Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Partições Conexas Balanceadas de Grafos

Arthur Correia Gomes

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisora: Profª. Drª. Yoshiko Wakabayashi

São Paulo
24 de Dezembro de 2021

Agradecimentos

Agradeço primeiramente aos meus pais, José Acacio Gomes e Maria José Fontes Correia Gomes e à minha irmã, Luísa Correia Gomes, que me apoiaram durante toda a minha trajetória, principalmente durante a elaboração deste trabalho.

Agradeço à professora Yoshiko, pela disciplina de Algoritmos de Aproximação, ministrada por ela com excelência, e por todas as orientações no desenvolvimento deste trabalho.

Resumo

Arthur Correia Gomes. **Partições Conexas Balanceadas de Grafos**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2021.

Estudamos neste trabalho dois problemas sobre q-partição conexa balanceada de um grafo conexo, onde q é um inteiro, $q \geq 2$. Em linhas gerais, esses problemas são definidos da seguinte forma: dado um grafo conexo $G = (V, E)$ e uma função peso $w : V \to \mathbb{Z}_+$, encontrar uma q-partição de V, de forma que cada classe da partição induza um subgrafo conexo de G, e essas classes tenham pesos os mais similares possíveis. O peso de uma classe é a soma dos pesos dos vértices pertencentes a essa classe. Para formalizar a ideia de balanceamento da partição, consideramos duas funções objetivos (que dão origem a dois problemas distintos): maximizar o peso da classe mais leve (MaxMin) ou minimizar o peso da classe mais pesada (MinMax). É sabido que esses dois problemas são NP-difíceis. Mencionamos diversos resultados existentes na literatura, e apresentamos alguns algoritmos de aproximação para esses problemas (para casos especiais de q). Por fim, mostramos um resultado de inaproximabilidade para um desses problemas.

Palavras-chave: algoritmos de aproximação. partição conexa balanceada. inaproximabilidade. partição conexa de grafos. NP-difícil.

Abstract

Arthur Correia Gomes. **Balanced Connected Partitions of Graphs**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2021.

In this project we study two problems on balanced connected q-partition of a connected graph, where q is an integer, $q \geq 2$. Broadly speaking, these problems are defined as follows: given a connected graph $G = (V, E)$ and a weight function $w : V \to \mathbb{Z}_+$, find a q-partition of V, such that each class induces a connected subgraph of G, and the weights of these classes are as balanced as possible. The weight of a class is the sum of the weights of the vertices in the class. More formally, we define two objective functions to capture the idea of balancedness of the classes (which give rise to two different problems): maximize the weight of the lightest class (MaxMin) or minimize the weight of the heaviest class (MinMax). It is known that these problem are NP-hard. We mention several results known in the literature, and present some approximation algorithms for these problems (for special values of q). Finally, we present an inapproximability result for one of these problems.

Keywords: approximation algorithms. balanced connected partition of graphs. graph partition into connected classes. NP-hard.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Problemas de partição conexa balanceada de grafos conexos possuem aplicações em diversas áreas, como processamento de imagens, bancos de dados, redes metabólicas, logística, sistemas operacionais, análise de *cluster*, educação e robótica (Ronald I. Becker, Schach *et al.* (1982), Lucertini *et al.* (1989), Lucertini *et al.* (1993), Maravalle *et al.* (1997), X. Zhou *et al.* (2019) e D. Matić e Grbić (2020)).

Por conta dessas diversas aplicações, e também pelo interesse teórico que esses problemas suscitam, principalmente em termos de aproximabilidade, esses problemas têm sido estudados desde a década de 80. Neste trabalho vamos nos aprofundar no problema da q-**Partição Conexa Balanceada**, denotado por BCP_q.

Ao longo deste texto, q denota sempre um inteiro positivo maior que 1. Definimos a seguir o conceito de q-partição de um conjunto qualquer e de q-partição conexa de um grafo conexo.

Definição 1. *Seja A um conjunto não vazio e $A_1, A_2, \ldots, A_q$ subconjuntos não vazios de A de forma que os conjuntos A_i e A_j são disjuntos dois a dois para $1 \leq i < j \leq q$ e $A_1 \cup A_2 \cup \cdots \cup A_q = A$. Dizemos então que $\mathcal{P} = \{A_1, A_2, \ldots, A_q\}$ é uma q-**partição** de A.*

Definição 2. *Se $G = (V, E)$ é um grafo conexo e $\mathcal{P} = \{V_1, V_2, \ldots, V_q\}$ é uma q-partição de V tal que para $i = 1, 2, \ldots, q$ o subgrafo de G induzido por V_i é conexo, então $\mathcal{P}$ é uma q-**partição conexa** de G.*

Com isso, podemos definir o problema BCP_q da seguinte forma:

Definição 3 (BCP_q)**.** *Dado um par (G, w), onde $G = (V, E)$ é um grafo conexo, e $w : V \to \mathbb{Z}_+$ é uma função peso definida sobre seus vértices, queremos encontrar uma q-partição conexa $\mathcal{P} = \{V_i\}_{i \in [q]}$ de V tal que os pesos dessas classes V_i sejam os mais balanceados possíveis. Denotamos por $w(V_i)$ o peso de uma classe V_i, definido como a soma dos pesos dos vértices em V_i.*

Na literatura, existem diversas maneiras de se definir balanceamento. Cada uma dessas formas origina uma nova versão do problema. Nosso foco neste trabalho será nas versões MaxMin e MinMax do BCP_q, definidas a seguir.

Max-Min BCP$_q$

Dado um par (G, w), onde $G = (V, E)$ é um grafo conexo, e $w : V \to \mathbb{Z}_+$, encontrar uma q-partição conexa $\{V_i\}_{i\in[q]}$ de G que maximiza o peso da classe mais leve, isto é maximiza $\min_{i\in[q]}\{w(V_i)\}$.

Min-Max BCP$_q$

Dado um par (G, w), onde $G = (V, E)$ é um grafo conexo, e $w : V \to \mathbb{Z}_+$, encontrar uma q-partição conexa $\{V_i\}_{i\in[q]}$ de G que minimiza o peso da classe mais pesada, isto é minimiza $\max_{i\in[q]}\{w(V_i)\}$.

Dentre algumas outras maneiras de se definir balanceamento, mencionamos as duas seguintes funções objetivo (veja Chlebíková (1996)):

$$\text{minimizar} \max_{i,j\in[q]} |w(V_i) - w(V_j)|,$$

$$\text{maximizar} \prod_{i=1}^{q} w(V_i).$$

Quando $q = 2$ as versões MinMax e MaxMin são equivalentes, no sentido de que encontram uma mesma bipartição ótima, embora para cada uma delas o valor da função objetivo possa ser diferente. Para $q \geq 3$, é fácil construir instâncias para as quais as soluções ótimas diferem substancialmente (Lucertini *et al.* (1993)).

Quando trabalhamos com grafos com pesos unitários, denotamos os problemas como 1-MinMax BCP$_q$ e 1-MaxMin BCP$_q$.

Quando q não é fixado a priori, e faz parte da instância do problema, denotamos o problema correspondente por MinMax BCP ou MaxMin BCP.

O objetivo deste trabalho é fazer um estudo sobre o BCP$_q$, focando nas versões MinMax e MaxMin. No Capítulo 2 apresentamos alguns conceitos básicos e a notação utilizada neste texto. Esses conceitos estão apresentados nas subseções: Teoria dos Grafos, Complexidade Computacional e Algoritmos de Aproximação.

No Capítulo 3, citamos diversos resultados existentes na literatura para esses problemas. Primeiro mencionamos resultados referentes à complexidade desses problemas. Depois mencionamos resultados existentes sobre algoritmos exatos para eles. Alguns desses algoritmos possuem formulações combinatórias, enquanto outros utilizam formulações baseadas em programação linear inteira e programação linear inteira mista.

No Capítulo 4, nos aprofundamos em um algoritmo de aproximação para o 1-MinMax BCP$_3$ e um para o 1-MinMax BCP$_q$, ambos desenvolvidos por Y. Chen *et al.* (2019). Ademais, mencionamos um resultado recente de Moura *et al.* (2021), que generaliza esses algoritmos para as versões com pesos dos problemas.

Já no Capítulo 5, apresentamos em detalhes uma $\frac{4}{3}$-aproximação para o MaxMin BCP_2, proposta por Chlebíková (1996). Primeiro é descrito um algoritmo com essa razão para grafos 2-conexos, e depois um algoritmo que utiliza o anterior como rotina para resolver o problema para grafos conexos gerais.

No Capítulo 6 apresentamos alguns resultados existentes na literatura, referentes a esses problemas e que tratam de aproximabilidade. Finalmente, no Capítulo 7, apresentamos algumas considerações finais.

Capítulo 2

Preliminares

Neste capítulo apresentamos os conceitos e a notação que serão utilizados ao longo do texto. Ele está dividido em três partes: Seção 2.1 – Teoria dos Grafos; Seção 3.1 – Complexidade Computacional; Seção 2.3 – Algoritmos de Aproximação.

2.1 Teoria dos Grafos

Um **grafo** G é um par ordenado (V, E), onde V é um conjunto finito não vazio, e E é um conjunto formado por pares não ordenados de elementos distintos de V. Os elementos do conjunto V são chamados **vértices** e os elementos de E são chamados **arestas**.

Quando nos referimos a um grafo G, também denotamos o seu conjunto de vértices por $V(G)$ e conjunto de arestas por $E(G)$. Comumente definimos $n := |V|$ e $m := |E|$. Dizemos que n é a ordem do grafo G.

Seja e uma aresta definida pelos vértices $\{u, v\}$. Então denotamos e por $\{u, v\}$ ou simplesmente por uv. Dizemos que u e v são os **extremos** de e. Dois vértices u e v são **adjacentes** se existe uma aresta com extremos em u e v. Se cada vértice é adjacentes a todos os outros dizemos que o grafo é **completo**. Denotamos um grafo completo com n vértices por K_n. Quando G é completo, o número de arestas desse grafo é $m = \frac{1}{2}n(n-1)$. O tamanho de um grafo é o número $\langle G \rangle = |V| + |E|$.

Dizemos que um grafo é trivial se ele possui apenas um vértice.

Um grafo é **bipartido** se seu conjunto de vértices V pode ser dividido em dois conjuntos X e Y de forma que $V = X \cup Y$, $X \cap Y = \emptyset$ e todas as arestas do grafo possuem um extremo em X e outro em Y.

Um grafo $H = (V(H), E(H))$ é um **subgrafo** de $G = (V(G), E(G))$ se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$. Dizemos que H é um subgrafo **gerador** de G se $V(H) = V(G)$.

Dado um conjunto qualquer $X \subseteq V(G)$, o subgrafo de G **induzido** por X, denotado por $G[X]$, é o subgrafo formado por todos os vértices de X com conjunto

de arestas formado por todas as arestas que possuem ambos os extremos nos vértices de X, ou seja, o conjunto $\{uv \in E(G) | u, v \in X\}$.

Denotamos por $G - X$ o subgrafo obtido pela remoção de todos os vértices de X e de todas as arestas que possuem algum extremo em X. Quando X é formado por apenas um vértice v, denotamos esse subgrafo por $G - v$.

Dado um conjunto $X \in V$ de um grafo G, denotamos por $\delta(X)$ o conjunto de todas as arestas que possuem um extremo em X e outro em $V \setminus X$. Um **corte** é qualquer conjunto $\delta(X)$. Denotamos por $E(V_i, V_j)$ o conjunto de todas as arestas com um extremo em V_i e outro em V_j, para dois conjuntos V_i, V_j tal que $V_i \cap V_j = \emptyset$.

Um **passeio** é uma sequência $(v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k)$ em que $v_0, \ldots, v_k$ são vértices e $e_1, \ldots, e_k$ são arestas de forma que e_i tenha extremos em v_{i-1} e v_i. Dizemos que v_0 é a origem do passeio e v_k é o término. Quando a origem é igual ao término dizemos que o passeio é **fechado**, caso contrário é **aberto**. Muitas vezes denotamos um caminho apenas pelos seus vértices.

Um passeio que não possui arestas repetidas é chamado de **trilha**. E um passeio sem nenhum vértice repetido é chamado de **caminho**. Denotamos por P_n um grafo de n vértices formado apenas por um caminho com n vértices. Um **circuito** é um passeio fechado, no qual apenas os vértices iniciais e finais são iguais.

Um grafo é **conexo** se existe um caminho entre quaisquer dois de seus vértices. Dizemos que um grafo é **desconexo** se não é conexo. Cada subgrafo conexo maximal de um grafo é chamado de **componente**.

Uma **floresta** é um grafo que não contém circuitos. Uma **árvore** é uma floresta conexa. Para qualquer par de vértices x, y em uma árvore, existe apenas um caminho de x a y.

Seja $G = (V, E)$ um grafo conexo e k um número natural. Dizemos que G é **k-conexo** se $G - X$ é conexo para todo conjunto de vértices X tal que $|X| < k$. A **conexidade** de G é o menor inteiro positivo k tal que G é k-conexo.

Chamamos um vértice x de G um **vértice de corte**, se o número de componentes de $G - x$ é maior que o número de componentes de G. Também dizemos que x é um **ponto de articulação** de G.

Os subgrafos maximais de G que são 2-conexos ou que são isomorfos a K_2 são chamados **blocos** de G. Seja A o conjunto de pontos de articulação de G e seja B o conjunto de blocos. Definimos o **grafo de blocos** de G como o grafo bipartido com conjunto de vértices $A \cup B$ e conjunto de arestas ab em que $a \in A$ e $b \in B$.

2.2 Complexidade Computacional

Seja I uma instância de um problema. Denotamos por $\langle I \rangle$ o **tamanho da instância**.

Chamamos um problema cuja resposta deve ser SIM ou NÃO de **problema de decisão**. Similarmente, quando a resposta de um problema deve ser a melhor

possível dentro de algum conjunto soluções viáveis, chamamos de **problema de otimização**.

Se um algoritmo A resolve um problema Π em tempo $p(\langle I \rangle)$ para algum polinômio p, então esse algoritmo é dito **polinomial**.

Dizemos que se um algoritmo é polinomial, então ele é **eficiente**.

Todos os problemas de decisão que podem ser resolvidos polinomialmente formam uma classe, denotada por **P**.

A classe formada por todos os problemas para os quais existe um certificado para SIM que pode ser verificado em tempo polinomial é denotada por **NP**. Claramente, $\mathrm{P} \subseteq \mathrm{NP}$.

Uma **redução** de um problema Π a um problema Π' é um algoritmo A que resolve Π utilizando um outro algoritmo A', que resolve Π'.

Dizemos que um problema Π é **NP-completo** se todos os problemas em NP podem ser reduzidos a Π em tempo polinomial.

Dizemos que um problema Π é NP-difícil se a existência de um algoritmo polinomial para Π implica em $\mathrm{P} = \mathrm{NP}$. Não necessariamente um problema NP-difícil está em NP.

2.3 Algoritmos de Aproximação

Um problema de otimização é composto por um conjunto de **instâncias** $\mathcal{I}$, um conjunto de **soluções viáveis** $\mathrm{Sol}(I)$ para alguma instância $I \in \mathcal{I}$ e uma função que atribui um **valor** $\mathrm{val}(S)$ para cada solução viável S.

Existem problemas de **minimização**, que buscam encontrar as soluções viáveis de valor mínimo, e problemas de **maximização**, que buscam encontrar as soluções viáveis de valor máximo. Uma solução viável cujo valor é ótimo é chamada solução **ótima**.

Denotamos por S^* uma solução ótima. O valor de uma uma solução ótima para uma instância será denotado por $\mathbf{OPT}(I)$, ou simplesmente **OPT**. Temos que

$$\mathrm{OPT}(I) := \mathrm{val}(S^*).$$

A classe de problemas **NPO** é a extensão da classe NP para problemas de otimização. Ela é formada pelos problemas de otimização para os quais:

- existe um polinômio p tal que para toda instância I do problema e toda solução viável S de I, $\langle S \rangle \leq p(\langle I \rangle)$;
- existe um algoritmo polinomial que decide se uma dada palavra é uma instância válida do problema;

- existe um algoritmo polinomial que decide se uma dada solução é viável para uma certa instância do problema;
- existe um algoritmo polinomial que calcula $\text{val}(I, S)$ dados I e S.

Dado um problema de otimização em que $\text{val}(S) \geq 0$ para toda solução viável S de qualquer instância I do problema, seja A um algoritmo, que, para toda instância devolve uma solução viável $A(I)$ de I. Seja OPT o valor ótimo de uma solução ótima de uma instância desse problema. Se o problema é de minimização e para toda instância

$$\text{val}(A(I)) \leq \alpha \cdot \text{OPT},$$

dizemos que A é uma **α-aproximação** para o problema. Caso o problema seja de maximização e para toda instância

$$\text{val}(A(I)) \geq \alpha \cdot \text{OPT},$$

então A também é uma α-aproximação.

Dizemos que α é a **razão de aproximação** do algoritmo. Ademais, essa razão pode depender da instância I.

Um **algoritmo de aproximação** é uma α-aproximação para algum α.

Dizemos que a razão de aproximação α é **justa** para o algoritmo A, quando existe uma instância I tal que $\text{val}(A(I)) = \alpha \cdot \text{OPT}$.

Para um problema de otimização, se um algoritmo A recebe um valor $\epsilon > 0$ e uma instância I, e devolve uma aproximação com erro relativo de no máximo ϵ, ou seja,

$$(1 - \epsilon) \cdot \text{OPT} \leq \text{val}(I, A(\epsilon, I)),$$

se o problema for de maximização, e

$$\text{val}(I, A(\epsilon, I)) \leq (1 + \epsilon) \cdot \text{OPT}$$

se o problema for de minimização, então dizemos que A é um **esquema de aproximação** (*approximation scheme).*

Dizemos que A é um **esquema de aproximação polinomial** (*polynomial-time approximation scheme*) se ele é polinomial para qualquer ϵ fixo.

Se, além de ser polinomial, a quantidade de tempo consumida pelo algoritmo A for limitada por $p(\langle I \rangle, 1/\epsilon)$ para algum um polinômio p, dizemos que A é um **esquema de aproximação plenamente polinomial** (*fully polynomial-time approximation scheme*).

A classe **PTAS** é formada por todos os problemas em NPO que admitem um esquema de aproximação polinomial.

A classe **FPTAS** é formada por todos os problemas em NPO que admitem um esquema de aproximação plenamente polinomial.

Segue que

$$\text{FPTAS} \subseteq \text{PTAS} \subseteq \text{NPO}.$$

Capítulo 3

Resultados Conhecidos

Existem na literatura diversos resultados sobre os problemas MinMax BCP$_q$ e MaxMin BCP$_q$. Neste capítulo mencionamos alguns desses resultados.

Na Seção 3.1 apresentamos resultados sobre a complexidade computacional dos problemas MinMax BCP$_q$ e MaxMin BCP$_q$, tanto nas versões com pesos nos vértices quanto nas versões sem. Além disso, apresentamos um resultado quando o número de classes da partição não é fixo e faz parte da entrada do problema.

Já na Seção 3.2 mencionamos algoritmos exatos que existem para o MinMax BCP$_q$ e o MaxMin BCP$_q$. Alguns desses resultados são baseados em formulações combinatórias e restritos a classes específicas de grafos. Já outros, são baseados em formulações de programação linear inteira e programação linear inteira mista.

3.1 Complexidade Computacional e Aproximabilidade

Dyer e Frieze (1985) mostraram que os problemas 1-MaxMin BCP$_q$ e 1-MinMax BCP$_q$, são NP-difíceis quando o grafo de entrada é planar (até mesmo quando é bipartido). Disso, segue que as versões com pesos nos vértices dos problemas, MaxMin BCP$_q$ e MinMax BCP$_q$ também são NP-difíceis para grafos planares bipartidos. Ademais, Ronald I. Becker, Lari *et al.* (1998) mostraram que esses problemas também são NP-difíceis em grades com pelo menos 3 linhas.

Wu (2012) mostrou que o problema MaxMin BCP$_2$ restrito a grafos de intervalos admite FPTAS. Além disso, mostrou também que para, $q \geq 3$, o problema MaxMin BCP$_q$ também admite FPTAS para grafos de intervalos q-conexos. Já Wang *et al.* (2013) mostraram que para $q = 2, 3, 4$, existe FPTAS em grafos trapezoidais de grau limitado.

Quando q é parte da entrada do problema, Chataigner *et al.* (2007) mostraram que o problema MaxMin BCP não pode ser aproximado com razão de aproximação menor ou igual a $\frac{6}{5}$.

Os próximos resultados citados aqui serão provados com detalhes no Capítulo 6.

Chlebíková (1996) provou que é NP-difícil aproximar 1-MaxMin BCP_q com garantia de erro absoluto $|V|^{1-\epsilon}$ para qualquer $0 < \epsilon < 1$. Para grafos q-conexos, Chataigner *et al.* (2007) mostraram que MaxMin BCP_q é NP-difícil no sentido forte. Além disso, mostraram também que, a não ser que P = NP, não existe algoritmo para o problema BCP_2 que devolva uma $(1+\epsilon)$-aproximação quando $\epsilon \leq \frac{1}{|V|^2}$.

Os resultados mencionados estão resumidos na Tabela 3.1.

Tipo de Grafo	Problema	Resultado
Planares bipartidos e Grades não triviais	1-MinMax BCP_q, $q \geq 2$	NP-difícil
	1-MaxMin BCP_q, $q \geq 2$	NP-difícil
	MinMax BCP_q, $q \geq 2$	NP-difícil
	MaxMin BCP_q, $q \geq 2$	NP-difícil
Grafos de intervalos	MaxMin BCP_2	Admite FPTAS
Grafos de intervalos q-conexos	MaxMin BCP_q, $q \geq 2$	Admite FPTAS
Grafos trapezoidais de grau limitado	MaxMin BCP_q, $q = 2, 3, 4$	Admite FPTAS
Grafo q-conexo	MaxMin BCP_q, $q \geq 2$	NP-difícil no sentido forte
Grafos conexos	MaxMin BCP	NP-difícil $\leq \frac{6}{5}$-aproximação
	MaxMin BCP	NP-difícil erro absoluto $\leq |V|^{1-\epsilon}$

Tabela 3.1: *Complexidade computacional e aproximabilidade dos problemas*

3.2 Algoritmos Exatos

Nesta seção mencionamos alguns resultados sobre algoritmos polinomiais. Esses algoritmos podem ser divididos em dois tipos principais: algoritmos combinatórios, (para classes específicas de grafos) e algoritmos baseados em formulações lineares inteiras ou lineares inteiras mistas.

Estas formulações lineares e mistas fornecem algoritmos exatos para grafos gerais, no entanto, instâncias muito grandes desses problemas não podem ser resolvidas eficientemente.

3.2.1 Algoritmos combinatórios

O seguinte resultado, foi mostrado por Lovász (1977) e Györi (1978).

Teorema 1. *Seja G um grafo q-conexo de ordem n com $q \geq 2$ e sejam $n_1, n_2, \ldots, n_q$ números naturais tais que $n_1 + n_2 + \cdots + n_q = n$. Então G possui uma q-partição conexa $\{V_1, V_2, \ldots, V_q\}$ tal que $|V_i| = n_i$ para $i = 1, 2, \ldots, q$.*

Apesar de inúmeras tentativas, até hoje não se sabe se existe um algoritmo polinomial para obter a partição mencionada no Teorema 1. Para valores pequenos de q, alguns algoritmos foram obtidos, conforme mencionamos a seguir.

Hitoshi Suzuki *et al.* (1990) apresentaram um algoritmo linear para encontrar uma bipartição de grafos 2-conexos. No mesmo ano, Suzuki *et al.* (1990) apresentaram um algoritmo quadrático para encontrar uma 3-partição de grafos 3-conexos. Já Nakano *et al.* (1997) mostraram que em grafos planares 4-conexos existe algoritmo linear para encontrar uma 4-partição. Note que esses resultados são para o caso de pesos unitários.

Ronald I Becker e Perl (1983) desenvolveram um algoritmo para o problema MinMax BCP$_q$ restrito a árvores. Dada uma árvore T com raio r, esse algoritmo é polinomial e tem tempo computacional $O(q^3r + q|V|)$.

Mais tarde, Frederickson (1991) e Frederickson e S. Zhou (2017) melhoraram o tempo de execução do algoritmo de Ronald I Becker e Perl (1983), desenvolvendo algoritmos para os problemas MinMax BCP$_q$ e MaxMin BCP$_q$ que executam em tempo $O(|V|)$.

R. Becker *et al.* (2001) desenvolveram um algoritmo polinomial para o MaxMin BCP$_q$ restrito a escadas. Também foram desenvolvidos, por Alimonti e Calamoneri (1999), algoritmos polinomiais para o MaxMin BCP$_2$ restrito a grafos com no máximo dois vértices de corte.

Resumimos na Tabela 3.2 os resultados mencionados nesta seção.

Tipo de grafo	Problema
Árvores	MinMax BCP$_q$ e MaxMin BCP$_q$
Escadas	MaxMin BCP$_q$
Grafo com 1 ou 2 vértices de corte	MaxMin BCP$_2$
Grafo 2-conexo	1-BCP$_2$
Grafo 3-conexo	1-BCP$_3$
Grafo planar 4-conexo	1-BCP$_4$

Tabela 3.2: *Casos em que foram obtidos algoritmos polinomiais combinatórios*

3.2.2 Formulações lineares inteiras ou mistas

Dragan Matić (2014) apresentou uma formulação do problema baseada em programação linear inteira mista que possui um número polinomial (no tamanho do grafo de entrada) de variáveis e restrições. Essa formulação apresentada consegue resolver de maneira exata a maior parte de entradas pequenas e médias. Nos testes realizados pelo autor, grades com até 70 vértices e grafos aleatórios com até 50 vértices tiveram a maioria de suas instâncias resolvidas.

Alguns anos depois, X. Zhou *et al.* (2019) apresentaram outra formulação, também baseada em programação linear inteira mista. Essa formulação é baseada em um modelo

de fluxo e resolve o MinMax BCP$_q$ de forma exata. Nos testes realizados pelos autores, essa formulação conseguiu resolver de maneira exata grafos com até 200 vértices (para valores pequenos de q).

Foram apresentadas por Miyazawa *et al.* (2021) novas formulações baseadas em programação linear inteira e formulações baseadas em fluxos. Ambas tem tamanho polinomial no tamanho do grafo de entrada, tanto para o número de variáveis quanto para o número de restrições.

Os testes realizados pelos autores mostraram que essas formulações conseguem resolver exatamente instâncias até 400 vezes maiores que o tamanho das maiores instâncias resolvidas por Dragan Matić (2014) e por X. Zhou *et al.* (2019). Ademais, os algoritmos dessas formulações são em média 5 vezes mais rápidos que os apresentados anteriormente. No entanto, todas essas formulações não demonstraram bom desempenho quando $q \geq 5$ (ou quando foram consideradas instâncias reais).

Capítulo 4

Algoritmos de Aproximação para o MinMax BCP$_q$

Existem na literatura alguns poucos algoritmos de aproximação para o problema MinMax BCP$_q$. Neste capítulo vamos apresentar alguns deles.

Primeiramente, mencionamos os seguintes lemas para o 1-MinMax BCP$_q$ e para o MinMax BCP$_q$ (que são imediatos, mas serão úteis no que segue).

Lema 2. *Seja I uma instância do* 1-MinMax BCP$_q$*, que consiste de um grafo conexo $G = (V, E)$. Então,* $\mathrm{OPT}(I) \geq \frac{1}{q}|V|$.

Assim, podemos observar que existe uma q-aproximação trivial para o 1-MinMax BCP$_q$.

Lema 3. *Seja $I = (G, w)$ uma instância do* MinMax BCP$_q$*, que consiste de um grafo conexo $G = (V, E)$ e uma função peso $w : V \to \mathbb{Z}_+$. Então,* $\mathrm{OPT}(I) \geq \frac{1}{q}w(V)$.

4.1 Uma $\frac{3}{2}$-aproximação para o 1-MinMax BCP$_3$

O primeiro algoritmo que apresentamos foi obtido por Y. Chen *et al.* (2019). Esse algoritmo é uma $\frac{3}{2}$-aproximação para o 1-MinMax BCP$_3$.

Seja $\{V_1, V_2, V_3\}$ uma 3-partição conexa (também chamada viável) de um grafo conexo $G = (V, E)$ e suponha, sem perda de generalidade, que $|V_1| \leq |V_2| \leq |V_3|$. Seja $n = |V|$. Primeiro observamos que, como G é conexo, em qualquer partição viável $\{V_1, V_2, V_3\}$ de V, o conjunto V_3 deve ser adjacente a V_1 ou a V_2.

Usaremos as duas operações, definidas a seguir, no algoritmo.

Definição 4 (MERGE)**.** *Para que essa operação possa ser aplicada em $\{V_1, V_2, V_3\}$, devemos ter $|V_3| > \frac{1}{2}n$ e V_1 e V_2 devem ser adjacentes.*

A operação MERGE *devolve uma* 3*-partição conexa* $\{V_1 \cup V_2, V_3', V_3''\}$ *onde* $\{V_3', V_3''\}$ *é uma* 2*-partição conexa arbitrária de* V_3*.*

Lema 4. *Dado um grafo conexo* $G = (V, E)$ *e uma* 3*-partição conexa* $\{V_1, V_2, V_3\}$ *de* V *com* $|V_3| > \frac{1}{2}n$*, a operação* MERGE *gera uma* 3*-partição conexa de* V *e mais balanceada que* $\{V_1, V_2, V_3\}$*.*

Demonstração. Como $V_3 > \frac{1}{2}n$, o tamanho da nova classe $V_1 \cup V_2$ será $|V_1| + |V_2| < \frac{1}{2}n$. Além disso, claramente os tamanhos de V_3' e V_3'' são menores que $|V_3|$, com isso provamos o que queríamos.

□

Definição 5 (PULL)**.** *Para que essa operação possa ser aplicada em* $\{V_1, V_2, V_3\}$*, devemos ter* $|V_3| > \frac{1}{2}n$*. Além disso, deve existir um subconjunto* U *de* V_3 *tal que* $G[U]$ *e* $G[V_3 \setminus U]$ *são conexos,* U *é adjacente a* V_i *e* $|V_i| + |U| < |V_3|$*, com* $i \in \{1, 2\}$*.*

A operação PULL *devolve uma* 3*-partição conexa* $\{V_3 \setminus U, V_i \cup U, V_j\}$ *com* $j = 3 - i$*.*

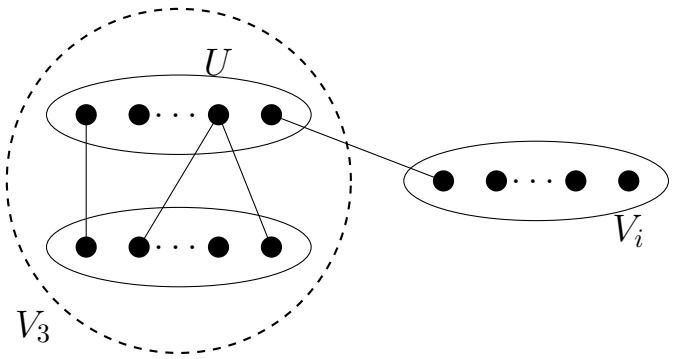

Figura 4.1: *Ilustração da operação* PULL*. Cada região delimitada representa uma parte conexa do grafo e* $i \in \{1, 2\}$*.*

Lema 5. *Dado um grafo conexo* $G = (V, E)$ *e uma* 3*-partição conexa* $\{V_1, V_2, V_3\}$ *de* V *com* $|V_3| > \frac{1}{2}n$*, a operação* PULL *gera uma* 3*-partição conexa de* V *que é mais balanceada que* $\{V_1, V_2, V_3\}$*.*

Demonstração. Pelas pré-condições impostas para que a operação possa ser aplicada, observamos que a 3-partição gerada é conexa. Ademais, como $|V_i| + |U| < |V_3|$, $|V_j| < \frac{1}{2}n < |V_3|$ e $|V_3 \setminus U| < |V_3|$, então a partição obtida é mais balanceada que a anterior e a prova está completa.

□

O próximo lema fornece informações sobre $\{V_1, V_2, V_3\}$ quando não podemos aplicar nem MERGE nem PULL nessa partição.

Lema 6. *Dado um grafo conexo* $G = (V, E)$ *e uma* 3*-partição conexa* $\{V_1, V_2, V_3\}$ *de* V *com* $V_3 > \frac{1}{2}n$*. Seja* uv *uma aresta de* $E(V_3, V_1)$*. Se a partição não satisfizer as condições impostas para a aplicação nem de* MERGE *nem de* PULL*, então*

(a) $|V_1|+|V_2| < \frac{1}{2}n$. *Além disso,* V_1 *e* V_2 *não são adjacentes, e portanto são adjacentes a* V_3.

(b) $G[V_3 \setminus \{u\}]$ *não é conexo. Sejam* $G[V_{31}^u]$, $G[V_{32}^u]$, ..., $G[V_{3l}^u]$ *as componentes de* $G[V_3 \setminus \{u\}]$, *então para todo* $i \in \{1, 2, ..., l\}$, $|V_{3i}^u| \leq |V_1|$ *e* V_{3i}^u *e* V_1 *não são adjacentes.*

(c) Nenhum vértice de $V_1 \cup V_2$ *é adjacente a nenhum outro vértice de* V_3 *além de* u.

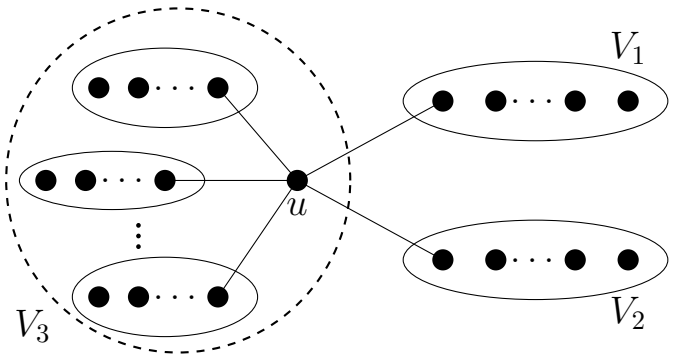

Figura 4.2: *Exemplo de um grafo em que nem a operação* PULL *nem a operação* MERGE *podem ser executadas.*

Demonstração. Sabemos que $|V_1|+|V_2| < \frac{1}{2}n$ e como $|V_1| \leq |V_2|$, temos que $|V_1| < \frac{1}{4}n$.

Provamos agora cada item:

(a) Como não é possível aplicar a operação MERGE, então V_1 e V_2 não são adjacentes. Logo, como G é conexo, tanto V_1 quanto V_2 devem ser adjacentes a V_3.

(b) Se $G[V_3 \setminus \{u\}]$ fosse conexo, então poderíamos aplicar PULL na 3-partição existente. Ademais, se $|V_{3i}^u| > |V_1|$ para algum $i \in \{1, 2, ..., l\}$, então também seria possível aplicar PULL com $U = V_{3i}^u$ e V_1, já que $|V_3 \setminus V_{3i}^u| + |V_1| < |V_3|$.

Por último, se V_{3i}^u e V_1 fossem adjacentes para algum i, então poderíamos aplicar PULL com V_{3i}^u e V_1, pois $|V_{3i}^u| + |V_1| \leq 2|V_1| < \frac{1}{2}n < |V_3|$.

(c) Pelo item (b) temos que u é o único vértice que pode ser adjacente a um vértice de V_1. Vamos mostrar que o mesmo vale para V_2.

Assuma que V_2 é adjacente a V_{3i}^u para algum $i \in \{1, 2, ..., l\}$. Então, como $|V_{3i}^u| \leq |V_1|$, $|V_{3i}^u| + |V_2| \leq |V_1| + |V_2| < \frac{1}{2}n < V_3$, poderíamos aplicar PULL, uma contradição. Assim, u é o único vértice que pode ser adjacente a V_2.

□

Os Lemas 4, 5 e 6, sugerem um algoritmo, apresentado por Y. Chen *et al.* (2019).

Algoritmo 1 APPROX-1

Entrada: Grafo conexo $G = (V, E)$
Saída: 3-partição conexa $\{V_1, V_2, V_3\}$ de V

```
1: Construa uma partição conexa {V1, V2, V3} de G tal que |V1| ≤ |V2| ≤ |V3|
2: n ← |V|
3: enquanto |V3| > (1/2)n faça
4:     se {V1, V2, V3} satisfaz as condições de MERGE então
5:         {V1, V2, V3} ← MERGE(V1, V2)
6:     senão se {V1, V2, V3} satisfaz as condições de PULL então
7:         Seja U ⊆ V3
8:         {V1, V2, V3} ← PULL(U, Vi)
9:     senão
10:        devolva {V1, V2, V3}
11: devolva {V1, V2, V3}
```

Teorema 7. *O algoritmo* APPROX-1 *é uma* $\frac{3}{2}$*-aproximação para o* 1-MinMax BCP$_3$ *e executa em tempo* $O(|V||E|)$*. Ademais, essa razão de aproximação é justa.*

Demonstração. Seja $G = (V, E)$ o grafo de entrada do algoritmo. No fim da execução, se $|V_3| \leq \frac{1}{2}n$, então, pelo Lema 2,

$$\text{OPT} \geq \frac{n}{3} \geq \frac{2|V_3|}{3}.$$

Logo,

$$|V_3| \leq \frac{3}{2}\text{OPT}.$$

Por outro lado, se $|V_3| > \frac{1}{2}n$ no fim da execução, então $|V_1| + |V_2| < \frac{1}{2}n$, e como $|V_1| \leq |V_2|$, temos que $|V_1| < \frac{1}{4}n$.

Assim, pelos Lemas 4 e 6, V_1 e V_2 não são adjacentes e devem ser adjacentes a V_3. Seja u, pelo Lema 6, o único vértice de V_3 que pode ser adjacente aos vértices de V_1 e de V_2. Concluímos então, por esse lema, que o subgrafo $G[V_3 \setminus \{u\}]$ é desconexo.

Logo, existem componentes $G[V_{31}^u]$, $G[V_{32}^u]$, ..., $G[V_{3l}^u]$, com $l \geq 2$ tal que V_{3i}^u não é adjacente a V_1 nem a V_2 e $|V_{3i}^u| \leq |V_1|$ para todo $i \in \{1, 2, ..., l\}$. Isto é, como podemos observar na Figura 4.2, o grafo G possui, nesse caso, uma estrutura parecida com uma estrela, isto é, as componentes são não adjacentes duas a duas, mas todas são adjacentes ao vértice u.

Claramente, numa 3-partição conexa ótima, a parte que contém u deve ter tamanho pelo menos V_3, logo a 3-partição encontrada é ótima.

Agora, vamos mostrar que o algoritmo executa em tempo $O(|V||E|)$.

Para executar uma operação PULL podemos fazer uma busca no subgrafo $G[V_3 \setminus \{u\}]$ para primeiro determinar se ele é conexo, e, caso não seja, explorar suas componentes. Podemos realizar tal busca em tempo $O(|V| + |E|)$. Da mesma

forma, uma operação MERGE também é executada em tempo $O(|V|+|E|)$. O número máximo de operações PULL e MERGE aplicadas pelo algoritmo é $O(|V|)$. Assim, APPROX-1 executa em tempo $O(|V||E|)$ e a prova do teorema está concluída.

□

Recentemente, Moura *et al.* (2021) apresentaram uma $\frac{3}{2}$-aproximação para o MinMax BCP$_3$ que executa em tempo pseudo-polinomial. Fazendo uso de *scaling*, esses autores obtiveram um algoritmo polinomial com razão $\frac{3}{2}+\epsilon$, onde ϵ pode ser tomado tão pequeno quanto se queira. Esse algoritmo é similar ao que descrevemos nessa seção e possui uma implementação e análise um pouco mais simples do que as do algoritmo descrito por G. Chen *et al.* (2020), também para o MinMax BCP$_3$ e que também possui razão de aproximação $\frac{3}{2}$.

4.2 Uma $\frac{q}{2}$-aproximação para o 1-MinMax BCP$_q$

O algoritmo apresentado na Seção 4.1 pode ser generalizado para o problema 1-MinMax BCP$_q$, gerando uma $\frac{q}{2}$-aproximação para ele. Esse algoritmo também mostrado por Y. Chen *et al.* (2019), é descrito na prova do teorema a seguir.

Teorema 8. *O problema* 1-MinMax BCP$_q$ *admite uma* $\frac{q}{2}$*-aproximação, para* $q \geq 3$*, que pode ser encontrada em tempo* $O(|V||E|)$.

Demonstração. Seja $G = (V, E)$ um grafo conexo. Podemos aplicar o algoritmo APPROX-1 em G para obter uma 3-partição $\{V_1, V_2, V_3\}$ de V de forma que $|V_1| \leq |V_2| \leq |V_3|$.

Se $|V_3| \leq \frac{1}{2}n$, então podemos sempre particionar a maior parte em 2 até que tenhamos q partes. Notamos que a maior parte não é maior que $\frac{1}{2}n$.

Se $|V_3| > \frac{1}{2}n$, seja u o único vértice de V_3 que pode ser adjacente aos vértices de V_1 e de V_2.

Como $G[V_3 \setminus \{u\}]$ é desconexo, observamos que existem $k \geq 4$ componentes conexas em $G[V \setminus \{u\}]$, que são V_1, V_2 e as demais (pelo menos duas) existentes em $G[V_3 \setminus \{u\}]$. A maior delas, $G[V_2]$, é tal que $|G[V_2]| \leq \frac{1}{2}n$, e as outras terão tamanho menor que $\frac{1}{4}n$.

Quando $q \leq k$, podemos manter as $q-1$ maiores componentes e fazer a união de todas as restantes a partir do vértice u. Assim, teremos uma q-partição conexa tal que a classe de tamanho máximo não é maior que $\max\{V_2, \text{OPT}\}$, pois em uma q-partição ótima, a parte que contém o vértice u é maior ou igual à última parte que foi construída.

Já quando a partição que queremos construir deverá ter mais partes do que o número de componentes que possuímos, ou seja, $q > k$, podemos usar a mesma técnica utilizada quando $|V_3| \leq \frac{1}{2}n$:

Começamos com a partição gerada pelas componentes de $G[V \setminus \{u\}]$ e sempre dividimos a maior classe em duas, até que tenhamos q classes de forma que a maior classe tenha tamanho menor que $\frac{1}{2}n$.

Assim, encontramos uma partição ótima ou uma q-partição em que o tamanho da maior classe é menor que $\frac{1}{2}n$. Seja V_i tal parte. Então, pelo Lema 2,

$$|V_i| < \frac{1}{2}n \leq \frac{1}{2} \cdot q \cdot \text{OPT}.$$

Assim,

$$|V_i| < \frac{q}{2}\text{OPT},$$

e esse algoritmo é uma $\frac{q}{2}$-aproximação para o 1-MinMax BCP$_q$.

No entanto, podemos observar que essa aproximação não é justa, pois quando $q \geq 4$, se a q-partição obtida não for ótima, o tamanho de sua maior classe é menor que $\frac{1}{2}n$.

Executamos uma vez, no início desse algoritmo, APPROX-1, que executa em tempo $O(|V||E|)$. O número de iterações que faremos nesse algoritmo após essa execução é no máximo q, que é menor que $|V|$.

Assim, como gastamos $O(|E|)$ por iteração para realizar o particionamento de cada classe, o algoritmo pode ser executado em tempo $O(|V||E|)$ e é polinomial.

□

Da mesma forma que o algoritmo citado no fim da Seção 4.1, Moura *et al.* (2021), descreveram uma $\frac{q}{2}$-aproximação polinomial para o MinMax BCP$_q$, com $q \geq 4$. Esse algoritmo foi inspirado na $\frac{q}{2}$-aproximação para o 1-MinMax BCP$_q$ desenvolvida por Y. Chen *et al.* (2019) e que foi detalhado acima, nessa seção.

Esse novo algoritmo utiliza uma técnica de *scaling* para lidar com pesos que sejam possivelmente muito grandes.

Quando os pesos dos vértices do grafo de entrada são limitados por uma função que é um polinômio no tamanho do grafo, então esse algoritmo possui razão de aproximação exatamente $\frac{q}{2}$.

Já quando esses pesos são maiores que esse polinômio, a técnica de *scaling* utilizada faz com que a razão de aproximação atingida pelo algoritmo seja de $\frac{q}{2} + \epsilon$ para algum $\epsilon > 0$ arbitrariamente pequeno.

Assim, o algoritmo polinomial de Moura *et al.* (2021) devolve uma $(\frac{q}{2} + \epsilon)$-aproximação para o MinMax BCP$_q$.

Para concluir o capítulo, mencionamos aqui um último resultado, apresentado por Y. Chen *et al.* (2019) que mostra que o algoritmo desenvolvido por Chlebíková (1996), que será exposto na Seção 5.1, além de ser uma $\frac{4}{3}$-aproximação para o MaxMin BCP$_2$, também é uma $\frac{5}{4}$-aproximação para o MinMax BCP$_2$.

Os algoritmos de aproximação que mencionamos para o 1-MinMax BCP$_q$ e para o MinMax BCP$_q$ são os melhores conhecidos na literatura para grafos conexos arbitrários.

Capítulo 5

Algoritmos de Aproximação para o MaxMin BCP$_q$

Os algoritmos para o MaxMin BCP$_q$ são, em geral, mais complicados do que os algoritmos para o MinMax BCP$_q$. Por conta disso, neste capítulo vamos apresentar dois algoritmos, obtidos por Chlebíková (1996) para o MaxMin BCP$_2$. O primeiro é para grafos 2-conexos. Chlebíková mostrou como este algoritmo pode ser utilizado para resolver MaxMin BCP$_2$ em grafos conexos arbitrários.

A primeira observação que fazemos é o seguinte lema para o MaxMin BCP$_q$, análogo ao que vimos no capítulo anterior para o MinMax BCP$_q$.

Lema 9. *Seja $I = (G, w)$ uma instância do* MaxMin BCP$_q$*, que consiste de um grafo conexo $G = (V, E)$ e uma função peso $w : V \to \mathbb{Z}_+$. Então,* $\mathrm{OPT}(I) \leq \frac{1}{q} w(V)$.

5.1 Uma $\frac{4}{3}$-aproximação para o MaxMin BCP$_2$

Os algoritmos que apresentamos nesta seção foram desenvolvidos por Chlebíková (1996). O primeiro, que chamamos de APPROX-2 lida com grafos 2-conexos e o segundo usa APPROX-2 para lidar com grafos conexos arbitrários

5.1.1 Grafos 2-conexos

Antes de apresentar o algoritmo apresentamos um lema auxiliar.

Lema 10. *Seja $G = (V, E)$ um grafo 2-conexo e $\{V_1, V_2\}$ uma bipartição conexa de V tal que $|V_2| \geq 2$. Existem dois vértices distintos $u_1, u_2 \in V_2$ tais que tanto $\{V_1 \cup \{u_1\}, V_2 \setminus \{u_1\}\}$ quanto $\{V_1 \cup \{u_2\}, V_2 \setminus \{u_2\}\}$ são partições conexas de V.*

Agora definimos a operação PULL-2 que será utilizada no algoritmo.

Definição 6 (PULL-2)**.** *Para que essa operação possa ser aplicada em uma bipartição $\{V_1, V_2\}$, devemos ter $w(V_1) < \frac{3}{8} w(V)$. Escolhemos um vértice $u \in V_2$ tal que u*

seja o vértice de menor peso entre aqueles que podem passar de V_2 para V_1, mantendo a bipartição conexa. Além disso, para aplicar essa operação precisamos que $w(u) < w(V) - 2w(V_1)$.

A operação PULL-2 *devolve uma bipartição conexa $\{V_1 \cup \{u\}, V_2 \setminus \{u\}\}$.*

Descrevemos o algoritmo a seguir.

Algoritmo 2 APPROX-2

Entrada: Grafo 2-conexo $G = (V, E)$ e função peso $w : V \to \mathbb{Z}_+$
Saída: 2-partição conexa $\{V_1, V_2\}$ de V

1: Ordene os vértices de V de maneira que $w(v_1) \geq w(v_2) \geq \cdots \geq w(v_n)$
2: $\{V_1, V_2\} \leftarrow \{\{v_1\}, V \setminus \{v_1\}\}$
3: **se** $w(v_1) \geq \frac{1}{2}w(V)$ **então**
4: **devolva** $\{V_1, V_2\}$
5: **enquanto** $w(V_1) < \frac{3}{8}w(V)$ **faça**
6: **se** $\{V_1, V_2\}$ satisfaz as condições de PULL-2 **então**
7: $u \leftarrow \min\{w(v) | v \in V_2 \text{ e } \{V_1 \cup \{u\}, V_2 \setminus \{u\}\}\}$
8: $\{V_1, V_2\} \leftarrow$ PULL-2(u, V_1)
9: **devolva** $\{V_1, V_2\}$

Teorema 11. *O algoritmo* APPROX-2 *é uma $\frac{4}{3}$-aproximação para o* MaxMin BCP$_2$ *que executa em tempo $O(|V||E|)$. Ademais, a razão $\frac{4}{3}$ é justa para esse algoritmo.*

Demonstração. Seja $G = (V, E)$ um grafo 2-conexo com uma função peso $w : V \to \mathbb{Z}_+$. Suponha que $|V| = n$ e $w(v_1) \geq w(v_2) \geq \cdots \geq w(v_n)$. Seja $\{V_1, V_2\}$ a partição devolvida pelo algoritmo APPROX-2.

Quando $w(v_1) \geq \frac{1}{2}w(V)$, claramente a bipartição encontrada é ótima.

Já quando $w(v_1) < \frac{1}{2}w(V)$, observamos que $|V| \geq 3$. Assim, para qualquer vértice u escolhido para executar PULL-2 temos $w(u) \leq w(v_3)$, pois $v_1 \notin V_2$, e pelo Lema 10, V_2 tem pelo menos dois vértices que podem migrar para V_1 e manter a bipartição resultante conexa.

O algoritmo para em dois casos diferentes. Se ele parar quando $w(V_1) < \frac{1}{2}w(V)$ e $w(u) \geq w(V) - 2w(V_1)$, então temos que

$$\frac{1}{2}w(V) > w(V_1) \geq \frac{1}{2}(w(V) - w(u)) \geq \frac{1}{2}(w(V) - w(v_3)).$$

Por outro lado, se ele parar quando $w(V_1) \geq \frac{1}{2}w(V)$

$$w(u) < w(V) - 2w(V_1 \setminus \{u\}) = w(V) - 2w(V_1) + 2w.$$

Logo,

$$\frac{1}{2}w(V) \leq w(V_1) < \frac{1}{2}(w(V) + w(u)).$$

Finalmente, nesse segundo caso,

$$\frac{1}{2}w(V) \geq w(V_2) > \frac{1}{2}(w(V) - w(u)) \geq \frac{1}{2}(w(V) - w(v_3)).$$

Dessa forma, em ambos os casos, como $w(v_3) \leq \frac{1}{3}w(V)$, temos

$$\min\{V_1, V_2\} \geq \frac{1}{2}(w(V) - w(v_3)) \geq \frac{1}{3}w(V),$$

Mostramos agora que OPT $\leq \frac{4}{3}\min\{V_1, V_2\}$. Pelo Lema 9, sabemos que OPT $\leq \frac{1}{2}w(V)$. No entanto, conseguimos melhorar essa desigualdade para

$$\text{OPT} \leq (1 - 2\beta)w(V), \tag{5.1}$$

com $w(v_3) = \beta w(V)$ e $\beta \in [1/4, 1/3]$.

Essa nova desigualdade vale, pois, dada qualquer bipartição $\{V_1', V_2'\}$ de V, uma das classes V_1 e V_2 contém pelo menos dois vértices entre v_1, v_2 e v_3. Assim, o peso dessa classe V_i', $i \in \{1, 2\}$, será pelo menos $2\beta w(V) \geq \frac{1}{2}w(V)$.

Para concluir a prova do teorema, vamos analisar dois casos: quando $w(v_3) \leq \frac{1}{4}w(V)$ e quando $w(v_3) \geq \frac{1}{4}w(V)$.

- Se $w(v_3) \leq \frac{1}{4}w(V)$, então $min\{V_1, V_2\}$. Pelo Lema 9, temos

$$\text{OPT} \leq \frac{4}{3}\min\{V_1, V_2\};$$

- Se $w(v_3) \leq \frac{1}{4}w(V)$, então $w(v_3) = \beta w(V)$ para algum $\beta \in [1/4, 1/3]$. Segue então que $\min\{V_1, V_2\} \geq \frac{1-\beta}{2}w(V)$, e usando a desigualdade 5.1, temos

$$\text{OPT} \leq \frac{2(1 - 2\beta)}{1 - \beta}\min\{V_1, V_2\}.$$

  Tomamos uma função $f(\beta) \coloneqq \frac{2(1-2\beta)}{1-\beta}$ e observamos que no intervalo $(-\infty, 1)$ ela é decrescente, e que seu valor máximo no intervalo $[1/4, 1/3]$ é em $\beta = \frac{1}{4}$, que resulta em $f(\beta) = \frac{4}{3}$. Assim, nesse caso,

$$\text{OPT} \leq \frac{4}{3}\min\{V_1, V_2\}.$$

Como nos dois casos temos a mesma razão, mostramos que o algoritmo APPROX-2 é uma $\frac{4}{3}$-aproximação para o MaxMin BCP$_2$.

Para mostrar que essa razão é justa, podemos observar a execução do algoritmo no grafo 2-conexo desenhado na Figura 5.1 e com pesos descritos na Tabela 5.1.

Ao executar o algoritmo APPROX-2 no grafo da Figura 5.1, na primeira iteração

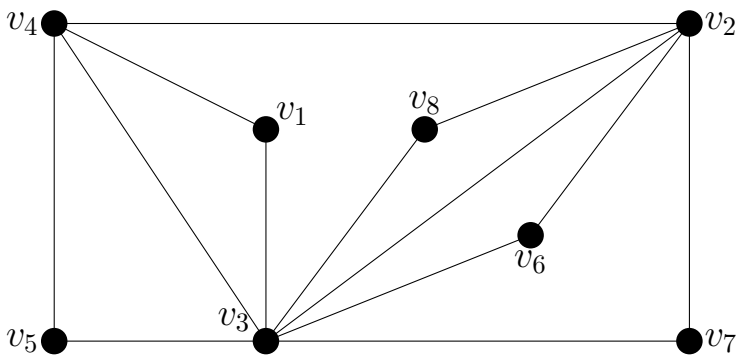

Figura 5.1: *Exemplo de grafo* 2*-conexo em que o algoritmo APPROX-*2 *devolve uma* $\frac{4}{3}$*-aproximação.*

v	$w(v)$
v_1	6
v_2	6
v_3	6
v_4	2
v_5	1
v_6	1
v_7	1
v_8	1

Tabela 5.1: *Pesos para os vértices do grafo desenhado na figura 5.1*

obtemos uma partição $V_1 = \{v_1\}$ e $u = \{v_4\}$. Na segunda iteração temos $V_1 = \{v_1, v_4\}$ e $u = \{v_5\}$ e na terceira, $V_1 = \{v_1, v_4, v_5\}$ e $u = \{v_3\}$.

No entanto, nesse caso, não satisfazemos uma condição de PULL-2 e o algoritmo devolve a bipartição $\{V_1, V_2\}$, com $V_1 = \{v_1, v_4, v_5\}$ e $V_2 = \{v_2, v_3, v_6, v_7, v_8\}$.

Observamos que $\min\{w(V_1), w(V_2)\} = 9$ e que nesse caso o ótimo seria OPT $= 12$. Assim, a razão de aproximação dessa instância é $\frac{12}{9} = \frac{4}{3}$.

Quanto ao consumo de tempo do algoritmo, fazemos no máximo $|V_2|$ operações PULL-2 e observamos que cada operação PULL-2 leva tempo $O(|E|)$. Assim, o tempo de execução do algoritmo é $O(|V||E|)$, e concluímos a prova do teorema.

□

5.1.2 Grafos conexos

Na Seção 5.1.1 mostramos como encontrar uma $\frac{4}{3}$-aproximação para o MaxMin BCP$_2$ restrito a grafos 2-conexos. Mostramos agora como resolver esse problema para grafos conexos quaisquer, mantendo a mesma razão de aproximação. O algoritmo que apresentamos aqui foi desenvolvido por Chlebíková (1996).

Antes de descrevê-lo, precisamos apresentar as seguintes definições e resulta-

dos:

Definição 7. *Seja $G = (V, E)$ um grafo conexo e $w : V \to \mathbb{Z}_+$ uma função peso. Seja A_G o conjunto de pontos de articulação de G e $\mathcal{H}_G$ o conjunto de blocos de G. Definimos $T_G = (V(T_G), E(T_G))$ como a árvore de blocos de G em que $V(T_G) := A_G \cup \mathcal{H}_G$ e $E(T_H) := \{\{a, H\} | H \in \mathcal{H}_G \text{ e } a \in V(H) \cap A_G\}$.*

Seja $\mathcal{B}(H, v) := \{\bigcup V(\tilde{H}) | \tilde{H} \in \mathcal{H}_G$ e $\tilde{H}$ pertence à subárvore de $T_G - \{v, H\}$ que contém $v\}$, em que $H \in \mathcal{H}$ e $a \in V(H) \cap A_G$.

Para cada bloco $H \in \mathcal{H}_G$, definimos uma função peso w_H nos vértices de H da seguinte forma:

$$w_H(v) = \begin{cases} w(v) & \text{se } v \in V(H) \setminus A_G \\ \{\sum_u w(u) | u \in \mathcal{B}_G(H, v)\} & \text{se } v \in V(H) \cap A_G \end{cases} \tag{5.2}$$

Lema 12. *Seja $G = (V, E)$ um grafo conexo com $|V| \geq 2$ e $\{V_1, V_2\}$ uma bipartição conexa de V. Existe um único bloco $H = (V(H), E(H))$ de G tal que ambos $V_1 \cap V(H)$ e $V_2 \cap V(H)$ não são vazios. Ademais, $\{V_1 \cap V(H), V_2 \cap V(H)\}$ é uma bipartição conexa de H.*

Lema 13. *Seja $G = (V, E)$ um grafo conexo com $|V| \geq 2$, H um bloco de G e $\{\tilde{V}_1, \tilde{V}_2\}$ uma bipartição conexa de H. Então existe apenas uma bipartição conexa $\{V_1, V_2\}$ de V tal que $\tilde{V}_1 = V_1 \cap V(H)$ e $\tilde{V}_2 = V_2 \cap V(H)$.*

Os dois Lemas acima sugerem um algoritmo para resolver o MaxMin BCP$_2$ para grafos conexos quaisquer, reduzindo-os a problemas do mesmo tipo em seus blocos. O algoritmo está descrito a seguir.

Algoritmo 3 APPROX-3

Entrada: Grafo conexo $G = (V, E)$ e função peso $w : V \to \mathbb{Z}_+$
Saída: 2-partição conexa $\{V_1, V_2\}$ de V

1: Seja A_G o conjunto de pontos de articulação de G
2: Seja $\mathcal{H}_G$ o conjunto de blocos de G
3: Construa T_G usando A_G e $\mathcal{H}_G$
4: Associe o peso w_H a cada $H \in \mathcal{H}_G$
5: **para** cada $H \in \mathcal{H}_G$ **faça**
6: $\quad \{V_1^H, V_2^H\} \leftarrow$ APPROX-2(H, w_H)
7: Seja H_0 o bloco que maximiza $\min\{w_H(V_1^H, V_2^H)\}$
8: Reconstrua $\{V_1, V_2\}$ de V tal que $V_i^{H_0} = V_i \cap H_0$, $i = 1, 2$
9: **devolva** $\{V_1, V_2\}$

Teorema 14. *Para qualquer grafo $G = (V, E)$ com $|V| \geq 2$ e função peso $w : V \to \mathbb{Z}_+$, o algoritmo APPROX-3 devolve uma $\frac{4}{3}$-aproximação para o problema MaxMin BCP$_2$ em tempo polinomial.*

Demonstração. Claramente o algoritmo pode ser executado em tempo polinomial.

Chamemos de m_H o valor $\min\{w(V_1^H), w(V_2^H)\}$ da bipartição $\{V_1, V_2\}$ devolvida por APPROX-2 e seja OPT_H o peso da menor classe de uma bipartição ótima de H e w_H. Pelo Teorema 11, temos que $\text{OPT}_H \leq \frac{4}{3} m_H$.

Seja $\{V_1^*, V_2^*\}$ uma bipartição ótima para G e w. Pelo Lema 12, existe um bloco $H^* = (V^*, E^*)$ de G tal que $\{V_1^* \cap V(H^*), V_2^* \cap V(H^*)\}$ é uma bipartição conexa de H^*.

Pela definição de w_H na equação 5.2, temos que

$$\text{OPT} = \min\{w(V_1^* \cap V(H^*)), w(V_2^* \cap\ V(H^*))\} = \text{OPT}_{H^*}.$$

Então, se H_0 é o bloco escolhido por APPROX-3, temos que $\min\{V_1, V_2\} = m_{H_0} \geq m_{H^*}$. Portanto,

$$\text{OPT} = \text{OPT}_{H^*} \leq \frac{4}{3} m_{H^*} \leq \frac{4}{3} m_{H_0} = \frac{4}{3} \min\{V_1, V_2\}.$$

Dessa forma, o algoritmo APPROX-3 é uma $\frac{4}{3}$-aproximação para o MaxMin BCP$_2$.

□

Podemos observar que se, em vez de usarmos o algoritmo APPROX-2 como uma rotina de APPROX-3, usarmos qualquer outro algoritmo para grafos 2-conexos com razão de aproximação α, o algoritmo APPROX-3 também terá razão α. Por conta disso, temos o seguinte corolário

Corolário 1. *Se existe uma α-aproximação para o problema* MaxMin BCP$_2$ *restrito a grafos 2-conexos, então existe uma α-aproximação para grafos conexos arbitrários.*

Capítulo 6

Resultados de Inaproximabilidade

Neste capítulo apresentaremos alguns resultados de inaproximabilidade existentes na literatura para os problemas considerados neste texto.

As provas aqui apresentadas usam de reduções polinomiais e o problema que será utilizado para tais reduções é definido da seguinte forma:

Definição 8 (EXACT COVER)**.** *Dada uma coleção $\mathcal{C} = \{C_1, C_2, ..., C_{3q}\}$ de subconjuntos de um conjunto $X = \{x_1, x_2, ..., x_{3q}\}$ com $|C_j| = 3$, $j \in [3q]$ e cada elemento de X ocorrendo exatamente três vezes nos subconjuntos de $\mathcal{C}$, queremos saber se existe uma subcoleção $\mathcal{C}' = \{C_{j_1}, C_{j_2}, ..., C_{j_q}\}$ de $\mathcal{C}$ que cobre X exatamente, ou seja, cada elemento de X aparece exatamente uma vez em algum dos subconjuntos de $\mathcal{C}'$.*

Garey e Johnson (1990) mostraram que o problema acima é NP-difícil.

O primeiro resultado que apresentamos foi mostrado por Chlebíková (1996) e trata do problema BCP_2 em grafos conexos com pesos uniformes. Esse resultado afirma que aproximar o 1-MaxMin BCP_2 com garantia de erro absoluto $|V|^{1-\epsilon}$ para qualquer $\epsilon > 0$ é NP-difícil.

Vamos mostrar isso apresentando uma redução de EXACT COVER para o 1-MaxMin BCP_2. Primeiro mostramos como fazer essa redução, depois provamos alguns resultados intermediários, e, finalmente, provamos o resultado principal.

Definição 9. *Dada uma instância $\mathcal{C} = \{C_1, C_2, \ldots, C_{3q}\}$ e X de* EXACT COVER*, algum $\epsilon > 0$ e um inteiro $k > \frac{4}{\epsilon}$, construímos um grafo $G = (V, E)$ da seguinte forma:*

Primeiro construímos um grafo $\tilde{G} = (\tilde{V}, \tilde{E})$ de forma que o conjunto de vértices $\tilde{V}$ seja

$$\tilde{V} = X \cup \mathcal{C} \cup \{x_0, z\},$$

em que x_0 e z são novos vértices, e o conjunto de arestas $\tilde{E}$ seja

$$\tilde{E} = \bigcup_{j=1}^{3q}(\{(C_j, x_i)|x_i \in C_j\} \cup \{(C_j, z) \cup (C_j, x_0)\}).$$

Isto é, os vértices referentes a cada conjunto C_j, $j = 1, \ldots, 3q$ serão conectados com todos os elementos desse conjunto. Ademais, todos os vértices referentes aos conjuntos C_j são conectados tanto ao vértice x_0 quanto ao vértice z.

Para simplificar a escrita, definimos as seguintes variáveis:

$$r \coloneqq 4q^k,$$

$$p \coloneqq 8q^{k+1},$$

$$s \coloneqq 24q^{k+2} + 4q^{k+1}.$$

Finalmente, para obter o grafo G, adicionamos caminhos únicos em cada vértice de $\tilde{V}$ da seguinte forma:

- *Em cada vértice x_i, $i = 1, 2, \ldots, 3q$ conectamos um caminho*

$$(\beta_{i,2}, \beta_{i,3}, \ldots, \beta_{i,p}).$$

- *Em cada vértice C_j, $j = 1, 2, \ldots, 3q$ conectamos um caminho*

$$(\gamma_{j,2}, \gamma_{j,3}, \ldots, \gamma_{j,r}).$$

- *No vértice z conectamos um caminho*

$$(\alpha_2, \alpha_3, \ldots, \alpha_s).$$

Assim, observamos que o grafo $G = (V, E)$ formado é conexo, bipartido e tem número de vértices

$$|V| = s + (3q + 1)p + 3qr =$$
$$= 24q^{k+2} + 4q^{k+1} + (3q + 1) \cdot 8q^{k+1} + 3q \cdot 4q^k =$$
$$= 48q^{k+2} + 24qk + 1.$$

Essa redução pode ser feita em tempo polinomial.

Dados $\epsilon > 0$ e inteiro $k > \frac{4}{\epsilon}$ como na Definição 9, existe um número finito de instâncias para as quais a desigualdade

$$(48q^{k+2} + 24q^{k+1})^{1-\epsilon} < q^k \tag{6.1}$$

não vale. Podemos, então, nos restringir às instâncias do problema em que a desigualdade vale.

Afirmação 1. *Em uma instância X e $\mathcal{C}$ de* EXACT COVER*, se existe subcoleção $\mathcal{C}'$ de $\mathcal{C}$ que cobre X exatamente, então existe uma 2-partição conexa $\{V_1^*, V_2^*\}$ de G, gerado como na Definição 9, tal que $|V_1^*| = |V_2^*| = \frac{|V|}{2}$.*

Demonstração. Podemos criar uma 2-partição conexa $\{\{z\} \cup (\mathcal{C} \setminus \mathcal{C}'), \{x_0\} \cup X \cup \mathcal{C}\}$ de $\tilde{V}(\tilde{G})$. A partir dela, podemos tomar a 2-partição $\{V_1^*, V_2^*\}$ de $V(G)$ tal que $\{z\} \cup (\mathcal{C} \setminus \mathcal{C}') \subseteq V_1^*$ e $\{x_0\} \cup X \cup \mathcal{C} \subseteq V_2^*$. Além disso, os caminhos únicos adicionados a cada vértice de $\tilde{G}$ para formar o grafo G devem ficar na mesma classe V_1^* ou V_2^* que o vértice inicial do caminho.

Temos

$$|V_1^*| = 2q \cdot r + s = 24q^{k+2} + 12q^{k+1}$$
$$|V_2^*| = (3q+1) \cdot p + q \cdot r = 24q^{k+2} + 12q^{k+1}.$$

Concluímos então que $|V_1^*| = |V_2^*| = \frac{|V|}{2}$.

□

Afirmação 2. *Seja $\mathcal{C}$ e X uma instância de* EXACT COVER *e G o grafo formado como na redução apresentada na Definição 9. Seja $\{V_1, V_2\}$ uma 2-partição conexa desse grafo tal que $\min\{|V_1|, |V_2|\} \geq \frac{|V|}{2} - r + 1$. Ademais, considere que $z \in V_1$. Então, $\mathcal{C} \cap V_2$ é uma subcoleção que cobre X exatamente.*

Demonstração. Observamos que $\{x_0\} \cup X \subseteq V_2$, pois caso contrário, como z está em V_1 teríamos que

$$|V_1| \geq s + p + r = \frac{|V|}{2} + r.$$

Logo, $|V_2| \leq \frac{V}{2} - r$, uma contradição.

Como $X \subseteq V_2$, por conta da conexidade do subgrafo $G[V_2]$, temos que $|\mathcal{C} \cap V_2| \geq q$, sendo que $|\mathcal{C} \cap V_2| = q$ é possível apenas quando $\mathcal{C} \cap V_2$ cobre X exatamente.

No entanto, podemos observar que $|\mathcal{C} \cap V_2| > q$ implicaria em $|\mathcal{C} \cap V_1| \leq 2q - 1$, então $|V_1| \leq s + (2q-1) \cdot r = \frac{|V|}{2} - r$, uma contradição.

Dessa forma, $|\mathcal{C} \cap V_2| = q$ e $\mathcal{C} \cap V_2$ cobre X exatamente.

□

Afirmação 3. *Seja A um algoritmo que recebe como entrada um grafo conexo bipartido $G = (V, E)$ e devolve uma aproximação para o* 1-MaxMin BCP_2 *com garantia de erro absoluto $|V|^{1-\epsilon}$.*

Ao aplicar A no grafo G, obtido pela redução de uma instância $\mathcal{C}$ e X de EXACT COVER*, podemos decidir a existência de $\mathcal{C}' \subseteq \mathcal{C}$ que cobre X exatamente da seguinte forma: Se $A(G) \geq \frac{|V|}{2} - \frac{3r}{4}$, então existe tal $\mathcal{C}'$, por outro lado, se $A(G) \leq \frac{|V|}{2} - \frac{r}{4}$, tal $\mathcal{C}'$ não existe.*

Ademais, o algoritmo A não devolve nenhum valor no intervalo aberto $\left(\frac{|V|}{2} - \frac{3r}{4}, \frac{|V|}{2} - \frac{r}{4}\right)$.

Ou seja, o algoritmo A resolve o problema NP-completo EXACT COVER.

Demonstração. Seja $G = (V, E)$ o grafo obtido pela redução de uma instância $\mathcal{C}$, X de EXACT COVER. Seja $\{V_1, V_2\}$ uma 2-partição conexa de V.

Pela desigualdade 6.1 temos

$$|V|^{1-\epsilon} = (48q^{k+2} + 24q^{k+1})^{1-\epsilon} < q^k = \frac{r}{4}.$$

Se existe uma cobertura exata para $\mathcal{C}$, então, pela Afirmação 1, o valor máximo de $\min\{V_1, V_2\}$ é $\frac{|V|}{2}$ e a saída do algoritmo, $A(G)$, deve satisfazer

$$A(G) \geq \frac{|V|}{2} - |V|^{1-\epsilon} > \frac{|V|}{2} - \frac{r}{4}.$$

Se não existe cobertura exata para $\mathcal{C}$, então, pela Afirmação 2, o valor máximo de $\min\{V_1, V_2\}$ é menor ou igual a $\frac{|V|}{2} - r$ e a saída do algoritmo deve satisfazer

$$A(G) \leq \frac{|V|}{2} - r + |V|^{1-\epsilon} < \frac{|V|}{2} - \frac{3r}{4}.$$

□

A partir dessas afirmações, Chlebíková (1996) provou o seguinte teorema.

Teorema 15. *Seja G um grafo conexo com pesos unitários nos vértices.*

Para qualquer $\epsilon > 0$, aproximar $\max\min\{V_1, V_2\}$ *para uma partição conexa $\{V_1, V_2\}$, com garantia de erro absoluto $|V|^{1-\epsilon}$, é NP-difícil (inclusive para grafos bipartidos).*

Demonstração. Seja $\mathcal{C}$ e X uma instância de EXACT COVER. Pelas Afirmações 1, 2 e 3 concluímos que podemos resolver EXACT COVER por meio de uma redução polinomial a aproximar 1-MaxMin BCP_2 com garantia de erro absoluto $|V|^{1-\epsilon}$.

Como sabemos que EXACT COVER é NP-Completo, então segue que esse problema de aproximação é NP-difícil.

□

Agora mostramos o seguinte resultado obtido por Chataigner *et al.* (2007) também sobre o BCP_2, que trata de grafos 2-conexos.

Teorema 16. *A versão de decisão do BCP_2 é NP-completo no sentido forte para grafos 2-conexos.*

Demonstração. Para provar esse Teorema apresentaremos uma redução do problema EXACT COVER para o BCP_2.

Dada uma instância $(X, \mathcal{C})$ de EXACT COVER, seja $G = (V, E)$ o grafo gerado da seguinte forma:

$$V = X \cup \mathcal{C} \cup \{a, b\}$$

$$E = \bigcup_{j=1}^{3q} [\{C_j x_i | x_i \in C_j\} \cup \{C_j a\} \cup \{C_j b\}]$$

É fácil ver que G é 2-conexo e pode ser construído em tempo polinomial. Considere a função peso $w : V \to \mathbb{Z}_+$ tal que

$$w(v) = \begin{cases} 9q^2 + 2q & \text{se } v = a \\ 3q & \text{se } v = b \\ 3q & \text{se } v = x_i, i = 1, ..., 3q \\ 1 & \text{se } v = C_j, i = 1, ..., 3q \end{cases}$$

Assim, podemos observar que $w(V) = 2(9q^2 + 4q)$.

Vamos mostrar que $\mathcal{C}$ contém uma cobertura exata de X se e somente se G possui uma 2-partição conexa $\{V_1, V_2\}$ tal que $\min\{V_1, V_2\} \geq \frac{w(V)}{2}$.

Primeiro, mostramos que, dada uma cobertura exata $\mathcal{C}'$, temos uma 2-partição que respeita a condição estabelecida.

Seja $\mathcal{C}'$ uma cobertura exata de X e considere a 2-partição $\{V_1, V_2\}$ de V de maneira que $V_1 = \{a\} \cup \{C_j : C_j \notin \mathcal{C}\}$ e $V_2 = \{b\} \cup \{C_j, x_i : C_j \in \mathcal{C}' \text{ e } x_i \in C_j\}$.

Como $\mathcal{C}'$ possui q subconjuntos, então $w(V_1) = w(a) + 3q - q = 9q^2 + 4q = \frac{w(V)}{2}$ e $w(V_2) = \frac{w(V)}{2}$.

Agora, mostramos que dada uma 2-partição conexa de G, $\mathcal{C}$ contém uma cobertura exata.

Seja $\{V_1, V_2\}$ uma 2-partição conexa de G tal que $\min\{V_1, V_2\} \geq \frac{w(V)}{2}$, então, claramente, $\min\{V_1, V_2\} = \frac{w(V)}{2}$. Notamos que a e b não podem estar na mesma classe V_i, pois $w(a) + w(b) = 9q^2 + 5q > \frac{w(V)}{2}$. Suponha então, sem perda de generalidade, que $a \in V_1$ e $b \in V_2$. Nenhum vértice de X está em V_1, pois senão teríamos $w(V_1) \geq w(a) + 3q > \frac{w(V)}{2}$. Dessa forma, V_1 contém a e alguns vértices de $\mathcal{C}$.

Como $w(V_1) = \frac{w(V)}{2} = 9q^2 + 4q$, V_1 deve conter exatamente $2q$ vértices de $\mathcal{C}$. Assim, V_2 possui q vértices de $\mathcal{C}$. Como esses q vértices são independentes e os vértices em $X \cup \{b\}$ são independentes, é fácil verificar que esses q vértices de $\mathcal{C}$ em V_2 configuram uma cobertura exata de X.

□

Vamos chamar de DBCP$_q$ a versão de decisão do problema BCP$_q$. Podemos generalizar o Teorema anterior para grafos q-conexos e obter o seguinte resultado:

Teorema 17. *Para todo $q \geq 2$, a versão de decisão do BCP$_q$ é NP-completo no sentido forte para grafos q-conexos.*

Demonstração. Como já provamos para $q = 2$, suponha que $q \geq 3$. Vamos mostrar, por indução em q, que DBCP$_{q-1}$ pode ser reduzido a DBCP$_q$, ou seja, podemos resolver o problema de encontrar $q-1$ classes conexas balanceadas em uma grafo $(q-1)$-conexo, resolvendo o problema de encontrar q classes conexas balanceadas em um grafo q-conexo.

Seja $I = (G, w, m)$ uma instância de DBCP$_{q-1}$ que consiste de um grafo $(q-1)$-conexo $G = (V, E)$, uma função peso $w : V \to \mathbb{Z}_+$ e um inteiro positivo m. O objetivo é responder se a instância possui solução com medida pelo menos m, em que a medida é o peso da classe mais leve.

Precisamos construir uma instância $I' = (G', w', m)$ de $DBCP_q$. Faremos isso da seguinte forma:

O grafo G' terá conjunto de vértices V' e conjunto de arestas E'

$$V' = V \cup \{v'\} \text{ com um vértice } v' \text{que não está em V}$$

$$E' = E \cup \{v'u : u \in V\}$$

Além disso, vamos definir a função $w' : V \to \mathbb{Z}_+$ tal que $w'(v') = \frac{w(V(G))}{q-1}$ e $w'(v) = w(v) \forall v \in V$.

É fácil ver que podemos construir G' em tempo polinomial no tamanha da instância I. Além disso, observamos que, como v' está conectado com todos os vértices de G, G' é q-conexo.

Vamos mostrar que I de DBCP$_{q-1}$ possui uma $(q-1)$-partição conexa com medida pelo menos m se e somente se I' de $DBCP_q$ possui q-partição conexa com medida também pelo menos m.

Primeiro provamos que, se I possui uma $(q-1)$-partição conexa com medida pelo menos m, então I' possui q-partição conexa com medida pelo menos m.

Seja $P = \{V_1, ..., V_{q-1}\}$ uma $(q-1)$-partição conexa de G com medida pelo menos m. Então, claramente, $\{V_1, ..., V_{q-1}, \{v'\}\}$ é uma q-partição conexa de G' que possui medida pelo menos m, pois $\{v'\}$ será a classe de peso mínimo apenas se todas as outras tiverem exatamente o mesmo peso.

Agora vamos provar que se I' possui uma q-partição conexa com medida m' tal que $m' \geq m$, então I possui uma $(q-1)$-partição conexa com medida pelo menos m.

Suponha que $P' = \{V'_1, ..., V'_q\}$, e, sem perda de generalidade, que v' está na classe V'_q. Além disso, assumimos que a primeira classe é a mais leve, isto é, $w'(V'_1) \leq w'(V'_i)$ com $2 \leq i \leq q-1$. Como $w'(V'_q) \geq \frac{w(V')}{q}$, temos que essa classe só é a mais leve se todas as outras tiverem o mesmo peso que ela, assim $m' = w'(V'_1)$.

Vamos considerar uma nova classe $R = V'_q \setminus \{v'\}$. Caso R seja vazio, então podemos simplesmente remover a classe V'_q e, assim, $\{V'_1, ..., V'_{q-1}\}$ é uma $(q-1)$-partição conexa de $G = (V, E)$ com medida m'. Já se $R \neq \emptyset$, como G é $(q-1)$-conexo, podemos distribuir os vértices de R entre os conjuntos V'_i, com $1 \leq i \leq q-1$. Isso é feito de

forma a formar novos conjuntos $V_i' \cup R_i$ que induzem subgrafos conexos em G tal que $R = \bigcup_{i=1}^{q-1} R_i$.

Nesse caso, $\{V_1' \cup R_1, ..., V_{q-1}' \cup R_{q-1}\}$ será uma $(q-1)$-partição conexa de G. Essa partição terá medida pelo menos m', e como $m' \geq m$, a prova da afirmação está completa.

Para completar a prova, basta observar que para $q = 2$, a prova já foi feita no Teorema 16. Então, para $q \geq 3$, pela redução que foi apresentada, podemos concluir, por indução, que para todo $q \geq 2$, a versão de decisão do BCP_q é NP-completo no sentido forte para grafos q-conexos.

□

Podemos observar que do resultado obtido no Teorema 17, temos o seguinte resultado de inaproximabilidade

Corolário 2. *Para todo $q \geq 2$, o problema BCP_q restrito a grafos q-conexos não admite* FPTAS*, a não ser que* P = NP.

Um outro resultado apresentado por Chataigner *et al.*, 2007 é relacionado ao Teorema 15. Esse Teorema considera o erro absoluto do valor encontrado em relação ao ótimo. No entanto, podemos considerar a razão entre esse valor e o ótimo. Assim, obtemos um novo Teorema.

Para provar esse próximo Teorema, faremos uma redução de uma instância $(X, \mathcal{C})$ de EXACT COVER a um grafo $G = (V, E)$ e uma função peso $w : V \to \mathbb{Z}_+$ da seguinte maneira:

$$V = \mathcal{C} \cup X \cup \{a, b\}$$

$$E = \bigcup_{j=1}^{3q} [\{C_j x_i | x_i \in C_j\} \cup \{C_j a\} \cup \{C_j b\}]$$

$$w(v) = \begin{cases} 6q^3 + q^2 & \text{se } v = a \\ 2q^2 & \text{se } v = b \\ 2q^2 & \text{se } v = x_i, i = 1, ..., 3q \\ q & \text{se } v = C_j, i = 1, ..., 3q \end{cases}$$

A partir disso, provamos duas afirmações que serão importantes para a prova do Teorema.

Afirmação 4. *Se existe subcoleção $\mathcal{C}'$ de $\mathcal{C}$ que cobre X exatamente, então G possui uma 2-partição conexa com medida $\frac{w(V)}{2}$*

Demonstração. Dada uma cobertura exata $\mathcal{C}'$ de X, considere a 2-partição conexa $\{V_1, V_2\}$ de G com $V_1 = \{a\} \cup (\mathcal{C} \setminus \mathcal{C}')$ e $V_2 = \{b\} \cup X \cup \mathcal{C}'$. Claramente, temos que $w(V_1) = 6q^3 + 3q^2 = \frac{w(V)}{2} = w(V_2)$.

□

Afirmação 5. *Se G possui 2-partição conexa $\{V_1, V_2\}$ com medida pelo menos $\frac{w(V)}{2} - q + 1$, então existe subcoleção de $\mathcal{C}$ que cobre X exatamente.*

Demonstração. Seja $\{V_1, V_2\}$ uma 2-partição conexa de G com medida m tal que $m \geq \frac{w(V)}{2} - q + 1$.

Observamos que os vértices a e b não podem fazer parte da mesma classe V_i com $i \in \{1, 2\}$, pois caso isso ocorresse, a, b e C_j para algum $j \in \{1, ..., 3q\}$, pertenceriam ao mesmo conjunto, e então o peso desse conjunto seria pelo menos $6q^3+3q^2+q = \frac{w(V)}{2}+q$, só que, dessa forma, $m \leq \frac{w(V)}{2} - q$, uma contradição.

Suponha, sem perda de generalidade, que $a \in V_1$ e $b \in V_2$. Usando uma ideia similar, podemos provar que $X \subseteq V_2$. Como $G[V_2]$ deve ser conexo, $|\mathcal{C} \cap V_2| \geq q$, pois cada um dos $3q$ elementos de X vai estar conectado em no máximo 3 cláusulas C de $\mathcal{C}$.

No entanto, $|\mathcal{C} \cap V_2| > q$, então $|\mathcal{C} \cap V_2| \leq 2q - 1$e, nesse caso, $w(V_1) \leq 6q^3 + q^2 + (2q-1)q = 6q^3 + 3q^2 - q = \frac{w(V)}{2} - q$, que é uma contradição. Dessa forma, $|\mathcal{C} \cap V_2| = q$, logo $\mathcal{C} \cap V_2$ cobre X exatamente.

□

Dadas essas afirmações, obtemos o seguinte Teorema, mostrado por Chataigner *et al.* (2007), que é o mais forte conhecido até hoje.

Teorema 18. *Não existe algoritmo para o BCP_2, que devolve uma $(1+\epsilon)$-aproximação, com $\epsilon \leq \frac{1}{n^2}$, em que n é o número de vértices do grafo da entrada, a não ser que $P = NP$.*

Demonstração. Suponha que exista algoritmo A para BCP_2 para grafos com n vértices tal que A seja uma $(1 + \epsilon)$-aproximação com $\epsilon \leq \frac{1}{n^2}$.

Vamos mostrar que podemos usar esse algoritmo para resolver o problema EXACT COVER, que é uma contradição, a não ser que P = NP.

Seja $(X, \mathcal{C})$ uma instância de EXACT COVER com $\mathcal{C} = \{C_1, C_2, ..., C_{3q}\}$ e $X = \{x_1, ..., x_3q\}$.

Contruímos um grafo como descrito acima. Assim, observamos que $w(V) = 2(6q^3 + 3q^2)$ e $|V| = 6q + 2$.

Seja B o seguinte algoritmo, que resolve EXACT COVER:

Dado, $(X, \mathcal{C})$, seja $G = (V, E)$ um grafo construído como descrito acima. Aplique o algoritmo A em G. Se o algoritmo encontra uma 2-partição conexa com medida pelo menos $\frac{w(V)}{2} - q + 1$, então $\mathcal{C}$ tem cobertura exata. Já se encontra uma 2-partição com medida menor que essa, $\mathcal{C}$ não possui cobertura exata.

Vamos mostrar que esse algoritmo resolve EXACT COVER.

Suponha que $\mathcal{C}$ possui cobertura exata. Então, pela Afirmação 4, G possui partição 2-conexa com medida $\frac{w(V)}{2}$. Como A é uma $(1+\epsilon)$-aproximação, com $\epsilon \leq \frac{1}{n^2}$ e $n = |V|$,

então A devolve solução com medida $m \geq \frac{n^2}{n^2+1} \cdot \text{OPT}$. Como $\text{OPT} = \frac{w(V)}{2}$, segue que

$$m \geq \left(\frac{n^2}{n^2+1}\right) \cdot \frac{w(V)}{2} = \left(1 - \frac{1}{n^2+1}\right) \cdot \frac{w(V)}{2} = \frac{w(V)}{2} - \frac{w(V)}{2(n^2+1)}$$

Como $w(V) = 2(6q^3 + 3q^2)$ e $n = 6q + 2$, então $\frac{w(V)}{2(n^2+1)} < q$.

Se $\mathcal{C}$ não possuir cobertura exata, então, pela Afirmação 5, G possui apenas soluções com medida menor que $\frac{w(V)}{2} - q + 1$.

Dessa forma, claramente, o algoritmo A devolve uma solução com medida menor que $\frac{w(V)}{2} - q + 1$.

Como G pode ser construído em tempo polinomial no tamanho de G, segue que B é polinomial no tamanho de $(X, \mathcal{C})$ e isso conclui a prova do teorema.

□

Capítulo 7

Conclusão

Neste texto discutimos diversos resultados e algoritmos para os problemas MinMax BCP_q e o MaxMin BCP_q, tanto em suas versões sem pesos, quanto em suas versões com pesos. No entanto, não mencionamos todos os resultados existentes na literatura, e tratamos com detalhes apenas alguns, pois é um assunto amplamente estudado, e portanto, muito vasto.

Dos algoritmos e resultados de aproximabilidade que foram apresentados em detalhes, buscamos reproduzir as provas originais de maneira didática, para ajudar o leitor na compreensão dessas provas.

Nos Capítulos 4 e 5, buscamos descrever os algoritmos em alto nível, mas de maneira que não fosse difícil transformá-los em códigos funcionais. Já no Capítulo 6, nos concentramos em apresentar provas de alguns dos principais resultados de inaproximabilidade existentes na literatura.

Esse trabalho possibilitou-nos estudar diversos artigos sobre os tópicos aqui apresentados. Com isso, levou-nos a aprender diversos novos algoritmos e novas técnicas para se provar resultados de aproximabilidade e inaproximabilidade.

Esperamos que este trabalho seja útil não só para aqueles que têm interesse nos problema tratados aqui, mas também para aqueles que têm interesse em algoritmos de aproximação e provas de inaproximabilidade.

Referências

[Alimonti e Calamoneri 1999] P. Alimonti e T. Calamoneri. "On the complexity of the max balance problem". Em: *Argentinian Workshop on Theoret. Comput. Sc. (WAIT'99)*. 1999, pgs. 133–138 (citado na pg. 13).

[R. Becker *et al.* 2001] R. Becker, I. Lari, M. Lucertini e B. Simeone. "A polynomial-time algorithm for max-min partitioning of ladders". Em: *Theory of Computing Systems* 34.4 (2001), pgs. 353–374 (citado na pg. 13).

[Ronald I Becker e Perl 1983] Ronald I Becker e Yehoshua Perl. "Shifting algorithms for tree partitioning with general weighting functions". Em: *Journal of Algorithms* 4.2 (1983), pgs. 101–120. ISSN: 0196-6774 (citado na pg. 13).

[Ronald I. Becker, Lari *et al.* 1998] Ronald I. Becker, Isabella Lari, Mario Lucertini e Bruno Simeone. "Max-min partitioning of grid graphs into connected components". Em: *Networks* 32.2 (1998), pgs. 115–125 (citado na pg. 11).

[Ronald I. Becker, Schach *et al.* 1982] Ronald I. Becker, Stephen R. Schach e Yehoshua Perl. "A shifting algorithm for min-max tree partitioning". Em: *J. ACM* 29.1 (1982), pgs. 58–67 (citado na pg. 1).

[Chataigner *et al.* 2007] Frédéric Chataigner, Liliane R. B. Salgado e Yoshiko Wakabayashi. "Approximation and inapproximability results on balanced connected partitions of graphs". Em: *Discrete Mathematics & Theoretical Computer Science* 9.1 (2007) (citado nas pgs. 11, 12, 32, 35, 36).

[G. Chen *et al.* 2020] Guangting Chen *et al.* "Approximation algorithms for the maximally balanced connected graph tripartition problem". Em: *J. Comb. Optim.* (2020), pgs. 1–21. DOI: 10.1007/s10878-020-00544-w (citado na pg. 19).

[Y. Chen *et al.* 2019] Yong Chen, Zhi-Zhong Chen, Guohui Lin, Yao Xu e An Zhang. "Approximation algorithms for maximally balanced connected graph partition". Em: *International Conference on Combinatorial Optimization and Applications*. Springer. 2019, pgs. 130–141 (citado nas pgs. 2, 15, 17, 19, 20).

[Chlebíková 1996] Janka Chlebíková. "Approximating the maximally balanced connected partition problem in graphs". Em: *Information Processing Letters* 60.5 (1996), pgs. 225–230 (citado nas pgs. 2, 3, 12, 20, 23, 26, 29, 32).

[Dyer e Frieze 1985] M.E. Dyer e A.M. Frieze. "On the complexity of partitioning graphs into connected subgraphs". Em: *Discrete Applied Mathematics* 10.2 (1985), pgs. 139–153 (citado na pg. 11).

[Frederickson 1991] Greg N. Frederickson. "Optimal algorithms for tree partitioning". Em: *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '91. San Francisco, California, USA: Society for Industrial e Applied Mathematics, 1991, pgs. 168–177. ISBN: 0897913760 (citado na pg. 13).

[Frederickson e S. Zhou 2017] Greg N. Frederickson e Samson Zhou. "Optimal parametric search for path and tree partitioning". Em: *CoRR* abs/1711.00599 (2017). arXiv: 1711.00599. URL: http://arxiv.org/abs/1711.00599 (citado na pg. 13).

[Garey e Johnson 1990] Michael R. Garey e David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman I& Co., 1990. ISBN: 0716710455 (citado na pg. 29).

[Györi 1978] E. Györi. "On division of graph to connected subgraphs". Em: *Combinatorics (Proc. Fifth Hungarian Colloq., Koszthely, 1976)*. Vol. 18. Colloq. Math. Soc. János Bolyai. 1978, pgs. 485–494 (citado na pg. 12).

[Lovász 1977] László Lovász. "A homology theory for spanning tress of a graph". Em: *Acta Mathematica Academiae Scientiarum Hungarica* 30 (1977), pgs. 241–251 (citado na pg. 12).

[Lucertini *et al.* 1989] Mario Lucertini, Yehoshua Perl e Bruno Simeone. "Image enhancement by path partitioning". Em: *Recent Issues in Pattern Analysis and Recognition*. Springer Berlin Heidelberg, 1989, pgs. 12–22 (citado na pg. 1).

[Lucertini *et al.* 1993] Mario Lucertini, Yehoshua Perl e Bruno Simeone. "Most uniform path partitioning and its use in image processing". Em: *Discrete Applied Mathematics* 42.2 (1993), pgs. 227–256 (citado nas pgs. 1, 2).

[Maravalle *et al.* 1997] Maurizio Maravalle, Bruno Simeone e Rosella Naldini. "Clustering on trees". Em: *Computational Statistics & Data Analysis* 24.2 (1997), pgs. 217–234 (citado na pg. 1).

[D. Matić e Grbić 2020] D. Matić e M. Grbić. "Partitioning weighted metabolic networks into maximally balanced connected partitions". Em: *2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH)*. 2020, pgs. 1–6 (citado na pg. 1).

[Dragan Matić 2014] Dragan Matić. "A mixed integer linear programming model and variable neighborhood search for maximally balanced connected partition problem". Em: *Applied Mathematics and Computation* 237 (2014), pgs. 85–97. ISSN: 0096-3003. DOI: https://doi.org/10.1016/j.amc.2014.03.098. URL: https://www.sciencedirect.com/science/article/pii/S0096300314004603 (citado nas pgs. 13, 14).

[Miyazawa *et al.* 2021] Flávio K. Miyazawa, Phablo F.S. Moura, Matheus J. Ota e Yoshiko Wakabayashi. "Partitioning a graph into balanced connected classes: formulations, separation and experiments". Em: *European Journal of Operational Research* 293.3 (2021), pgs. 826–836. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2020.12.059. URL: https://www.sciencedirect.com/science/article/pii/S0377221720311218 (citado na pg. 14).

[Moura *et al.* 2021] Phablo F. S. Moura, Matheus J. Ota e Yoshiko Wakabayashi. *Approximation and parameterized algorithms to find balanced connected partitions of graphs*. 2021. arXiv: 2108.10398 [cs.DS] (citado nas pgs. 2, 19, 20).

[Nakano *et al.* 1997] Shin-ichi Nakano, Md.Saidur Rahman e Takao Nishizeki. "A linear-time algorithm for four-partitioning four-connected planar graphs". Em: *Information Processing Letters* 62.6 (1997), pgs. 315–322 (citado na pg. 13).

[Suzuki *et al.* 1990] H Suzuki, N Takahashi, T Nishizeki, H Miyano e S Ueno. "An algorithm for tripartitioning 3-connected graphs". Em: *Journal of Information Processing Society of Japan* 31.5 (1990), pgs. 584–592 (citado na pg. 13).

[Hitoshi Suzuki *et al.* 1990] Hitoshi Suzuki, Naomi Takahashi e Takao Nishizeki. "A linear algorithm for bipartition of biconnected graphs". Em: *Information Processing Letters* 33.5 (1990), pgs. 227–231. ISSN: 0020-0190. DOI: https://doi.org/10.1016/0020-0190(90)90189-5. URL: https://www.sciencedirect.com/science/article/pii/0020019090901895 (citado na pg. 13).

[Wang *et al.* 2013] Lele Wang, Zhao Zhang, Di Wu, Weili Wu e Lidan Fan. "Max-min weight balanced connected partition". Em: *Journal of Global Optimization* 57.4 (dez. de 2013), pgs. 1263–1275. ISSN: 1573-2916. DOI: 10.1007/s10898-012-0028-8. URL: https://doi.org/10.1007/s10898-012-0028-8 (citado na pg. 11).

[Wu 2012] Bang Ye Wu. "Fully polynomial-time approximation schemes for the max-min connected partition problem on interval graphs". Em: *Discrete Mathematics, Algorithms and Applications* 04.01 (2012), pg. 1250005 (citado na pg. 11).

[X. Zhou *et al.* 2019] Xing Zhou, Huaimin Wang, Bo Ding, Tianjiang Hu e Suning Shang. "Balanced connected task allocations for multi-robot systems: an exact flow-based integer program and an approximate tree-based genetic algorithm". Em: *Expert Systems with Applications* 116 (2019), pgs. 10–20 (citado nas pgs. 1, 13, 14).