pre-branching* propostos neste trabalho.
- No Capítulo 5, são apresentados os resultados dos experimentos computacionais realizados.

2 FUNDAMENTAÇÃO TEÓRICA E REVISÃO DA LITERATURA

Neste capítulo, são apresentados alguns conceitos que envolvem o trabalho desenvolvido, como Teoria dos Grafos e o Método *branch-and-bound* para resolução de PLIM.

2.1 Teoria dos Grafos

Nesta seção, apresentamos uma breve introdução sobre Teoria dos Grafos. Um estudo mais detalhado sobre Teoria dos Grafos pode ser encontrado no livro de Bondy e Murty (2008). Para realizar o estudo das subseções 2.1.2 e 2.1.3, tivemos como base o livro Cormen et al (2009).

2.1.1 Conceitos Básicos

Um grafo é um par ordenado $G = (V, E)$ que consiste em um conjunto de vértices não vazio V e um conjunto de arestas E, sendo cada aresta e formada por um par de vértices $i, j \in V$, assim, uma aresta é representada por $e(i, j)$. Grafos são chamados assim porque são representados graficamente, e é essa representação que nos ajuda a entender muitas propriedades. A Figura 2.1 mostra um grafo $G = (V, E)$ com $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$ e $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}$ e um grafo $H = (V, E)$ com $V = \{1, 2, 3, 4, 5, 6\}$ e $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$.

Pode-se dizer que um grafo é simples se ele não possuir *laços* ou arestas *paralelas*. Os grafos G e H da figura 2.1 são grafos simples. Uma aresta $e(v, u)$ com extremidades iguais $v = u$ é chamada de *laço*. Duas ou mais arestas com o par de extremidades iguais são arestas *paralelas*. A Figura 2.2 mostra um grafo com um *laço*, que é a aresta e_1, as arestas e_5 e e_6 são arestas *paralelas*.

Pode-se dizer que dois vértices v e u de um grafo $G = (V, E)$ são *adjacentes*, se

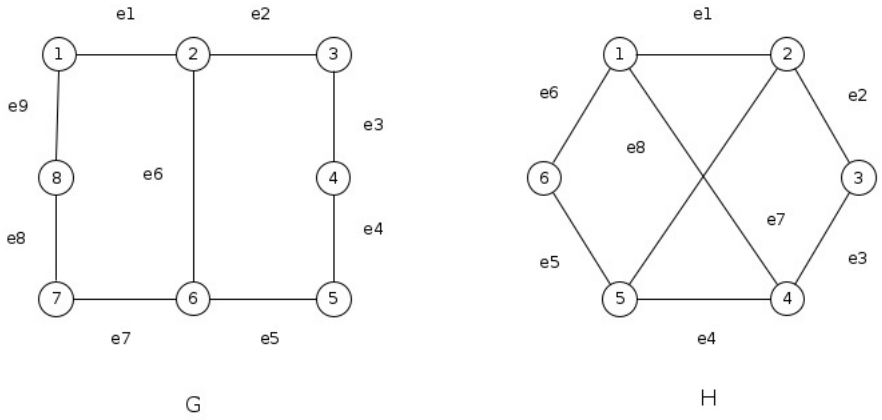

Figura 2.1: Exemplo de grafos G e H.

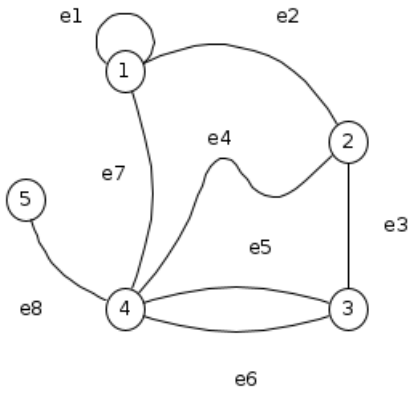

Figura 2.2: Exemplo de grafo com *laço* e arestas *paralelas*.

e somente se existir uma aresta $e(v,u) \in E$. Neste caso, a aresta $e(v,u)$ é *incidente* aos vértices v e u e os vértices v e u são as extremidades da aresta $e(v,u)$. Denota-se um conjunto de *vizinhos* de um vértice v no grafo G por $N_G(v)$. O grau de um vértice v, $\delta(v)$, em grafo simples, é igual ao número de arestas que incidem em v, portanto, $\delta(v) = |N_G(v)|$.

Alguns tipos de grafos desempenham funções importantes na Teoria dos Grafos. Um grafo *completo* é um grafo simples, no qual, quaisquer dois vértices do grafo são *adjacentes*. Pode-se dizer que um grafo é *bipartido* se seu conjunto de vértices puder ser particionado em um subconjunto X e um subconjunto Y; essa partição (X,Y) é chamada de bipartição do grafo e, X e Y suas partes. Um grafo *bipartido* G com bipartição (X,Y) é denotado por $G[X,Y]$. $G[X,Y]$ é dito um grafo *bipartido completo* se for simples e todos os vértices em X forem unidos a todos os vértices em Y. Uma *estrela* é um $G[X,Y]$ com $|X| = 1$ ou $|Y| = 1$. A Figura 2.3 mostra um grafo *completo*, um grafo *bipartido completo* e um *estrela*.

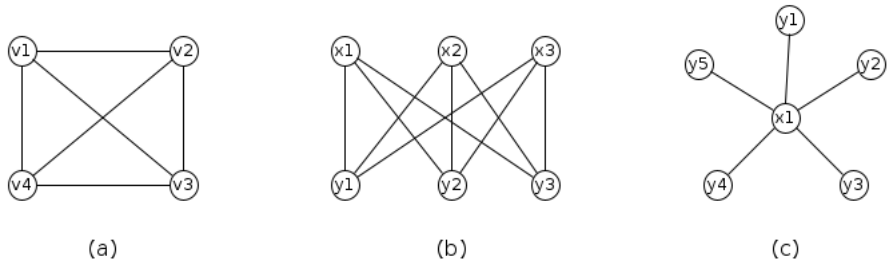

Figura 2.3: (a) grafo *completo*, (b) grafo *bipartido completo*, (c) grafo *estrela*.

O grafo $H = (V, E)$, onde $V(H)$ é o conjunto de vértices de H e $E(H)$ o conjunto de arestas de H, é um subgrafo do grafo $G = (V, E)$, onde $V(G)$ é o conjunto de vértices de G e $E(G)$ o conjunto de arestas de G, se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$. Se $V(H) = V(G)$, então H é um subgrafo gerador de G. Seja S, um subconjunto de vértices, onde $S \subseteq V$, o subgrafo de G é definido pelos vértices de S e pelas arestas de G que incidem sobre dois vértices quaisquer de S, onde este subgrafo é induzido por S. A Figura 2.4 mostra um grafo G e um grafo H, que é subgrafo de G.

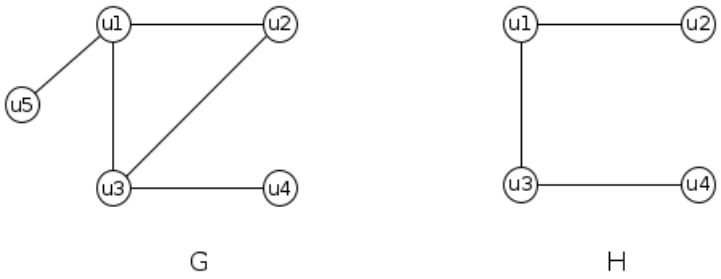

Figura 2.4: Exemplo de subgrafo.

Em um grafo G, um *caminho* é representado por uma sequência de vértices distintos $u_1, u_2, ..., u_k$ onde os vértices u_i e u_{i+1} são *vizinhos*, $\forall i \in \{1, 2, ..., k-1\}$. Pode-se dizer que um *ciclo* é um *caminho* fechado que dada uma sequência de vértices $u_1, u_2, ..., u_k$, u_i e u_{i+1} são *vizinhos*, $\forall i \in \{1, 2, ..., k-1\}$, onde todos os vértices dessa sequência são distintos, menos o vértice inicial e o vértice final, que são iguais. A Figura 2.5 mostra um *caminho* de tamanho três e um *ciclo* de tamanho cinco.

Pode-se dizer que um grafo G é *conexo* quando existir um caminho em G com extremidades em v e u, para dois vértices v e u quaisquer. Caso contrário, G é um grafo *desconexo*. A Figura 2.6 exemplifica um grafo *conexo* e um *desconexo* respectivamente.

Um grafo que não contém ciclos é um grafo *acíclico*, assim, uma *árvore* T é um grafo conexo que não possui ciclos. Então, dada uma *árvore* T, ela pode ser dita uma

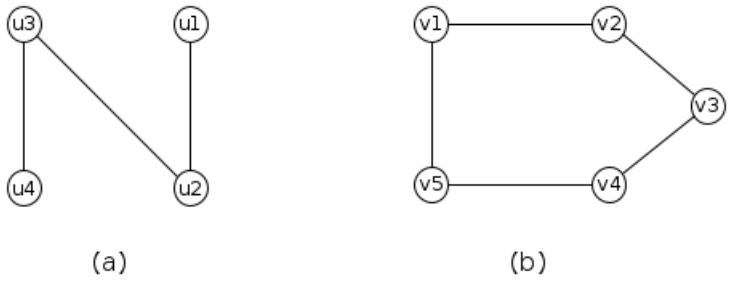

Figura 2.5: (a) *Caminho* de tamanho três, (b) *Ciclo* de tamanho cinco.

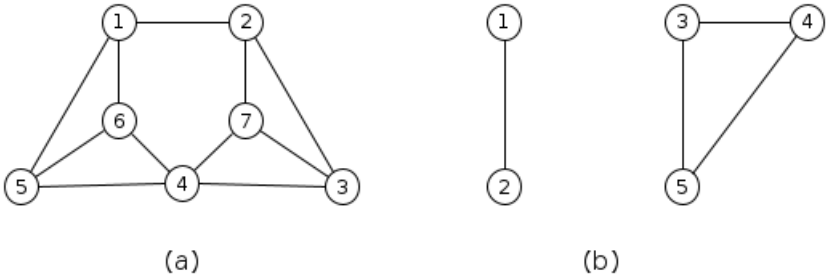

Figura 2.6: (a) grafo *conexo*, (b) grafo *desconexo*.

árvore geradora de um grafo G se esta for um subgrafo gerador de G.

2.1.2 Representação de Grafos

Um grafo $G = (V, E)$ pode ser representado computacionalmente através de dois modos padrão. *Lista de adjacências* ou como uma *matriz de adjacências*. Esses dois modos podem representar grafos orientados ou não orientados.

Uma *lista de adjacências* consiste em um arranjo de $|V|$ listas, sendo uma para cada vértice $v \in V$. A *lista de adjacências* de um vértice v contém todos vértices $u \in V$ tal que exista uma aresta $(v, u) \in E$, ou seja, a *lista de adjacências* de v é formada por todos os vértices adjacentes a ele. A Figura 2.7 ilustra a *lista de adjacências*.

Uma desvantagem da *lista de adjacências* é verificar se uma aresta (v, u) está no grafo G, pois, tem que ser feito uma busca linear na *lista de adjacências* de v verificando se $u \in Adj[v]$, onde sua complexidade é $O(n)$.

Para representação de um grafo $G = (V, E)$ através de uma *matriz de adjacências*, supondo que os vértices são enumerados $1, 2, ..., |V|$ de forma arbitrária, então, a *matriz de adjacências* de um grafo consiste em uma matriz $|V| \times |V|$ $A = (a_{ij})$ tal que

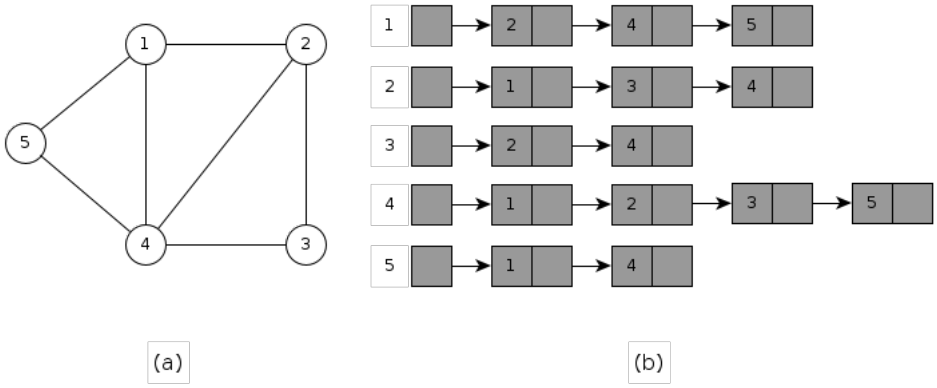

Figura 2.7: (a) Grafo exemplo, (b) *lista de adjacências*.

$$a_{ij} = \begin{cases} 1 & \text{se } (i,j) \in E, \\ 0 & \text{em caso contrário.} \end{cases}$$

A Tabela 2.1 representa a *matriz de adjacências* do grafo exemplo da Figura 2.7 (a).

Tabela 2.1: Matriz de adjacências A_{ij}

A_{ij}	1	2	3	4	5
1	0	1	0	1	1
2	1	0	1	1	0
3	0	1	0	1	0
4	1	1	1	0	1
5	1	0	0	1	0

Matriz de adjacências pode contornar essa desvantagem que a *lista de adjacências* tem ao verificar se uma aresta (v, u) está no grafo G. Desse modo, podemos verificar se (v, u) está em G em $O(1)$, pois, basta consultar a linha v e a coluna u da matriz. Porém, uma desvantagem da *matriz de adjacências* é listar os vizinhos de um vértice v de G, onde a complexidade é $O(n)$.

2.1.3 Algoritmos de Busca em Grafos

A seguir, serão apresentados dois algoritmos básicos para realizar busca em grafo. Quando falamos em realizar uma busca, isso significa que devemos percorrer de forma sistemática as arestas do grafo afim de visitar todos o vértices.

A busca em largura é um algoritmo simples para realizar busca em um grafo. Dado um grafo $G = (V, E)$ e um vértice raiz s, a busca em largura explora as arestas de G

a partir de s descobrindo cada vértice que pode ser alcançado. O algoritmo calcula a distância de s até cada vértice alcançado e produz uma "árvore de busca em largura". Para qualquer vértice v que pode ser alcançado a partir de s, o caminho simples de s até v corresponde a um *caminho mínimo* de s a v em G.

Para se obter o controle no funcionamento do algoritmo, cada vértice pode ser marcado de branco, cinza ou preto. No início do algoritmo, todos os vértices são brancos, e a partir do momento que vão sendo visitados se tornam cinza ou preto. Um vértice preto significa que todos os seus vértices adjacentes já foram visitados, já os vértices com a cor cinza podem ter vértices adjacentes brancos, ou seja, não visitados.

Como dito anteriormente, o algoritmo constrói uma árvore em largura, que inicialmente contém apenas a raiz s. Sempre que um vértice v branco é encontrado na lista de adjacência de um vértice u já visitado, o vértice v e a aresta (u, v) são adicionadas à árvore. O procedimento do BFS mostrado no Algoritmo 1 supõe que o grafo $G = (V, E)$ de entrada é representado com a utilização de listas de adjacências.

Algoritmo 1 BFS

Entrada: Grafo G e um vértice raiz s

```
1:  para cada vértice u ∈ V[G] − {s} faça
2:      u.cor = BRANCO
3:      u.d = ∞
4:      u.π = NIL
5:  s.cor = CINZA
6:  s.d = 0
7:  s.π = NIL
8:  Q = ∅
9:  ENFILEIRAR(Q,s)
10: enquanto Q ≠ ∅ faça
11:     u = DESENFILEIRAR(Q)
12:     para cada v = Adj[u] faça
13:         se v.cor == BRANCO então
14:             v.cor = CINZA
15:             u.d = u.d + 1
16:             v.π = u
17:             ENFILEIRAR(Q,v)
18:         u.cor = PRETO
```

Descrevemos o funcionamento do algoritmo a seguir. As linhas 1-4, com exceção do vértice raiz s, cada vértice $u \in G$ é definido como branco, $u.d$ como infinito e $u.\pi$ com NIL. Nas linhas 5-7, s é definido como cinza, $s.d = 0$ e $s.\pi$ com NIL, ou seja, a busca em largura começará a partir do vértice raiz s.

O laço das linhas 10-18 executa até que a fila Q seja vazia, ou seja, enquanto houver vértices cinzas. Na linha 11, u recebe o primeiro vértice da fila Q e o remove da fila. No laço das linhas 12-17, considera cada vértice v adjacente a u. Na linha 11, se v é branco quer dizer que ainda não foi visitado, então v é marcado de cinza, $v.d$ é incrementado e o pai de v é u. O vértice v é inserido na fila Q (linha 17). Por fim, u é marcado de preto, o que significa que todos os vértices da sua lista de adjacência já foram visitados. O tempo de execução total do algoritmo BFS é $O(V + E)$, assim, a busca em largura é executada em tempo linear em relação ao tamanho da representação por listas de adjacências de G.

A Figura 2.8 ilustra a execução do BFS em um grafo exemplo não orientado.

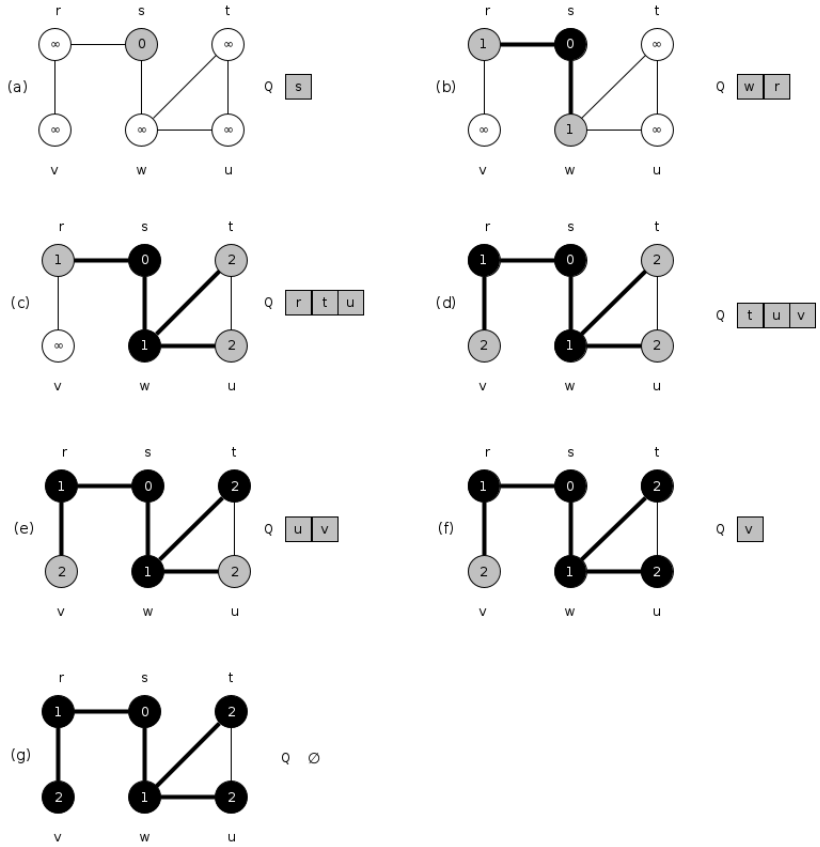

Figura 2.8: Execução do algoritmo BFS.

A estratégia adotada pela busca em profundidade é ir "mais fundo" no grafo sempre que possível. O algoritmo explora as arestas a partir do vértice v recém visitado. Depois de explorar todas as arestas de v a busca retorna pelo mesmo caminho até o vértice predecessor de v para explorar as arestas que partem deste vértice. Esse procedimento se repete até que todos os vértices possíveis sejam visitados a partir do vértice raiz inicial. Caso reste vértices que ainda não foram visitados, a busca em profundidade seleciona um deles como raiz e inicia o mesmo processo a partir deste vértice selecionado.

A busca em profundidade marca cada vértice para indicar seu estado atual, seguindo o mesmo processo que ocorre na busca em largura. Os vértices iniciam brancos, são marcados de cinza quando visitado e preto quando termina de visitar todos os seus vértices adjacentes. Ao encontrar um vértice v na lista de adjacências de u, o "pai" de v será u, ou seja, o atributo $v.\pi = u$. Os atributos $u.d$ e $u.f$ representam dois carimbos de tempo, $u.d$ registra quando o vértice é visitado pela primeira vez e $u.f$ registra o término da busca na lista de adjacências deste vértice.

A seguir, o algoritmo básico de busca em profundidade é representado (Algoritmo 2). O DFS "chama" a sub-rotina DFS-visit que é descrita no Algoritmo 3.

Algoritmo 2 DFS

Entrada: Grafo G

```
1: para cada vértice u ∈ V[G] faça
2:     u.cor = BRANCO
3:     u.π = NIL
4: tempo = 0
5: para cada vértice u ∈ V[G] faça
6:     se u.cor == BRANCO então
7:         DFS-Visit(G, u)
```

Algoritmo 3 DFS-Visit

Entrada: Grafo G e vértice u

```
1: tempo = tempo + 1
2: u.d = tempo
3: u.cor = CINZA
4: para cada vértice v ∈ Adj[u] faça
5:     se v.cor == BRANCO então
6:         v.π = u
7:         DFS-Visit(G, v)
8: u.cor = PRETO
9: tempo = tempo + 1
10: u.f = tempo
```

Nas linhas 1-3 todo vértice $u \in G$ é marcado de branco e seus "pai" são inicializados como NIL. Verifica-se cada vértice de $v \in V[G]$ e ao encontrar um vértice branco a sub-rotina DFS-Visit é chamada para visitar este vértice (5-7). Quando a sub-rotina é chamada na linha 7, o vértice u se torna a raiz de uma nova árvore na floresta em profundidade.

Na linha 1 do DFS-visit, a váriavel global *tempo* é incrementada. Já nas linhas 2 e 3 registra o novo valor de $u.d$ e u é marcado de cinza (visitado). Nas linhas 4-7 percorre a lista de adjacências de u e ao encontrar um vértice v branco, sua variável $u.\pi$ recebe u e a

sub-rotina DFS-Visita é chamada para visitar esse vértice. Este loop encerra quando não estiver mais vértices brancos adjacentes de u. Em seguida u é marcado de preto, *tempo* é incrementado e $u.f$ recebe *tempo*. O custo total do tempo de execução do algoritmo DFS é $\Theta(V + E)$.

A Figura 2.9 apresenta o funcionamento do algoritmo DFS. Ao iniciar o algoritmo, todos os vértices do grafo são marcados de branco, como pode ser visto em (a). Em seguida, o vértice s é escolhido de forma aleatória verificando se sua cor é branco; se sim, o *DFS-Visit* é chamado para visitar este vértice marcando-o de cinza (b). Em (c), percorre a lista de adjacência de s e o *DFS-Visit* é chamado para visitar o vértice r. O vértice v adjacente a r é visitado (d). Em (e) e (f), todos os vértices adjacentes a v e r já foram visitados, então ambos são marcados de preto. Ao percorrer a lista de adjacência de s o vértice w ainda não foi visitado, então o *DFS-Visit* é chamado para visitá-lo (h). Em (k), ao percorrer os adjacentes de w, o vértice t é visitado; em seguida, o vértice u é visitado ao percorrer a lista de adjacência de t (i). Por fim, em (j), (k), (l) e (m), os vértices u, t, w e s são marcados de preto respectivamente.

2.2 Método *Branch-and-Bound* para Resolução de PLIM

Alguns problemas exigem que suas variáveis sejam inteiras, com isso, surgem problemas de Programação Linear Inteira (PLI). Nesta seção, apresentamos uma visão geral do método *branch-and-bound* (B&B). Para o estudo desta seção utilizamos os livros de Goldbarg e Luna (2000) e Arenales et al (2015).

De maneira inteligente, o *branch-and-bound* enumera pontos candidatos à solução ótima inteira de um problema. Um conceito fundamental utilizado por este método é a relaxação linear, que consiste em substituir $x \in \mathbb{Z}^+$ por $x \in \mathbb{R}^+$, ou seja, tornando um Problema Inteiro (PI) em um Problema Linear (PL). Como vemos a seguir.

$$\text{(PI)} = \text{Maximizar } \{\text{cx} \mid \text{Ax} = \text{b}, \text{x} \geq 0, \text{x} \in \mathbb{Z}^+\}$$

o problema relaxado,

$$\text{(PL)} = \text{Maximizar } \{\text{cx} \mid \text{Ax} = \text{b}, \text{x} \geq 0, \text{x} \in \mathbb{R}^+\}$$

Definindo $\overline{z}$ como valor ótimo da função objetivo do problema relaxado e z do problema inteiro, temos que o problema relaxado é um limitante superior do problema inteiro, isto é, $\overline{z} \geq z$.

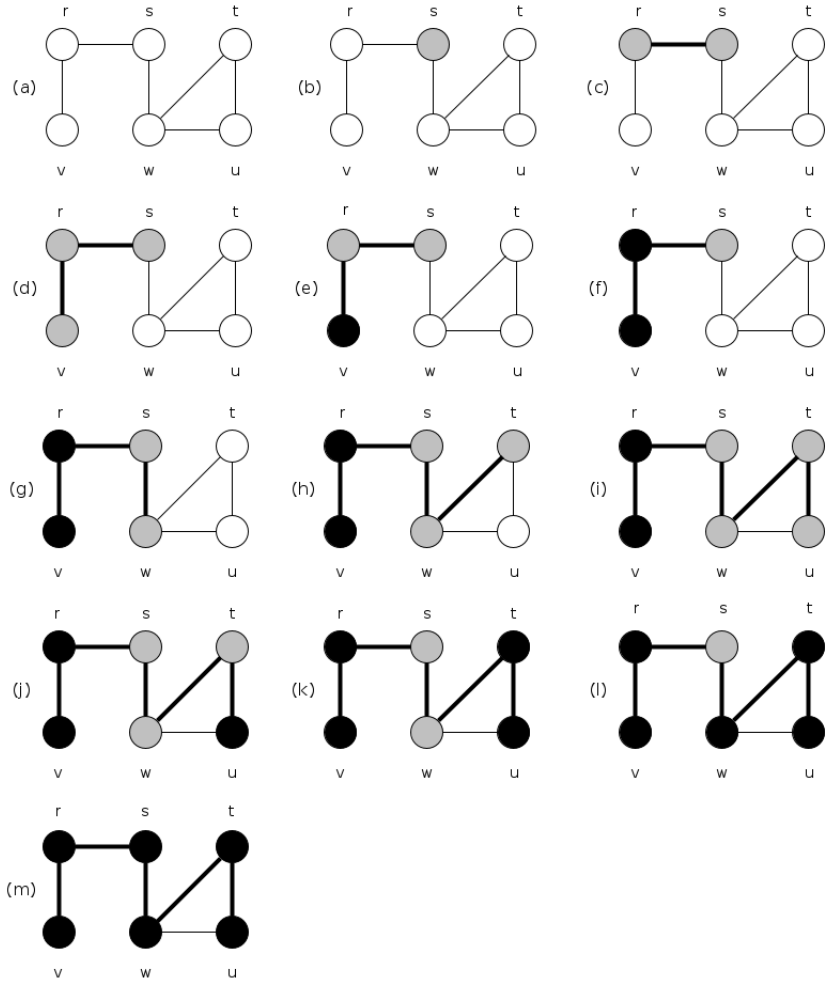

Figura 2.9: Execução do algoritmo DFS.

Seja $\overline{x}$ uma solução ótima de (PL) tal que a variável $\overline{x}_j$ é não inteiro, temos:

$$\overline{x}_j \geq \lfloor \overline{x}_j \rfloor + 1 \text{ ou } \overline{x}_j \leq \lfloor \overline{x}_j \rfloor$$

Reduzir o espaço de busca usando informações do problema de PL é uma estratégia usada pelo método *branch-and-bound*. Este procedimento é denominado *enumeração implícita*, em que subconjuntos são considerados e descartados. Tais subconjuntos são obtidos através da separação do problema original em problemas menores, que normalmente são mais fáceis de serem resolvidos, o que leva a reduzir o esforço computacional. Para esta separação do problema é usada a estratégia *divisão e conquista*.

A seguir, abordamos os conceitos definidos acima através do exemplo (P).

(P)

$$\text{Maximizar z} = 6x_1 + 9x_2$$

$$\begin{aligned} \text{Sujeito a.} \quad & x_1 + x_2 \leq 5 \\ & 4x_1 + 9x_2 \leq 36 \\ & x_1, x_2 \in \mathbb{R}^+ \end{aligned}$$

A Figura 2.11 representa a solução gráfica do exemplo (P). Pode ser visto que os valores das variáveis contínuas $\overline{x}_1 = 1{,}8$ e $\overline{x}_2 = 3{,}2$, leva o valor da função objetivo a $\overline{z} = 39{,}6$.

Dividir o exemplo original (P) em dois problemas menores (P_1) e (P_2) em relação a variável x_2, produz uma nova restrição em cada subproblema. Como podemos ver a seguir:

$$x_2 \leq \lfloor 3{,}2 \rfloor \Rightarrow x_2 \leq 3 \qquad\qquad x_2 \geq \lfloor 3{,}2 \rfloor + 1 \Rightarrow x_2 \geq 4$$

(P_1) (P_2)

$$\begin{aligned} \max z = {} & 6x_1 + 9x_2 \\ \text{s. a.} \quad & x_1 + x_2 \leq 5 \\ & x_2 \leq 3 \\ & 4x_1 + 9x_2 \leq 36 \\ & x_1, x_2 \in \mathbb{R}^+ \end{aligned} \qquad\qquad \begin{aligned} \max z = {} & 6x_1 + 9x_2 \\ \text{s. a.} \quad & x_1 + x_2 \leq 5 \\ & x_2 \geq 4 \\ & 4x_1 + 9x_2 \leq 36 \\ & x_1, x_2 \in \mathbb{R}^+ \end{aligned}$$

Este particionamento de um problema original em problemas menores pode ser facilmente representado através de uma árvore, onde cada nó da árvore representa um problema. O problema original (P) é a *raiz*; (P_1) e (P_2), os *filhos* de (P). A Figura 2.13 representa uma árvore B&B.

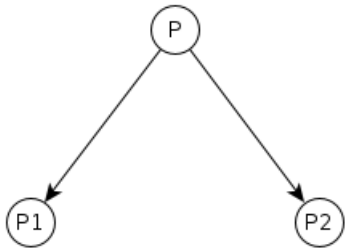

Figura 2.10: Árvore B&B.

Considerando as novas restrições, o exemplo original (P) será reduzido a dois novos problemas, (P_1) e (P_2), produzindo novos espaços de soluções factíveis (Figura 2.12).

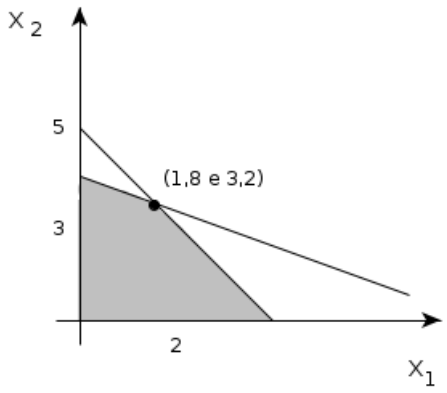

Figura 2.11: Solução gráfica do exemplo (P).

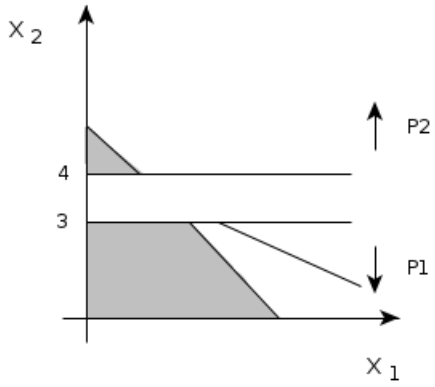

Figura 2.12: Espaço de soluções de (P_1) e (P_2).

No exemplo descrito acima, usamos a estratégia de separação em função da variável x_2. Esta estratégia também pode ser reaplicada em função da variável x_1 quando, e somente quando, a variável for contínua. Após o término do procedimento o problema com solução ótima inteira e o maior valor da função objetivo é retornado. As soluções contínuas são um limite superior para o valor de $\overline{z}$, enquanto as soluções com todas as variáveis inteiras geram um limite inferior. Podemos ver na árvore B&B na Figura 2.14, que (P) tem uma solução contínua $\overline{z} = 39,6$ e (P_1) possui uma solução inteira $z = 39$. O problema (P_2) não precisaria ser resolvido, uma vez que entre $39,6$ e 39 não tem outra solução inteira melhor que 39 ($39 \leq \overline{z} \leq 39,6$).

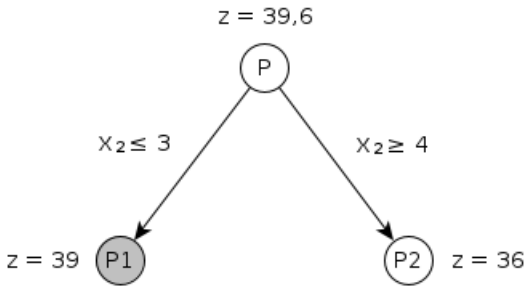

Figura 2.13: Árvore B&B do problema (P).

3 MÉTODOS DA LITERATURA PARA O PCGRM

Este capítulo está dividido da seguinte forma, na Seção 3.1, apresentamos os trabalhos relacionados, na Seção 3.2, abordamos alguns modelos de PLIM, e algumas heurísticas presentes na literatura são descritas na Seção 3.3.

3.1 Trabalhos Relacionados

No trabalho de Xu e Faragó (2019) é apresentado o cenário do PCGRM, fazendo uma revisão aprofundada do problema, fornecendo novas ideias, apresentando algumas definições e corrigindo alguns erros existentes na literatura. Principalmente, o trabalho aborda a relação do que chamam de versão não-sobreposta e versão sobreposta. A versão não-sobreposta é onde cada aresta possui apenas um rótulo, e na versão sobreposta, cada aresta possui múltiplos rótulos. Os autores proporam um método para transformar uma versão sobreposta em uma versão não sobreposta que garante a conectividade de cobertura do grafo.

A tese de Silva (2019) aborda diversos problemas de conectividade definidos sobre grafos com arestas rotuladas, tendo um maior foco no Problema da Árvore Geradora com Rotulação Mínima (PAGRM) e sua versão generalizada, Problema Generalizado da Árvore Geradora com Rotulação Mínima (PGAGRM). O trabalho introduziu novos conceitos, definições, propriedades e teoremas úteis em relação a grafos com arestas rotuladas, como também realizou um estudo poliédrico sobre o PGAGRM. Ainda como contribuição do trabalho, o autor propôs novas heurísticas e métodos exatos para solucionar o PGAGRM. Os experimentos mostraram que as abordagens introduzidas no trabalho alcançaram os melhores resultados para métodos heurísticos e exatos em comparação com a literatura, segundo o autor.

Um problema que está fortemente relacionado ao PCGRM é o Problema do Corte s-t Rotulado Mínimo (PCstRM), que pode ser definido da seguinte forma: dado um grafo $G = (V, E, L)$ orientado ou não orientado; um vértice origem s e um vértice destino t; e seja L' um subconjunto de rótulos ($L' \subseteq L$), o objetivo do problema é minimizar $|L'|$ tal que com a remoção das arestas associadas aos rótulos em L', desconecta s e t. O PCstRM foi proposto por Jha et al (2002) e os autores provaram que o problema é NP-Hard.

O PCstRM tem diversas aplicações, e uma das principais áreas de aplicação é em segurança de computadores (Jha et al, 2002; Sheyner et al, 2002; Sheyner e Wing, 2003). Em particular, o que motivou os estudos foi a geração de grafos de ataque e detecção de intrusos. Os vértices do grafo de ataque representam diversos estados, as arestas representam a transição de um estado para outro e os possíveis ataques são representados pelos rótulos.

Os artigos de Coudert et al (2007, 2016) abordam o PCstRM e o PCGRM quando o objetivo é mensurar a conectividade da rede quando seus links possuem riscos compartilhados de falha. Fellows et al (2010) também estudou o PCstRM, mas relacionado a sua complexidade parametrizada, demostrando que o problema é W[1]-Hard para grafos com largura de caminho menor ou igual a 4, e W[2]-Hard quando grafos com largura de caminho menor ou igual a 3.

Vários algoritmos de aproximação foram propostos para o PCstRM (Zhang et al, 2011; Tang e Zhang, 2012; Zhang, 2014). Broersma et al (2005) propôs um algoritmo exato para o PCstRM com tempo de execução $O(n^2|L|!)$, onde L é o conjunto de rótulos. Em Zhang e Tang (2019) foi feito um estudo de dois programas lineares para o PCstRM e provaram que ambos tem largos *gaps* de integralidade, nomeando-os, $\Omega(m)$ e $\Omega(m^{1/3-\varepsilon})$, onde m é o número de arestas em um grafo.

Sucupira et al (2017) apresenta o estudo da complexidade do Problema do Corte Global Máximo (PCGM) e do Problema do Corte Colorido, que é um caso particular do PCGM. Os autores provaram que ambos os problemas são NP-completo mesmo em grafos completos, planar e com largura de árvore limitada. Mostraram também que o PCGM é tratável com parâmetro fixo com relação aos parâmetros $|E|$ e $|L|$.

3.2 Modelos de Programação Linear Inteira Mista para o PCGRM

No trabalho de Silva et al (2016) foram propostos três modelos de programação linear inteira mista para o PCGRM, além de técnicas exatas para resolvê-los. Os modelos

são: o Modelo Baseado em Particionamento ($PART_2$), o Modelo Baseado em Agrupamento de Vértices (P3E) e o Modelo Baseado em Eliminação de Árvores (EAC).

3.2.1 $PART_2$

O Modelo $PART_2$ tem como objetivo gerar duas partições de vértices: S e $\overline{S} = V \backslash S$. Para qualquer conjunto $S \subset V$, $S \neq \emptyset$, a remoção de todas as arestas com um extremo em S e outro em $\overline{S}$ desconecta o grafo. Seja e_{ij} uma aresta com extremos nos vértices i e j, $L(e_{ij})$ o rótulo associado a aresta e_{ij}; seja z_ℓ, $\forall \ell \in L$, uma variável que assume valor 1 se e somente se o rótulo ℓ participa do corte; e w_v, $\forall v \in V$ uma variável que assume valor 1 se e somente se o vértice v faz parte do conjunto S; o modelo está descrito nas expressões (3.1-3.7):

$$PART_2 = \text{Minimize} \sum_{\ell \in L} z_\ell \tag{3.1}$$

$$\text{s.a.} \quad \sum_{v \in V} w_v < |V|, \tag{3.2}$$

$$w_1 = 1, \tag{3.3}$$

$$z_{L(e_{ij})} \geq w_i - w_j, \qquad \forall e_{ij} \in E, \tag{3.4}$$

$$z_{L(e_{ij})} \geq w_j - w_i, \qquad \forall e_{ij} \in E, \tag{3.5}$$

$$z_\ell \geq 0, \qquad \forall \ell \in L, \tag{3.6}$$

$$w_v \in \{0, 1\}, \qquad \forall v \in V. \tag{3.7}$$

A função objetivo (3.1) minimiza o número de rótulos necessários para desconectar o grafo; a restrição (3.2) força que $S \subset V$; a restrição (3.3) além de garantir que $S \neq \emptyset$, elimina a simetria das soluções ao escolher um vértice arbitrário para participar de S; as restrições (3.4) e (3.5) ativam as cores (ou rótulos) das arestas do corte; e as restrições (3.6) e (3.7) definem o domínio das variáveis.

3.2.2 P3E

O modelo P3E é baseado no problema de agrupamento de vértices por edição de arestas. Seja K_n um grafo completo e E o conjunto de arestas de K_n; onde cada aresta $e_{ij} \in E$ possui um custo positivo ou negativo associado, o objetivo deste modelo é remover as arestas, contabilizando o custo associado, de modo que o grafo resultante seja um particionamento de subcliques. Então, para transformar um grafo G de entrada em um grafo K_n, é adicionado a G uma aresta com custo de remoção 0 para cada par de vértices i, j tal que $e_{ij} \notin E$.

Seja $\overline{E}$ o conjunto de arestas que não pertecem a G, de modo que $E \cup \overline{E}$ origina em um grafo completo; e seja $x_e, \forall e \in E$, uma variável que assume valor 1 se e somente se a aresta e participa do corte; e z_ℓ, $\forall \ell \in L$, uma variável que assume valor 1 se e somente se o rótulo ℓ participa do corte. As expressões (3.8-3.15) apresentam o modelo P3E.

A função objetivo (3.8) minimiza o custo dos rótulos da solução; o conjunto do restrições (3.9) liga as variáveis de aresta com as variáveis de rótulos; o conjunto de inequações (3.10-3.12) é o clássico conjunto de eliminação de P_3, ou seja, caminhos com 3 vértices; a restrição (3.13) garante que a solução não seja vazia; e as restrições (3.14) e (3.15) definem o domínio das variáveis de decisão.

$$\text{P3E} = \text{Minimize} \sum_{\ell \in L} z_\ell \tag{3.8}$$

$$\text{s.a.} \quad z_{\ell(e)} \geq x_e, \qquad \forall e \in E, \tag{3.9}$$

$$+x_{ij} + x_{jk} - x_{ki} \geq 0, \qquad \forall i, j, k \in V, i \neq j, j \neq k, k \neq i, \tag{3.10}$$

$$+x_{ij} - x_{jk} + x_{ki} \geq 0, \qquad \forall i, j, k \in V, i \neq j, j \neq k, k \neq i, \tag{3.11}$$

$$-x_{ij} + x_{jk} + x_{ki} \geq 0, \qquad \forall i, j, k \in V, i \neq j, j \neq k, k \neq i, \tag{3.12}$$

$$\sum_{e \in E} x_e \geq 1, \tag{3.13}$$

$$z_\ell \geq 0, \qquad \forall \ell \in L, \tag{3.14}$$

$$x_e \in \{0, 1\}, \qquad \forall e \in E \cup \overline{E}. \tag{3.15}$$

Dado que o grafo G de entrada é conexo, a corretude do modelo baseia-se no fato de que G é um particionamento de cliques se e somente se P_3 não é subgrafo de G, P_3 sendo um caminho formado por 3 vértices. Por fim, foi proposto o algoritmo *branch-and-cut* para resolução do modelo.

3.2.3 EAC

O modelo EAC é baseado na eliminação de árvores. Dado que uma árvore de cobertura é um grafo conexo minimal com respeito ao seu número de arestas, temos que a proibição de todas as árvores de cobertura de G resulta em um grafo desconexo. Seja $\mathcal{T}$ o conjunto de todas as árvores de cobertura do grafo G; seja $L(T)$, $T \in \mathcal{T}$, o conjunto de rótulos utilizados pelas arestas de T; e seja z_ℓ, $\forall \ell \in L$, variáveis de decisão. As expressões de (3.16-3.18) apresentam o modelo:

$$\text{EAC} = \text{Minimize} \sum_{\ell \in L} z_\ell \tag{3.16}$$

$$\text{s.a.} \quad \sum_{\ell \in L(T)} z_\ell \geq 1, \qquad \forall T \in \mathcal{T}, \tag{3.17}$$

$$z_\ell \in \{0, 1\}, \qquad \forall \ell \in L. \tag{3.18}$$

A funcão objetivo (3.16) minimiza o custo dos rótulos da solução; o conjunto de restrições (3.17) garante a desconectividade do grafo solução pela proibição de todas as árvores de cobertura de G; e as restrições (3.18) definem o domínio das váriaveis.

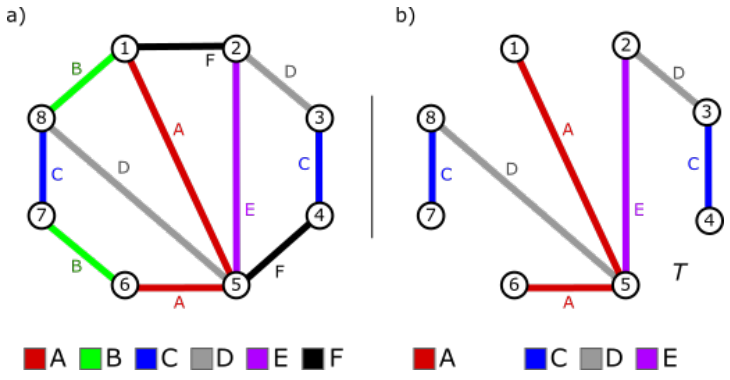

Figura 3.1: Árvore de cobertura contida na instância exemplo do PCGRM.

A Figura 3.1(a) ilustra uma instância exemplo do PCGRM e a Figura 3.1(b) apresenta uma árvore de cobertura T extraída da instância exemplo. Logo, T não pode continuar existindo para garantir que o grafo G seja desconexo, para isto, a adição da inequação

$$z_A + z_C + z_D + z_E \geq 1$$

garante que pelo menos um dos rótulos de T faz parte do corte.

Por fim, o trabalho apresenta testes computacionais que demonstram que os modelos são capazes de solucionar instâncias de pequeno e médio porte com esforço computacional razoável. O $PART_2$ obteve o melhor desempenho dos três modelos propostos no trabalho, o que se deve, segundo os autores, ao número linear de inequações do modelo e ao processo de *branch-and-bound*.

3.3 Heurísticas para o PCGRM

Apresentamos a seguir, algumas heurísticas para solucionar o PCGRM. Modelos exatos são muitas vezes limitados em aplicações práticas. Com isso, torna-se necessário utilizar métodos heurísticos para resolução de problemas de otimização combinatória, que tem um custo computacional alto, embora não seja garantida a solução ótima para o problema.

3.3.1 Algoritmo Genético, VNS e *MultiVND*

Silva et al (2018) propõe três heurísticas para solucionar o problema. A primeira heurística é baseada nos conceitos de Algoritmo Genético, onde X é o conjunto de todas as soluções para uma instância do PCGRM e a função de avaliação: $X \rightarrow \mathbb{Z}^+$. O algoritmo busca uma solução $x \in X$ com uma função de avaliação(x) mínima. Esta versão inicia com uma população aleatória, executa o elitismo e as iterações são reiniciadas até o limite de tempo ser atingido.

Cada solução é uma bipartição do conjunto V, sendo essas partições chamadas de *left* e *right*. Cada indivíduo x é representado por um vetor binário, onde $x[i] = 0$, indica que o vértice $i \in P_\ell^x$, no qual é a partição *left* da solução x. Por outro lado, $i \in P_r^x$ é a partição *right* de x. Portanto, $L^x = \{L_{i,j} | (i,j) \in E\}$ e $x[i] \neq x[j]\}$ e *fitness*(x) $= |L^x|$.

A população inicial $\mathcal{P}$ é composta por $|N|$ soluções aleatórias. Para cada $x \in \mathcal{P}$, um número de vértices é retirado para ser inserido em P_ℓ^x, enquanto os demais são inserido

em P_r^x. Seja $\mathcal{P}_e$ que represente 20% dos melhores indivíduos de $\mathcal{P}$, e $\mathcal{P}_{\bar{e}}$ represente os indivíduos restantes. O elitismo é feito por operações de cruzamento, onde N é novos indivíduos gerado e adicionados a $\mathcal{P}$ pelo cruzamento de duas soluções $x^e \in \mathcal{P}_e$ e $x^{\bar{e}} \in \mathcal{P}_{\bar{e}}$. Para realização da operação de cruzamento é ultilizado o conhecido *random single cut point crossover*. O resultado gerado é adicionado a nova população $\mathcal{P}^+$.

Por fim, são eliminados metade dos indivíduos de $\mathcal{P}^+$ na fase de reinício da população. Em seguida, 30% dos piores indivíduos que permanacem em $\mathcal{P}^+$ são substituidos por novas soluções aleatórias. O algoritmo retorna a melhor solução x^* após ser atingida a condição de parada.

A segunda heurística proposta é uma adaptação da *Variable Neighborhood Search* (VNS) (Mladenović e Hansen, 1997), que é uma meta heurística para explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança durante o processo de busca com o objetivo de evitar ótimos locais. Para representação da solução foi definido um par de partições $P^G = \{P_\ell^G, P_r^G\}$, onde P_ℓ^G representa o conjunto de vértices na partição *left* e P_r^G o conjunto de vértices na partição *right*.

Se $|P_\ell^G| > 0$ e $|P_r^G| > 0$, tem-se a garantia de um corte válido. Seja o conjunto $L^G = \{L_{i,j} \mid i \in P_\ell^G, j \in P_r^G \text{ e } (i,j) \in E\}$ para representar a solução G. Para o procedimento de busca local, os autores proporam duas estruturas de vizinhança. Em todas essas estruturas, é considerada uma pré-solução viável de $G^{'}$, formado por um par de partições válidas $P^{G'}$.

Na mudança de partição, se $|P_\ell^{G'}| > 1$, move um vértice $v \in P_\ell^{G'}$ para a partição $P_r^{G'}$ ou se $|P_r^{G'}| > 1$, move um vértice $v \in P_r^{G'}$ para $P_\ell^{G'}$. Depois, é atualizado o conjunto de cores no corte L^G. Na parte da heurística que os autores chamam de *interchange-partition* ou partição de intercâmbio, quando $|P_\ell^{G'}| > 1$ e $|P_r^{G'}| > 1$, move um vértice $v \in P_\ell^{G'}$ para a partição $P_r^{G'}$ e um vértice $u \in P_r^{G'}$ para $P_\ell^{G'}$. Depois, é atualizado o conjunto de cores no corte L^G. Por fim, foi proposto uma estrutura de dados auxiliar para avaliação das soluções.

A terceira heurística desenvolvida no trabalho é a *MultiVND*, que consiste em uma heurística *multistart* simples. Em cada iteração *multistart* é gerado uma solução aleatória e é aplicado o método *Variable Neighborhood Descent* (VND) como pesquisa local. A melhor solução encontrada é retornada.

Portanto, segundos os autores, a meta heurística VNS obteve os melhores resultados em comparação aos outros métodos, sendo capaz de produzir melhores resultados em média para todos os casos estudados. O *MultiVND* funcionou melhor com instâncias mais simples,

e em alguns casos alcançando o VNS; porém, em intâncias mais difíceis, o *MultiVND* obteve resultados muito ruins. Por fim, o Algoritmo Genético para instâncias difíceis foi tão bom quanto o VNS.

3.3.2 *VNS-Greedy* e *VNS-Probabilistic*

Outro trabalho que propõe heurísticas para resolução do PCGRM é o de Bordini et al (2017), que apresenta uma abordagem determinística *VNS-Greedy* e uma abordagem probabilística *VNS-Probabilistic*. Os autores proporam um algoritmo geral (Algoritmo 4) como estrutura básica para essas duas abordagens. Algumas sub-rotinas desse algoritmo como *Generate-Initial-Solution*, *New-Solution* e *Local-search* têm uma "versão *greedy*" e uma "versão *probabilistic*". Portanto, executando o algoritmo geral usando as versões *greedy* de tais sub-rotinas produz o algoritmo *VNS-Greedy*, enquanto executando usando as versões *probabilistic* produz o algoritmo *VNS-Probabilistic*.

Algoritmo 4 General algorithm

Entrada: Graph $G = (V, E, C)$, Where $C = \{c(e) \mid e \in E\}$
1: Generate-Initial-Solution($BestS$)
2: $MaxNeighborhood \leftarrow |C| - |BestS|$
3: **repita**
4: New-Solution(S)
5: **enquanto** $|S| > |BestS|$ **faça**
6: $BestS \leftarrow S$
7: $MaxNeighborhood \leftarrow |C| - |BestS|$
8: New-Solution(S)
9: $k - 1$
10: **enquanto** $k < MaxNeighborhood$ **faça**
11: $S' \leftarrow S$
12: Shake(S', k)
13: **se** Number-of-Components(S') = 1 **então**
14: Fix(S')
15: Local-Search(S')
16: **se** $|S'| > |S|$ **então**
17: $S' \leftarrow S$
18: $k \leftarrow 1$
19: **senão**
20: $k \leftarrow k + 1$
21: **se** $|S| > |BestS|$ **então**
22: $|BestS| \leftarrow S$
23: $MaxNeighborhood \leftarrow |C| - |BestS|$
24: **até que** stop condition is true

A variável $BestS$ representa a melhor solução atual, $MaxNeighborhood$ é uma

variável que controla a vizinhança, S e S' são soluções auxiliares e *Number-of-Components*(S') é a função padrão que retorna o número de componentes conexas da solução S'. Na linha 1, uma solução inicial é gerada; $MaxNeighborhood$ é um conjunto com o número de cores que não estão em $BestS$ (linha 2) e o loop principal (3 a 24) é executado até que a condição de parada seja aceita.

Nas linhas 4 a 7, uma nova solução candidata S é gerada no início de uma nova iteração. Primeiro, S é gerado utilizando a sub-rotina New-Solution(S) (linha 4). Se S for maior que $BestS$, $BestS$ e $MaxNeighborhood$ são atualizados e outra solução candidata S é gerada. O loop (linha 5 a 9) termina quando o número de cores da solução candidata não for maior que o número de cores da melhor solução atual. As linhas 10 a 23 contêm o núcleo da estratégia básica da meta heurística VNS.

Os resultados das heurísticas propostas foram comparados com os resultados obtidos pelos três métodos exatos propostos por Silva et al (2016), que foram brevemente abordados no Capítulo 3. Segundo os autores, as abordagens *VNS-Greedy* e *VNS-Probabilistic*, foram capazes de alcançar todas as soluções ótimas, em instâncias com $|V| \leq 200$, em um baixo tempo computacional. Já com instâncias com o valor ótimo desconhecido, as abordagens geraram uma solução em um tempo computacional razoável. Por fim, *greedy* e *probabilistics*, exibiram o mesmo comportamento em termos de qualidade da solução, e nenhuma diferença significativa em termos de tempos computacionais.

4 MODELO $\mathcal{MF}_d$

Neste capítulo, apresentamos o modelo $\mathcal{MF}_d$, que é a contribuição central deste trabalho. O $\mathcal{MF}_d$ é uma nova família de formulações matemáticas capaz de solucionar o PCGRM. Para construir o modelo $\mathcal{MF}_d$ nos baseamos em um modelo presente na literatura chamado $PART_2$, proposto no trabalho de Silva et al (2016), e utilizamos o conceito de fecho cromático, que é definido na Seção 4.1. Na Seção 4.2, apresentamos a formulação do modelo, e nas Seções 4.3 e 4.4, os algoritmos B&C e estratégias *pré-branching*, respectivamente.

4.1 Fecho cromático

Nesta seção, são apresentadas duas definições úteis sobre os GARs: o fecho transitivo e o fecho cromático, que nos permite alterar a representação de um GAR de entrada. Estas definições vêm do trabalho de Silva (2019), sendo estendidas e adaptadas para o objetivo desta pesquisa. Então, utilizando o conceito de fecho cromático, podemos transformar um grafo $G = (V, E, L)$ de entrada em um grafo $H = (V, E', L)$ com um novo conjunto de arestas E'.

Assim, para apresentarmos as duas definições de fecho cromático a seguir, declaramos o subgrafo de um GAR G como sendo: $G[\{\ell\}] = (V, E(\{\ell\}), \{\ell\})$, onde $E(\{\ell\}) = \{e \in E \mid L(e) = \ell\}$.

4.1.1 Definição

Definição 1. *O fecho transitivo de um rótulo $\ell \in L$, denotado por $\mathcal{F}(\{\ell\})$, representa a expansão do conjunto de arestas $E(\{\ell\})$ tal que se houver um caminho entre os vértices u e v no grafo $G[\{\ell\}]$, então a aresta $e = (u, v) \in \mathcal{F}(\{\ell\})$.*

Definição 2. *Por extensão, o fecho transitivo cromático, abreviado fecho cromático, de um GAR $G = (V, E, L)$, denotado por $\mathcal{F}(G)$, é o grafo $H = (V, E', L)$, que tem o mesmo conjunto de vértices e rótulos de G, e cujo conjunto de arestas é definido pela união do fecho transitivo $\mathcal{F}(\{\ell\})$, $\forall \ell \in L$. Formalmente, $E' = \bigcup_{\ell \in L} \mathcal{F}(\{\ell\})$.*

A Figura 4.1 ilustra os conceitos de fecho transitivo e fecho cromático. A Figura 4.1(a) apresenta um pequeno GAR $G = (V, E, L)$. Fig. 4.1(b) destaca $\mathcal{F}(\{F\})$, o fecho transitivo do rótulo $F \in L$, e a Fig. 4.1(c) apresenta $\mathcal{F}(G)$, o fecho cromático do grafo G.

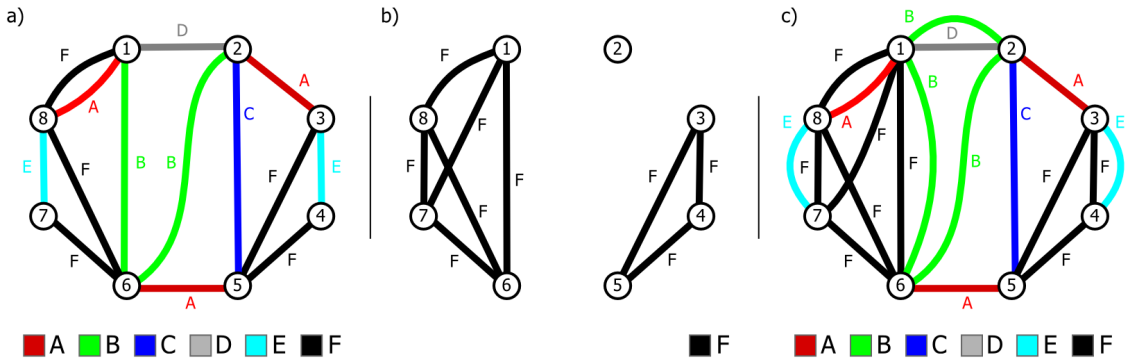

Figura 4.1: Exemplos de fechos transitivo e cromático. **(a)** O grafo de arestas rótuladas $G = (V, E, L)$. **(b)** O fecho transitivo do rótulo F. **(c)** O fecho cromático do grafo G.

Proposição 1. *Resolver o PCGRM para um GAR G é equivalente à resolver o problema para o grafo $H = \mathcal{F}(G)$.*

Demonstração. Seja C qualquer subconjunto de L. Provamos esta proposição demonstrando que $H[C] = (V, E', C)$ é desconexo se e somente se $G[C] = (V, E, C)$ é desconexo.

($\Rightarrow$) Suponha que $H[C]$ é desconexo e $G[C]$ é conexo. Já que o fecho transitivo é uma extensão do conjunto de arestas de G, $H[C]$ é o grafo $G[C]$ com arestas adicionais. A remoção de arestas não pode conectar o grafo $G[C]$, uma contradição.

($\Leftarrow$) Suponha que $G[C]$ é desconexo e $H[C]$ é conexo. Neste caso $G[C]$ tem dois conjuntos separados de vértices $S, \overline{S} \subset V$, tal que $u \in S$, $v \in \overline{S}$, $e' = (u, v) \in E$, mas não há caminho entre u e $v \in G[C]$. Contudo, da Definição 2, se $e' = (u, v) \in E'$, então há um caminho entre u e v em $G[C]$, uma contradição. □

Corolário 1. *Dado um GAR G, adicionar uma aresta que forme um ciclo monocromático em G não muda o valor da função objetivo em relação ao PCGRM.*

Corolário 2. *Dado um GAR G, qualquer ciclo monocromático pode ser quebrado arbitrariamente pela remoção de uma de suas arestas sem mudar o valor da função objetivo em relação ao PCGRM.*

4.1.2 Extensão do Fecho Cromático

A seguir, será apresentado algumas extensões da definição geral de fecho cromático, no qual, a partir destas, obtemos o fecho cromático d, o fecho cromático 0 e o fecho cromático $*$.

Definição 3. *Seja $\delta(i,j)$ a distância entre os vértices $i,j \in G(\{\ell\})$ e extrapolando a definição do fecho transitivo $\mathcal{F}(\{\ell\})$, temos que $\mathcal{F}_d(\{\ell\})$ é o fecho transitivo de distância d, onde $d \geq 1$, que expande o conjunto de arestas $E(\{\ell\})$ tal que se houver um caminho entre os vértices u e v no grafo $G[\{\ell\}]$ com $\delta(u,v) \leq d$, então a aresta $e = (u,v) \in \mathcal{F}_d(\{\ell\})$.*

Definição 4. *De forma análoga, o fecho transitivo cromático de distância d, abreviado fecho cromático d, de um GAR $G = (V,E,L)$, denotado por $\mathcal{F}_d(G) = (V,E_d,L)$, é definido pela união do fecho transitivo $\mathcal{F}_d(\{\ell\})$, $\forall \ell \in L$. Formalmente, $E_d = \bigcup_{\ell \in L} \mathcal{F}_d(\{\ell\})$.*

Definição 5. *Extrapolando a definição do fecho transitivo $\mathcal{F}(\{\ell\})$, temos que $\mathcal{F}_0(\{\ell\})$ é o fecho transitivo 0, que reduz o conjunto de arestas $E(\{\ell\})$ tal que se houver um ciclo ℓ-cromático $C \subset G[\{\ell\}]$, então remove uma aresta e de rótulo ℓ, onde $e \in C$.*

Definição 6. *Por extensão, o fecho transitivo cromático 0, abreviado fecho cromático 0, de um GAR $G = (V,E,L)$, denotado por $\mathcal{F}_0(G)$, é o grafo $H = (V,E_0,L)$, que tem o mesmo conjunto de vértices e rótulos de G, e cujo conjunto de arestas é definido pela união do fecho transitivo $\mathcal{F}_0(\{\ell\})$, $\forall \ell \in L$. Formalmente, $E_0 = \bigcup_{\ell \in L} \mathcal{F}_0(\{\ell\})$.*

Definição 7. *Extrapolando a definição do fecho transitivo $\mathcal{F}(\{\ell\})$, temos que $\mathcal{F}_*(\{\ell\})$ é o fecho transitivo $*$, que transforma o conjunto de arestas $E(\{\ell\})$ tal que se houver uma componente ℓ-cromática $K_\ell \subset G[\{\ell\}]$, onde $|V(K_\ell)| > 2$, então K_ℓ é transformada em uma estrela S_ℓ ℓ-cromática cujo vértice central v é um vértice arbitrário de $V(K_\ell)$.*

Definição 8. *Por extensão, o fecho transitivo cromático $*$, abreviado fecho cromático $*$, de um GAR $G = (V,E,L)$, denotado por $\mathcal{F}_*(G)$, é o grafo $H = (V,E_*,L)$, que tem o mesmo conjunto de vértices e rótulos de G, e cujo conjunto de arestas é definido pela união do fecho transitivo $\mathcal{F}_*(\{\ell\})$, $\forall \ell \in L$. Formalmente, $E_* = \bigcup_{\ell \in L} \mathcal{F}_*(\{\ell\})$.*

Para obter um melhor entendimento das definições acima, a Figura 4.2 ilustra o conceito de fecho cromático d, fecho cromático 0 e fecho cromático $*$. A Figura 4.2(a) apresenta um GAR $G = (V, E, L)$. Fig. 4.2(b) destaca $\mathcal{F}_4(G)$, o fecho cromático 4, a Fig. 4.2(c) apresenta o fecho cromático $\mathcal{F}_3(G)$, e a Fig. 4.2(d) apresenta o fecho cromático $\mathcal{F}_2(G)$.

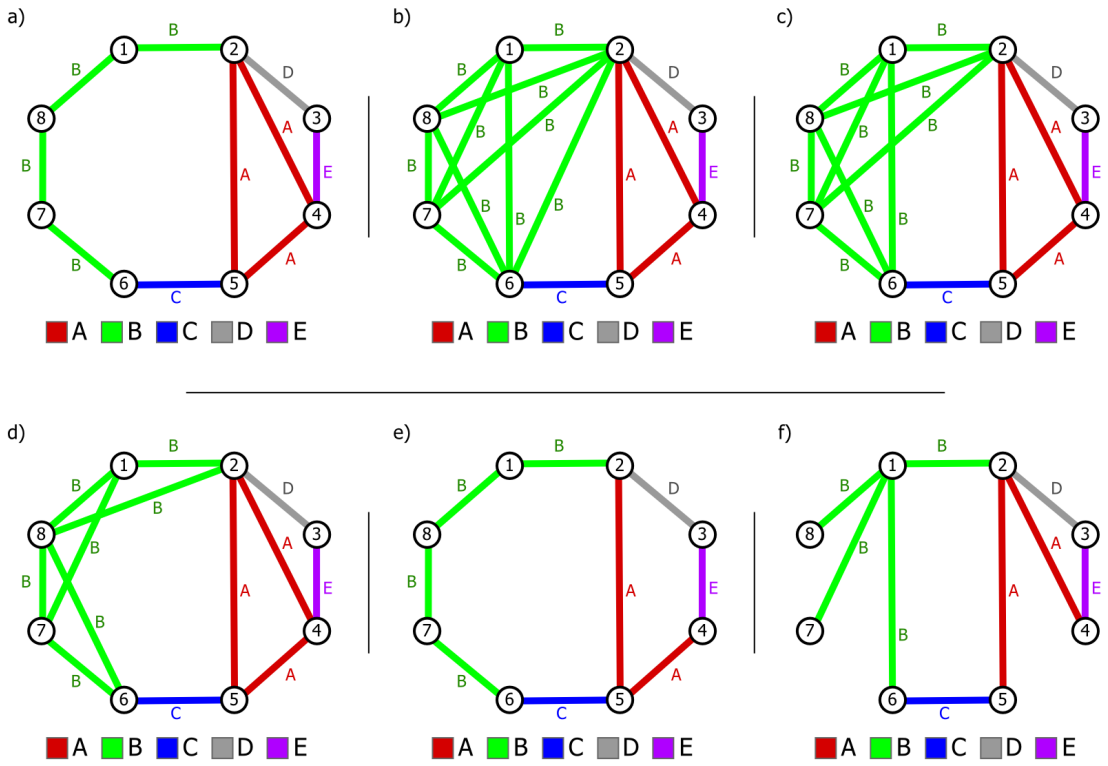

Figura 4.2: Exemplos de fechos transitivo e cromático d. **(a)** O grafo de arestas rótuladas $G = (V, E, L)$. **(b)** O fecho cromático 4. **(c)** O fecho cromático 3. **(d)** O fecho cromático 2. **(e)** O fecho cromático 0. **(f)** O fecho cromático *.

É possível perceber que o fecho cromático com $d = 1$, ou seja, o $\mathcal{F}_1(G)$, é o próprio GAR G ilustrado na Fig. 4.2(a). As Figuras 4.2(b),(c) e (d) representam o fecho cromático com $d = 4$, $d = 3$ e $d = 2$, respectivamente. Podemos observar na Fig. 4.2(e), o $\mathcal{F}_0(G) = (V, E_0, L)$, que é o grafo formado pelos vértices e rótulos de G, e o conjunto de arestas E_0 formado pelas arestas de E removendo uma aresta de rótulo ℓ de cada ciclo ℓ-cromático $C \subset G$. E por fim, o $\mathcal{F}_*(G) = (V, E_*, L)$, que é o grafo formado pelos vértices e rótulos de G, e o conjunto de arestas E_* formado pelas arestas de E trocando as arestas de cada componente ℓ-cromática $K_\ell \subset G$, onde $|V(K_\ell)| > 2$, pelas arestas de uma estrela S_ℓ ℓ-cromática cujo vértice central v é um vértice arbitrário de $V(K_\ell)$. A Fig. 4.2(f) ilustra o grafo $\mathcal{F}_*(G)$, onde a componente K_B é substituída pela estrela S_B com vértice central 1 e a componente K_A é substituída pela estrela S_A com vértice central 2.

Dados os exemplos e definições de fechos transitivos, apresentamos a seguir, os algoritmos e técnicas utilizadas para construção destes.

Para construírmos o fecho transitivo $\mathcal{F}_d(\{\ell\})$ criamos a matriz D^ℓ. Então, dado um rótulo $\ell \in L$ e os vértices $v, u \in V(G[\{\ell\}])$, e seja $\delta^\ell(v,u)$ a menor distância entre v e u no grafo $G[\{\ell\}]$, supomos que os vértices são numerados $1, 2, ..., |V|$ arbitrariamente, então, definimos a matriz $|V| \times |V|$ D^ℓ tal que

$$D^\ell_{vu} = \begin{cases} \delta^\ell(v,u) & \text{se existe caminho de } v \text{ até } u \text{ em } G[\{\ell\}], \\ \infty & \text{em caso contrário.} \end{cases}$$

Para obter a matriz D^ℓ aplicamos o algoritmo de busca em largura $D^\ell_v = BFS(G[\{\ell\}], v)$ para todo $v \in V$ como vértice inicial da busca sobre o subgrafo $G[\{\ell\}]$, retornando o vetor das distâncias $\delta^\ell(v,u)$ para todo $u \in V$.

Algoritmo 5 Gerar matriz de distância

Entrada: Subgrafo $G[\{\ell\}]$
1: $D^\ell = \emptyset$
2: **para cada** vértice $v \in V(G[\{\ell\}])$ **faça**
3: $\quad D^\ell_v = BFS(G[\{\ell\}], v)$
4: **Retorne** D^ℓ

O Algoritmo 5 constroe a matriz D^ℓ dado um subgrafo $G[\{\ell\}]$ de entrada. Na linha 1, a matriz D^ℓ é inicializada vazia. O loop principal do algoritmo inicia na linha 2, onde para cada vértice $v \in V(G[\{\ell\}])$ será feito uma busca em largura. Por fim, o algoritmo retorna a matriz de distância D^ℓ preenchida. Como exemplo, podemos observar que a Tabela 4.1 ilustra a matriz D^B do subgrafo $G[\{B\}] \subset G$, cujo GAR G está ilustrado na Figura 4.2(a).

Tabela 4.1: Matriz de distâncias D^B do subgrafo $G[\{B\}]$

D^B_{ij}	1	2	3	4	5	6	7	8
1	∞	1	∞	∞	∞	3	2	1
2	1	∞	∞	∞	∞	4	3	2
3	∞	∞	∞	∞	∞	∞	∞	∞
4	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞
6	3	4	∞	∞	∞	∞	1	2
7	2	3	∞	∞	∞	1	∞	1
8	1	2	∞	∞	∞	2	1	∞

De posse da matriz D^ℓ, obtêm-se o fecho transitivo $\mathcal{F}_d(\{\ell\}) = \{(v,u) \mid v, u \in$

V e $D^{\ell}_{vu} \leq d\}$. Seja o grafo de entrada $G = (V, E, L)$, como o BFS, de complexidade $O(|V| + |E|)$, é executado $|V|$ vezes, a construção da matriz D^{ℓ} se dá em tempo $O(|V|^2 + |E| \cdot |V|)$. Já a obtenção do fecho cromático $\mathcal{F}_d(G)$ constrói a matriz D^{ℓ} para todo $\ell \in L$, logo, sua complexidade é $O(|L| \cdot |V|^2 + |L| \cdot |E| \cdot |V|)$.

Algoritmo 6 Construção do $\mathcal{F}_*(\{\ell\})$

Entrada: Subgrafo $G[\{\ell\}]$
1: $\mathcal{T}^{\ell} = \text{DFS}(G[\{\ell\}])$
2: **para cada** árvore T $\in \mathcal{T}^{\ell}$ **faça**
3: $\quad r \leftarrow aleatorio(V(T))$
4: $\quad$ **para cada** vértice $v \in V(T) - \{r\}$ **faça**
5: $\quad\quad adj[r] \leftarrow v$

Para a construção dos fechos transitivos $\mathcal{F}_0(\{\ell\})$ e $\mathcal{F}_*(\{\ell\})$, aplica-se o algoritmo de busca em profundidade $\mathcal{T}^{\ell} = DFS(G[\{\ell\}])$ que retorna uma floresta de profundidade $\mathcal{T}^{\ell} \subset G[\{\ell\}]$ associada à busca. Sendo assim, $\mathcal{F}_0(\{\ell\})$ é composto por toda aresta $(i, j) \in \mathcal{T}^{\ell}$. Já para a construção de $\mathcal{F}_*(\{\ell\})$, como pode ser observado no Algoritmo 6, onde para toda árvore de profundidade $T \in \mathcal{T}^{\ell}$ obtemos uma árvore estrela $S = (V(T), E^*, \{\ell\})$ de raiz $r \in V(T)$, onde r é obtido de forma aleatória da árvore T, e $E^* = \{(r, v) \mid v \in V(T) - \{r\}\}$, adicionando E^* em $\mathcal{F}_*(\{\ell\})$.

Por fim, dado o grafo de entrada $G = (V, E, L)$, a construção dos fechos cromáticos $\mathcal{F}_0(G)$ e $\mathcal{F}_*(G)$ aplica o algoritmo DFS, de complexidade $O(|V| + |E|)$, para todo $\ell \in L$, logo, sua complexidade é $O(|L| \cdot |V| + |L| \cdot |E|)$.

4.2 Formulação $\mathcal{MF}_d$

Seja o fecho cromático $\mathcal{F}_d(G) = (V, E_d, L)$ de entrada, o modelo $\mathcal{MF}_d$ tem como objetivo particionar o conjunto de vértices em dois grupos: S e $\overline{S} = V \backslash S$. Para qualquer conjunto $S \subset V$, $S \neq \emptyset$, a remoção de todas as arestas com um extremo em S e outro em $\overline{S}$ desconecta o grafo. Seja $e_{ij} \in E_d$ uma aresta com extremos nos vértices i e j, $L(e_{ij})$ um rótulo associado a uma aresta; seja z_{ℓ}, $\forall \ell \in L$, uma variável que assume valor 1 se e somente se o rótulo ℓ participa do corte; e w_v, $\forall v \in V$ uma variável que assume valor 1 se e somente se o vértice v faz parte do conjunto S; o modelo $\mathcal{MF}_d$ está descrito nas expressões (4.1-4.7):

$$\mathcal{MF}_d = \text{Minimize} \sum_{\ell \in L} z_\ell \tag{4.1}$$

$$\text{s.a.} \quad \sum_{v \in V} w_v \geq 1, \tag{4.2}$$

$$w_1 = 0, \tag{4.3}$$

$$z_{L(e_{ij})} \geq w_i - w_j, \qquad \forall e_{ij} \in E_d, \tag{4.4}$$

$$z_{L(e_{ij})} \geq w_j - w_i, \qquad \forall e_{ij} \in E_d, \tag{4.5}$$

$$z_\ell \geq 0, \qquad \forall \ell \in L, \tag{4.6}$$

$$w_v \in \{0, 1\}, \qquad \forall v \in V. \tag{4.7}$$

A função objetivo (4.1) minimiza o número de rótulos necessários para desconectar o grafo; as restrições (4.2) e (4.3) forçam que $S \subset V$ e $S \neq \emptyset$, e além disso é possível perceber que toda solução representada pela váriavel w também pode ser representada pelo complemento dos valores de w, portanto, a restrição (4.3) quebra essa simetria da representação da solução; o conjunto de restrições (4.4) e (4.5) ativam as cores (ou rótulos) do corte associadas as arestas $e_{ij} \in E_d$. Por fim, as restrições (4.6) e (4.7) definem o domínio das variáveis.

O modelo $\mathcal{MF}_d$ define um conjunto de modelagens que se diferenciam pelo fecho cromático de entrada. Como exemplo, o modelo $\mathcal{MF}_3$ recebe como entrada o fecho cromático 3 para ativar as cores do corte associadas às suas arestas. Já o $\mathcal{MF}_0$ é o modelo que recebe um grafo livre de ciclos monocromáticos para ativar as cores do corte associadas às suas arestas. É possível perceber que o modelo $\mathcal{MF}_1$ que recebe o fecho cromático $\mathcal{F}_1(G)$ é o próprio modelo $PART_2$, como definido em Silva et al (2016).

4.3 Algoritmo *branch-and-cut*

As versões dos modelos matemáticos $\mathcal{MF}_d$ propostos são solucionados pelo resolvedor CPLEX 12.7.1 através de um algoritmo clássico *branch-and-cut*. As desigualdades (4.4) e (4.5) são inseridas de forma dinâmica durante a resolução dos nós, sendo as mesmas inicialmente relaxadas e checada sua violação quando, e somente quando, uma solução in-

teira ou fracionada é encontrada. É importante destacar que esta estratégia adotada neste trabalho difere da estratégia adotada em Silva et al (2016), pois, o $PART_2$, insere todas as desigualdades de uma só vez.

A seguir, descrevemos a estratégia citada acima:

Seja $(\bar{z}, \bar{w})$ uma solução fracionada encontrada durante a otimização. Verifica-se, da seguinte forma, se alguma inequação dos conjuntos (4.4) ou (4.5) foi violada: Para toda aresta $e = (v, u) \in E$ cujo $|\bar{w}_v - \bar{w}_u| > 1 - \epsilon$, então se $\bar{w}_v > \bar{w}_u$ e $\bar{z}_{L(e_{vu})} \leq \bar{w}_v - \bar{w}_u$, adiciona-se o seguinte corte:

$$z_{L(e_{vu})} \geq w_v - w_u,$$

caso contrário, se $\bar{w}_u > \bar{w}_v$ e $\bar{z}_{L(e_{uv})} \leq \bar{w}_u - \bar{w}_v$, adiciona-se o seguinte corte:

$$z_{L(e_{uv})} \geq w_u - w_v.$$

Por fim, salienta-se que todo o mecanismo de *branching* é gerenciado pelo resolvedor adotado.

4.4 Estratégias *pré-branching*

Nesta seção, é apresentado duas estratégias *pré-branching*, que no qual, podem fornecer uma perfomance melhor para solucionar o modelo $\mathcal{MF}_d$. Então, a ideia principal é guiar o resolvedor antes dele iniciar o *branching*. Ambas as estratégias, chamadas $\mathcal{PB}_1$ e $\mathcal{PB}_2$, utilizam a heurística *Maximum Vertex Covering Algorithm* ($MVCA$) (Krumke e Wirth, 1998) para obter uma árvore geradora colorida T. Seja $L = \{A, B, C, D\}$ o conjunto de rótulos da árvore T, a estratégia $\mathcal{PB}_1$ consiste em explorar o espaço de soluções possíveis para os rótulos A, B, C, D. Considerando que T gera uma árvore de cobertura, a inequação a seguir garante que T não exista, conforme abordado na eliminação de árvores (Seção 3.2):

$$\sum_{\ell \in L(T)} \ell \geq 1,$$

Como podemos ver na Figura 4.3, começamos da direita para a esquerda, explorando todas as soluções fixas, tais que $A = B = C = 0$ e $D = 1$. Ao fixarmos $D = 1$, dizemos que o rótulo D tem que está no corte, ou seja, tem que estar na solução ótima do Problema 1; e ao fixarmos os rótulos $A = B = C = 0$, dizemos que estes rótulos não podem ser cortados. Se o Problema 1 obtiver uma solução viável, esta será um limitante superior para o restante dos problemas. Após isso, deixamos D livre, colocando a responsabilidade deste rótulo com o resolvedor, e exploramos as soluções fixando $A = B = 0$ e $C = 1$, ou seja, o rótulo C estará na solução ótima do Problema 2. Então, se o Problema 2 obtiver uma solução viável melhor que o problema anterior, será o limitante superior para o restante dos problemas, e assim sucessivamente até percorremos todas as soluções possíveis para os rótulos de T. Por fim, a solução final será a melhor entre as encontradas pelos problemas.

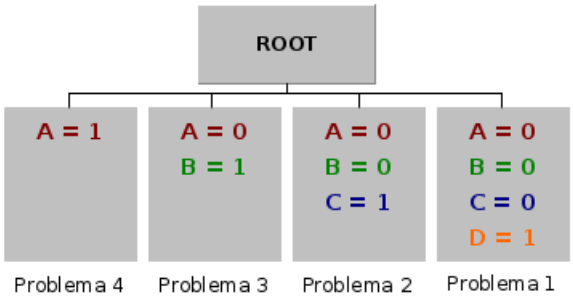

Figura 4.3: Exemplo do algoritmo *pre-branching*.

Na segunda estratégia, dado $L = \{A, B, C\}$ o conjunto de rótulos de uma árvore geradora T, consiste em testar todas as combinações booleanas possíveis entre os rótulos, com exceção de $A = B = C = 0$, pelo princípio descrito acima. Esta estratégia segue o mesmo conceito da primeira.

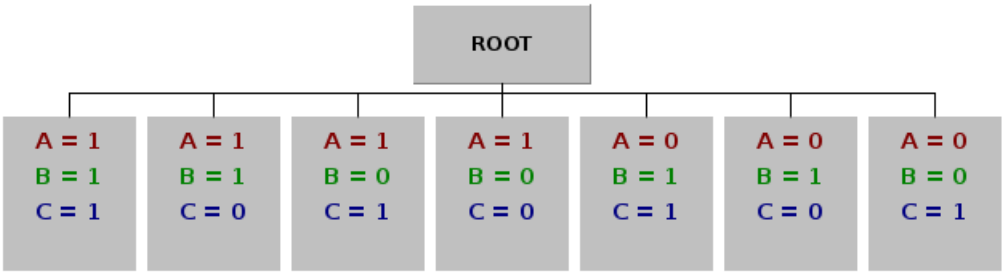

Figura 4.4: Exemplo do algoritmo *pre-branching* percorrendo todas a soluções possíveis.

Por fim, realizamos todos os testes utilizando as duas estratégias e não obtivemos

resultados satisfatórios esperados, ficando então um melhor estudo desse algoritmo para trabalhos futuros. No Capítulo 5 apresentamos todos os resultados obtidos do algoritmo.

5 EXPERIMENTOS COMPUTACIONAIS

Todos os modelos propostos neste trabalho foram desenvolvidos em linguagem C++, utilizando o compilador g++ 5.4.0 (com otimização -O3) e com o auxílio do resolvedor matemático CPLEX 12.7.1. Os experimentos foram executados em um computador com processador *Intel Core* I7-3770, com 16Gb de RAM e rodando o sistema operacional Linux Ubuntu 16.04. Os experimentos foram realizados com 1 *thread* e tempo limite de uma hora.

Para todos os experimentos realizados, adotamos o conjunto de instâncias definidas em Silva et al (2018). Estes grafos são divididos em dois conjuntos principais com $|V| = 100$ e $|V| = 200$. Cada conjunto tem 30 grafos diferentes com densidades selecionadas de $d = \{ld = 0.2; md = 0.5; hd = 0.8\}$, e o número de rótulos $|L|$ variando de $\frac{|V|}{2}$ até $2|V|$.

5.1 Comparação entre $PART_2$ e $\mathcal{MF}_d$

A Tabela 5.1 apresenta os resultados computacionais dos modelos $PART_2$ e $\mathcal{MF}_1$. Os resultados são as médias de um grupo de 10 instâncias de mesma densidade. Como vimos em capítulos anteriores, os dois modelos têm o mesmo grafo de entrada, porém, a estratégia $PART_2$ adciona todas restrições do modelo desde o início de sua execução, já a estratégia $\mathcal{MF}_1$ adiciona as restrições sob demanda, como definido em seu plano de cortes. Na coluna instância, d representa a densidade de arestas do grafo e $|V|$ o número de seus vértices. As váriaveis $tempo_r$ e $tempo$, dadas em segundos, representam o tempo na raiz e o tempo de resolução de cada modelo, respectivamente. Podemos observar que o $\mathcal{MF}_1$ obteve um menor tempo para solucionar todas as instâncias com $d = ld$ e $d = md$; $|V| = 100$ e $|V| = 200$. No entanto, as instâncias com $d = hd$ e $|V| = 200$, o $PART_2$ obteve um menor tempo, executando-as em um tempo médio de $371,0$ segundos, enquanto

o $\mathcal{MF}_1$ levou 480, 89 segundos para executar o mesmo conjunto de instâncias.

Tabela 5.1: Resultados computacionais dos modelos $PART_2$ e $\mathcal{MF}_1$

Instância		$PART_2$		$\mathcal{MF}_1$	
d	$\|V\|$	***tempo***$_r$	***Tempo(s)***	***tempo***$_r$	***Tempo(s)***
ld	100	0,41	2,15	0,30	**0,98**
ld	200	2,32	28,40	2,83	**15,82**
md	100	0,92	6,85	0,88	**3,79**
md	200	6,05	99,67	9,49	**65,67**
hd	100	1,27	10,63	1,40	**6,14**
hd	200	10,36	**371,00**	15,18	480,89

As Tabelas de 5.2 a 5.7 apresentam a comparação dos resultados computacionais de todos os modelos abordados neste trabalho. Cada tabela representa a média dos resultados de um grupo de 10 instâncias de mesma densidade. Cada linha representa o resultado das variáveis, sendo *LB*, o *lower bound* obtido na raiz do algoritmo *branch-and-bound*; *#cortes*$_r$, o número de cortes adicionados pelo usuário na resolução da raiz; *tempo*$_r$, o tempo computacional em segundos para a resolução da raiz; *#nós*, o número de nós resolvidos do algoritmo *branch-and-bound* para resolver a instância; *#cortes*, o número de desigualdades adicionadas pelo usuário no algoritmo *branch-and-cut*; *tempo*, o tempo de execução de cada modelo em segundos, e por fim, *#arestas*, o número de arestas do fecho cromático de entrada $\mathcal{F}_d(G) = (V, E_d, L)$. As colunas são divididas em 8 grupos: $PART_2$, que é o modelos proposto em Silva et al (2016), e as colunas $\mathcal{MF}_1$, $\mathcal{MF}_{n-1}$, $\mathcal{MF}_2$, $\mathcal{MF}_3$, $\mathcal{MF}_4$, $\mathcal{MF}_*$, $\mathcal{MF}_0$, representam a solução do algoritmo *branch-and-cut* proposto sobre o conjunto de modelos de mesmo nome proposto.

Tabela 5.2: Resultados computacionais para as instâncias $|V| = 100$ e $d = ld$

	$PART_2$	$\mathcal{MF}_1$	$\mathcal{MF}_{n-1}$	$\mathcal{MF}_2$	$\mathcal{MF}_3$	$\mathcal{MF}_4$	$\mathcal{MF}_*$	$\mathcal{MF}_0$
LB	0,24	0,18	0,18	0,18	0,18	0,18	0,18	0,18
#cortes$_r$	0,00	1961,70	4647,80	2900,90	3373,10	3929,30	1940,40	1959,30
tempo$_r$	0,41	0,30	0,65	0,41	0,47	0,53	0,27	0,28
#nós	189,90	194,20	194,30	193,80	194,10	194,20	194,40	194,40
#cortes	0,00	1963,50	4650,30	2906,20	3373,10	3941,80	**1944,30**	1961,10
tempo	2,15	0,98	1,97	1,31	1,47	1,61	**0,90**	0,92
#arestas	990,00	990,00	2342,50	1463,90	1701,40	1981,60	988,80	988,80

Todos os modelos chegaram ao ótimo em todas as instâncias, entretanto, o modelo

Tabela 5.3: Resultados computacionais para as instâncias $|V| = 100$ e $d = md$

	$PART_2$	$\mathcal{MF}_1$	$\mathcal{MF}_{n-1}$	$\mathcal{MF}_2$	$\mathcal{MF}_3$	$\mathcal{MF}_4$	$\mathcal{MF}_*$	$\mathcal{MF}_0$
LB	0,40	0,40	0,40	0,40	0,40	0,40	0,40	0,40
#cortes$_r$	0,00	4895,50	19720,30	7709,70	10063,80	12729,40	4688,50	4854,50
tempo$_r$	0,92	0,88	2,58	1,35	1,70	2,00	0,69	0,85
#nós	199,40	199,80	198,10	198,80	198,30	198,40	198,60	199,50
#cortes	0,00	5030,10	19982,70	7805,50	10120,00	12803,40	**4735,80**	5071,10
tempo	6,85	3,79	10,03	5,73	6,81	7,89	**3,02**	3,61
#arestas	2475,00	2475,00	9970,40	3894,70	5086,60	6430,40	2454,50	2454,50

$\mathcal{MF}_{n-1}$ obteve tempos computacionais muito elevados em comparação ao restante dos modelos. Como pode ser observado na Tabela 5.6, para as instâncias $\{|V| = 200; d = md\}$, foi gasto em média $977,07$ segundos para serem resolvidas pelo $\mathcal{MF}_{n-1}$, enquanto o $\mathcal{MF}_1$ resolveu estas mesmas instâncias em um tempo de $65,67$ segundos de média. Esse tempo computacional elevado do $\mathcal{MF}_{n-1}$ pode ser explicado pelo alto número de arestas inseridas no grafo original, produzindo um maior número de cortes adicionados no modelo. Para estas instâncias é possível observar que o fecho cromático $\mathcal{F}_1(G)$ tem uma média de $9950,00$ arestas, já o fecho $\mathcal{F}_{n-1}(G)$ tem em média $102457,00$ arestas.

Tabela 5.4: Resultados computacionais para as instâncias $|V| = 100$ e $d = hd$

	$PART_2$	$\mathcal{MF}_1$	$\mathcal{MF}_{n-1}$	$\mathcal{MF}_2$	$\mathcal{MF}_3$	$\mathcal{MF}_4$	$\mathcal{MF}_*$	$\mathcal{MF}_0$
LB	0,60	0,59	0,60	0,60	0,60	0,60	0,60	0,59
#cortes$_r$	0,00	7834,30	58662,70	13198,00	18980,10	26505,10	7116,70	7658,70
tempo$_r$	1,27	1,40	9,26	2,27	3,43	4,31	1,00	1,28
#nós	250,70	261,10	195,80	206,00	207,80	203,10	206,80	279,10
#cortes	0,00	8132,00	59539,00	13455,80	19355,80	26957,90	**7335,20**	8098,20
tempo	10,63	6,14	33,44	9,39	13,47	16,80	**4,07**	5,96
#arestas	3960,00	3960,00	29645,20	6669,70	9596,80	13397,40	3872,20	3872,20

Tabela 5.5: Resultados computacionais para as instâncias $|V| = 200$ e $d = ld$

	$PART_2$	$\mathcal{MF}_1$	$\mathcal{MF}_{n-1}$	$\mathcal{MF}_2$	$\mathcal{MF}_3$	$\mathcal{MF}_4$	$\mathcal{MF}_*$	$\mathcal{MF}_0$
LB	0,17	0,17	0,17	0,17	0,17	0,17	0,17	0,17
#cortes$_r$	0,00	7915,30	23338,10	11491,70	13876,80	16485,50	7813,10	7898,30
tempo$_r$	2,32	2,83	5,83	4,76	5,37	5,14	2,39	2,81
#nós	395,60	395,70	395,90	395,70	394,90	395,30	396,00	395,70
#cortes	0,00	8081,00	23931,40	11646,30	14005,30	17058,70	**7992,00**	7989,00
tempo	28,40	15,82	35,08	25,67	28,77	27,80	**13,88**	14,76
#arestas	3980,00	3980,00	11734,00	5778,90	6979,80	8290,50	3971,50	3971,50

Ainda com base nos resultados, podemos ver que o modelo $\mathcal{MF}_*$ obteve o menor tempo em todas as médias dos conjuntos de instâncias. Como por exemplo, na Tabela 5.7,

para as instâncias $\{|V| = 200; d = hd\}$ $\mathcal{MF}_1$ precisou de $480,89$ segundos de média para resolvê-las, já o $\mathcal{MF}_*$ obteve $93,30$ segundos de média para resolver o mesmo conjunto de instâncias, obtendo uma redução superior a 80% do tempo computacional. Essa redução no tempo se deve a adoção do fecho cromático $\mathcal{F}_*(G)$, o que gera uma diminuição no número de cortes adicionados ao modelo, podendo ser visto nas linhas *#cortes*, e ao algoritmo *branch-and-cut* proposto.

Tabela 5.6: Resultados computacionais para as instâncias $|V| = 200$ e $d = md$

	$PART_2$	$\mathcal{MF}_1$	$\mathcal{MF}_{n-1}$	$\mathcal{MF}_2$	$\mathcal{MF}_3$	$\mathcal{MF}_4$	$\mathcal{MF}_*$	$\mathcal{MF}_0$
LB	0,40	0,40	0,40	0,40	0,40	0,40	0,40	0,40
#cortes$_r$	0,00	20138,10	203793,91	34476,40	49836,50	75049,40	18840,00	19742,30
tempo$_r$	6,05	9,49	69,53	16,41	21,13	27,46	6,91	9,31
#nós	403,80	405,80	388,00	400,40	398,80	406,30	398,20	406,30
#cortes	0,00	21270,60	208709,80	35355,20	50974,90	77668,80	**19349,60**	20194,80
tempo	99,67	65,67	977,07	112,88	175,58	304,94	**48,63**	65,62
#arestas	9950,00	9950,00	102457,00	17327,30	25060,40	37733,30	9769,80	9769,80

A Tabela 5.8, apresenta os resultados do algoritmo *pre-branching*, onde cada linha da tabela é a média de um grupo de 10 instâncias de mesma densidade. O $\mathcal{PB}_1$, representa a estratégia onde $T = \{A, B, C, D\}$ e $\ell \in T$, que consiste em explorar o espaço de soluções possíveis para os rótulos A, B, C, D. Já o $\mathcal{PB}_2$, representa a segunda estratégia, onde dado uma árvore geradora colorida $T = \{A, B, C\}$, consiste em testar todas as combinações booleanas possíveis entre os rótulos, com exceção de $A = B = C = 0$. Pode-se notar que a primeira estratégia obteve melhores resultados na maior parte dos conjuntos de instâncias, porém, no geral, o algoritmo *pre-branching* não conseguiu tempos satisfatórios, chegando a aumentar o tempo de resolução do modelo.

Por fim, as Tabelas 5.9 e 5.10 apresentam o valor obtido pelo método $\mathcal{MF}_*$ para cada instância dos conjuntos com $|V| = 100$ e $|V| = 200$, respectivamente. A coluna d define a densidade do grafo original; *#id*, o identificador da instânca; *opt*, o valor ótimo alcançado; e $tempo(s)$, o tempo computacional em segundos do método para resolver a instância.

Tabela 5.7: Resultados computacionais para as instâncias $|V| = 200$ e $d = hd$

	$PART_2$	$\mathcal{MF}_1$	$\mathcal{MF}_{n-1}$	$\mathcal{MF}_2$	$\mathcal{MF}_3$	$\mathcal{MF}_4$	$\mathcal{MF}_*$	$\mathcal{MF}_0$
LB	0,62	0,62	0,62	0,62	0,62	0,62	0,62	0,62
#cortes$_r$	0,00	32515,50	338069,91	53743,70	79652,00	116871,40	30549,80	31875,90
tempo$_r$	10,36	15,18	124,44	24,04	36,99	49,48	10,74	16,16
#nós	6009,10	8349,30	400,40	2468,30	626,90	696,80	754,60	9306,90
#cortes	0,00	33104,60	345123,31	56245,20	82607,10	119636,40	**31202,00**	32980,70
tempo	371,01	480,89	1893,44	402,23	420,76	811,94	**93,30**	295,43
#arestas	15920,00	15920,00	169869,80	27003,80	40016,30	58721,50	15607,70	15607,70

Tabela 5.8: Resultados do algoritmo *pre-branching*

Instância		$\mathcal{PB}_1$	$\mathcal{PB}_2$	$\mathcal{MF}_*$
d	$\|V\|$	***Tempo(s)***	***Tempo(s)***	***Tempo(s)***
ld	100	1,87	2,15	**0,90**
ld	200	32,00	33,51	**13,88**
md	100	5,05	5,46	**3,02**
md	200	90,70	88,85	**48,63**
hd	100	6,40	7,49	**4,07**
hd	200	159,93	126,34	**93,30**

Tabela 5.9: Resultados computacionais do $\mathcal{MF}_*$ para as instâncias com $|V| = 100$

Instâncias			$\mathcal{MF}_*$	**Instâncias**			$\mathcal{MF}_*$	**Instâncias**			$\mathcal{MF}_*$
d	***#id***	***opt***	***Tempo(s)***	***d***	***#id***	***opt***	***Tempo(s)***	***d***	***#id***	***opt***	***Tempo(s)***
	0	13	0,95		0	38	3,40		0	31	2,08
	1	14	0,94		1	37	3,29		1	47	4,45
	2	15	0,88		2	35	2,99		2	40	4,65
	3	13	0,71		3	23	1,46		3	42	6,19
ld	4	13	0,90	md	4	35	3,57	hd	4	37	2,98
	5	15	0,99		5	32	2,51		5	42	3,95
	6	14	0,95		6	37	3,91		6	38	4,95
	7	14	0,88		7	30	1,93		7	36	3,94
	8	14	0,89		8	39	3,43		8	36	3,63
	9	14	0,91		9	35	3,76		9	38	3,86

Tabela 5.10: Resultados computacionais do $\mathcal{MF}_*$ para as instâncias com $|V| = 200$

Instâncias		$\mathcal{MF}_*$		**Instâncias**		$\mathcal{MF}_*$		**Instâncias**		$\mathcal{MF}_*$	
d	***#id***	***opt***	***Tempo(s)***	***d***	***#id***	***opt***	***Tempo(s)***	***d***	***#id***	***opt***	***Tempo(s)***
	0	29	11,89		0	51	24,75		0	85	82,08
	1	27	13,69		1	73	62,03		1	95	106,37
	2	33	16,97		2	68	55,01		2	75	63,24
	3	29	13,85		3	43	43,84		3	92	109,15
ld	4	31	15,31	md	4	75	71,31	hd	4	69	99,43
	5	29	13,43		5	64	46,08		5	86	76,81
	6	26	12,42		6	52	30,33		6	102	118,09
	7	30	14,01		7	68	62,77		7	93	106,15
	8	30	13,42		8	63	44,09		8	89	106,03
	9	25	13,86		9	71	46,12		9	83	65,64

6 CONCLUSÃO

Neste trabalho, abordamos o Problema do Corte Global Rotulado Mínimo (PCGRM). O PCGRM é um problema de otimização combinatória e sua complexidade ainda continua como uma questão teórica em aberto (Xu e Faragó, 2019). Este problema está intimamente relacionado à vulnerabilidade de redes multicamadas e tem como objetivo medir a conectividade da rede quando esta possue riscos compartilhados de falhas. O PCGRM não está apenas relacionado a redes, mas também pode ser aplicado em sistemas de planejamento de transportes. Então, com o objetivo de solucionar o PCGRM, desenvolvemos algumas estratégias de resolução exata e uma nova família de formulações matemáticas, que para desenvolvê-la, nos baseamos em um modelo presente na litaratura chamado $PART_2$.

A seguir, de forma breve, podemos elencar algumas contribuições deste trabalho:

- Desenvolvemos o modelo $\mathcal{MF}_d$, e como dito anteriormente, é uma nova família de formulações matemáticas. Para obter o $\mathcal{MF}_d$, estendemos e adaptamos o conceito de fecho cromático definido em Silva (2019), e aplicamos no grafo G de entrada, gerando um novo conjunto de arestas $E^{'}$;

- Para resolver o $\mathcal{MF}_d$, desenvolvemos um algoritmo *branch-and-cut*. As desigualdades do modelo são inseridas de forma dinâmica durante a resolução dos nós, sendo inicialmente relaxadas e checada sua violação quando, e somente quando, uma solução inteira ou fracionada é encontrada;

- Com o intuito de melhorar o $\mathcal{MF}_d$, desenvolvemos um algoritmo chamado *pré-branching*. Esse algoritmo tem como objetivo guiar o resolvedor antes que ele inicie o *branching*.

Dessa forma, realizamos uma análise comparativa dos resultados dos modelos $\mathcal{MF}_d$ e $PART_2$. Diante dos resultados obtidos, podemos notar, que o $\mathcal{MF}_1$ conseguiu melhorar o tempo computacional em $83,3\%$ dos conjuntos de instâncias resolvidos comparado com o modelo $PART_2$. Os dois modelos têm o mesmo grafo de entrada, porém, o que os diferem é o algoritmo *branch-and-cut* utilizado para resolver o $\mathcal{MF}_1$. Outro ponto interessante que podemos destacar, são os resultados do modelo $\mathcal{MF}_*$. O $\mathcal{MF}_*$ conseguiu ser melhor que o $PART_2$ em todos os conjuntos de instâncias resolvidos. Para termos uma melhor ideia disso, o $\mathcal{MF}_*$ obteve uma redução de cerca de 75% no tempo computacional para resolver o conjunto de instâncias com $\{|V| = 200;d = hd\}$ comparado ao $PART_2$.

Com isso, observamos que o modelo $\mathcal{MF}_*$ foi bem superior em comparação aos outros modelos abordados, o que se deve ao algoritmo *branch-and-cut* e ao fecho cromático $\mathcal{F}_*(G)$. Portanto, concluimos que o conceito de fecho cromático foi um fator importante para obter os resultados deste trabalho, e consideramos um tema promissor para futuras pesquisas. Estudos mais aprofundados sobre o algoritmo *pre-branching*, sobre a escolha do vértice raiz da estrela monocromática no $\mathcal{F}_*(G)$ e qual escolha da aresta que quebra o ciclo monocromático no $\mathcal{F}_0(G)$ ficarão para trabalhos futuros.

REFERÊNCIAS BIBLIOGRÁFICAS

ARENALES, M.; MORABITO, R.; ARMENTANO, V.; YANASSE, H. *Pesquisa operacional: para cursos de engenharia*. Elsevier Brasil, 2015.

BONDY, J. A.; MURTY, U. S. R. *Graph Theory*, volume 244 of *Graduate Texts in Mathematics*. Springer London, London, 2008.

BORDINI, A.; PROTTI, F.; DA SILVA, T. G.; FILHO, G. F. D. S. New algorithms for the minimum coloring cut problem. *International Transactions in Operational Research*, v. 0, n. 0. doi: 10.1111/itor.12494. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12494`, 2017.

BROERSMA, H.; LI, X.; WOEGINGER, G. J.; ZHANG, S. Paths and cycles in colored graphs. *Australasian J. Combinatorics*, v. 31, p. 299–312, 2005.

CARRABS, F.; CERULLI, R.; DELL'OLMO, P. A mathematical programming approach for the maximum labeled clique problem. *Procedia-Social and Behavioral Sciences*, v. 108, p. 69–78, 2014.

CARRABS, F.; CERULLI, R.; GENTILI, M. The labeled maximum matching problem. *Computers & Operations Research*, v. 36, n. 6, p. 1859–1871, 2009.

CHANG, R.-S.; LEU, S.-J. The minimum labeling spanning trees. *Inf. Process. Lett.*, v. 63, n. 5, p. 277–282. ISSN 0020-0190. doi: 10.1016/S0020-0190(97)00127-0. URL `http://dx.doi.org/10.1016/S0020-0190(97)00127-0`, 1997.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. *Introdução aos algoritmos*, 2009.

COUDERT, D.; DATTA, P.; PÉRENNES, S.; RIVANO, H.; VOGE, M.-E. Shared risk resource group complexity and approximability issues. *Parallel Processing Letters*, v. 17, n. 02, p. 169–184, 2007.

COUDERT, D.; PÉRENNES, S.; RIVANO, H.; VOGE, M.-E. Combinatorial optimization in networks with shared risk link groups, 2016.

FARAGÓ, A. A graph theoretic model for complex network failure scenarios. *Proceedings of the Eighth INFORMS Telecommunications Conference, Dallas, Texas*, 2006.

FELLOWS, M. R.; GUO, J.; KANJ, I. The parameterized complexity of some minimum label problems. *Journal of Computer and System Sciences*, v. 76, n. 8, p. 727–740, 2010.

GOLDBARG, M. C.; LUNA, H. P. L. Otimização combinatória e programação linear. *Editora Campus*, v. 2, 2000.

GRANATA, D.; CERULLI, R.; SCUTELLÀ, M. G.; RAICONI, A. Maximum flow problems and an np-complete variant on edge-labeled graphs. *Handbook of Combinatorial Optimization*, p. 1913–1948. Springer, 2013.

JHA, S.; SHEYNER, O.; WING, J. Two formal analyses of attack graphs. *Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15*, p. 49–63. IEEE, 2002.

KRUMKE, S. O.; WIRTH, H.-C. On the minimum label spanning tree problem. *Information Processing Letters*, v. 66, n. 2, p. 81–85, 1998.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. *Comput. Oper. Res.*, v. 24, n. 11, p. 1097–1100. ISSN 0305-0548, 1997.

SHEYNER, O.; HAINES, J.; JHA, S.; LIPPMANN, R.; WING, J. M. Automated generation and analysis of attack graphs. *Proceedings 2002 IEEE Symposium on Security and Privacy*, p. 273–284. IEEE, 2002.

SHEYNER, O.; WING, J. Tools for generating and analyzing attack graphs. *International Symposium on Formal Methods for Components and Objects*, p. 344–371. Springer, 2003.

SILVA, T. G. The minimum labeling spanning tree and related problems. *Anais do XXXII Concurso de Teses e Dissertações*. SBC, 2019.

SILVA, T. G. D.; DE SOUSA FILHO, G. F.; OCHI, L. S.; MICHELON, P.; GUEYE, S.; CABRAL, L. A. Métodos exatos aplicados ao problema do corte global rotulado mınimo. *Anais do XLVIII Simpósio Brasileiro de Pesquisa Operacional*, p. 4152–4162, 2016.

SILVA, T. G. D.; SOUSA FILHO, G. F. D.; MALHEIROS, I. A.; MLADENOVIC, N.; CABRAL, L. A.; OCHI, L. S.; ALOISE, D. Efficient heuristics for the minimum labeling global cut problem. *Electronic Notes in Discrete Mathematics*, v. 66, p. 23 – 30. ISSN 1571-0653. doi: https://doi.org/10.1016/j.endm.2018.03.004. URL `http://www.sciencedirect.com/science/article/pii/S1571065318300507`. 5th International Conference on Variable Neighborhood Search, 2018.

SILVESTRI, S.; LAPORTE, G.; CERULLI, R. The rainbow cycle cover problem. *Networks*, v. 68, n. 4, p. 260–270, 2016.

SUCUPIRA, R.; FARIA, L.; KLEIN, S.; SAU, I.; SOUZA, U. S. Maximum cuts in edge-colored graphs. *Electronic Notes in Discrete Mathematics*, v. 62, p. 87 – 92. ISSN 1571-0653. doi: https://doi.org/10.1016/j.endm.2017.10.016. URL `http://www.sciencedirect.com/science/article/pii/S1571065317302548`. LAGOS'17 – IX Latin and American Algorithms, Graphs and Optimization, 2017.

TANG, L.; ZHANG, P. Approximating minimum label s-t cut via linear programming. *Latin American Symposium on Theoretical Informatics*, p. 655–666. Springer, 2012.

XIONG, Y.; GOLDEN, B.; WASIL, E. The colorful traveling salesman problem. *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*, p. 115–123. Springer, 2007.

XU, R.; FARAGÓ, A. The landscape of minimum label cut (hedge connectivity) problem. *arXiv preprint arXiv:1908.06541*, 2019.

ZHANG, P. Efficient algorithms for the label cut problems. *International Conference on Theory and Applications of Models of Computation*, p. 259–270. Springer, 2014.

ZHANG, P.; CAI, J.-Y.; TANG, L.-Q.; ZHAO, W.-B. Approximation and hardness results for label cut and related problems. *Journal of Combinatorial Optimization*, v. 21, n. 2, p. 192–208, 2011.

ZHANG, P.; TANG, L. Minimum label st cut has large integrality gaps. *arXiv preprint arXiv:1908.11491*, 2019.