SELEÇÃO DE REPRESENTANTES PARA COBERTURA DE COMPONENTES CONEXAS EM GRAFOS

ANDERSON LEMOS DA SILVA

SELEÇÃO DE REPRESENTANTES PARA COBERTURA DE COMPONENTES CONEXAS EM GRAFOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Vinícius Fernandes dos Santos
Coorientadora: Olga Goussevskaia

Belo Horizonte
Dezembro de 2018

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Silva, Anderson Lemos da

S586s Seleção de representantes para cobertura de componentes conexas em grafos / Anderson Lemos da Silva — Belo Horizonte, 2018.
xix, 61 p.:il.; 29 cm.

Dissertação (mestrado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.

Orientador: Vinícius Fernandes dos Santos
Coorientadora: Olga Nikolaevna Goussevskaia

1. Computação – Teses. 2. Redes de computadores. 3. Teoria dos grafos. 4. Algoritmos de aproximação. 5. Problemas NP-completo. I. Orientador. II. Coorientadora. III. Título.

CDU 519.6*22(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Seleção de representantes para cobertura de componentes conexas em grafos

ANDERSON LEMOS DA SILVA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Vinícius Fernandes dos Santos - Orientador
Departamento de Ciência da Computação - UFMG

Profa. Olga Nikolaevna Goussevskaia - Coorientadora
Departamento de Ciência da Computação - UFMG

Prof. Sebastián Alberto Urrutia
Departamento de Ciência da Computação - UFMG

Prof. Maurício de Lemos Rodrigues Collares Neto
Departamento de Matemática - UFMG

Belo Horizonte, 17 de dezembro de 2018.

À minha mãe.

Agradecimentos

Agradeço primeiramente à Deus por todas as bençãos que me tem concedido.

Agradeço aos meus pais, Toin e Dilene, por terem me dado a educação e apoio necessários para minha formação e por terem sempre sido meus heróis. Agradeço ao meu irmão, Wendell, pela amizade e companheirismo que sempre tivemos.

Agradeço a Lisiani, minha companheira, que sempre esteve comigo, nos bons e maus momentos dessa caminhada. Que me deu força a todo tempo, e que é a pessoa mais incrível que conheço. Minha parcerita, obrigado.

Agradeço ao professor Vinicius Fernandes pela orientação neste trabalho. Da mesma forma, agradeço a professora Olga Goussevskaia, pela coorientação neste trabalho. Obrigado aos dois por terem contribuído na minha formação acadêmica, por me ajudarem tanto e por serem os melhores orientadores que um aluno pode ter. Uma honra trabalhar com vocês.

Agradeço aos professores Arthur Araruna, Davi Romero e Wladimir Tavares, da Universidade Federal do Ceará, por, na minha graduação, terem me ajudado muito com orientações de TCC e bolsa e por terem me recomendado à UFMG. Agradeço aos professores Sebastián Urrutia, Jussara Almeida, Luiz Chaimowicz, Geraldo Robson Mateus e Wagner Meira por terem contribuído na minha formação acadêmica no mestrado. Agradeço ao professor Mauricio Collares por aceitar participar da banca desta dissertação. Agradeço às pessoas da secretaria do PPGCC pela enorme ajuda em tudo.

Agradeço aos meus amigos que de alguma forma me ajudaram no tempo em que estive no mestrado: Lucas, Douglas, João, Guilherme, Armando, Evelin, Evellyn, Luiza, Ricardo, Bárbara, Matheus, Junior, Araújo, Daniel, Kerley, Yago, Alysson, Rodrigo, Patrícia, Alberta, Tércio e todos que conviveram comigo ao longo da minha formação.

E a todos que direta ou indiretamente fizeram parte da minha formação.

"Challenge accepted!"
(Barney Stinson - How I Met Your Mother)

Resumo

A partir da generalização de problemas de conectividade em redes de comunicação *ad hoc*, amplamente estudados na literatura, introduzimos um novo problema de cobertura, chamado *Component Cover by Vertices* (CCV). Neste problema, são dados um grafo G e uma partição A, B de $V(G)$, e nosso objetivo é encontrar o menor subconjunto B' de B tal que, para cada componente conexa C de $G[A]$, há um vértice $v \in C$ tal que v tem um vizinho em B'. Estudamos a complexidade deste problema e damos resultados positivos e negativos. Em particular, mostramos que o problema CCV é NP-Completo para várias classes de grafos, como cordais, bipartidos com grau de vértice no máximo três, *Unit Disk Graphs* (UDGs) planares e grades. Além disso, mostramos que o problema também é difícil de aproximar. Também discutimos uma conexão entre CCV e o problema *Reb-Blue Dominating Set*, deduzindo que CCV é $W[2]$-Completo em grafos gerais. Por outro lado, apresentamos algoritmos polinomiais para as classes de grafos de blocos, cografos, cactos e *outerplanares*. Mostramos que o problema é FPT se parametrizado por largura arbórea ou por tamanho da solução e grau máximo. Por fim, mostramos dois algoritmos aproximativos para grafos gerais e um algoritmo aproximativo de fator constante para a classe de UDGs.

Palavras-chave: Cobertura em grafos, NP-Completude, Algoritmos aproximativos, Parametrização.

Abstract

Based on the generalization of connectivity problems in ad hoc communication networks, widely studied in the literature, we introduce a new covering problem, called Componet Cover by Vertices (CCV). In this problem, we are given a graph G and a partition A, B of $V(G)$, and our goal is to find the smallest subset B' of B such that, for every connected component C of $G[A]$, there is a vertex $v \in C$ such that v has a neighbor in B'. We study the complexity of this problem and give positive and negative results. In particular, we show that the CCV problem is NP-Complete for several classes of graphs, such as chordal, bipartite with vertex degree at most three, planar Unit Disk Graphs (UDGs) and grids. Moreover, we show that the problem is also hard to approximate. We also discuss a connection between CCV and the so called Reb Blue Dominating Set problem, deducing that CCV W[2]-Complete in general graphs. On the other hand, we present polynomial algorithms for the classes of block graphs, cographs, cactus and outerplanar graphs. We show that the problem is FPT if parameterized by treewidth or by solution size and maximum degree. Finally, we show two approximation algorithms for general graphs and a constant-factor approximation algorithm for the class of UDGs.

Keywords: Graph covering, NP-Completeness, Approximation algorithms, Parameterization.

Lista de Figuras

Sumário

Capítulo 1

Introdução

A comunicação tem se tornado indispensável, sendo cada vez mais crucial a necessidade da sua obtenção de forma rápida e precisa. Em meio a essa evolução da necessidade de comunicação, informação e velocidade, os meios digitais tem se tornado uma alternativa cada vez mais utilizada. A comunicação por meios digitais se beneficia de dispositivos e componentes que se conectam por meio de uma rede. Porém, com tantos dispositivos e componentes, surgiram os problemas de rede e de conectividade.

Quando se têm tantos dispositivos que devem se comunicar é importante encontrar uma boa forma de conectá-los levando em consideração critérios como custo, velocidade de comunicação, entre outros, e isso pode não ser trivial. Alguns desses fatores são abordados no trabalho desenvolvido por Ferreira Neto et al. [2017], que estuda o problema de infraestrutura de acesso em redes sem fio obstruídas — do inglês, *Obstructed Wireless Network Backbone Cover* (OWN-BC).

Em seu trabalho, Ferreira Neto et al. [2017] provam que OWN-BC é NP-Completo, isto é, está entre os problemas mais difíceis conhecidos na teoria da computação. Ferreira Neto et al. [2017] também mostram que OWN-BC pode ser modelado por meio de grafos. Essa abstração nos permite estudar generalizações do problema que podem ser relevantes.

Nesta dissertação, objetivamos o estudo das variações do problema *Component Cover by Vertices* (CCV) — em tradução oficial, Cobertura de Componentes por Vértices —, que é um novo problema de cobertura em grafos, motivado pela generalização do problema OWN-BC. Iremos analisar o problema CCV em diferentes tipos de grafos e desenvolver algoritmos eficientes para resolução do mesmo. O problema CCV é definido da seguinte maneira: dado um grafo G e uma partição A, B dos vértices de G, nosso objetivo é encontrar o menor subconjunto B' de B tal que, para cada componente conexa composta por vértices de A, existe um vértice da mesma que é vizinho

de algum vértice de B'.

Intuitivamente, podemos visualizar o problema CCV da seguinte forma: considere um cenário onde existam dois conjuntos de elementos: A e B. Elementos em A são capazes de formar caminhos de comunicação entre si. Elementos de B são capazes de prover algum serviço, tal como conectividade para uma infraestrutura global, aos elementos de A, contanto que haja um caminho passando só por elementos de A entre um elemento de A e um elemento de B. O objetivo consiste em encontrar um (pequeno) subconjunto B' de elementos de B, de tal forma que todos os elementos de A tenham acesso ao serviço fornecido, ou seja, há um caminho através dos vértices de A, conectando cada elemento em A, a pelo menos um elemento em B'. Uma instância do problema CCV é ilustrada na Figura 1.1, na qual círculos representam vértices de A, triângulos representam vértices de B e vértices sombreados representam uma solução para o problema.

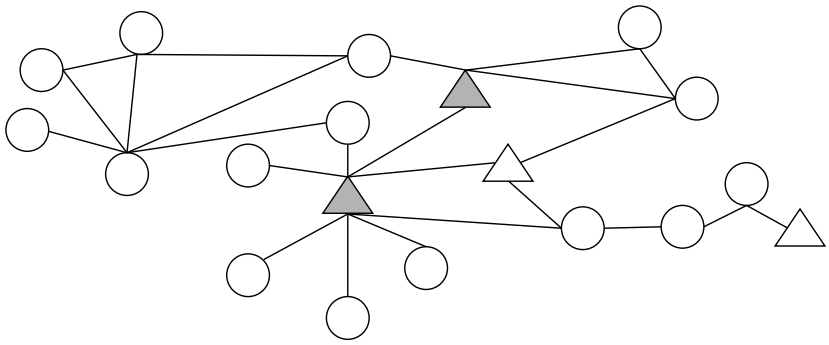

Figura 1.1: Exemplo de instância do problema CCV.

Um exemplo de aplicação deste problema foi estudado em Ferreira Neto et al. [2017], com ênfase nas propriedades de conectividade em redes sem fio obstruídas, aspectos geométricos e distribuições aleatórias de nós. O cenário estudado consiste em um conjunto de dispositivos de comunicação e um conjunto de pontos de acesso a um *backbone* que fornece conectividade à internet. Os dispositivos têm um raio de transmissão e destinam-se a formar uma topologia conexa. Em cenários onde o raio de transmissão não tem alcance suficiente para conectar todos os dispositivos uns aos outros, ou seja, a rede de dispositivos é dividida em várias componentes conexas, os pontos de acesso se tornam úteis. Dois dispositivos podem se comunicar se houver um caminho entre eles, isto é, se ambos pertencerem a mesma componente conexa, ou se houver um caminho entre cada dispositivo e algum ponto de acesso do *backbone*. O objetivo é, então, encontrar o menor subconjunto de pontos de acesso de modo que todos os dispositivos estejam conectados a pelo menos um ponto de acesso. Mapeando este problema para o CCV, o conjunto A seria composto dos dispositivos, e o conjunto

B seria composto pelos pontos de acesso. Pares de vértices de A e B seriam conectados por arestas se os dispositivos ou pontos de acesso correspondentes estivessem dentro do alcance de comunicação um do outro. Queremos então encontrar o menor subconjunto B' de pontos de acesso, para que todos os dispositivos em A estejam conectados a pelo menos um ponto de acesso em B' por meio de um caminho.

1.1 Sobre esta dissertação

Neste Capítulo mostra-se uma introdução e motivação do problema a ser estudado. Vemos aqui a conexão do problema CCV com o problema OWN-BC, estudado por Ferreira Neto et al. [2017]. O resto da dissertação é organizado como descrito a seguir.

O Capítulo 2 define conceitos e notações importantes utilizadas neste trabalho, ou seja, é mostrada a fundamentação teórica. Mostramos definições e notações de grafos, componentes conexas, classes de grafos, definição de algoritmos de complexidade parametrizada e algoritmos aproximativos.

No Capítulo 3 discutimos problemas relacionados ao CCV, como *Vertex Cover*, *Set Cover* e *Red-Blue Dominating Set*. Também mostramos trabalhos relacionados a este, tais como Levin [2008] e Feige [1998], que estudam aproximação de algoritmos para o problema *Set Cover*. Esses algoritmos são relevantes para o nosso trabalho, pois, a partir deles, conseguimos provar resultados de aproximação para o problema CCV.

No Capítulo 4 definimos e formalizamos o problema estudado neste trabalho, o *Component Cover by Vertices*. Definimos a versão de decisão e de otimização do problema. Também fazemos uma relação do problema com outros problemas de cobertura estudados na literatura.

O Capítulo 5 contém resultados negativos encontrados para o problema em relação a NP-Completude para determinadas classes de grafos. São resultados em que não conseguimos melhorias, mas provamos que o problema é difícil mesmo se impormos restrições de uma classe ao grafo de entrada. Demonstramos que CCV é NP-Completo para várias classes de grafos: bipartidos com máximo grau 3, grafos cordais, *Unit Disk Graphs* (UDGs) planares e grades.

Ao contrário do Capítulo 5, em que mostramos que o problema é NP-Completo para várias classes de grafos, no Capítulo 6 mostramos algoritmos polinomiais exatos para outras classes de grafos. Mostramos que CCV pode ser resolvido em tempo polinomial para árvores, blocos, cografos, cactos e grafos *outerplanares*.

O Capítulo 7, além de definir classes de complexidade parametrizada, apresenta

resultados positivos e negativos sobre parametrização para o problema CCV. Como resultado negativo, mostramos que o problema é $W[2]$-Completo se parametrizado pelo parâmetro natural k. No entanto, como resultados positivos, mostramos também que CCV em grafos gerais pode ser resolvido em tempo *Fixed-Parameter Tractable* (FPT) se parametrizado pela largura arbórea (*treewidth*) ou pela combinação do parâmetro natural k e grau máximo do grafo. Além disso, para grafos planares, o mesmo pode ser resolvido em tempo FPT, se parametrizado pelo parâmetro natural k.

Os resultados sobre algoritmos aproximativos são mostrados no Capítulo 8. mostramos que se o fator de aproximação for baseado no número c de componentes que queremos cobrir, é impossível encontrar um algoritmo com fator melhor que $O(\log c)$ (a menos que P = NP). Porém, apresentamos também três algoritmos aproximativos: dois para grafos gerais, com fatores de aproximação $O(\log c)$ e Δ_A (número máximo de vértices de B que podem cobrir uma componente de A) e um para UDGs com fator de aproximação $1,79$.

Finalmente, o Capítulo 9 conclui mostrando o que foi feito na dissertação, dando as considerações finais e falando sobre possíveis trabalhos futuros.

Capítulo 2

Notações e Definições

Neste Capítulo, mostraremos notações e definições importantes para este trabalho. A maioria dessas, são baseadas em Cormen et al. [2009] e Bondy et al. [1976]. Tais notações e definições são baseadas na teoria dos grafos.

Um grafo modela conexões entre duas entidades, onde tais entidades são chamadas de vértices e uma conexão entre duas delas é uma aresta — como um exemplo, os vértices podem ser cidades e uma aresta entre dois vértices pode significar que há uma estrada entre as duas cidades. Se um vértice x tem aresta pra um vértice y, dizemos que x é vizinho de y, ou que x é adjacente a y. Representamos como $\boldsymbol{G = (V, E)}$ um grafo G com o conjunto de vértices V e o conjunto de arestas E. Grafos podem ser direcionados, isto é, a relação tem uma direção — x estar conectado com y não significa que y está conectado com x — e também podem ser ponderados — por exemplo, um peso em uma aresta pode representar a distância entre duas cidades. Neste trabalho assumimos que os grafos são não ponderados e não orientados a menos que se diga o contrário.

Assim sendo, dado um grafo $G = (V, E)$, dizemos que $\boldsymbol{V(G)}$ é o conjunto de vértices do grafo G, onde $\boldsymbol{n} = |V(G)|$, e $\boldsymbol{E(G)}$ é conjunto de arestas do grafo G, onde $\boldsymbol{m} = |E(G)|$. Dizemos também que, se $v \in V(G)$, a vizinhança aberta de v, denotada por $\boldsymbol{N(v)}$, é o conjunto de vértices vizinhos de v, ou seja, $\forall u \in V(G)$, com $u \neq v$, temos que $(u, v) \in E(G)$ se e somente se $u \in N(v)$. Definimos então sua vizinhança fechada $\boldsymbol{N[v]}$ que é o conjunto de seus vizinhos e ele mesmo, ou seja, $N[v] = N(v) \cup \{v\}$. Chama-se grau de um vértice v, o número de vizinhos que o mesmo possui, ou $|N(v)|$, e denota-se o grau de v por $\boldsymbol{d(v)}$. Assim, o grau mínimo de um grafo é o menor grau entre os seus vértices, denotado por $\boldsymbol{\delta(G)}$ e o grau máximo de um grafo é o maior grau entre os seus vértices, denotado por $\boldsymbol{\Delta(G)}$.

Um **caminho** é uma sequência de vértices distintos $P = <v_1, v_2, ..., v_{|P|}>$ tal que

existe aresta entre todos os pares de vértices consecutivos do caminho, ou seja, $\forall v_i, v_{i+1}$ com $i \in \{1, 2, ..., |P| - 1\}$ temos que $(v_i, v_{i+1}) \in E$.

Dado um conjunto de elementos S, Um conjunto $S' \subseteq S$ é dito **maximal** em relação a uma propriedade π se não existir um conjunto $S'' \supset S'$ tal que S'' contém a propriedade π. Em outras palavras, S' é maximal se ao adicionar qualquer coisa em S' ele perde a propriedade π. De forma análoga, um conjunto $S' \subseteq S$ é dito **minimal** em relação a uma propriedade π se ao remover qualquer coisa de S' ele perde a propriedade π. Existem também notações importantes para subconjuntos de vértices e arestas. Por exemplo, dado um subconjunto de arestas $Y \subseteq E(G)$, $\boldsymbol{V(Y)}$ é o conjunto de todos os vértices de G que são extremidades de pelo menos alguma aresta em Y. De forma similar, dado um subconjunto de vértices $X \subseteq V(G)$, $\boldsymbol{E(X)}$ é o conjunto de todas as arestas de G que têm extremidades em vértices de X. Com essas definições podemos falar sobre subgrafos e subgrafos induzidos.

Dizemos que um grafo $G' = (V', E')$ é um **subgrafo** de um grafo $G = (V, E)$ se é formado por vértices e arestas de G, ou seja, $V' \subseteq V$ e $E' \subseteq E$. Um **subgrafo induzido** por um conjunto de vértices X, denominado $\boldsymbol{G[X]}$, é um grafo com os vértices de X e as arestas em $E(X)$. Analogamente, um subgrafo induzido por um conjunto de arestas Y, denotado $\boldsymbol{G[Y]}$, é um grafo com as arestas de Y e os vértices em $V(Y)$. A partir disto, definimos uma **componente conexa** como sendo um subgrafo em que dois vértices pertencem a ela se e somente se há pelo menos um caminho entre eles e nenhum deles é conectado a nenhum vértice fora da componente conexa. Um grafo é conexo se o mesmo é uma componente conexa.

Um conjunto independente em um grafo G é um subconjunto $I \subseteq V(G)$ tal que o grafo induzido pelos vértices em I não possui arestas, ou seja, $\forall u, v \in I$ temos que $(u, v) \neq E(G)$. De forma oposta, uma clique em um grafo G é um subconjunto de vértices $K \subseteq V(G)$ tal que o grafo induzido pelos vértices em K é completo, ou seja, $\forall u, v \in K$ com $u \neq v$ temos que $(u, v) \in E(G)$. Denota-se por K_t uma clique de tamanho t, por exemplo, K_4 é uma clique de tamanho 4 [Bondy et al., 1976].

Um conceito importantíssimo para este trabalho é o de **contração de arestas**. Contrair uma aresta significa removê-la do grafo e fazer com que os vértices em suas extremidades se tornem um só vértice que tem todas as arestas incidentes aos dois vértices originais. Dito de outra forma, dado um grafo $G = (V, E)$ e uma aresta $e = (u, v) \in E$, contrair e em G gera o grafo $\boldsymbol{G/e} = (V/e, E/e)$ construído da seguinte forma:

1. Seja um vértice $w \notin V$ e seja $E_{uv} \subseteq E$ o conjunto de arestas em que pelo menos uma de suas extremidades é u ou v.

2. Construímos um novo conjunto de arestas E_w tal que $\forall(u,x) \in E_{uv}$ com $x \neq v$ então $(w,x) \in E_w$ e $\forall(v,x) \in E_{uv}$ com $x \neq u$ então $(w,x) \in E_w$. Note que assim, o número de arestas diminui, pois pelo menos uma aresta não gera uma nova aresta em E_w, logo $|E_w| < |E_{uv}|$.

3. $G/e = (V/e, E/e)$ é construído de forma que $V/e = (V - \{u,v\}) \cup \{w\}$ e $E/e = (E - E_{uv}) \cup E_w$ [Wolle & Bodlaender, 2004]. Assim, temos que $|V/e| = |V| - 2 + 1 = |V| - 1$ e $|E/e| < |E|$, ou seja, ao contrair uma aresta, o número de vértices diminui em uma unidade e o número de arestas diminui em pelo menos uma unidade.

Assim, dizemos que $\boldsymbol{G/e}$ é o grafo resultante da contração da aresta e em G. Se Y é um conjunto de arestas, então $\boldsymbol{G/Y}$ é o grafo resultante da contração de todas as arestas de Y em G. É importante frisar que a ordem em que as arestas de Y são contraídas não é relevante. Por fim, se X é um conjunto de vértices, então $\boldsymbol{G/X}$ é o grafo resultante da contração de todas as arestas de $E(X)$ em G. É importante frisar também que a contração de uma aresta $e = (u,v)$ pode ser feita em tempo polinomial, pois basicamente precisamos apenas conhecer os vizinhos de u e v, no pior caso, com complexidade $O(n+m)$. Assim, contrair todas as arestas de um grafo tem complexidade no máximo $m \cdot O(n+m) = O(n+m^2)$, polinomial.

Algoritmos parametrizados são usados para resolver problemas NP-Completos mais eficientemente do que pela força bruta simples, uma vez que a exponencialidade é em função de um parâmetro definido anteriormente k, menor que o tamanho da entrada. Algoritmos com complexidade de tempo da forma $f(k) \cdot n^c$, onde c é uma constante e f uma função computável, são chamados algoritmos de *Fixed-Parameter Tractable* (FPT), ou simplesmente, parâmetro fixo tratável. Dizemos que esses algoritmos são executados em tempo FPT. Como c é constante, a parte que é uma função de n é polinomial e a exponencialidade depende de $f(k)$. Um problema é dito FPT se existe um algoritmo FPT que o resolva exatamente. FPT é a classe de complexidade dos problemas FPT e XP é uma classe de problemas para os quais um algoritmo com complexidade $f(k) \cdot n^{g(k)}$ é conhecido, com f e g sendo funções computáveis [Cygan et al., 2015].

Classes de complexidade parametrizadas: Como nas classes de problema NP-completas, há classes de complexidade parametrizada que são divididas em uma hierarquia W: $FPT = W[0] \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq W[P] \subseteq XP$. Similarmente às reduções de NP-completude, fazer uma redução FPT de um problema P para um problema P' significa reduzir qualquer instância (n,k) de P em uma instância (n',k') de P' em tempo FPT, porém com algo a mais, que é garantir que $k' \leq g(k)$ para alguma

função computável g. Por fim, temos que garantir também que P' tenha resposta SIM se, e somente se, P tiver responda SIM [Cygan et al., 2015; Downey & Fellows, 2013].

Algoritmos de aproximação ou **algoritmos aproximativos** são sempre executados em tempo polinomial e geram uma solução S, que pode não ser exatamente a solução ótima S^*, mas podemos garantir que o valor de S está no máximo a um fator de aproximação α longe do valor da solução ótima. Isto é, S^* é no máximo α vezes melhor que S, onde α pode ser uma constante ou uma função do tamanho da entrada.

O último conceito importante para este trabalho é o de **Classes de grafos**. Uma classe de grafo pode ser definida como um um conjunto de grafos que satisfazem uma propriedade específica daquela classe [dos Santos, 2013]. Por exemplo, a classe dos grafos cúbicos possui os grafos que todos os vértices possuem grau exatamente igual a 3. Algumas classes de grafos utilizadas neste trabalho são definidas a seguir:

- **Árvores:** uma árvore é um grafo conexo sem ciclos. Assim sendo, há um e somente um caminho entre cada par de vértices de uma árvore. Outra propriedade é que o número de arestas de uma árvore é exatamente o número de vértices menos um [Bondy et al., 1976]. Dois exemplos de árvores são mostrados na Figura 2.1.

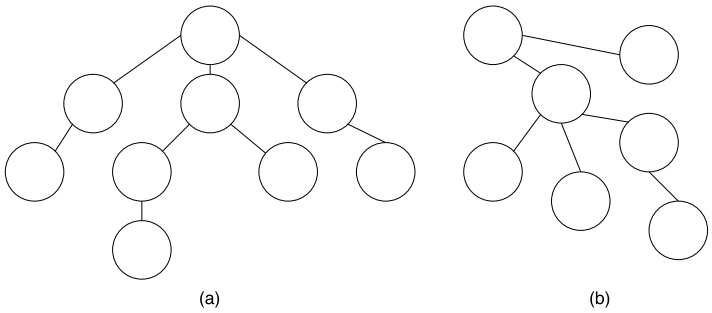

Figura 2.1: Exemplos de árvores.

- **Blocos:** uma **componente biconexa** é uma componente conexa na qual todos os pares de vértices estão conectados por pelo menos dois caminhos de vértices distintos. Um **grafo bloco** é um grafo na qual todas as componentes biconexas são cliques.

- **Bipartidos:** um grafo G é dito bipartido se é possível dividir seus vértices em uma partição com duas partes tal que não haja arestas entre vértices de uma mesma parte da partição, ou seja, é possível dividir os vértices em dois subconjuntos $V_1, V_2 \in V(G)$ em que $V_1 \cup V_2 = V(G)$ e $V_1 \cap V_2 = \emptyset$ onde $\forall u, v \in V_i$ com $i \in \{1, 2\}$ temos que $(u, v) \notin E(G)$. Seja $|V_1| = x$ e $|V_2| = y$, denota-se por $K_{x,y}$ o grafo bipartido em que há arestas de cada vértice de uma parte da partição

para todos os vértices da outra parte da partição [Bondy et al., 1976]. Um Teorema muito útil provado em Bondy et al. [1976] é que um grafo é bipartido se e somente se não tem ciclos de tamanho ímpar. A Figura 2.2 mostra exemplos de grafos bipartidos, onde o primeiro exemplo é um grafo $K_{3,3}$ na qual podemos ver facilmente a partição do vértices e no segundo exemplo, apesar de não ser tão fácil ver a partição, é fácil ver que não há ciclos de tamanho ímpar, logo é um grafo bipartido.

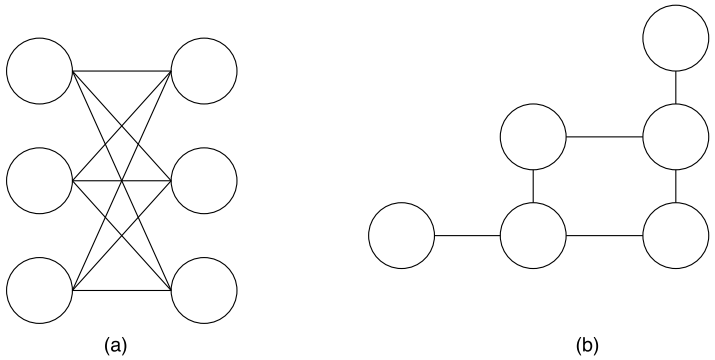

Figura 2.2: Exemplos de grafos bipartidos.

- **Split:** um grafo *split* G tem a propriedade que seus vértices podem ser particionados em dois subconjuntos $K, I \subseteq V(G)$, $K \cup I = V(G)$ e $K \cap I = \emptyset$, onde K é uma clique e I é um conjunto independente. Um exemplo de grafo *split* é apresentado na Figura 2.3.

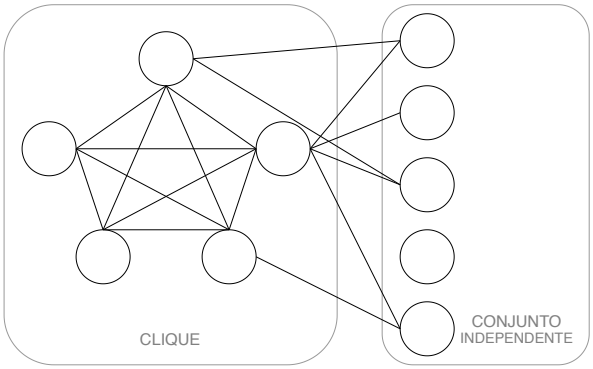

Figura 2.3: Exemplo de grafo split.

- **Cordais:** um grafo G é dito cordal se cada ciclo em G com mais de três vértices possui pelo menos uma corda. Isto é, se G é cordal, para cada ciclo C em G, seja n_c o número de vértices de C, se $n_c > 3$ então o número de arestas do grafo induzido pelo vértices de C é maior que n_c. Um detalhe importante a se notar é que todo grafo *split* é cordal, que implica dizer que a classe dos grafos *split* é

uma subclasse da classe dos grafos cordais [Brandstadt et al., 1999]. A Figura 2.4 mostra um exemplo de grafo cordal.

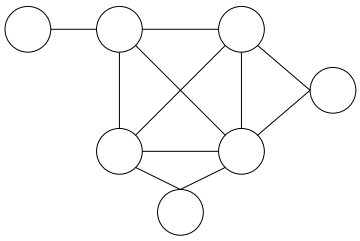

Figura 2.4: Exemplo de grafo cordal.

- **Planares:** um grafo é dito planar se é possível desenhar o mesmo em um plano de tal forma que nenhum par de arestas se cruze [Bondy et al., 1976]. Uma propriedade importante mostrada em Bondy et al. [1976] é que um grafo G é planar se e somente se não contém K_5 ou $K_{3,3}$ como grafo minor. Um grafo minor é um grafo gerado a partir do original apenas deletando vértices e arestas e contraindo arestas. Dois exemplos de grafos planares são mostrados na Figura 2.5: o primeiro mostra um grafo K_4 desenhado de forma planar e o segundo mostra um grafo correspondente a um cubo, também desenhado de forma planar.

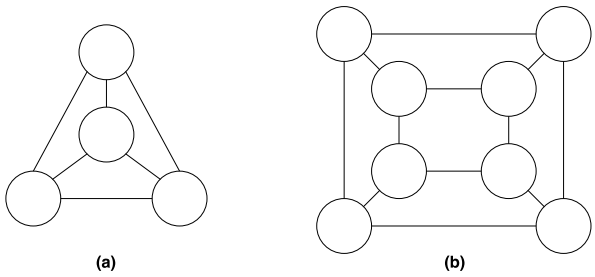

Figura 2.5: Exemplos de grafos planares.

- ***Outerplanares*:** um grafo é dito *outerplanar* ou **periplanar** se ele é planar e todos os vértices fazem parte da face externa do plano. Em outras palavras, nenhum vértice está "preso" por outros vértices e arestas [Bondy et al., 1976]. Uma propriedade importante mostrada em Bondy et al. [1976] é que um grafo G é *outerplanar* se e somente se não contém K_4 ou $K_{2,3}$ como grafo minor.

- **UDGs:** um *Unit Disk Graph* (UDG) é um grafo em que os vértices podem ser representados por discos (círculos) em um plano, tal que todos os discos tem o mesmo raio e há arestas entre dois vértices se há interseção entre os discos que representam tais vértices (assume-se que discos tangentes têm interseção) [Brandstadt et al., 1999; Bondy et al., 1976]. A Figura 2.6 mostra um exemplo de

como é representado um UDG (à esquerda) e de sua representação convencional (à direita).

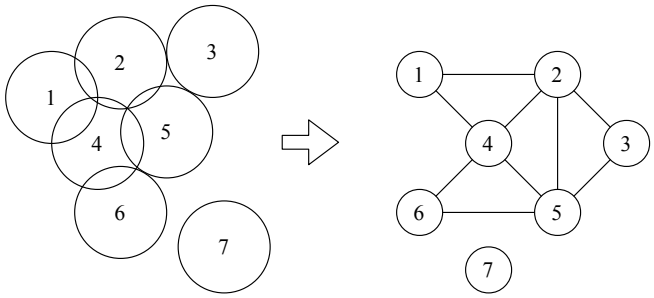

Figura 2.6: Exemplo de UDG.

- **Grades e grades parciais:** uma grade de dimensão $n \times m$ é um produto cartesiano de um caminho de tamanho n e um caminho de tamanho m. A Figura 2.7 (A) mostra uma grade 3×5. Uma grade parcial é um subgrafo de uma grade. Note que uma grade parcial pode ou não ser um subgrafo induzido de uma grade, como exemplificado na Figura 2.7 (B).

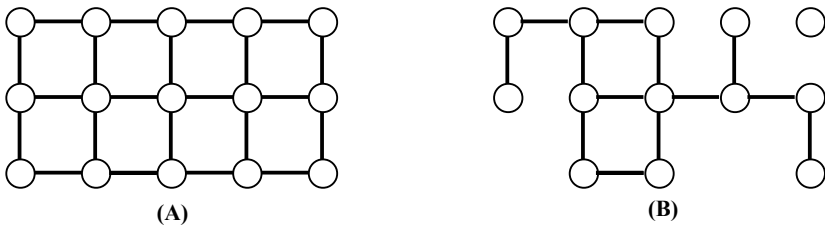

Figura 2.7: Exemplos de grade e grade parcial.

- **Cactos:** um grafo é um cacto se cada aresta faz parte de no máximo um ciclo simples. Em outras palavras, dois ciclos simples de um cacto não tem mais de um vértice em comum.

- **Cografos:** um cografo é um grafo definido recursivamente da seguinte forma:
  - Um grafo com um vértice e sem arestas é um cografo.
  - Se um conjunto de grafos disjuntos $\{G_1, G_2, \ldots, G_r\}$ são cografos, então a união desses grafos $G_1 \cup G_2 \cup \ldots \cup G_r$ é um cografo. Isto é, um novo grafo com os mesmos vértices e arestas do conjunto de grafos original.
  - O complemento de um cografo G também é um cografo. O complemento $\overline{G}$ de um grafo G é dado por $V(\overline{G}) = V(G)$ e para cada par de vértices distintos $u, v \in V(G)$ teremos que $(u, v) \in E(\overline{G})$ se e somente se $(u, v) \notin E(G)$ [Corneil et al., 1981].

Um cografo também pode ser definido por operações de união e de junção: um vértice isolado é um cografo, a união de dois cografos é um cografo (mesmos vértices e arestas) e a junção de dois cografos é um cografo. Nesse caso, a operação de junção de dois grafos G_1 e G_2 é colocar no novo cografo G_3, todos os vértices e arestas de G_1 e G_2 e $\forall v \in V(G_1) \forall u \in V(G_2)$ inserimos a aresta (u, v) em G_3. Uma ***cotree*** é uma árvore que representa a construção de um cografo. Nós folhas da *cotree* são os nós do cografo, nós não folha representam qual operação está sendo aplicada em seus filhos, se é uma junção ou uma união, e representa o cografo gerado por aquela operação [Corneil et al., 1985].

Capítulo 3

Trabalhos Relacionados

O problema CCV tem semelhança com alguns problemas clássicos de cobertura, como *Vertex Cover* (VC), *Set Cover* (SC) e *Hitting Set*. Outro problema relacionado ao CCV é o *Red-Blue Dominating Set* (RBDS) [Downey et al., 1999], no qual os vértices do grafo são particionados em vermelhos e azuis, e queremos cobrir todos os vértices vermelhos com um conjunto mínimo de vértices azuis. Nas subseções seguintes, serão detalhadas as semelhanças e diferenças do CCV com alguns desses trabalhos e problemas.

Red-Blue Dominating Set: RBDS e CCV não são iguais, mas o primeiro é um caso especial de CCV, onde A e B são conjuntos independentes. Por outro lado, dada uma instância de CCV, é possível obter uma instância equivalente para o *Red-Blue Dominating Set*, excluindo as arestas entre vértices de B (que são irrelevantes para o problema) e contraindo arestas entre vértices de A, como mostraremos no Lema 1. Essas observações implicam que os resultados negativos para o RBDS aplicam-se automaticamente ao CCV, por exemplo, NP-completude em grafos planares [Alber et al., 2000]. Por outro lado, se um resultado negativo for válido para CCV em uma classe de grafo fechada para exclusões e contrações de arestas, ele também será válido para o RBDS. Além disso, os resultados positivos para o CCV também são válidos para o RBDS. De uma perspectiva de complexidade parametrizada, RBDS é $W[2]$-Completo em geral, implicando a $W[2]$-completude para CCV.

Infraestrutura de Acesso de Redes Sem Fio Obstruídas: Em Ferreira Neto et al. [2017], foi formulado e estudado um caso especial do problema CCV, chamado de *Obstructed Wireless Network* (OWN) *Backbone Cover problem* (OWN-BC), ou em tradução oficial, problema da Infraestrutura de Acesso de Rede Sem Fio Obstruída. A ênfase estava nas propriedades de conectividade da rede, como a faixa crítica de transmissão para conectividade e a aplicação no domínio de redes sem fio obstruídas.

Ferreira Neto et al. [2017] definem o problema OWN-BC da seguinte forma: dada

uma malha quadrada regular, ou seja, um plano quadrado com linhas e colunas de blocos quadrados separados por retas — como em uma cidade, onde os blocos são os quarteirões e as retas são as ruas —, e dispositivos distribuídos ao longo dessas retas, deseja-se saber qual o número mínimo de pontos de acesso é necessário para conectar todos os dispositivos e onde colocar esses pontos de acesso. Dispositivos só podem se comunicar se estiverem conectados a pontos de acesso ou a outros dispositivos que já estejam conectados. Deve-se levar em consideração o alcance dos dispositivos, e se é possível conectá-los ou não — só é possível conectar dois dispositivos, ou um dispositivo com um ponto de acesso se não houver blocos entre eles. Por simplicidade, Ferreira Neto et al. [2017] supõem que os dispositivos sempre estão posicionados no meio das ruas.

Mais formalmente, OWN-BC é modelada em um *grid*. Uma instância do problema tem quatro parâmetros: g, ϵ, r e $\mathcal{D}$. O parâmetro g é a dimensão do *grid*, que assim, tem g linhas e g colunas, que representam as ruas. Cada rua tem largura 2ϵ (sendo $0 < 2\epsilon < 1$). A largura da rua é limitada por 1, pois a distância normalizada do centro de duas ruas paralelas é igual a 1. Os dispositivos têm alcance de comunicação igual a r, logo, um dispositivo só pode se comunicar com outro ou com um ponto de acesso se os dois estiverem a uma distância de no máximo r e tiverem linha de visão desobstruída. Por fim, $\mathcal{D}$ representa o conjunto de dispositivos, e cada $d_i \in \mathcal{D}$ tem sua posição (x_i, y_i) no *grid*. A Figura 3.1 do trabalho de Ferreira Neto et al. [2017] mostra um exemplo do modelo de OWN-BC.

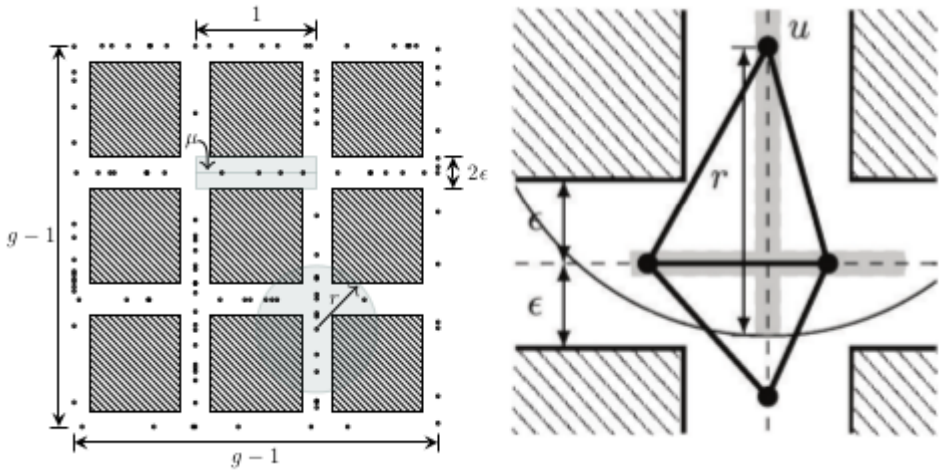

Figura 3.1: Modelo OWN-BC.

Definida a instância do problema, definimos agora a solução como sendo um conjunto de pontos de acesso P tal que todo $p_j \in P$ tem sua posição (x_j, y_j) no *grid* e $\forall d_i \in \mathcal{D}$, ou d_i está conectado diretamente a algum $p_k \in P$ — ou seja, a distância de

d_i a $p_k \leq r$ e não há obstrução de blocos entre os mesmos —, ou d_i está conectado de alguma forma — diretamente ou através de outros dispositivos — a algum $d_l \in \mathcal{D}$ tal que d_l está conectado diretamente a algum $p_k \in P$, e além disso $|P|$ é mínimo.

O problema OWN-BC pode ser modelado usando grafos. Utilizar uma modelagem em grafos, nos permite abstrair conceitos técnicos do problema como a obstrução entre os dispositivos, o raio de comunicação dos mesmos, a disposição dos dispositivos e pontos de acesso no *grid*, entre outros fatores e essa abstração pode facilitar a busca por algoritmos eficientes. Por outro lado, a modelagem é feita de forma a preservar as características da modelagem geométrica do *grid*, mantendo as restrições do problema, o que nos permite trabalhar na modelagem em grafos de forma que resolver o problema em grafos é equivalente a resolver o problema no *grid*. Para isso, dada uma instância $\{\mathcal{D}, g, \epsilon, r\}$ de OWN-BC criamos um grafo $G = (V, E)$ seguindo os seguintes passos:

1. Criamos o grafo $G = (V, E)$ inicialmente vazio. O conjunto de vértices do grafo será construído em uma partição com duas partes disjuntas, $A(G)$ e $B(G)$ — $A(G) \cup B(G) = V(G)$ e $A(G) \cap B(G) = \emptyset$. O conjunto de vértices $A(G)$ representarão os dispositivos e o conjunto de vértices $B(G)$ representarão os pontos de acesso candidatos a estarem na solução.

2. Para cada dispositivo $d_i \in \mathcal{D}$, criamos um vértice a_i correspondente a d_i que é inserido em $A(G)$. Após isso, para todo par dispositivos $d_i, d_j \in \mathcal{D}, d_i \neq d_j$ que conseguem se conectar — distância de d_i e $d_j \leq r$, e não há obstrução entre eles — insere-se uma aresta entre os vértices a_i e a_j correspondentes a esses dispositivos. Vale frisar que os dispositivos são sempre conectados aos dispositivos mais próximos, assim, um dispositivo está ligado diretamente a no máximo 4 outros dispositivos — isso é possível quando o mesmo se encontra em uma interseção. Assim, se o grafo gerado for conexo significa que todos os dispositivos podem se conectar, caso contrário, é preciso conectar as componentes conexas do grafo com os pontos de acesso, e passamos ao passo 3.

3. Ferreira Neto et al. [2017] mostram um algoritmo com o qual é possível calcular quais dispositivos podem se comunicar usando um mesmo ponto de acesso — distância entre os dois dispositivos é menor ou igual a $2r$ e não há obstrução entre os mesmos. Ferreira Neto et al. [2017] também mostra que no melhor caso, no máximo, apenas quatro dispositivos satisfazem essas condições para um único ponto de acesso — ou outro dispositivo —, que seria um caso onde esses dispositivos estão próximos a uma interseção no *grid*. Com isso, há três possibilidades de conexões entre as componentes:

a) Se nenhum dispositivo de uma componente C_θ de $A(G)$ consegue se conectar a um dispositivo de outra componente, quer dizer que C_θ é uma componente isolada, logo é necessário um ponto de acesso p_θ para aquela componente. Criamos um vértice b_θ correspondente a p_θ, o inserimos em $B(G)$ e inserimos a aresta (a_θ, b_θ) no grafo G, sendo a_θ um vértice de $A(G)$ correspondente a um dispositivo em C_θ.

b) Em cada segmento, duas componentes $C_j, C_k \in A(G)$ podem ter dispositivos com distância $\leq 2r$ e sem obstrução, logo, inserimos um ponto de acesso p_i candidato a estar na solução e que cobre essas duas componentes. Assim, seja $a_j \in C_j$ o vértice de C_j correspondente ao dispositivo mais próximo de C_k e $a_k \in C_k$ o vértice de C_k correspondente ao dispositivo mais próximo de C_j, criamos um vértice b_i correspondente a p_i que é inserido em $B(G)$ e inserimos as arestas (a_j, b_i) e (a_k, b_i) no grafo G.

c) Por fim, no caso das interseções, até quatro componentes de $A(G)$ podem ter dispositivos que podem se conectar com um único ponto de acesso. Assim, fazemos algo semelhante ao passo anterior: obtemos o conjunto de vértices X correspondentes aos dispositivos dessas componentes mais próximos das outras componentes, criamos um vértice b_i correspondente ao ponto de acesso candidato a cobrir essas componentes, inserimos b_i em $B(G)$ e $\forall a_j \in X$ inserimos a aresta (a_j, b_i) no grafo G. Como no máximo quatro componentes podem se conectar utilizando um único ponto de acesso, a exemplo do que acontece nos vértices de $A(G)$, o grau máximo entre os vértices de $B(G)$ é ≤ 4, consequentemente o grau máximo de G é ≤ 4.

4. Por fim, o problema será encontrar o subconjunto B' dos vértices de $B(G)$ tal que cada componente conexa de $A(G)$, esteja conectada a pelo menos um vértice de B', ou seja, todo conjunto isolado de dispositivos esteja conectado a pelo menos um ponto de acesso, e ainda, queremos minimizar o tamanho de B'.

Em seu trabalho, Ferreira Neto et al. [2017] provaram que essa versão geométrica do problema, onde os nós de comunicação são implementados em uma estrutura de grade regular e conectividade é determinada por critérios de proximidade e visibilidade, é NP completa, por uma redução do problema *Exact Cover by 3-Sets* (X3C) [Garey & Johnson, 2002]. Versões determinísticas e probabilísticas do problema OWN-BC foram estudadas. Um algoritmo 2-aproximativo foi proposto para o problema determinístico. Finalmente, um limite inferior na probabilidade de obter soluções ótimas (usando o último algoritmo) foi calculado como uma função das propriedades geométricas de

topologias de rede aleatórias. Em Ferreira Neto et al. [2017] foi mostrado que CCV é NP-Completo mesmo para grafos com grau de vértices ≤ 4.

Unweighted *d*-Set Cover Problem: O trabalho de Levin [2008] é baseado no problema *d-Set Cover* sem custos. Levin [2008] define o problema *Weighted Set Cover* da seguinte forma: dado um conjunto de elementos X e uma família $\mathcal{Y}$ de subconjuntos de X, tal que cada $Y_i \in \mathcal{Y}$ tem um custo c_i, deseja-se cobrir todos os elementos de X com conjuntos de $\mathcal{Y}$ de forma a minimizar o custo da cobertura, isto é, deseja-se encontrar um subconjunto $\mathcal{Y}' \subseteq \mathcal{Y}$ tal que todo elemento de X esteja em algum conjunto de $\mathcal{Y}'$, e queremos minimizar a soma dos custos dos conjuntos em $\mathcal{Y}'$. Assim, no problema *Unweighted Set Cover*, não há custos nos conjuntos de $\mathcal{Y}$, e queremos minimizar o tamanho de $\mathcal{Y}'$. Por fim, Levin [2008] define o problema *Unweighted d-Set Cover* como o problema *Unweighted Set Cover* com a restrição de que todos os conjuntos em $\mathcal{Y}$ tem tamanho no máximo d.

Uma pesquisa sobre algoritmos aproximativos para o problema *d-Set Cover* foi apresentada em Levin [2008]. Começando com o algoritmo guloso clássico, cujo fator de aproximação é H_d (o d-ésimo número harmônico) [Chvátal, 1979], os autores discutem algoritmos relacionados, que realizam otimizações e alcançam melhores fatores de aproximação. Em particular, eles discutem o algoritmo proposto em Duh & Fürer [1997], que obtém um fator de aproximação de $H_d - \frac{1}{2}$, fazendo algumas otimizações locais dentro do algoritmo guloso. Finalmente, os autores apresentam um novo algoritmo, baseado no algoritmo proposto em Duh & Fürer [1997], com o fator de aproximação $H_d - \frac{196}{390}$.

Neste trabalho, mostraremos que o problema a ser estudado aqui, o *Component Cover by Vertices* tem uma relação muito particular com o problema *Unweighted d-Set Cover*. Assim, o trabalho de Levin [2008] é relevante, uma vez que mostra o algoritmo com melhor fator de aproximação para *Unweighted d-Set Cover* e utilizaremos tal algoritmo para obter um fator de aproximação de $1,79$ para CCV em UDGs.

Aproximação $O(\log n)$ para Set Cover: Em seu trabalho, Feige [1998] também estuda o problema *Set Cover*. Usando a mesma definição anterior de Levin [2008], o trabalho de Feige [1998] é focado no problema sem custos nos conjuntos de $\mathcal{Y}$, isto é, o problema *Unweighted Set Cover* e sem restrições aos tamanhos de conjuntos em $\mathcal{Y}$.

Como mostrado por Levin [2008], se o tamanho dos conjuntos em $\mathcal{Y}$ é limitado por um inteiro d — problema do *Unweighted d-Set Cover* —, conseguimos um algoritmo aproximativo com fator de aproximação constante igual a $H_d - \frac{196}{390}$. No entanto, se não se restringe o tamanho dos conjuntos de $\mathcal{Y}$, o melhor algoritmo conhecido, que pode ser visto em Vazirani [2013], tem fator de aproximação $O(\log n)$, sendo n o número de elementos em X. Como esse fator de aproximação depende do tamanho da entrada,

seria interessante encontrar um algoritmo mais eficiente com fator de aproximação constante. No entanto, Feige [1998] mostra que o problema *Unweighted Set Cover* é inaproximável a um fator de aproximação melhor que $O(\log n)$, a menos que P = NP.

Para o nosso trabalho, o trabalho de Feige [1998] é relevante porque, a partir do resultado de inaproximabilidade mostrado por ele, podemos mostrar um resultado de inaproximabilidade também para o problema aqui estudado, o *Component Cover by Vertices*.

Capítulo 4

Definição do Problema

Partindo da modelagem em grafos do problema OWN-BC [Ferreira Neto et al., 2017], temos o problema que chamamos aqui de *Component Cover by Vertices* — em tradução oficial, **Cobertura de Componentes por Vértices** —, ou simplesmente CCV.

Formalmente, a versão de decisão do problema é definida da seguinte forma: dados um inteiro k e um grafo $G = (V, E)$, em que $V(G)$ é dividido em uma partição $A(G), B(G) \subseteq V(G)$ com $A(G) \cap B(G) = \emptyset$ e $A(G) \cup B(G) = V(G)$, deseja-se saber se existe um subconjunto de vértices $B' \subseteq B(G)$ tal que $|B'| \leq k$ e para todo $a_i \in A(G)$ existe caminho passando somente por vértices de $A(G)$ a partir a_i até algum vértice de B'. Em outras palavras, para toda componente conexa $C \in G[A(G)]$ existe $c_i \in V(C)$ e um $b_j \in B'$ tal que $(c_i, b_j) \in E(G)$. Em sua versão de otimização, queremos minimizar o tamanho de B'.

A partir dessa definição, podemos pensar em variações para o problema CCV, por exemplo, uma versão em que os vértices da solução B' precisam necessariamente estar conectados. Uma outra versão do problema que pode ser relevante é imaginar que um vértice de $A(G)$ só pode se conectar a algum vértice de $B(G)$ se estiver a uma distância restrita, ou seja, pensar em uma versão com restrição de diâmetro. Tais versões não são abordadas neste trabalho e poderão ser estudadas em trabalhos futuros.

Vale frisar que o grafo gerado pela instância de OWN-BC tem grau máximo igual a quatro [Ferreira Neto et al., 2017], uma classe de grafos muito restrita. No nosso trabalho, focamos no estudo do problema para grafos gerais, o que torna o problema mais difícil. Assim, tivemos como desafios encontrar algoritmos polinomiais que resolvem o problema para distintas classes de grafos e mostrar a NP-Completude do problema para outras classes. Também foi nosso objetivo encontrar algoritmos eficientes para essas classes de grafos em que o problema é NP-Completo.

O problema CCV tem grande similaridade com alguns problemas NP-Completos

clássicos da literatura como por exemplo o *Vertex Cover*. No problema *Vertex Cover* queremos cobrir todas as arestas de um grafo G com os vértices do mesmo, assim, podemos imaginar uma transformação de G em que suas arestas se transformam em componentes que queremos cobrir e os vértices de G se tornam vértices com as quais cobriremos as componentes, e com isso, se resolvermos o problema CCV resolveremos o problema *Vertex Cover*. Outros problemas de cobertura podem ser reduzidos a CCV como o Conjunto Dominante, o *Set Cover* e o *Red-Blue Dominating Set*.

O *Set Cover* e o *Red-Blue Dominating Set* são equivalentes entre si, pois podemos reduzir um problema ao outro. Por exemplo, se X é o conjunto de elementos que se quer cobrir no SC, podemos pensar em uma redução em que elementos de X são vértices vermelhos e os subconjuntos de X que podem ser usados para cobrir são vértices azuis. Além disso, há arestas de um vértice azul para um vermelho se o elemento representado pelo vértice vermelho pertencia ao subconjunto representado pelo vértice azul. Assim, se uma solução cobre todos os elementos de X, se selecionarmos os vértices azuis que representam os conjuntos dessa solução, ele será uma solução para RBDS. É fácil ver que a transformação de uma solução de RBDS para uma solução de SC também é válida, utilizando a mesma estratégia. Também é fácil ver que podemos transformar uma instância de RBDS para SC fazendo a mesma coisa: vértices vermelhos se transformam em elementos, vértices azuis se transformam em subconjuntos, que têm os elementos que representam seus vértices vermelhos vizinhos.

O *Red-Blue Dominating Set* é também equivalente ao CCV em grafos bipartidos com partição (A, B), como mostraremos no Teorema 3, onde é feita uma redução a partir de um caso particular do *Set Cover* e o grafo resultante é equivalente a uma instância de RBDS. Também mostraremos que, com o Lema 1, podemos reduzir CCV a RBDS. Portanto, o problema aqui estudado tem uma forte conexão com o *Set Cover*, uma vez que os três problemas são equivalentes. Ainda assim, podemos utilizar a estrutura do grafo de entrada para ajudar a resolver o problema de maneira mais eficiente, se beneficiando de determinadas especificidades que o grafo pode ter. Por exemplo, grafos de determinadas classes podem nos dar informações úteis para que possamos encontrar algoritmos polinomiais para o problema aplicado a grafos dessa classe. Ademais, se uma determinada classe não é fechada em relação as operações de remoção ou contração de arestas (isso quer dizer, após uma dessas operações, o grafo resultante não necessariamente continuará pertencendo àquela classe), um resultado obtido para o CCV não necessariamente se aplicará a RBDS ou SC.

Podemos também, interpretar a estrutura de um grafo pertencente a uma determinada classe como impondo restrições nas instâncias de *Set Cover*. Por exemplo, se o problema é aplicado a grafos com grau máximo 3, isso quer dizer que o tamanho dos

conjuntos da instância de SC tem tamanho no máximo 3. Outra observação interessante é uma abordagem diferente, sugerida por um dos membros da banca avaliadora deste trabalho, que utilizaria uma interpretação desta relação com *Set Cover* através de duas etapas:

1. a caracterização de qual tipo de instância de grafo bipartido poderia ser obtida se aplicássemos a grafos de uma determinada classe as regras de redução de CCV para RBDS;

2. a determinação da complexidade para subclasses de grafos bipartidos.

Capítulo 5

NP-Completude

Neste Capítulo, mostram-se os resultados obtidos sobre a complexidade do problema CCV, provando sua NP-Completude não só para grafos gerais, mas também para classes de grafos específicas, tais como bipartidos com grau máximo igual a três, *splits*, cordais, UDGs planares, grades e grades parciais.

Primeiramente mostramos o Lema 1 que é um resultado importante sobre contração de arestas em grafos de entrada do problema CCV. Esse Lema será utilizado para muitas outras provas deste Trabalho.

Lema 1 *Dada uma instância* $I = \{G, A \text{ e } B, k\}$ *de CCV e uma instância* $I' = \{G/A, k\}$ *sendo que* G/A *é o grafo gerado pelas contração das arestas em* $E(A(G))$ *no grafo* G, B' *é uma solução válida do problema para* I' *se e somente se* B' *é também uma solução válida do problema para* I.

Demonstração O subgrafo induzido $G_A = G[A(G)]$ forma várias componentes conexas, as quais queremos conectar com vértices de $B(G)$. Ao contrair todas as arestas de $E(G_A)$ temos um grafo apenas com vértices isolados, que juntos aos vértices de $B(G)$ mais as arestas não contraídas formam o grafo G/A onde não há aresta entre nenhum par de vértices de $A(G/A)$. Se B' é uma solução válida para a entrada I', para que todos os vértices de $A(G/A)$ tenham caminho até algum vértice de B' passando somente por vértices de $A(G/A)$ então é necessário que todo vértice de $A(G/A)$ tenha aresta para algum vértice de B', pois não havendo arestas entre os vértices de $A(G/A)$ o único modo desses vértices se conectarem é diretamente por uma aresta até B'. Seja v um vértice qualquer em $A(G/A)$. Por construção, v representa uma componente conexa em G_A. Como há uma aresta entre v e algum vértice $b_i \in B'$, qualquer vértice que estivesse na componente conexa representada por v teria um caminho até b_i passando somente por vértices de $A(G)$, ou seja, a componente toda estaria coberta por b_i. Como

essa propriedade é válida não só para v mas para qualquer vértice de $A(G/A)$ então todas as componentes conexas originais são cobertas por B', assim sendo, B' também é uma solução válida para a instância I. Por outro lado, se B' é uma solução para a instância I então todas as componentes de G_A têm um vértice vizinho de algum vértice de B'. Seja C uma componente qualquer de G_A. Por construção, há um vértice v que representa C em G/A. Após a contração de arestas, todos os vizinhos de qualquer vértice em C serão vizinhos de v, logo, o vértice de B' que cobre C, é vizinho de v e o cobre. Como essa propriedade é válida para todas as componentes de G_A então todas as novas componentes (que são vértices únicos) em G/A serão cobertas. Logo, B' também é uma solução válida para a instância I. ■

Uma observação importante, também muito utilizada nas provas de NP-Completude e nos algoritmos aqui encontrados, é que dada uma instância de CCV $\{G, A$ e $B, k)$, adicionar ou remover arestas entre os vértices de B não altera o resultado da solução. Isso é claro, uma vez que não estaremos inserindo nenhuma aresta de um vértice de A para um vértice de B, portanto, não criando novas soluções. Se contrairmos todas as arestas entre os vértices de A e removermos todas as arestas entre os vértices de B, teremos uma instância de *Red-Blue Dominating Set*. Chamaremos esse procedimento de **simplificação** do grafo.

A partir do Lema 1 e dessa observação, conseguiremos fazer as provas de NP-Completude do problema CCV. Inicialmente, precisamos provar que o problema é NP e depois provar que é NP-Completo. Uma maneira de provar que um problema pertence a classe NP é mostrar que ele é um problema de decisão e que existe um algoritmo polinomial que verifica se uma solução dada é válida [Cormen et al., 2009], e isso é mostrado no seguinte Lema:

Lema 2 *CCV é NP.*

Demonstração A questão da versão de decisão do problema é: dados um inteiro k, um grafo $G = (V, E)$ e uma partição $A(G), B(G)$ de $V(G)$, existe um subconjunto de vértices $B' \subseteq B(G)$ tal que $|B'| \leq k$ e para todo $a_i \in A(G)$ existe caminho passando somente por vértices de $A(G)$ a partir de a_i até algum vértice de B'? O Algoritmo 1 verifica uma solução em tempo polinomial. O algoritmo usa a função *DFS*, que é uma busca em profundidade, clássica em teoria dos grafos e é explicada com mais detalhes em Cormen et al. [2009]. Uma observação importante é que recursivamente essa busca em profundidade só verificará vértices de $A(G)$. A busca em profundidade tem complexidade $O(n + m)$. Além da busca, o algoritmo verifica se todos os vértices de $A(G)$ foram alcançados através dos vértices de B', passando por todos os vértices

de $A(G)$, complexidade $O(n)$, logo a complexidade total do algoritmo é $O(n + m)$, polinomial.

Algorithm 1 VERIFICA-CCV

Entrada G, A e B, k, B', sendo G uma instância válida de CCV
Saída SIM se B' for uma solução válida e NÃO, caso o contrário

```
1: se |B'| > k então
2:     retorne NÃO
3: para cada b ∈ B' faça
4:     se cor[b] = BRANCO então
5:         DFS(G, b)
6: se ∀a ∈ A(G), cor[a] = PRETO então
7:     retorne SIM
8: retorne NÃO
```

■

Após provarmos que CCV é NP, precisamos provar sua NP-Completude. Para provar que um problema P é NP-Completo devemos mostrar que P está em NP e que existe um problema P' conhecidamente NP-Completo que pode ser reduzido polinomialmente a P. Reduzir P' polinomialmente a P significa que através de uma instância genérica de P', aplicando operações em tempo polinomial geramos uma instância de P e através da saída de P geramos a saída de P' também em tempo polinomial, e a resposta será SIM para P' se e somente se a resposta for SIM para P. Pelo Lema 2 temos que CCV é NP. Por fim, reduziremos um problema NP-Completo a ele.

Utilizaremos como base para nossa redução o problema *Set Cover* — em tradução livre, Cobertura de Conjuntos —, ou simplesmente SC. Esse problema recebe como entrada um inteiro k, um conjunto de elementos $X = \{x_1, x_2, ..., x_{|X|}\}$ e uma família de conjuntos $\mathcal{Y} = \{Y_1, Y_2, ..., Y_{|\mathcal{Y}|}\}$, sendo que $\forall Y_i \in \mathcal{Y}$ temos $Y_i \subseteq X$, e deseja-se saber se é possível cobrir todos os elementos de X com no máximo k conjuntos de $\mathcal{Y}$, ou seja, deseja-se saber se existe um subconjunto $\mathcal{Y}' \subseteq \mathcal{Y}$ tal que $|\mathcal{Y}'| \leq k$ e $\forall x_j \in X$ existe um $Y_i \in \mathcal{Y}'$ no qual $x_j \in Y_i$ [Cygan et al., 2015]. O problema *d-Set Cover* é igual ao SC com a restrição de que o tamanho dos conjuntos em $\mathcal{Y}$ é no máximo d. Segundo Duh & Fürer [1997], para qualquer $d \geq 3$, *d-Set Cover* é NP-Completo. Assim, o problema escolhido para a redução do Teorema 3 foi o *3-Set Cover* (3-SC), onde esse d é igual a 3 e os conjuntos em $\mathcal{Y}$ tem tamanho no máximo 3.

Teorema 3 *CCV é NP-Completo.*

Demonstração Criamos a instância $I = \{G, A \text{ e } B, k^{CCV}\}$ de CCV a partir de uma instância genérica de 3-SC. Seja $I' = \{X, \mathcal{Y}, k^{3-SC}\}$ uma instância genérica de 3-SC.

Para cada elemento $x_j \in X$ cria-se um vértice a_j correspondente que é inserido em $A(G)$. Para cada $Y_i \in \mathcal{Y}$ cria-se um vértice b_i correspondente que é inserido em $B(G)$, e para cada elemento $x_j \in Y_i$, obtemos o vértice a_j correspondente a x_j e inserimos a aresta (a_j, b_i) em $E(G)$. Assim, uma aresta entre um par de vértices (a_j, b_i) indica que no problema 3-SC, o elemento x_j correspondente a a_j está contido no conjunto Y_i correspondente a b_j. Por fim, fazemos $k^{CCV} = k^{3-SC}$. Note que como cada Y_i tem no máximo três elementos, então o grau de todo vértice em $B(G)$ é no máximo três, e como não criamos arestas entre os vértices de $A(G)$, nem entre vértices de $B(G)$, o grafo G é bipartido. No grafo gerado pela transformação, as partes da bipartição dos vértices são os conjuntos $A(G)$ e $B(G)$. A Figura 5.1 mostra um exemplo dessa transformação, onde os elementos de X se tornam vértices de $A(G)$ e são representados por círculos e conjuntos de $\mathcal{Y}$ se tornam vértices de $B(G)$ e são representados pro triângulos. A seguir, provaremos a equivalência das instâncias.

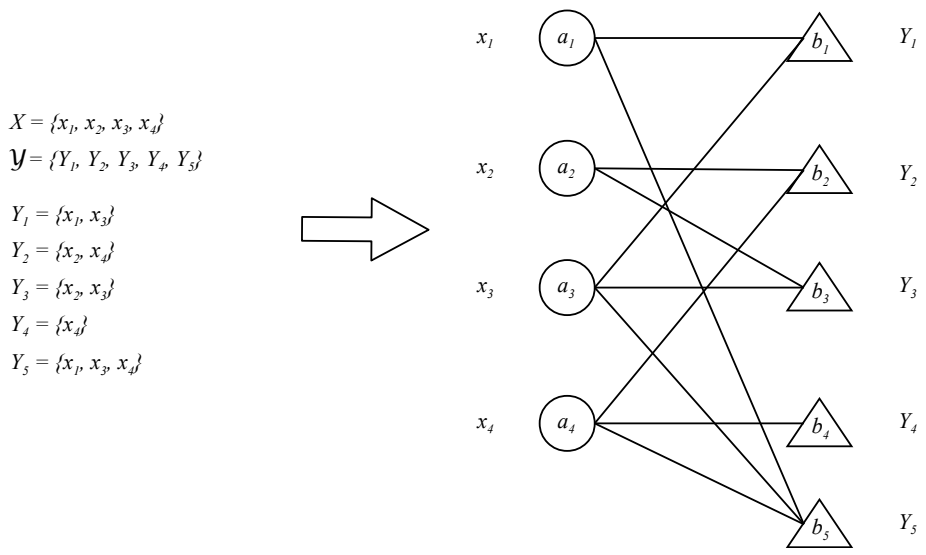

Figura 5.1: Exemplo de uma redução de 3-SC para CCV.

($\Rightarrow$) Se 3-SC tem resposta SIM então CCV tem resposta SIM: se 3-SC tem resposta SIM, existe um conjunto $\mathcal{Y}' \subseteq \mathcal{Y}$, tal que todo elemento de X está em pelo menos um conjunto de $\mathcal{Y}'$ e $|\mathcal{Y}'| \leq k^{3-SC}$. Por construção, como os vértices de $A(G)$ representam os elementos de X, se obtermos os vértices de $B(G)$ correspondentes aos conjuntos em $\mathcal{Y}'$ e os colocarmos em um conjunto $B' \subseteq B(G)$, então existe pelo menos uma aresta de cada vértice em $A(G)$ para um vértice em B'. Como, por construção, não há arestas entre os vértices de $A(G)$ então cada vértice forma uma componente conexa isolada, e como há arestas de todos eles a algum vértice de B' então B' cobre todas as componentes conexas em $A(G)$. Como $|B'| = |\mathcal{Y}'| \leq k^{3-SC} = k^{CCV}$ então $|B'| \leq k^{CCV}$.

($\Leftarrow$) Se CCV tem resposta SIM então 3-SC tem resposta SIM: se CCV tem resposta SIM, existe um conjunto $B' \subseteq B(G)$, tal que $|B'| \leq k^{CCV}$ e B' cobre todas as componentes conexas de $A(G)$. Como, por construção, não há arestas entre os vértices de $A(G)$ então cada vértice forma uma componente conexa isolada, logo, como B' cobre todas as componentes conexas de $A(G)$, B' cobre todos os vértices de $A(G)$, ou seja, todo vértice de $A(G)$ tem pelo menos uma aresta até um vértice de B'. Obtemos os conjuntos de $\mathcal{Y}$ correspondentes aos vértices em B' e os colocarmos em um conjunto $\mathcal{Y}' \subseteq \mathcal{Y}$. Por construção, uma aresta entre um vértice $a_j \in A$ e $b_i \in B$ representa que x_j correspondente a a_j está em Y_i correspondente a b_i. Como os vértices de $A(G)$ representam todos os elementos de X, e há aresta de todo vértice de $A(G)$ até algum vértice de B' então todo elemento de X está em pelo menos um conjunto de $\mathcal{Y}'$. Como $|\mathcal{Y}'| = |B'| \leq k^{CCV} = k^{3-SC}$ então $|\mathcal{Y}'| \leq k^{3-SC}$. ■

Corolário 4 *CCV é NP-Completo mesmo se o grafo de entrada for bipartido.*

Teorema 5 *CCV é NP-Completo mesmo se o grafo de entrada for bipartido e tiver grau máximo* 3.

Demonstração O grafo G criado na transformação do 3-SC para o CCV é bipartido, o grau máximo dos vértices de $B(G)$ é igual a 3 e os graus dos vértices de $A(G)$ podem ser maiores que 3. Seja a um vértice em $A(G)$ tal que $d = d(a) > 3$ e seja $N(a) = \{b_1, b_2, ..., b_d\}$ os vizinhos de a. Criamos mais dois vértices a_2 e a_3, que são inseridos em $A(G)$, e inserimos as arestas (a, a_2) e (a_2, a_3) em $E(G)$, ou seja, os três vértices pertencem a $A(G)$ e fazem parte da mesma componente conexa. Depois, desconectamos dois vizinhos de a e os fazemos vizinhos de a_3: $N(a) = \{b_3, b_4, ..., b_d, a_2\}$, $N(a_2) = \{a, a_3\}$ e $N(a_3) = \{a_2, b_1, b_2\}$. Assim o que fizemos foi inserir dois novos vértices com graus ≤ 3, pois $d(a_2) = 2$ e $d(a_3) = 3$, e diminuímos o grau de a em uma unidade, $|N(a)| = d(a) = d - 1$. Essa operação é ilustrada na Figura 5.2, onde vértices em $A(G)$ são representados por círculos e vértices em $B(G)$ são representados por triângulos. Repare que após essa operação o grafo continua bipartido, mas $A(G)$ e $B(G)$ não são as partes da bipartição natural do grafo bipartidos.

Note que ao fazer isso, uma solução B'_1 para o grafo original continua sendo uma solução para o problema, pois se anteriormente a era coberto por b_1 ou b_2 esses vértices cobrem a_3, consequentemente cobrindo a e a_2, pois há caminho para os mesmos passando por a_3. De forma similar, se a ainda tem aresta para algum vértice de $b_i \in B'_1$, b_i também cobre a_2 e a_3 pois há caminho para os mesmos passando por a. E claramente, uma solução B'_2 para o novo grafo também é solução para o grafo original, pois algum

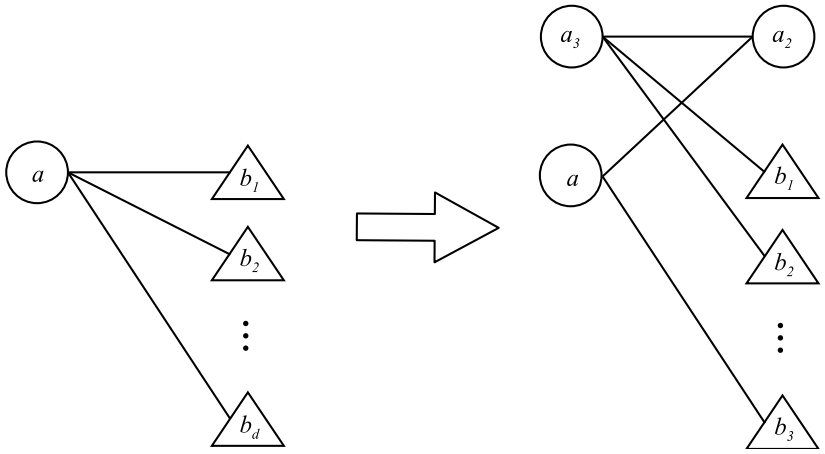

Figura 5.2: Decrementando o grau de vértices de $A(G)$ de um grafo bipartido.

$b_i \in B'_2$ precisa cobrir a componente conexa que contém a, a_2 e a_3, logo, b_i cobriria a no grafo original.

Note também que o procedimento não gera ciclos de tamanho ímpar, pois se a fazia parte de algum ciclo que usava as arestas (a, b_1) ou (a, b_2), agora o ciclo pode apenas ter aumentado de tamanho em duas arestas, (a, a_2) e (a_2, a_3), ou seja, o ciclo continua de tamanho par. Logo, o grafo continua bipartido e diminuímos em um o grau de algum vértice com grau maior que três. Se repetirmos o procedimento até que não haja mais vértices com grau maior que 3 em $A(G)$, teremos um grafo bipartido G' onde todos os vértices tem grau máximo 3, e que uma solução B' para G' também é solução para G.

Como a operação pode ser feita em tempo polinomial apenas passando por todos os vértices e verificando se há algum com grau maior que 3, e criar novos vértices, inseri-los, e trocar inserir e remover um número constante de arestas leva tempo constante, temos que essa operação leva tempo polinomial. ■

Assim como para grafos bipartidos, o problema CCV também é NP-Completo para uma outra classe de grafo muito relevante na teoria dos grafos, os grafos cordais. Para provar que CCV é NP-Completo para grafos cordais, provaremos que o problema é NP-Completo para uma de suas subclasses: os grafos *split*. Tanto grafos *split* quanto grafos cordais são definidos no Capítulo 2 e como os grafos *split* são grafos cordais, se o problema é NP-Completo para grafos *split*, consequentemente o problema também é NP-Completo para grafos cordais.

Teorema 6 *CCV é NP-Completo para grafos split.*

Demonstração Seja G o grafo gerado a partir da transformação da instância do 3-SC para a instância de CCV. Como mostrado no Teorema 3, CCV é NP-Completo para o grafo G que é bipartido, e não possui arestas entre os vértices de uma mesma parte da partição. Logo, $A(G)$ e $B(G)$ são conjuntos independentes. Se criamos um grafo $G' = G$ e adicionarmos todas as arestas possíveis entre vértices de $B(G')$, então G' será um grafo *split* pois $B(G')$ agora é uma clique e $A(G')$ é um conjunto independente. Por fim, CCV tem resposta SIM para a instância com o grafo G se e somente se tem resposta SIM para a instância com o grafo G'. É evidente notar isso, uma vez que adicionar arestas entre os vértices de $B(G)$ não altera o resultado do problema, pois queremos conectar as componentes conexas de $A(G)$ a vértices de $B(G)$ e adicionar arestas neste último conjunto não cria nem remove conexões entre vértices de partes distintas da partição. ■

Como todo grafo *split* também é cordal [Brandstadt et al., 1999], temos o seguinte corolário:

Corolário 7 *CCV é NP-Completo para grafos cordais.*

Duas classes de grafos também bastante estudadas são os grafos planares e os *Unit Disk Graphs* (UDGs), que são definidos no Capítulo 2. Mostraremos aqui que CCV é NP-Completo mesmo se o grafo for um UDG planar. Para esta prova, usaremos o seguinte Lema auxiliar:

Lema 8 *[Clark et al., 1990; Valiant, 1981] Um grafo planar G com grau máximo igual a 4 pode ser desenhado em um plano, usando uma área de tamanho $O(n)$, tal que todos os vértices são situados em coordenadas inteiras do plano e todas as arestas podem ser desenhadas sobre as linhas do plano, isto é, qualquer ponto dessa reta tem pelo menos uma de suas coordenadas inteiras. Uma ilustração desse processo pode ser vista na Figura 5.3, onde à esquerda temos um grafo planar com grau máximo igual a 4 e, à direita, o mesmo grafo desenhado no grid.*

Usando o Lema acima, provaremos que CCV é NP-Completo para UDGs planares fazendo uma redução a partir do problema *Vertex Cover* (VC), uma vez que este problema é conhecidamente NP-Completo para grafos planares com grau máximo igual a 3 [Bar-Yehuda & Even, 1985; Garey & Johnson, 2002; Clark et al., 1990]. No problema VC recebemos como entrada um grafo G e um inteiro k e desejamos saber se existe um subconjunto de vértices de tamanho no máximo k que cobre todas as arestas do grafo, ou seja, um subconjunto $V' \subseteq V(G)$ em que $|V'| \leq k$ e $\forall (u, v) \in E(G)$ temos que $V' \cap \{u, v\} \neq \emptyset$.

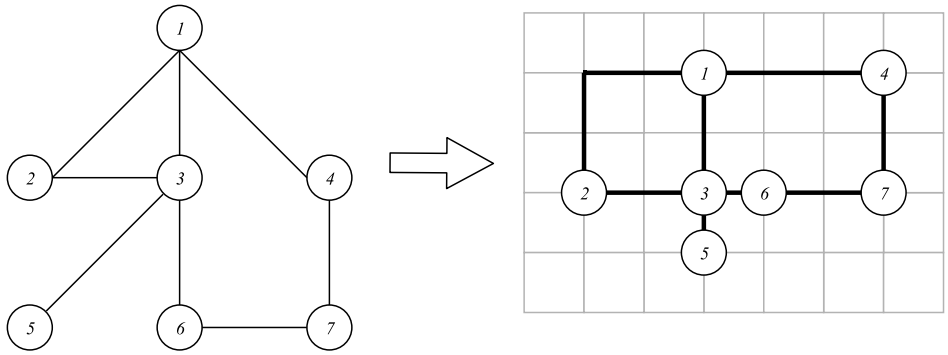

Figura 5.3: Exemplo de utilização do Lema 8.

Teorema 9 *CCV é NP-Completo para UDGs planares.*

Demonstração Criamos uma instância $\{G^{CCV}, A \text{ e } B, k^{CCV}\}$ de CCV, na qual G^{CCV} será um UDG. Seja $\{G^{VC}, k^{VC}\}$ um instância de VC e G^{VC} é um grafo planar de grau máximo 3. Uma ilustração desta redução é mostrada na Figura 5.4. Desenhamos G^{VC} em um plano como especificado no Lema 8, e trabalharemos sempre a partir de G^{VC} nesse plano, que é um *grid* que chamaremos de g_1. Define-se que a medida do lado de cada célula nesse *grid* é igual a 1. Como G^{CCV} deverá ser um UDG, ele também será desenhado em um plano, que também será um *grid* g_2 de tamanho igual ao de g_1 e com células com lado de tamanho 1.

Copiamos os vértices de G^{VC} para $B(G^{CVV})$ e os desenhamos em g_2, sendo que, como esses vértices devem ser discos, seus centros serão exatamente nas coordenadas em que estão situados em g_1 e terão raio igual a $\frac{1}{4}$. Após isso, substituiremos as arestas em G^{VC} por componentes conexas formadas por vértices intermediários: $\forall e_j \in E(G^{VC})$ cria-se uma componente conexa C_j com t_j vértices que são desenhados em g_2 seguindo o caminho que a aresta e_j segue em g_1 e como esses vértices são discos, eles terão raio igual a $\frac{1}{4}$ e serão colocados sempre tangentes aos vértices vizinhos. Os vértices em C_j são inseridos em $A(G^{CCV})$.

Note que como colocamos os vértices tangentes, eles serão vizinhos, ou seja, serão conexos e também serão vizinhos dos vértices a qual a aresta conectava. Note também que como G^{VC} é planar, suas arestas estão desenhadas de forma que não se cruzam e como a célula do *grid* tem lado de tamanho 1 e os discos tem raio $\frac{1}{4}$, nenhum disco de uma componente criada tem interseção com algum disco de outra componente. Vale frisar que o grafo gerado também é planar, pois todos os discos são apenas tangentes, não tem interseção, logo, não tem arestas que se cruzam. Por fim, fazemos $k^{CCV} = k^{VC}$. Provaremos então a equivalência dos problemas.

Se VC tem resposta SIM então existe um conjunto $V' \subseteq V(G^{VC})$ tal que V' cobre

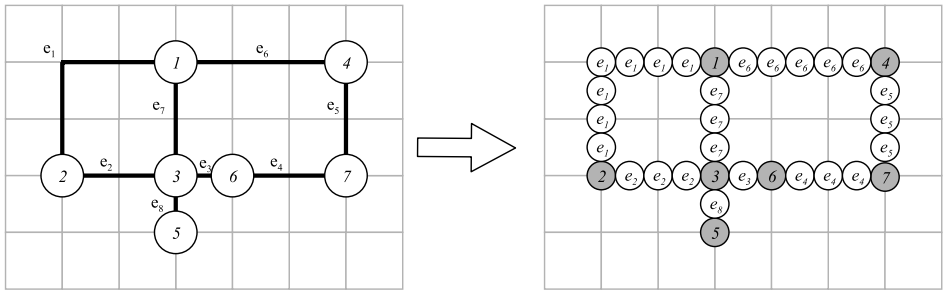

Figura 5.4: Exemplo de redução de VC em grafos planares com grau ≤ 4 para CCV em UDGs planares.

todas as arestas de G^{VC} e $|V'| \leq k^{VC}$. Obtemos o conjunto $B' \subseteq B(G^{CCV})$ de vértices que foram criados correspondentes aos vértices de V', ou seja, são as cópias de V' no grafo G^{CCV}. Por construção, cada componente conexa C_j de $A(G^{CCV})$ foi criada a partir de uma aresta $e_j = (u_j, v_j)$ de G^{VC} e C_j só é ligada a dois vértices b_j^u e b_j^v de $B(G^{CCV})$ que são as cópias de u_j e v_j. Como V' cobre todas as arestas temos que $u_j \in V'$ ou $v_j \in V'$ o que implica que $b_j^u \in B'$ ou $b_j^v \in B'$, logo C_j é coberta por B'. Como isso é válido para todas as arestas e componentes criadas a partir delas, B' cobre todas as componentes em $A(G^{CCV})$. Como $|B'| = |V'| \leq k^{VC} = k^{CCV}$ então $|B'| \leq k^{CCV}$.

Se CCV tem resposta Sim então existe um conjunto $B' \subseteq B(G^{CCV})$ que cobre todas as componentes de $A(G^{CCV})$ e $|B'| \leq k^{CCV}$. Obtemos o conjunto $V' \subseteq V(G^{VC})$ de vértices originais dos quais os vértices em B' foram criados. Por construção, cada componente conexa C_j de $A(G^{CCV})$ foi criada a partir de uma aresta $e_j = (u_j, v_j)$ de G^{VC} e C_j só é ligada a dois vértices b_j^u e b_j^v de $B(G^{CCV})$ que são as cópias de u_j e v_j. Como B' cobre todas as componentes temos que $b_j^u \in B'$ ou $b_j^v \in B'$ o que implica que $u_j \in V'$ ou $v_j \in V'$, logo a aresta e_j é coberta por V'. Como isso é válido para todas as arestas e componentes criadas a partir delas, V' cobre todas as arestas de G^{VC}. Como $|V'| = |B'| \leq k^{CCV} = k^{VC}$ então $|V'| \leq k^{VC}$.

Se existisse um algoritmo que resolvesse CCV para UDGs planares em tempo polinomial, poderíamos usá-lo para o grafo gerado nessa redução e resolver VC para grafos planares com grau máximo 3 em tempo polinomial, o que não é possível a menos que P = NP. Logo, CCV é NP-Completo para UDGs planares. ■

Como os grafos mostrados acima são todos grades parciais — definidas no Capítulo 2 —, o Teorema 9 deriva o seguinte corolário:

Corolário 10 *CCV é NP-Completo para grades parciais.*

Tendo em vista que o problema estudado é NP-Completo para grades parciais, é interessante pensar se o problema continua NP-Completo mesmo para grades. Vale frisar que grades são um tipo de grafo muito restrito e específico. Muitos problemas NP-Completos da literatura se tornam polinomiais, lineares ou até mesmo constantes quando o grafo de entrada se restringe a uma grade. No entanto, mostramos no Teorema a seguir que CCV continua NP-Completo mesmo que o grafo de entrada seja uma grade.

Teorema 11 *CCV é NP-Completo para grades.*

Demonstração Seja $\{G, A \text{ e } B, k\}$ uma instância de CCV na qual G é uma grade parcial, com um *grid* $n \times m$. Denotaremos os vértices de $V(G)$ por $v_{i,j}$, para $0 \leq i < n$ e $0 \leq j < m$. Por simplicidade, assumimos que, para cada par i, j, o vértice $v_{i,j}$ existe. Se esse não for o caso, podemos adicionar um novo vértice $v_{i,j}$ em $V(G)$ que pertencerá a $B(G)$. Como não adicionamos arestas, essa operação claramente não mudará a resposta do problema para essa instância.

Define-se que um vértice $v_{i,j}$ pode ser adjacente de um vértice $v_{k,l}$ somente se os dois são "vizinhos" na grade, ou em outras palavras, não há nenhum vértice entre eles. Mais formalmente, eles podem ser adjacentes somente se $i = k$ e $|j - l| = 1$ ou $|i - k| = 1$ e $j = l$.

Construiremos um novo grafo G', instância de CCV, que será a transformação de G em uma grade. G' terá dimensões $3n - 2 \times 3m - 2$. Para cada vértice $v_{i,j} \in V(G)$, criamos e mapeamos um vértice $w_{3i,3j}$ e inserimos em $V(G')$, e $w_{3i,3j}$ pertencerá a $A(G')$ se e somente se $v_{i,j} \in A(G)$ — caso contrário, pertencerá a $B(G')$.

Assim, podemos assumir que G' é uma grade parcial na qual faltam apenas arestas. Podemos ver que o *grid* de G' é um *grid* maior: entre cada par de colunas do *grid* original, há duas novas colunas. De forma similar, entre cada par de linhas do *grid* original, há duas novas linhas. Assim, entre cada par de vértices do grafo original, deveremos inserir dois novos vértices. Além disso, dentro de cada célula do *grid* original, existirão quatro novos vértices, por causa das interseções das novas linhas e colunas. Inserimos esses quatro novos vértices e os colocamos em $B(G')$. Ou seja, para cada $i, j \in [0, n-2] \times [0, m-2]$, são criados quatro vértices $w_{3i+1,3j+1}$, $w_{3i+1,3j+2}$, $w_{3i+2,3j+1}$ e $w_{3i+2,3j+2}$ e inseridos em $B(G')$. Inserimos também em $E(G')$ as quatro arestas necessárias entre eles, ou seja, as arestas $(w_{3i+1,3j+1}, w_{3i+1,3j+2})$, $(w_{3i+1,3j+1}, w_{3i+2,3j+1})$, $(w_{3i+1,3j+2}, w_{3i+2,3j+2})$ e $(w_{3i+2,3j+1}, w_{3i+2,3j+2})$. Como, por enquanto, eles só tem arestas entre si, eles não criam novas soluções.

Inseriremos então os vértices entre cada par de vértices $v_{i,j}, v_{i,j+1}$ do grafo original. Nesse caso, são vértices adjacentes do grafo original que estão na mesma linha, mas

o processo abaixo pode ser generalizado para os vértices adjacentes na mesma coluna (mudando apenas o índice dos vértices para $v_{i,j}, v_{i+1,j}$). Criamos os vértices $w_{3i,3j+1}$ e $w_{3i,3j+2}$ e os inserimos em $V(G')$. Inserimos também em $E(G')$ as arestas necessárias, ou seja, as arestas $(w_{3i,3j}, w_{3i,3j+1})$, $(w_{3i,3j+1}, w_{3i,3j+2})$ e $(w_{3i,3j+3})$. Lembrando que os vértices $w_{3i,3j}$ e $w_{3i,3j+3}$ já existem em $V(G')$ pois são os vértices mapeados a $v_{i,j}$ e $v_{i,j+1}$, respectivamente. A parte da partição de G' a qual $w_{3i,3j+1}$ e $w_{3i,3j+2}$ irão pertencer depende das partes da partição de $v_{i,j}$ e $v_{i,j+1}$ no grafo original e se há aresta entre eles. Teremos então seis casos:

- Se $v_{i,j}, v_{i,j+1} \in A(G)$ e $(v_{i,j}, v_{i,j+1}) \in E(G)$, então $w_{3i,3j+1}$ e $w_{3i,3j+2}$ pertencerão à $A(G')$, esse caso é mostrado na Figura 5.5. Assim, como em G, $v_{i,j}$ e $v_{i,j+1}$ faziam parte da mesma componente, em G' eles também farão parte e um vértice $v_{k,l} \in B(G)$ que cobrisse $v_{i,j}$ e $v_{i,j+1}$ no grafo original, tem um vértice $w_{3k,3l} \in B(G')$ correspondente que pode cobrir $w_{3i,3j}$ e $w_{3i,3j+3}$ em G', além de cobrir $w_{3i,3j+1}$ e $w_{3i,3j+2}$, o que faz sentido, já que estão na mesma componente.

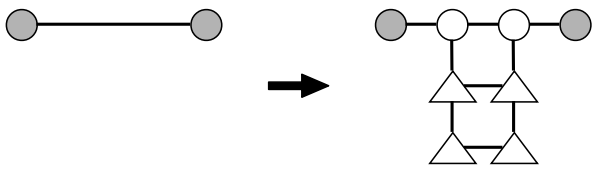

Figura 5.5: Redução de grade parcial para grade: caso 1.

- Se $v_{i,j}, v_{i,j+1} \in A(G)$ e $(v_{i,j}, v_{i,j+1}) \notin E(G)$, então $w_{3i,3j+1}$ e $w_{3i,3j+2}$ pertencerão à $B(G')$, como mostrado na Figura 5.6. Podemos criar assim novas soluções, mas não melhores que as já existentes. Isso porque se $w_{3i,3j+1}$ estiver na solução ele só estará cobrindo a componente de $w_{3i,3j}$, mas no grafo original, algum vértice $v_{k,l} \in B(G)$ cobriria a componente de $v_{i,j}$, então podemos colocar seu correspondente $w_{3k,3l}$ na solução no lugar de $w_{3i,3j+1}$. O mesmo procedimento é válido para caso $w_{3i,3j+2}$ gere uma nova solução.

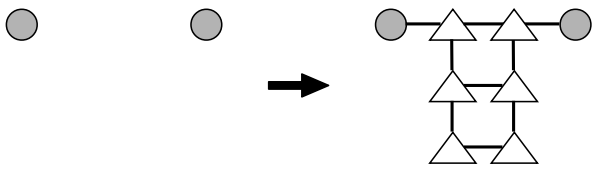

Figura 5.6: Redução de grade parcial para grade: caso 2.

- Se $v_{i,j} \in A(G)$ e $v_{i,j+1} \in B(G)$ (ou o contrário, ou seja, os vértices estejam em partes diferentes da partição), utilizamos um dos casos abaixo. Sem perda de

generalidade, assumimos que $v_{i,j} \in A(G)$, caso contrário, basta fazer o procedimento abaixo trocando os índices:

– Se $(v_{i,j}, v_{i,j+1}) \in E(G)$, então $w_{3i,3j+1}$ e $w_{3i,3j+2}$ pertencerão à $A(G')$, caso ilustrado na Figura 5.7. Assim, se em G a componente de $v_{i,j}$ fosse coberta por $v_{i,j+1}$, ela será coberta por $w_{3i,3j}$ em G'.

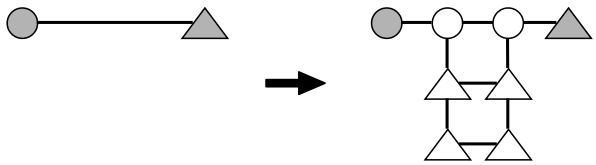

Figura 5.7: Redução de grade parcial para grade: caso 3.

– Se $(v_{i,j}, v_{i,j+1}) \notin E(G)$, então $w_{3i,3j+1}$ e $w_{3i,3j+2}$ pertencerão à $B(G')$, esse caso é mostrado na Figura 5.8. Podemos criar assim novas soluções, mas não melhores que as já existentes. Isso porque uma solução pode conter $w_{3i,3j+1}$, mas, neste caso, ele só estará cobrindo a componente de $w_{3i,3j}$. No entanto, no grafo original, a componente de $v_{i,j}$ era coberta por algum vértice $v_{k,l} \in B(G)$, então podemos colocar seu correspondente $w_{3k,3l}$ na solução no lugar de $w_{3i,3j+1}$.

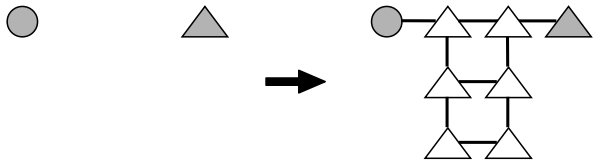

Figura 5.8: Redução de grade parcial para grade: caso 4.

- Os últimos dois casos são se $v_{i,j}, v_{i,j+1} \in B(G)$ e $(v_{i,j}, v_{i,j+1}) \in E(G)$ ou $\notin E(G)$. Nos dois casos, $w_{3i,3j+1}$ e $w_{3i,3j+2}$ pertencerão à $B(G')$, como mostrado na Figura 5.9. Nesses caso, claramente não criaremos novas soluções, pois não inserimos nenhuma aresta entre vértices de partições distintas em G'.

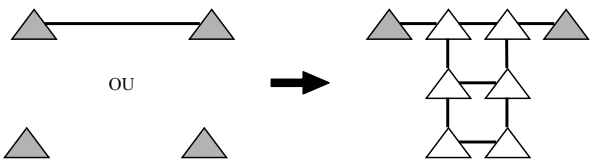

Figura 5.9: Redução de grade parcial para grade: casos 5 e 6.

Note que os novos vértices de $B(G')$ que foram inseridos dentro das células do *grid* original ($w_{3i+1,3j+1}$, $w_{3i+1,3j+2}$, $w_{3i+2,3j+1}$ e $w_{3i+2,3j+2}$) podem criar novas soluções, mas

não melhores que as originais. Isso porque um w_t desses vértices pode estar ligado a um vértice $w_p \in A(G')$ da sua mesma linha ou de sua mesma coluna, mas se isso acontece e ele está na solução, ele cobre só essa componente, e no grafo original existiria um vértices $v_r \in B(G)$ que cobriria a componente de w_p (lembre-se que todos os vértices inseridos em $A(G')$ fazem parte de uma componente do grafo original), então podemos trocar w_t por w_r. Também há o caso de w_t estar ligado a dois vértices $w_p, w_q \in A(G')$ da sua mesma linha e mesma coluna cada. Mas isso só acontece se:

- Os dois estão ligados a um vértice $w_s \in A(G')$ que é correspondente a um vértice $v_s \in A(G)$ do grafo original. Assim, algum vértice $v_r \in B(G)$ cobriria v_s no grafo original. Podemos então colocar seu correspondente w_r na solução no lugar de w_t, pois w_r cobre a componente de w_s.

- Os dois estiverem ligados a um vértice $w_s \in B(G')$ que é correspondente a um vértice $v_s \in B(G)$, do grafo original. Podemos então trocar w_t por w_s, simplesmente.

A redução pode claramente ser feita em tempo polinomial. Um exemplo dessa redução é mostrado na Figura 5.10, onde círculos representam vértices de A, triângulos representam vértices de B e vértices sombreados representam uma solução para o problema. Note que as componentes conexas induzidas por $A(G')$ têm uma correspondência um-para-um com as componentes conexas induzidas por $A(G)$. Note também que em cada um dos casos, podem ser criadas novas soluções, mas não melhores do que as já existentes. Como após a redução, G' é uma grade, então CCV é NP-Completo para grades.

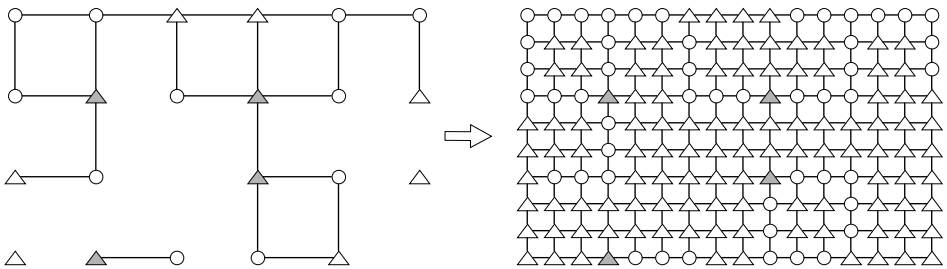

Figura 5.10: Exemplo de redução de uma grade parcial para uma grade.

■

Capítulo 6

Algoritmos Polinomiais Exatos

Um dos objetivos deste trabalho é encontrar algoritmos polinomiais que resolvam o problema CCV para determinadas classes de grafos, se beneficiando da estrutura das mesmas para obter tais algoritmos. No entanto, não é trivial encontrar algoritmos polinomiais para classes específicas, pois o problema é bastante difícil como mostrado anteriormente que o mesmo é NP-Completo mesmo para classes de grafos muito restritas como, por exemplo, grafos bipartidos com máximo grau 3 ou grades. Mostramos neste Capítulo que é possível resolver CCV em tempo polinomial para árvores, grafos bloco, cografos, cactos e grafos *outerplanares*.

Para a maioria dos algoritmos polinomiais exatos, dada uma instância de CCV $\{G, A \text{ e } B, k\}$ são utilizadas algumas regras em comum. Às chamaremos de **regras de redução básicas**. São elas:

- Se o grafo é desconexo, podemos aplicar o algoritmo para cada componente conexa separadamente. Isso acontece porque o resultado de uma componente claramente não influencia no resultado de outra. Assim, sem perda de generalidade, assumimos que o grafo de entrada é conexo.

- Se $k < 0$, respondemos NÃO.

- Se $|A(G)| = 0$ e $k \geq 0$, respondemos SIM.

- Se $|A(G)| > 0$ e $k = 0$ ou $|B(G)| = 0$, respondemos NÃO.

- Se $|B(G)| = 1$, esse vértice tem que estar na solução, pois ele cobre a componente conexa inteira. O colocamos na solução e diminuímos k em uma unidade.

Após aplicar essas regras o máximo possível, temos que o algoritmo já deu a resposta ou assumiremos que estamos trabalhando com um grafo conexo e que $k > 0$, $|A(G)| > 0$ e $|B(G)| > 1$.

Para alguns algoritmos polinomiais também utilizamos a seguinte regra de redução, que chamamos de **regra do grau 1**: seja $\{G, A \text{ e } B, k\}$ uma instância de CCV. Aplicamos o processo de simplificação em G e obtemos o grafo G_s. Após essa operação, vértices de $A(G_s)$ só tem arestas para vértices de $B(G_s)$ e vice-versa. Seja $u \in V(G_s)$ um vértice de grau 1 (se existir). Seja v o vizinho de u.

- se $u \in B(G_s)$ então $v \in A(G_s)$. Podemos então remover u do grafo pois, como assumimos que $|B(G_s)| > 1$ e que o grafo é conexo, existe algum outro vértice $b \in B(G_s)$ que pode cobrir v, logo, se existisse uma solução que continha o vértice u ele está cobrindo somente v e podemos trocá-lo por b.

- se $u \in A(G_s)$ então $v \in B(G_s)$ e v é o único vértice que pode cobrir u. Então colocamos v na solução, diminuímos k em uma unidade e removemos do grafo u, v e quaisquer outros vértices de $A(G_s)$ cobertos por v.

Podemos aplicar essa regra de redução até encontrar a resposta, ou ao fim da mesma, obter um grafo com grau no mínimo 2.

Mostraremos agora um algoritmo polinomial para árvores:

Teorema 12 *CCV pode ser resolvido em tempo polinomial em árvores.*

Demonstração Seja $\{G, A \text{ e } B, k\}$ uma instância de CCV e G é uma árvore. Aplicamos o procedimento de simplificação em G, gerando o grafo G_s. Note que contrair arestas não gera ciclos, então isso não desfaz a estrutura de árvore. Remover arestas pode desconectar o grafo, mas cada componente conexa do grafo ainda será uma árvore. Utilizamos as regras de redução básicas, assim, podemos assumir que estamos trabalhando com uma árvore em que vértices de $A(G_s)$ só tem vizinhos em $B(G_s)$ e vice-versa. Como o grafo é uma árvore, algum vértice é folha e tem grau 1. Se aplicarmos a regra do grau 1 diminuiremos o tamanho do grafo em pelo menos uma unidade e o grafo resultante continua sendo uma árvore. Se aplicarmos essas regras repetidamente, encontraremos uma solução ótima. ■

A partir do algoritmo polinomial para árvores, podemos desenvolver um algoritmo polinomial para grafos bloco, mostrado a seguir:

Teorema 13 *CCV pode ser resolvido em tempo polinomial em grafos bloco.*

Demonstração Seja $\{G, A \text{ e } B, k\}$ uma instância de CCV e G é um bloco. Aplicamos as regras de redução básicas, assim podemos assumir que trabalhamos com um grafo conexo. Aplicamos o processo de simplificação em G obtendo o grafo G_s. Repare que, cada componente biconexa de G é uma clique. Após a contração, cada uma dessas componentes biconexas possuirão no máximo um vértice de $A(G_s)$. Note que após a simplificação, não há mais ciclos no grafo. Isso porque, como só há arestas entre vértices de partes diferentes da partição (A, B) de G_s teríamos que ter um ciclo de tamanho par, onde os vértices se alternam entre vértices de $A(G_s)$ e de $B(G_s)$. Mas esse ciclo seria uma componente biconexa, e teria que ser uma clique. O que que quer dizer, que vértices de $A(G_s)$ desse ciclo teriam arestas que seriam contraídas. Logo, o grafo G_s é conexo e sem ciclos, ou seja, é uma árvore. Basta utilizar o mesmo algoritmo utilizado em árvores. ■

Teorema 14 *CCV pode ser resolvido em tempo polinomial em cografos.*

Demonstração Seja $\{G, A \text{ e } B, k\}$ uma instância de CCV e G é um cografo. Primeiramente utilizamos as regras de redução básicas. Podemos então assumir que G é conexo, $|A(G)| > 0$, $|B(G)| > 1$ e $k > 0$. Mostraremos que para cada componente conexa de um cografo, se ela tem algum vértice de B, esse vértice cobre todas as componentes de vértices de A. Para demonstrar isso, obtemos a *cotree* do cografo. Como mostrado em Cournier & Habib [1994], a obtenção da *cotree* pode ser feita em tempo linear. Podemos assumir que a *cotree* é binária, pois caso não seja, é fácil ver que junções e uniões são aplicadas em pares de cografos, ou seja, é só criarmos mais nós internos, e isso pode claramente ser feito em tempo linear. Podemos assumir também que a raiz da *cotree* é uma junção, como assumimos que o grafo é conexo, a última operação necessariamente foi uma junção. Assim, digamos que a raiz é a junção de dois cografos G_1 e G_2. Se há vértices de $A(G)$ nos dois lados da *cotree*, isto é, em $A(G_1)$ e em $A(G_2)$: como a raiz representa uma junção, todos os vértices de $A(G_1)$ têm arestas para todos os vértices de $A(G_2)$, logo, é uma única componente conexa, e como o grafo é conexo, podemos escolher qualquer vértice de $B(G)$ vizinho de algum vértice de $A(G)$ para estar na solução. Se há vértices de $A(G)$ em somente um lado da *cotree*, digamos em $A(G_1)$, e além de saber que existe pelo menos um vértice em $B(G)$, sabemos também que $V(G_2) \neq \emptyset$, então existe pelo menos um vértice em $b \in B(G_2)$. Como a raiz representa uma junção, todos os vértices de $A(G_1)$ têm aresta para b, logo, ele pode estar na solução que cobrirá todos os vértices de $A(G)$. ■

Teorema 15 *CCV pode ser resolvido em tempo polinomial em cactos.*

Demonstração Seja $\{G, A \text{ e } B, k\}$ uma instância de CCV e G é um cacto. Faremos então as regras de redução básicas, e assumiremos que estamos trabalhando com G conexo, $k > 0$, $|A(G)| > 0$ e $|B(G)| > 1$. Aplicamos então o processo de simplificação em G, obtendo o grafo G_s. Note que esse grafo também é um cacto, pois remover ou contrair arestas não faz com que uma aresta faça parte de mais de um ciclo simples, pois ao remover só poderíamos desconectar um ciclo e ao contrair só poderíamos diminuir o tamanho do ciclo. Assim, como só há arestas entre vértices de partes diferentes da partição, teremos um grafo formado por ciclos e caminhos onde os vértices se alternam em relação a qual parte da partição pertencem. Aplicamos a regra do grau 1. Após essa regra, ou o algoritmo já retornou a solução, ou todos os vértices do grafo tem grau maior que 1. Logo, o grafo é formado apenas por ciclos em que os vértices são alternados em relação à parte da partição a que pertencem. Note que esses ciclos têm um número par de vértices, pois como os vértices se alternam na sequência do ciclo, se o tamanho fosse ímpar, esse ciclo iniciaria e terminaria com vértices de uma mesma parte da partição, o que não é possível. Como o grafo é um cacto, todo par de ciclos tem no máximo um vértice em comum. Chamaremos os vértices em comum de ciclos diferentes de **vértices de corte**. Note que sempre há pelo menos um ciclo com no máximo um vértice de corte, pois se todos os ciclos tivessem pelo menos dois vértices de corte teríamos arestas que fariam parte de mais de um ciclo, não satisfazendo a condição de ser um cacto. Seja $C \subseteq V(G_s)$ o conjunto de vértices de um ciclo de G_s com no máximo um vértice $c_1 \in C$ de corte. Chamaremos de $A(C)$ e $B(C)$ os conjuntos de vértices de C que pertencem a $A(G_s)$ e $B(G_s)$, respectivamente. Se houver apenas um ciclo, fazemos c_1 ser qualquer vértice de $B(C)$, pois assim sabemos que se $c_1 \in A(C)$, temos mais de um ciclo no grafo. Sabemos que C tem um número par de vértices, então diremos que $|C| = 2t$. Nomearemos a sequência do ciclo C da seguinte forma: $C =< c_1, c_2, \ldots, c_{2t}, c_1 >$. Nessa sequência, os vértices se alternam em relação as partes $A(G_s)$ e $B(G_s)$ da partição, ou seja, sendo os conjuntos $C^i, C^j \subseteq C$ onde todo $c_{2i-1} \in C$ pertence a C^i e todo $c_{2j} \in C$ pertence a C^j e $i, j \in \{1, 2, \ldots, t\}$, temos que C^i pertence a uma parte da partição de G_s e C^j pertence a outra. Com isso, um vértice de $B(C)$ pode cobrir exatamente dois vértices de $A(C)$. Assim, precisaremos de exatamente $\lceil \frac{|C|}{4} \rceil$ vértices de $B(C)$ para cobrir todos os vértices de $A(C)$. Precisamos saber como escolher tais vértices.

- Se $c_1 \in B(G_s)$ é sempre melhor escolher os vértices $c_{4i-3} \in C^i$, com $i \in 1, 2, \ldots, \lceil \frac{t}{2} \rceil$, ou seja, incluindo c_1 na solução, pois como ele é um vértice de corte, ele pode cobrir mais vértices que os do ciclo. Removemos então do grafo todos os vértices do ciclo, mais os possíveis vértices cobertos por c_1 e diminuímos

k em $\lceil \frac{|C|}{4} \rceil$.

- Se $c_1 \in A(G_s)$, teremos dois casos:

  1. se $|C|$ é um múltiplo de 4, ou seja, $|C| = 4q$, um de seus vizinhos no ciclo tem que estar na solução. Para mostrar isso, escreveremos a sequência da seguinte forma: $C =< a_1, b_2, a_3, \ldots, a_{4q-1}, b_{4q}, a_1 >$, sendo $a_1 = c_1$. Excluindo a_1, para cobrir os $2q-1$ vértices restantes de $A(C)$, precisaremos de no mínimo $\lceil \frac{2q-1}{2} \rceil = q$ vértices de $B(C)$. Seja $j \in \{1, 2, \ldots, q\}$. Como sempre escolhemos os vértices de $B(C)$ alternadamente para a solução, se não escolhemos b_2, o primeiro escolhido é b_4. Então para termos q vértices, escolheremos os vértices $b_{4j} \in C$. No entanto, o último vértice é b_{4q}, que é vizinho de a_1. Se tentarmos fazer a escolha inversa e não escolhermos b_{4q}, o primeiro escolhido é b_{4q-2}. Então para termos q vértices, escolheremos os vértices $b_{4j-2} \in C$. No entanto, o primeiro vértice é b_2, que é vizinho de a_1. Logo, em qualquer caso teremos que colocar um vizinho de a_1 na solução. Escolhemos então um desses conjuntos, colocamos na solução, diminuímos k em q unidades e removemos o ciclo do grafo.

  2. se $|C|$ não é múltiplo de 4, ou seja, $|C| = 4q-2$, nenhum dos vizinhos de a_1 no ciclo tem que estar na solução. Para mostrar isso, escreveremos a sequência da seguinte forma: $C =< a_1, b_2, a_3, \ldots, a_{4q-3}, b_{4q-2}, a_1 >$, sendo $a_1 = c_1$. Se escolhermos colocar um dos vizinhos de a_1 na solução, ele cobrirá a_1 e no máximo outro vértice de $A(C)$. Assim, para cobrir os outros $2q-3$ vértices de $A(C)$ são necessários no mínimo $\lceil \frac{2q-3}{2} \rceil = q-1$ vértices. Como já escolhemos um vizinho de a_1, teríamos escolhido q vértices de $B(C)$ para a solução. No entanto, se não escolhermos nenhum vizinho de a_1, para cobrir os outros $2q-2$ vértices de $A(C)$, precisaremos de no mínimo $\lceil \frac{2q-2}{2} \rceil = q-1$ vértices. Como sabemos que há mais de um ciclo no grafo, então há pelo menos dois vizinhos $b^{a1}, b^{a2} \in B(G_s)$ que não pertencem a C e são vizinhos de a_1. Como não cobrimos a_1 com nenhum vértice de C, precisaremos colocar b^{a1} ou b^{a2} na solução, e assim, usando também q vértices para cobrir os vértices de $A(C)$. Mas note que b^{a1} ou b^{a2} podem ser vértices de corte de outros ciclos e cobrir não apenas dois vértices (o que aconteceria se escolhêssemos algum vizinho de a_1 em C). Logo, é sempre melhor a segunda opção. Então, sendo $j \in \{1, 2, \ldots, q-1\}$, se $|C| = 4q-2$, colocamos o conjunto de vértices $c_{4j} \in C$ na solução, diminuímos k em $q-1$ unidades e removemos o ciclo C do grafo, com exceção do vértice a_1.

Se repetirmos essas regras de redução consecutivamente, em algum momento retornaremos SIM ou NÃO. Note que sempre só removemos vértices e arestas, isso é claramente polinomial. Logo, podemos resolver o problema CCV em cactos em tempo polinomial. ■

Teorema 16 *CCV pode ser resolvido em tempo polinomial em grafos outerplanares.*

Demonstração Seja $\{G, A \text{ e } B, k\}$ uma instância de CCV e G é *outerplanar*. Aplicamos as mesmas regras de redução que aplicamos na prova para cactos, no Teorema 15: regras de redução básica e simplificação de G obtendo o grafo G_s. Podemos então assumir assim que $k > 0$, $|A(G_s)| > 0$ e $|B(G_s)| > 1$. Remoções e contrações de arestas não fazem com que vértices deixem de fazer parte da face externa. Logo, como o grafo é *outerplanar*, após essas regras ele continua *outerplanar*. Como na prova para cactos, teremos apenas caminhos e ciclos pares onde em ambos os vértices são alternados em relação à parte da partição a qual pertencem. Continuamos então aplicando as regras de redução do Teorema 15: aplicamos a regra do grau 1 e aplicamos a regra para ciclos conectados por um único vértice de corte. Basicamente, aplicamos todas as regras possíveis da prova para cactos. Após aplicar essas regras, se ainda existirem vértices no grafo, ele será composto por ciclos pares, onde os vértices são alternados em relação à parte da partição a qual pertencem e eles tem pelo menos dois vértices de corte (vértices que pertencem a mais de um ciclo), ou seja, o grafo restante é biconectado. Em grafos biconectados podemos fazer uma **decomposição em orelhas**.

Dado um grafo biconectado G e F um subgrafo de G, uma orelha de F em G é um caminho P em G tal que seus extremos (primeiro e último vértice de P) estão em F, mas os vértices internos de P não estão em F. Chamamos de decomposição em orelhas uma sequência $(G_0, G_1, \ldots, G_r)$ de subgrafos de G tal que G_0 é um ciclo e, para $0 \leq i < r$, seja Q_i uma orelha de G_i, temos que $G_{i+1} = G_i \cup Q_i$ e que $G_r = G$ [Aubry et al., 2016]. Em outras palavras, podemos dividir o grafo em um ciclo inicial e várias orelhas, que são caminhos conectados a esses ciclos. Em seu trabalho, Aubry et al. [2016] mostra grafos *outerplanares* biconectados tem uma decomposição em orelhas em que G_0 é qualquer face de G e os extremos de cada orelha são adjacentes em G, isto é, há arestas entre os extremos de cada orelha. Assim sendo, uma orelha nessa decomposição, mais a aresta entre seus extremos, é um ciclo de G. Um algoritmo polinomial para obter uma decomposição em orelhas de um grafo *outerplanar* biconectado é mostrado em Kazmierczak & Radhakrishnan [2000].

Usando a decomposição em orelhas, se usarmos a orelha Q_r, ela terá apenas os extremos em comum com o resto do grafo. E essa orelha, mais a aresta entre seus extremos, forma um ciclo. Chamaremos esse ciclo de $C =< a_1, b_2, a_3, b_4, \ldots, a_{t-1}, b_t, a_1 >$.

Como dito antes, C tem tamanho par (t é par) e seus vértices são alternados em relação a parte da partição a qual pertencem. Sabemos também que os extremos de Q_r pertencem a partes diferentes da partição. São eles a_1 e b_t. Note que a_1 é um vértice de corte, logo ele não necessariamente precisa ser coberto por algum vértice de $B(C)$. Seja A_q o conjunto dos vértices de Q_r pertencentes a $A(C)$ que não são vértices extremos de Q_r, ou seja, $A_q = A(C) - \{a_1\}$. Esses vértices necessariamente precisam ser cobertos por vértices de $B(C)$. Se t for múltiplo de 4, então existe um número inteiro x tal que $4x = t$. Assim sendo, $2x$ desses vértices são de $A(C)$, e $|A_q| = 2x - 1$. Para cobrir esses vértices, serão necessários $\lceil \frac{2x-1}{2} \rceil = x$ vértices de $B(C)$ e precisamos escolher esses vértices alternadamente. Seja $i \in \{1, 2, \ldots, x\}$. Se colocarmos os vértices $b_{4i-2} \in B(C)$ na solução, como $b_t = b_{4x}$ e o último vértice escolhido será b_{4x-2} então b_t não estará na solução e cobriremos não só A_q mas todos os vértices de $A(C)$. No entanto, se escolhermos os vértices b_{4i}, o último vértice escolhido será $b_{4x} = b_t$, que estará na solução e além de todos os vértices de $A(C)$ serem cobertos, por b_t ser um vértice de corte, ele pode cobrir mais vértices de $A(G)$ de outros ciclos. Logo, se t for múltiplo de 4, é sempre melhor escolher para a solução os vértices de $B(C)$ alternadamente iniciando de b_t. Se t não for múltiplo de 4, então existe um número inteiro y tal que $t = 4y - 2$. Assim, $|A(C)| = 2y - 1$ e $|A_q| = 2y - 2$. Logo, serão necessários $\frac{2y-2}{2} = y - 1$ vértices de $B(C)$ para cobrir A_q. Seja $j \in \{1, 2, \ldots, y - 1\}$. Nesse caso, se tentarmos incluir $b_t = b_{4y-2}$ na solução, teremos que colocar os vértices b_{4j+2}. No entanto, nesse caso, o primeiro vértice escolhido seria b_6, que cobriria a_5 e b_t cobriria a_1, mas a_3 não seria coberto. Então é melhor escolhermos os vértices b_{4j}, pois garantimos que todos os vértices de A_q serão cobertos de forma ótima. Nesse caso, removemos do grafo todos os vértices de C, menos a_1 e b_t, isso porque a_1 ainda não foi coberto. Mas nos dois casos, diminuímos o grafo em pelo menos uma orelha (lembrando sempre de diminuir o k quando incluirmos um vértice na solução). Se repetirmos o processo, encontraremos a solução ótima. ■

Capítulo 7

Parametrização de CCV

Como o problema CCV é NP-Completo para muitas classes de grafos, buscou-se alternativas para resolução usando as técnicas de parametrização e aproximação (esta última, mostrada no Capítulo 8). Neste Capítulo são mostrados os resultados obtidos sobre parametrização para o problema CCV. Como resultado negativo, mostramos que o problema é $W[2]$-Completo se parametrizado pelo parâmetro natural. Como resultados positivos, mostramos que o problema é FPT para grafos planares se parametrizado pelo parâmetro natural k e para grafos gerais se parametrizado pela largura arbórea (*treewidth*) ou pela combinação de dois parâmetros: k e grau máximo do grafo.

7.1 Complexidade Parametrizada

A parametrização é uma técnica que pode ser muito eficiente na resolução de problemas NP-Difíceis. Algoritmos de complexidade parametrizada são utilizados para resolver tais problemas de forma a obter a resposta exata e que, em geral, são mais eficientes do que utilizar uma força bruta, pois a exponencialidade na complexidade de tempo de um algoritmo parametrizado é em função de um parâmetro k definido previamente, menor que o tamanho da entrada n, consequentemente, melhor que utilizar um algoritmo exponencial em função de n [Cygan et al., 2015].

No Capítulo 2 são definidos alguns conceitos importantes sobre parametrização, como classes e redução de problemas parametrizados e classe FPT. Basicamente, algoritmos com complexidade de tempo da forma $f(k) \cdot n^c$, sendo c uma constante qualquer e f uma função computável, são denominados *fixed-parameter algorithms* — traduzindo para o português, algoritmos de parâmetro fixo —, ou algoritmos FPT. Em um algoritmo FPT, como c é uma constante, a parte da complexidade que depende de n é

polinomial, assim a exponencialidade da complexidade se dá apenas na função f que depende apenas do parâmetro k.

Diferentemente da classe de problemas NP-Completos, onde qualquer problema da classe é considerado tão difícil quanto o outro, as classes de complexidade de algoritmos parametrizados são divididas em uma hierarquia W de níveis, onde, $FPT = W[0] \subseteq W[1] \subseteq W[2] \subseteq ... \subseteq W[P] \subseteq XP$. Downey & Fellows [2013] e Cygan et al. [2015] explicam de forma mais detalhada essa hierarquia mas, basicamente, ela se baseia na menor cadeia de portas lógicas caras em um circuito necessárias para verificar uma solução, isto é, os problemas na classe $W[i]$ necessitariam de i portas caras para verificar uma solução.

Semelhantemente as reduções de provas de NP-Completude, fazer uma redução FPT de P para P' significa transformar uma instância (n, k) de P em uma instância (n', k') de P' em tempo FPT e garantir que $k' \leq g(k)$ para alguma função computável g. Além disso, P' tem resposta SIM se e somente se P tem resposta SIM [Cygan et al., 2015].

Para o problema CCV podemos pensar em diversas parametrizações: pelo parâmetro natural k, pelo grau máximo do grafo, por largura arbórea, entre outros. Também podemos pensar em parametrizações em classes de grafos distintas. Na Seção 7.2 serão mostradas três parametrizações: uma para a largura arbórea, em grafos gerais, outra para a combinação de dois parâmetros (grau máximo e o parâmetro natural), em grafos gerais, e a última para o parâmetro natural em grafos planares.

Nesta Seção, como resultados negativos, temos que se realizarmos uma parametrização pelo grau máximo do grafo, não obteremos resultados positivos, uma vez que, como mostrado no Teorema 5, mesmo que o grau máximo seja 3 o problema é NP-Completo. Também como resultado negativo, o problema CCV parametrizado pelo parâmetro natural k é $W[2]$-Completo. Isso pode ser comprovado fazendo uma redução a partir do problema *Red-Blue Dominating Set* que, por sua vez, é $W[2]$-Completo [Garnero et al., 2017]. Isso é claro pois RBDS é $W[2]$-Completo e é um caso particular de CCV, logo CCV que é uma generalização de RBDS também será $W[2]$-Completo.

7.2 Algoritmos de Complexidade Parametrizada

Nesta Seção, mostraremos alguns algoritmos FPT para o problema CCV. Foram escolhidas três parametrizações: grau máximo do grafo e parâmetro natural em grafos gerais, largura arbórea em grafos gerais e parâmetro natural em grafos planares.

Iniciamos com a parametrização por grau máximo e parâmetro natural. É inte-

ressante pensar nessa parametrização, uma vez que não é conveniente parametrizar por cada um desses fatores individualmente (NP-Completo para grafos com grau máximo 3 e $W[2]$-Completo para parâmetro natural). Para esta demonstração, encontraremos um *kernel* de tamanho FPT.

Um algoritmo de kernelização para um problema parametrizado qualquer é um algoritmo polinomial que dada uma instância $\{I, k\}$ do problema retorna uma instância equivalente $\{I', k'\}$ onde $|I'| \leq f(k)$ e $k' \leq k$, para alguma função f computável [Downey & Fellows, 2013; Cygan et al., 2015]. Dizemos então que $\{I', k'\}$ é um *kernel* para o problema. Um fator importante é que um problema parametrizado é FPT se e somente se admite um *kernel* [Downey & Fellows, 2013]. Mostraremos então um *kernel* no Teorema 17 e com ele, derivaremos o Corolário 18.

Teorema 17 *CCV, se parametrizado pelo grau máximo do grafo d e pelo parâmetro natural k, admite um kernel com no máximo $2^{kd} + kd$ vértices e no máximo $kd \cdot 2^{kd}$ arestas.*

Demonstração Seja $\{G, A \text{ e } B, k\}$ uma instância de CCV. Seja $d = \Delta(G)$ o grau máximo de G. Primeiramente, aplicamos o processo de simplificação em G, obtendo um novo grafo G_s, que é basicamente uma instância de *Red-Blue Dominating Set*. Essas reduções são válidas, como dito no Capítulo 3. Note que remover as arestas entre os vértices de $B(G)$ não aumenta o grau de nenhum vértice, já que estamos apenas removendo arestas. Note também que, no entanto, a contração das arestas entre os vértice de $A(G)$ pode aumentar o grau dos vértices dessa partição. Queremos cobrir então todos os vértices de $A(G_s)$ com vértices de $B(G_s)$. O grau máximo dos vértices de $B(G_s)$ é d, assim, um vértice $b \in B(G_s)$ cobre no máximo d vértices de $A(G_s)$. Se podemos usar no máximo k vértices de $B(G_s)$, é possível cobrir no máximo kd vértices de $A(G_s)$. Assim, se $|A(G_s)| > kd$, a resposta para o problema é automaticamente Não. A próxima regra de redução é a seguinte: sejam os vértices $b_i, b_j \in B(G_s)$ e os conjuntos $A_i, A_j \subseteq A(G_s)$, onde $A_i \supset A_j$. Se b_j pode cobrir A_j e b_i pode cobrir A_i, em uma solução que contém o vértice b_j, podemos trocá-lo pelo vértice b_i. Isto é, se dois vértices de $B(G_s)$ podem cobrir os mesmos vértices de $A(G_s)$ podemos remover um deles do grafo (o que cobre menos vértices) e ficar com o outro. Com essa regra de redução, garantimos que todos os vértices de $B(G_s)$ podem cobrir um conjunto diferente de vértices de $A(G_s)$. Como cada vértice de $B(G_s)$ pode cobrir no máximo d vértices de $A(G_s)$ que por sua vez tem no máximo kd vértices, o número de subconjuntos distintos de $A(G_s)$ é limitado por $\sum_{i=1}^{d} \binom{kd}{i} \leq 2^{kd}$. Se esse é o número máximo de subconjuntos diferentes, e eliminamos os vértices que cobriam conjuntos que não fossem diferentes,

temos no máximo 2^{kd} vértices em $B(G_s)$. Assim o número de vértices em G_s é no máximo $2^{kd} + kd$. Como vértices de uma mesma parte da partição não têm arestas entre si, o número máximo de arestas é limitado por $kd \cdot 2^{kd}$. Logo, temos um *kernel* com no máximo $2^{kd} + kd$ vértices e no máximo $kd \cdot 2^{kd}$ arestas. Como ainda poderemos utilizar k vértices de $B(G_s)$ na solução, fazemos $k' = k$. ■

Encontrando o *kernel* como especificado no Teorema acima, um algoritmo de força bruta simples que enumera todos os subconjuntos de vértices de $B(G')$ com complexidade $O(2^{2^{kd}} \cdot n^{O(1)})$ resolve o problema em tempo FPT. Com isso, podemos derivar o seguinte Corolário:

Corolário 18 *CCV é FPT, se parametrizado pelo grau máximo do grafo e pelo parâmetro natural k.*

Muitos problemas NP-Completos em grafos gerais, podem ser resolvidos em tempo polinomial quando o grafo de entrada é uma árvore, inclusive CCV, como visto no Teorema 12. É interessante pensar que, para o problema em grafos gerais, se o grafo de entrada "parecer" uma árvore, ele pode ser resolvido eficientemente. Precisamos então de uma unidade de medida para saber o quão um grafo é parecido à uma árvore. A medida que utilizamos é a *treewidth*, ou traduzindo ao português, largura arbórea [Cygan et al., 2015].

Medimos a largura arbórea de um grafo através de uma decomposição em árvore. Uma decomposição em árvore de um grafo G é uma árvore $\mathcal{T}$ e uma família T_x de subconjuntos de $V(G)$. Chamamos esse subconjuntos de *bags*. Cada subconjunto está associado a um vértice x de $\mathcal{T}$. A união de todos os conjuntos de T_x é igual a $V(G)$, ou seja, todo vértice do grafo está em pelo menos uma *bag* e não necessariamente um vértice estará em uma única *bag*. Para cada $(u, v) \in E(G)$ deve existir algum x tal que $\{u, v\} \in T_x$, ou seja, para cada par de vértices adjacentes, esse par deve estar junto em pelo menos uma *bag*. Por fim, sejam $x, y, z \in V(\mathcal{T})$ e y está no único caminho entre x e z, então $T_x \cap T_z \subseteq T_y$, ou seja, se um vértice de G está em duas *bags* T_x e T_z, ele também tem que estar em todas as *bags* entre T_x e T_z. A largura arbórea de uma decomposição é o máximo valor de $|T_x| - 1$, ou seja, uma unidade a menos que o tamanho da maior *bag*. A largura arbórea de uma grafo é a menor largura de todas as suas possíveis decomposições em árvore [Downey & Fellows, 2013].

Teorema 19 *CCV é FPT, se parametrizado pela largura arbórea.*

Demonstração Primeiramente, reduzimos nossa instância $\{G, A \text{ e } B, k\}$ de CCV a uma instância de *Red-Blue Dominating Set*. Como vimos no Capítulo 3, no RBDS, o

grafo tem seus vértices particionados em vermelhos e azuis e queremos cobrir os vértices vermelhos com um número mínimo de vértices azuis. Para fazer essa redução, basta aplicar o processo de simplificação em G. Como já visto, essa operação cria um grafo bipartido, onde só há arestas entre vértices de partes diferentes da partição. Se usarmos agora este grafo como instância de RBDS podemos dizer que os vértices de $A(G)$ são vermelhos e os de $B(G)$ são azuis e queremos cobrir os vértices de $A(G)$ com os de $B(G)$. Claramente os problemas são equivalentes, e essas reduções, como já provadas, podem ser feitas em tempo polinomial. Em seu trabalho, Alber & Niedermeier [2002] mostraram um algoritmo que resolve RBDS em tempo $O(3^{tw}s)$, sendo tw a largura arbórea do grafo de entrada e s o número de *bags* da decomposição em árvore. O número de *bags* é no máximo o número de vértices, então, é limitado por n. Se transformarmos nosso grafo de CCV em uma instância de RBDS e utilizarmos o algoritmo de Alber & Niedermeier [2002], teremos um algoritmo para CCV em tempo $O(3^{tw} \cdot n^{O(1)})$, que é FPT. ■

Em grafos gerais, não há uma parametrização FPT pelo parâmetro natural k, como já citado anteriormente. No entanto, podemos pensar nessa parametrização para alguma classe de grafos. Mostramos no Teorema abaixo, que CCV é FPT parametrizado por k em grafos planares.

Teorema 20 *CCV em grafos planares, parametrizado pelo parâmetro natural k, admite um kernel linear, com no máximo $43k$ vértices.*

Demonstração Seja $\{G, A \text{ e } B, k\}$ uma instância de CCV e G é um grafo planar. Se removermos as arestas entre os vértices de $B(G)$ e contrairmos as arestas entre os vértices de $A(G)$, teremos um novo grafo G', que também é planar e é instância de *Red-Blue Dominating Set*. Note que $|V(G)| \geq |V(G')|$. Em Garnero et al. [2017] é mostrado como que *Red-Blue Dominating Set* admite um *kernel* com no máximo $43k$ vértices. Assim, se aplicarmos as regras de redução de Garnero et al. [2017] em G', teremos $|V(G')| \leq 43k$, que é um *kernel* linear. ■

Corolário 21 *CCV em grafos planares é FPT, se parametrizado pelo parâmetro natural k.*

Capítulo 8

Aproximação de CCV

Muitas vezes não é viável resolver um problema de forma a obter a solução ótima pois o mesmo pode demorar tempo exponencial em função do tamanho da entrada, como é o caso dos problemas NP-Completos. Nesses casos, pode ser interessante dispensar a otimalidade da solução em troca de velocidade em tempo de execução. No entanto, podemos querer garantir o quão pior a solução gerada é em relação a ótima. Para isso, podemos utilizar uma técnica chamada de aproximação. Neste Capítulo, mostraremos como resultado negativo um resultado de inaproximabilidade de CCV. Como resultados positivos, mostramos duas aproximações para grafos gerais e uma para *Unit Disk Graphs*.

Um algoritmo aproximativo é executado, geralmente, em tempo polinomial e gera uma solução S que pode não ser exatamente a solução ótima S^*, mas podemos garantir que ela está a um fator de aproximação α da solução ótima, isto é, S^* é no máximo α vezes melhor que S. Assim, se temos um problema de maximização e um algoritmo com fator de aproximação α nos fornece uma solução S, podemos garantir que $S \geq \frac{S^*}{\alpha}$. De forma similar, se o problema for de minimização, garantimos que $S \leq \alpha S^*$.

Assim como os problemas parametrizados, os problemas também são classificados em relação ao fator de aproximação dos mesmos. Vazirani [2013] e Crescenzi & Panconesi [1991] definem algumas dessas classes da seguinte forma:

- **APX:** problemas com fator de aproximação constante, isto é, $\alpha = O(1)$ — ex: *Vertex Cover*.
- **PTAS:** seja Π um problema de maximização, OPT o valor da solução ótima para Π, e $\epsilon > 0$ um parâmetro dado como entrada, $\mathcal{A}$ é um esquema de aproximação para Π se ele resolve o problema e gera uma solução $S \geq (1 - \epsilon)OPT$. Se o problema for de minimização, $\mathcal{A}$ é um esquema de aproximação se $S \leq (1 +$

$\epsilon)OPT$. Assim, dizemos que $\mathcal{A}$ é um *Polynomial-Time Approximation Scheme* — em tradução livre, esquema de aproximação em tempo polinomial —, ou PTAS, se é um esquema de aproximação e seu tempo de execução é limitado por um polinômio em função do tamanho da entrada de Π.

- **FPTAS**: um *Fully Polynomial-Time Approximation Scheme* — em tradução livre, esquema de aproximação em tempo completamente polinomial —, ou FPTAS, é um PTAS em que seu tempo de execução também é limitado por um polinômio em função de $1/\epsilon$.

- **NPO-Completo:** não possui aproximação com fator constante. Dentre esses temos duas subclasses: a classe $\log n$ que tem fator de aproximação $\alpha = O(\log n)$ — como o *Set Cover* —, e a classe n que tem fator de aproximação $\alpha = n$ — exemplo: Clique.

8.1 Inaproximabilidade de CCV

Nesta Seção, mostramos um resultado negativo de inaproximabilidade para CCV, e para isso, usaremos o resultado encontrado por Feige [1998] para o problema *Set Cover*. Primeiramente, mostramos a equivalência do problema CCV com o problema *Set Cover*, mostrando que um problema é redutível ao outro.

O problema *Set Cover*, ou simplesmente SC, definido no Capítulo 5, é muito conhecido na literatura e há muitos estudos e resultados para o mesmo. Esses resultados podem ser utilizados para o problema CCV, uma vez que CCV e SC são dois problemas com uma relação equivalência. Dizer que CCV e SC tem uma relação de equivalência é dizer que podemos reduzir um problema ao outro, isto é, podemos reduzir CCV a SC e vice-versa. Essa equivalência é mostrada nos Lemas 22 e 23.

Lema 22 *Set Cover é redutível a CCV.*

Demonstração A redução do SC para o CCV é idêntica a redução utilizada anteriormente a partir do 3-SC mostrado no Teorema 3, só não podemos afirmar nada sobre o grau máximo do grafo gerado, pois ele será igual ao tamanho do maior conjunto em $\mathcal{Y}$. Partindo do Teorema 3, é fácil concluir que podemos usar a mesma transformação e os mesmos argumentos para provar que a redução é válida, isto é, SC tem resposta SIM se e somente se CCV tem resposta SIM. Logo, SC é redutível a CCV. ■

Lema 23 *CCV é redutível a Set Cover.*

Demonstração Criamos uma instância $\{X, \mathcal{Y}, k^{SC}\}$ de SC a partir de uma instância genérica de CCV: seja $\{G, A \text{ e } B, k^{CCV}\}$ uma instância genérica de CCV. Aplicamos o procedimento de simplificação em G obtendo o grafo G_s. Com base no Lema 1 e na observação de que remover arestas de B não afeta o resultado, uma solução para o grafo G_s também é uma solução para o grafo G. Cada vértice $a_j \in A(G_s)$ representa uma componente conexa de $A(G)$, logo, só possui arestas para vértices de $B(G_s)$, porque se existissem dois vértices $a_1, a_2 \in A(G_s)$ tal que a aresta (a_1, a_2) existisse em G_s, essa aresta poderia ser contraída. Logo, como não há arestas entre vértices de $A(G_s)$ e removemos também todas arestas de $B(G_s)$, esse grafo é bipartido. Assim, construímos a instância de SC da seguinte forma: para cada $a_j \in A(G_s)$ cria-se um elemento x_j correspondente que é inserido em X. Para cada vértice $b_i \in B(G_s)$, sejam $N(b_i) = \{a_1^i, a_2^i, ..., a_l^i\}$ os vizinhos de b_i, cria-se um conjunto $Y_i = \{x_1^i, x_2^i, ..., x_l^i\}$, onde cada x_j^i é o elemento correspondente ao vértice a_j^i, e inserimos Y_i em $\mathcal{Y}$. Isso significa que se $x_j \in Y_i$ então o vértice a_j correspondente a x_j tem aresta para o vértice b_i correspondente a Y_i, o que implica também que b_i pode cobrir a componente conexa a qual a_j representa. Note que o tamanho do maior conjunto em $\mathcal{Y}$ é igual ao maior grau entre os vértice de $B(G_s)$. Por fim, fazemos $k^{SC} = k^{CCV}$.

CCV tem resposta Sim se e somente se SC tem resposta Sim:

($\Rightarrow$) Se CCV tem resposta Sim então SC tem resposta Sim: se CCV tem resposta Sim então existe um conjunto $B' \subseteq B(G)$ tal que B' cobre todas as componentes conexas de $A(G)$ e $|B'| \leq k^{CCV}$. Obtemos um subconjunto $\mathcal{Y}' \subseteq \mathcal{Y}$ onde os conjuntos em $\mathcal{Y}'$ correspondem aos vértices de B'. Por construção, cada componente conexa de $A(G)$ representa um elemento em X, pois contraímos as arestas em $A(G)$, transformando cada componente em um vértice, e para cada um desses criamos um elemento em X. Como B' cobre todas as componentes conexas de $A(G)$, por construção, cada elemento de X está em pelo menos um conjunto de $\mathcal{Y}'$, ou seja, $\mathcal{Y}'$ cobre todos os elementos de X. Como $|\mathcal{Y}'| = |B'| \leq k^{CCV} = k^{SC}$ então $|\mathcal{Y}'| \leq k^{SC}$.

($\Leftarrow$) Se SC tem resposta Sim então CCV tem resposta Sim: se SC tem resposta Sim então existe um conjunto $\mathcal{Y}' \subseteq \mathcal{Y}$ tal que $\mathcal{Y}'$ cobre todos os elementos de X e $|\mathcal{Y}'| \leq k^{SC}$. Obtemos um subconjunto $B' \subseteq B(G)$ onde os vértices de B' correspondem aos conjuntos em $\mathcal{Y}'$. Por construção, cada elemento em X corresponde a uma componente conexa de $A(G)$. Como cada elemento de X aparece pelo menos em algum conjunto de $\mathcal{Y}'$, cada componente conexa de $A(G)$ é coberta por pelo menos um vértice de B'. Como $|B'| = |\mathcal{Y}'| \leq k^{SC} = k^{CCV}$ então $|B'| \leq k^{CCV}$. ■

Seja $\{G, A \text{ e } B\}$ uma instância da versão de otimização de CCV e $C(G)$ o conjunto de componentes conexas no grafo induzido pelos vértices de $A(G)$ — ou seja, $C(G)$

é o conjunto de componentes conexas que queremos cobrir —, existe um algoritmo de aproximação para o problema CCV com um fator de aproximação $O(\log |C(G)|)$. Falaremos desse algoritmo na Seção 8.2. Mostraremos no Teorema a seguir que esse é o melhor fator de aproximação possível (em função do número de componentes):

Teorema 24 *Assumindo que $P \neq NP$, CCV é inaproximável para um fator de aproximação melhor que $O(\log |C(G)|)$.*

Demonstração Para provar, usaremos o fato de que *Set Cover* é inaproximável para fator de aproximação melhor que $O(\log n)$ [Feige, 1998]. Por contradição, suponha que existe um algoritmo $\mathcal{A}$ que resolve o problema CCV com fator de aproximação $\beta = o(\log |C(G)|)$ — estritamente melhor que $O(\log |C(G)|)$. Dada uma instância de SC, se a reduzimos a uma instância de CCV como mostrado no Lema 22, teremos um grafo G na qual $|C(G)| = |X| = n$. Se resolvemos o problema CCV para essa instância utilizando o algoritmo $\mathcal{A}$, teremos uma solução com fator de aproximação β. Se transformarmos a solução de CCV na solução do problema original SC teremos uma resposta com fator de aproximação β que é melhor que $O(\log n)$ o que é uma contradição, pois Feige [1998] mostra que isso não é possível a menos que P = NP. Logo, CCV é inaproximável para um fator de aproximação melhor que $O(\log |C(G)|)$. ■

8.2 Algoritmos Aproximativos para CCV

Em seu trabalho, Ferreira Neto et al. [2017] mostrou um algoritmo com fator de aproximação igual a 2 para o problema CCV quando o grau máximo do grafo é igual a 4. Nesta Seção serão mostrados outros algoritmos aproximativos para CCV. Mostraremos duas aproximações para grafos gerais: uma baseada na frequência em que uma componente é coberta (ou o "grau" de uma componente) e outro baseado no número de componentes do grafo. Também mostraremos uma aproximação com fator constante para o problema em UDGs.

Seja $\{G, A \text{ e } B\}$ uma instância da versão de otimização de CCV, G_s o grafo resultante do procedimento de simplificação aplicado em G e Δ_A o maior grau entre os vértices de $A(G_s)$, a primeira aproximação a ser mostrada tem fator igual a Δ_A. Para isso reduziremos o nosso problema ao problema SC, como mostrado no Lema 23. Nessa transformação, primeiramente é feita uma contração de arestas entre os vértices de $A(G)$, o que nos garante que todos os vizinhos de um vértice $a_j \in A(G_s)$ estão em $B(G_s)$. Para cada vértice $a_j \in A(G_s)$ é criado um elemento x_j e para cada vértice

$b_i \in B(G_s)$ é criado um conjunto Y_i que contém os elementos referentes aos vértices de $A(G_s)$ que são vizinhos de b_i. Assim, se um vértice a_j tem f vizinhos, x_j aparecerá em exatamente f conjuntos, ou seja, a frequência de um elemento nos conjuntos de $\mathcal{Y}$ é o grau de seu vértice correspondente em G_s. Assim, se Δ_A é o maior grau entre os vértices de $A(G_s)$ e f é a frequência do elemento mais frequente nos conjuntos de $\mathcal{Y}$, então $f = \Delta_A$. Vazirani [2013] mostra um algoritmo chamado de **algoritmo de camadas** com fator de aproximação igual a f, assim se reduzirmos nosso problema ao *Set Cover*, resolvermos usando o algoritmo de camadas e transformarmos a solução que tem fator de aproximação f na solução de CCV então teremos uma solução com fator de aproximação igual a Δ_A.

A outra aproximação a ser mostrada funciona de forma similar, utilizando redução ao problema *Set Cover*. Seja $\{G, A \text{ e } B\}$ uma instância da versão de otimização de CCV e $C(G)$ o conjunto de componentes conexas no grafo induzido pelos vértices de $A(G)$ — ou seja, $C(G)$ é o conjunto de componentes conexas que queremos cobrir —, no Lema 23, para cada $c_j \in C(G)$ é criado um elemento que é adicionado em X — pois o grafo é comprimido e cada c_j se torna apenas um vértice a_j que a partir deste se cria um elemento x_j correspondente —, assim, $|X| = |C(G)|$. Seja $n = |X|$, Vazirani [2013] mostra um algoritmo guloso que resolve o problema *Set Cover* com fator de aproximação $O(\log n)$. Assim, se fizermos uma redução da nossa entrada para o SC, resolvermos o problema usando o algoritmo guloso e transformarmos a solução que tem fator de aproximação $O(\log n)$ na solução de CCV, como $|C(G)| = |X| = n$, o fator de aproximação dessa solução será $O(\log |C(G)|)$.

Essa última aproximação é o melhor que conseguimos, se nos basearmos no número de componentes conexas, como mostrado na Seção 8.1. No entanto, podemos pensar em aproximações para determinadas classes de grafos. Os UDGs são grafos com várias restrições. Dentre elas, temos o fato de serem geométricos, e isso nos ajudará a provar o Teorema abaixo:

Teorema 25 *Existe um algoritmo com fator de aproximação 1,79 para CCV em Unit Disk Graphs.*

Demonstração Utilizaremos o fato de que, dada uma instância $\{G, A \text{ e } B\}$ da versão de otimização de CCV e G é um UDG, após contrair as arestas de $A(G)$ teremos um grafo $G' = G/A$ em que para todo vértice $b \in B(G')$, o número de vizinhos de b que pertencem a $A(G')$ é no máximo 5. Isso é verdade pois o grafo induzido por esses vértices é um conjunto independente, o que é claro pois se dois deles tivessem arestas, ela poderia ser contraída e um deles deixaria de existir. Por fim, Wu et al. [2006]

mostra que em um UDG a vizinhança de um vértice contém um conjunto independente de tamanho no máximo 5.

Sabendo que em G' os vértices de $B(G')$ tem no máximo 5 vizinhos que estão em $A(G')$, reduziremos o problema CCV em UDGs ao problema *d-Set Cover*. Pelo Lema 23, como cada vértice $b_i \in B(G')$ será um conjunto de Y_i, e os elementos de Y_i serão referentes aos vizinhos de b_i que estão em $A(G')$, que são no máximo 5, então o maior conjunto em $\mathcal{Y}$ terá tamanho igual a 5. Assim, utilizaremos a redução do Lema 23 e obteremos uma instância de 5-*Set Cover*. A partir disso, podemos utilizar o algoritmo desenvolvido por Levin [2008], que tem um fator de aproximação igual a $H_d - \frac{196}{390}$ para o problema *d-Set Cover* — sendo H_d o d-ésimo número harmônico, isto é, $H_d = \sum_{i=1}^{d} \frac{1}{i}$. Como nesse caso $k = 5$, teremos um fator de aproximação $\approx 1,7807 < 1,79$. Logo, temos um algoritmo com fator de aproximação igual a $1,79$. ■

Vale aqui, fazer uma relação com o problema OWN-BC de Ferreira Neto et al. [2017]. Em seu trabalho, é apresentado um algoritmo aproximativo com fator 2 para seu problema. Como OWN-BC é um caso particular de CCV, seu algoritmo poderia ser utilizado para resolver o problema em UDGs, obtendo um fato de aproximação igual a 3. Apesar de um fator de aproximação igual a 3 não ser ruim, pudemos melhorá-lo com o algoritmo de Levin [2008], e obter fator $1,79$. Em contrapartida, o algoritmo de Levin [2008] poderia ser utilizado no problema OWN-BC e seria obtido um fator de aproximação $\approx 1,58$. É importante ressaltar que o algoritmo de Ferreira Neto et al. [2017], apesar de ter um fator de aproximação pior, é mais simples de implementar e mais eficiente em termos de tempo de execução.

Capítulo 9

Conclusões

Pode-se concluir que o problema CCV é bastante complexo, uma vez que é NP-Completo para grafos muito específicos. Provamos que o problema permanece NP-Completo para classes de grafos muito estudadas como cordais, bipartidos, *splits* e planares. Provamos que mesmo para grafos muito restritos como bipartidos de máximo grau 3, grades, ou mesmo UDGs planares, o problema continua NP-Completo.

Com relação a algoritmos de complexidade parametrizada, pode-se ver que CCV também é consideravelmente difícil. A relação com o *Set Cover* e com o *Red-Blue Dominating Set* nos ajudou a provar que o problema em grafos gerais, se parametrizado pelo parâmetro natural k, é $W[2]$-Completo. Também mostramos que uma parametrização pelo grau máximo do grafo não seria eficiente, pois o problema é NP-Completo mesmo para grafos com grau máximo 3. Mostramos que podemos parametrizar o problema por uma combinação desses dois parâmetros (parâmetro natural k e grau máximo do grafo) e com isso conseguimos uma parametrização FPT. Mostramos também que é possível conseguir uma parametrização FPT se o parâmetro for a largura arbórea do grafo. Por fim, mostramos uma *kernel* linear, e consequentemente uma parametrização FPT, para o problema aplicado a grafos planares, utilizando o parâmetro natural k.

Sobre algoritmos aproximativos, também ficou provada a dificuldade do problema, pois mostramos um resultado de inaproximabilidade para um fator melhor do que o já conhecido. Também mostramos um algoritmo com esse fator, que é $O(\log |C(G)|)$, sendo $C(G)$ o número de componentes que queremos cobrir. Além disso, mostramos um algoritmo com fator de aproximação igual a número de vértices que poderiam cobrir as componentes, baseado em frequência. Por fim, mostramos um algoritmo com fator de aproximação $1,79$ para UDGs.

Como trabalhos futuros, temos como objetivo então, encontrar algoritmos aproximativos para mais classes de grafos nas quais o problema é NP-Completo, principal-

mente tentando encontrar algoritmos com fator de aproximação constante. Em relação a algoritmos parametrizados, temos também como objetivo buscar parametrizações para classes de grafos, tentando encontrar parametrizações de classes de problemas menores, como FPT ou $W[1]$ ou parametrizações por outros parâmetros. Também há a possibilidade de estudar o problema com algumas variações, por exemplo, o problema em que os vértices da solução precisam estar conectados, ou vértices de $A(G)$ podem ser incluídos em $B(G)$ e se tornarem parte da solução, ou até mesmo restrições de diâmetro para que um vértice consiga se conectar, entre outras restrições que possam ser relevantes. Por fim, pretendemos formular o problema como um problema de Programação Linear Inteira (PLI), fazendo sua modelagem e realizando experimentos para comparação das diferentes técnicas e algoritmos apresentados.

Referências Bibliográficas

Alber, J.; Bodlaender, H. L.; Fernau, H. & Niedermeier, R. (2000). Fixed parameter algorithms for p lanar d ominating s et and related problems. Em *Scandinavian Workshop on Algorithm Theory*, pp. 97--110. Springer.

Alber, J. & Niedermeier, R. (2002). Improved tree decomposition based algorithms for domination-like problems. Em *Latin american symposium on theoretical informatics*, pp. 613--627. Springer.

Aubry, Y.; Godin, J.-C. & Togni, O. (2016). Free choosability of outerplanar graphs. *Graphs and Combinatorics*, 32(3):851--859.

Bar-Yehuda, R. & Even, S. (1985). A local-ratio theorem for approximating the weighted vertex cover problem. *North-Holland Mathematics Studies*, 109:27--45.

Bondy, J. A.; Murty, U. S. R. et al. (1976). *Graph theory with applications*, volume 290. Citeseer.

Brandstadt, A.; Spinrad, J. P. et al. (1999). *Graph classes: a survey*, volume 3. Siam.

Chvátal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233--235.

Clark, B. N.; Colbourn, C. J. & Johnson, D. S. (1990). Unit disk graphs. *Discrete mathematics*, 86(1-3):165--177.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. & Stein, C. (2009). *Introduction to algorithms*. MIT press.

Corneil, D. G.; Lerchs, H. & Burlingham, L. S. (1981). Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163--174.

Corneil, D. G.; Perl, Y. & Stewart, L. K. (1985). A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926--934.

Cournier, A. & Habib, M. (1994). A new linear algorithm for modular decomposition. Em *Colloquium on Trees in Algebra and Programming*, pp. 68--84. Springer.

Crescenzi, P. & Panconesi, A. (1991). Completeness in approximation classes. *Information and Computation*, 93(2):241--262.

Cygan, M.; Fomin, F. V.; Kowalik, Ł.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M. & Saurabh, S. (2015). *Parameterized algorithms*, volume 3. Springer.

dos Santos, V. F. (2013). *Convexidades em grafos: intermediações, parâmetros e conversões*. Tese de doutorado, Universidade Federal do Rio de Janeiro.

Downey, R. G. & Fellows, M. R. (2013). *Fundamentals of parameterized complexity*, volume 4. Springer.

Downey, R. G.; Fellows, M. R.; Vardy, A. & Whittle, G. (1999). The parametrized complexity of some fundamental problems in coding theory. *SIAM Journal on Computing*, 29(2):545–26.

Duh, R.-c. & Fürer, M. (1997). Approximation of k-set cover by semi-local optimization. Em *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pp. 256--264. ACM.

Feige, U. (1998). A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634--652.

Ferreira Neto, M.; Goussevskaia, O. & dos Santos, V. F. (2017). Connectivity with backbone structures in obstructed wireless networks. *Computer Networks*, 127:266 – 281. ISSN 1389-1286.

Garey, M. R. & Johnson, D. S. (2002). *Computers and intractability*, volume 29. wh freeman New York.

Garnero, V.; Sau, I. & Thilikos, D. M. (2017). A linear kernel for planar red–blue dominating set. *Discrete Applied Mathematics*, 217:536--547.

Kazmierczak, A. & Radhakrishnan, S. (2000). An optimal distributed ear decomposition algorithm with applications to biconnectivity and outerplanarity testing. *IEEE Transactions on Parallel and Distributed Systems*, 11(2):110--118.

Levin, A. (2008). Approximating the unweighted k-set cover problem: greedy meets local search. *SIAM Journal on Discrete Mathematics*, 23(1):251--264.

Valiant, L. G. (1981). Universality considerations in vlsi circuits. *IEEE Transactions on Computers*, 100(2):135--140.

Vazirani, V. V. (2013). *Approximation algorithms*. Springer Science & Business Media.

Wolle, T. & Bodlaender, H. L. (2004). A note on edge contraction.

Wu, W.; Du, H.; Jia, X.; Li, Y. & Huang, S. C.-H. (2006). Minimum connected dominating sets and maximal independent sets in unit disk graphs. *Theoretical Computer Science*, 352(1-3):1--7.