Programação matemática e imersões métricas para aproximações em problemas de corte

Santiago Viertel [1]
André Luís Vignatti [1]

Data submissão: 19.08.2014
Data aceitação: 07.03.2015

Resumo: Neste trabalho, apresentamos um estudo teórico sobre problemas de otimização NP-difíceis de cortes em grafos e o estado da arte de algoritmos de aproximação que fazem uso de técnicas de programação matemática e de espaços métricos. Três problemas de cortes em grafos foram modelados por programas matemáticos inteiros, sendo esses últimos relaxados para admitirem soluções com valores contínuos. A solução desses programas descrevem vetores unitários, para o problema do corte máximo; pontos sobre um $(k-1)$-simplexo, para o problema do corte multisseparador mínimo; e uma métrica, para o problema do corte mais disperso. Cada algoritmo apresentado realiza uma tomada de decisão com base na solução do programa matemático.

Abstract: In this work, we present a theoretical study about NP-hard optimization problems of cuts in graphs and the state of the art in approximation algorithms that use techniques of mathematical programming and of metric spaces. Three problems of cuts in graphs were modeled by integer mathematical programs and they are relaxed to admit solutions with continuous values. The solution of these programs describes, respectively, unit vectors on the maximum cut problem, points in a $(k-1)$-simplex on the multiway cut problem, and a metric on the sparsest cut problem. Each presented algorithm performs a decision based on the mathematical program solution.
Keywords: NP-hard optimization problems, approximation algorithms, mathematical programming, metric spaces, metric embeddings, randomized algorithms.

1 Introdução

O presente artigo apresenta um estudo teórico sobre algoritmos de aproximação aplicados em problemas NP-difíceis de cortes em grafos. Tais algoritmos são analisados com o

[1]Departamento de Informática, UFPR, Caixa Postal 19081
{sviertel,vignatti}@inf.ufpr.br

uso de técnicas de programação matemática, aleatoriedade, espaços métricos e imersões métricas. Estas técnicas vêm sendo amplamente estudadas e aplicadas no desenvolvimento de algoritmos de aproximação, como aqueles desenvolvidos por Fakcharoenphol, Rao e Talwar [7] e por Arora, Lee e Naor [1], ambos abordados por Williamson e Shmoys [20].

Estudos redigidos na língua portuguesa e que apresentam técnicas que fazem uso de programação matemática, métricas e imersões são atualmente escassos e de difícil acesso. Além disso, são apresentados algoritmos aleatorizados, que podem apresentar soluções que, na média dos casos, são melhores em comparação àquelas adquiridas de algoritmos determinísticos. Esse artigo explica e exemplifica de forma didática a aplicação dos conceitos mencionados no desenvolvimento de algoritmos de aproximação. Esse conteúdo pode servir como uma fonte inicial de pesquisa para estudantes de ciência da computação que estão iniciando os seus estudos em análise de algoritmos.

Existe uma necessidade de criar e estudar algoritmos que possuem execução rápida para que seja possível o processamento de grandes cargas em um curto período de tempo. Informalmente falando, um algoritmo possui execução em tempo hábil se ele é um *algoritmo polinomial*. Um algoritmo é polinomial se a função que determina o seu tempo de execução é um polinômio em função do tamanho da entrada do algoritmo. Melhores explicações sobre a classificação de algoritmos com base no tempo de execução podem ser encontradas no livro escrito por Garey e Johnson [8].

Todos os problemas apresentados são considerados *problemas de otimização*. Os problemas de otimização possuem como objetivo encontrar a melhor resposta, também chamada de *resposta ótima*, dentro de um conjunto de respostas possíveis. Quando o conjunto domínio de um problema de otimização é formado por variáveis discretas, então se tem um *problema de otimização combinatória*. Mais explicações sobre a classificação de problemas de otimização podem ser encontradas no livro escrito por Papadimitriou e Steiglitz [16].

Um problema de otimização combinatória possui um conjunto de elementos resposta para cada entrada diferente, essa última também chamada de *instância*. Nessa classe de problemas, existe uma função que associa cada solução da instância com um valor numérico denominado *valor da solução*. Os valores das soluções são racionais e não negativos para os problemas apresentados no presente artigo. O objetivo de problemas de otimização combinatória é, dada uma instância, encontrar a resposta que possui o menor valor da solução, para *problemas de minimização*; ou o maior valor da solução, para *problemas de maximização*. Tais conceitos foram baseados naqueles apresentados por Williamson e Shmoys [20].

Muitos problemas existentes na atualidade podem ser entendidos como problemas de otimização combinatória, que podem ser também problemas NP-*difíceis*. Informalmente falando, Garey e Johnson classificam um problema de otimização como NP-difícil se ele é tão difícil de solucionar quanto os problemas mais difíceis na classe NP [8]. Ainda não são

conhecidas soluções polinomiais para nenhum problema NP-difícil.

São inicialmente apresentados conceitos sobre três problemas de cortes em grafos, algoritmos de aproximação, programação matemática, espaços métricos e imersões métricas. Logo após, são mostrados três algoritmos de aproximação aleatorizados, um que soluciona o problema do corte máximo, outro que soluciona o problema do corte multisseparador mínimo e, por último, um que soluciona o problema do corte mais disperso.

2 Problemas estudados

Um *corte* em um grafo $G = (V, E)$ corresponde a uma partição do conjunto de vértices V em dois subconjuntos disjuntos U e $W = V - U$. Um *k-corte* é uma variação do conceito de corte que corresponde a uma partição do conjunto de vértices V em k subconjuntos disjuntos $C_1, \ldots, C_k$. Um *conjunto de corte* é formado por todas as arestas que possuem um de seus vértices pertencente a um subconjunto disjunto C_i e o outro pertencente a outro subconjunto disjunto C_j, sendo $i \neq j$.

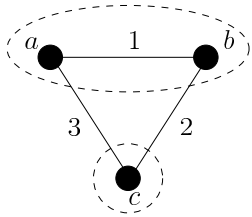
O *peso* do corte é definido como sendo a soma dos pesos das arestas pertencentes ao conjunto de corte. Será utilizado em alguns momentos o termo corte para denotar um k-corte e as notações *custo* e *capacidade* como sendo o peso.

Existem vários problemas que buscam cortes em grafos não direcionados. Um dos problemas mais conhecidos e estudados é o problema do corte máximo (*MAX CUT*).

Problema 1. *No* problema do corte máximo *(maximum cut problem) tem-se como entrada um grafo não direcionado $G = (V, E)$ e um peso não negativo w_e para cada aresta $e \in E$. O objetivo é particionar o conjunto de vértices V em dois subconjuntos U e $W = V - U$ de forma que seja maximizada a soma dos pesos das arestas que têm um vértice em U e o outro em W.*

A Figura 1 ilustra a explicação dada, onde as arestas que fazem parte do conjunto de corte F são $\{a, c\}$ e $\{b, c\}$ e seu peso é igual a 5. É possível perceber também que qualquer outro corte realizado no mesmo grafo possuirá peso menor do que 5.

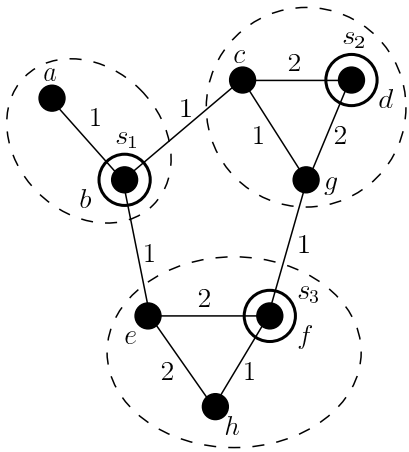
Este problema é NP-difícil, como provado por Karp [13], e pode ser aplicado, por exemplo, para criar grupos correlatos de pessoas em uma rede social. Cria-se um grafo que relaciona cada pessoa da rede com seus amigos, sendo que cada vértice corresponde a uma pessoa e cada aresta corresponde a uma relação de amizade entre duas pessoas. O peso das arestas determina o grau de proximidade entre as pessoas. Quanto menor o valor do peso da aresta, mais próximas as pessoas são. Ao executar o algoritmo do corte máximo nesse grafo, o algoritmo retornará dois grupos de pessoas que possuem mais proximidade. Outro problema estudado é o do corte multisseparador mínimo (*multiway cut*).



$U = \{a, b\}$ $W = \{c\}$
$F = \{\{a, c\}, \{b, c\}\}$

Figura 1. Corte máximo de um grafo

Problema 2. *No* problema do corte multisseparador mínimo *(multiway cut problem) é dado como entrada um grafo não direcionado $G = (V, E)$, custos não negativos c_e para cada aresta $e \in E$ e k vértices selecionados $s_1, \ldots, s_k \in V$. O objetivo é particionar o conjunto de vértices V em k subconjuntos disjuntos $C_1, \ldots, C_k$ de forma que cada vértice selecionado s_i pertença a um subconjunto C_i. Além disso, o k-corte encontrado deve ter o custo mínimo.*



$F = \{\{b, c\}, \{b, e\}, \{f, g\}\}$

Figura 2. Corte multisseparador mínimo de um grafo

Como existe um subconjunto para cada vértice selecionado, nesse problema busca-se não apenas uma bipartição do conjunto V, mas uma partição em k subconjuntos. A Figura

2 mostra um corte de custo 3 que particiona o conjunto de vértices V em três subconjuntos disjuntos diferentes: um que contém o vértice escolhido s_1, outro que contém o vértice s_2 e outro que contém o vértice s_3.

Dahlhaus, Johnson, Papadimitriou, Seymour e Yannakakis provaram que este problema é NP-difícil [5]. O problema do corte multisseparador mínimo tem aplicações na área de computação distribuída, como apresentado por Hogstedt, Kimelman, Rajan, Roth e Wegman [11]. Neste caso, cada vértice representa um objeto que deve ser armazenado em um computador. O custo de cada aresta representa a quantidade de comunicação necessária entre dois objetos. Todos os objetos precisam ser distribuídos em k computadores, em especial os objetos s_i que devem estar armazenados obrigatoriamente nos computadores i. Se dois objetos estão alocados em um mesmo computador, a comunicação entre eles é desconsiderada. O objetivo é distribuir os objetos nos k computadores de forma que a comunicação entre os computadores seja minimizada. Cada subconjunto C_i encontrado após a solução do problema lista os objetos que estarão armazenados em cada computador i. O último problema estudado é conhecido como o problema do corte mais disperso (*sparsest cut*).

Problema 3. *No* problema do corte mais disperso *(sparsest cut problem) é dado como entrada um grafo $G = (V, E)$ não direcionado, valores não negativos de capacidades c_e para cada aresta $e \in E$ e k pares de vértices selecionados (s_i, t_i) com demandas não negativas associadas d_i, sendo s_i e $t_i \in V$ para todo $1 \leq i \leq k$. Seja $F = \delta(S)$ o conjunto de corte formado por todas as arestas em E com somente um vértice em S, o objetivo é encontrar dois conjuntos de vértices $S \subseteq V$ e $V - S$ que minimiza a razão $\rho(S)$ apresentada na equação 1.*

$$\rho(S) = \frac{\sum_{e \in \delta(S)} c_e}{\sum_{i:|S \cap \{s_i, t_i\}|=1} d_i} \tag{1}$$

Analisando a equação 1, busca-se o corte com o valor mínimo da razão da capacidade total das arestas do conjunto de corte pela soma das demandas separadas por ele. Esse é um problema NP-difícil [8] e pode ser aplicado com o objetivo de identificar vias críticas que interligam um conjunto de várias cidades. As arestas correspondem às vias automobilísticas, os vértices correspondem às várias cidades interligadas e cada via e possui uma capacidade de fluxo de automóveis por minuto c_e. São presselecionados i pares de cidades polo que precisam ser interligadas (s_i, t_i) e é associada a cada par uma demanda de fluxo de carros d_i com origem na cidade s_i e destino na cidade t_i. É necessário identificar um conjunto de vias críticas que, se forem danificadas, o fluxo de carros que transita com origem e destino nas diferentes cidades polo decai substancialmente. Minimizar a razão entre o fluxo total cortado pelas vias críticas e o fluxo cortado entre as cidades polo garante a separação dos principais pares de cidades polo com a escolha do menor número de vias.

3 Fundamentação teórica

3.1 Algoritmos de aproximação

Definição 1. *Um* algoritmo α-aproximado *para um problema de otimização é um algoritmo polinomial que, para todas as instâncias do problema, ele produz uma solução cujo valor está em um fator α em comparação ao valor da solução ótima do problema.*

O fator α é denominado de *garantia de aproximação* do algoritmo. A garantia de aproximação expressa a proximidade entre o valor de uma resposta gerada pelo algoritmo e o valor da resposta ótima do problema. Os valores da garantia de aproximação são maiores do que 1 para problemas de minimização e menores do que 1 para problemas de maximização.

Seja $S(I)$ o valor de uma solução devolvida por um algoritmo de aproximação e $OPT(I)$ o valor da solução ótima. Caso o problema seja de maximização, então a inequação 2 é usada, caso seja de minimização, a inequação 3 é usada. As definições sobre algoritmos de aproximação se encontram formalizadas no livro escrito por Williamson e Shmoys [20].

$$\frac{S(I)}{OPT(I)} \geq \alpha, \ \forall I \in \mathcal{I} \tag{2}$$

$$\frac{S(I)}{OPT(I)} \leq \alpha, \ \forall I \in \mathcal{I} \tag{3}$$

Para exemplificar, um algoritmo $\frac{1}{2}$-aproximado é um algoritmo que soluciona um problema de maximização, que executa em tempo polinomial e gera uma solução que, para qualquer instância, tem valor de no mínimo metade do valor da solução ótima. Da mesma forma, um algoritmo 2-aproximado é um algoritmo que soluciona um problema de minimização, que executa em tempo polinomial e que, para qualquer instância, gera uma solução cujo valor é no máximo o dobro do valor da solução ótima.

3.2 Programação matemática

Existem problemas cujo objetivo é encontrar soluções que maximizam ou minimizam um determinado objetivo dados recursos limitados e restrições. Se for possível especificar o objetivo como uma função sobre variáveis que representam esses recursos e restrições em equações ou inequações sobre as mesmas variáveis, então possivelmente tem-se um problema de *programação matemática*.

Os tipos de programação matemática apresentados nesse documento são a *programação linear* e a *programação vetorial*. A **programação linear** possui a função objetivo

e as restrições definidas por funções lineares [16], enquanto que, na programação vetorial, podem existir restrições definidas por funções não lineares, além de características que serão melhor explicadas a seguir.

O principal objetivo da programação linear é encontrar um vetor não negativo x com valores em $\mathbb{Q}$ que minimiza ou maximiza uma dada função objetivo de x. Essa função objetivo deve respeitar um conjunto de restrições lineares que também são escritas em função de x. Dados um vetor c de tamanho n com valores pertencentes a $\mathbb{Q}$, um vetor b de tamanho m com valores pertencentes a $\mathbb{Q}$ e uma matriz $A = (a_{ij})$ de dimensão $m \times n$ com valores pertencentes a $\mathbb{Q}$, a *solução ótima* do programa linear

$$\begin{aligned} \text{minimizar (ou maximizar)} \quad & \sum_{j=1}^{n} c_j x_j & \\ \text{sujeito a} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, & i = 1, \ldots, m, \\ & x_j \geq 0, & j = 1, \ldots, n \end{aligned}$$

é um vetor x de tamanho n que minimiza a função objetivo $\sum_{j=1}^{n} c_j x_j$ sujeito ao conjunto de restrições $\sum_{j=1}^{n} a_{ij} x_j \geq b_i$, para $i = 1, \ldots, m$ e $x_j \geq 0$, para $j = 1, \ldots, n$.

O vetor x é chamado de *variável* do programa linear. Qualquer valoração de x que satisfaz todas as restrições é chamada de *solução factível*. Quando é encontrada uma solução factível que resulta no melhor valor possível para a função objetivo, então a solução encontrada é uma *solução ótima*. Quando uma solução ótima foi encontrada, o programa linear foi *solucionado*. Mais conceitos sobre programação linear podem ser encontrados no livro escrito por Papadimitriou e Steiglitz [16] e no livro escrito por Williamson e Shmoys [20].

A **programação vetorial** é um tipo de programação matemática cujas variáveis são vetores unidimensionais $v_i \in \mathbb{R}^n$ e as restrições e função objetivo são lineares sobre o produto escalar entre os vetores v_i. O valor n define tanto a dimensão dos vetores como a quantidade de vetores variáveis. No modelo de programação vetorial a seguir, o produto escalar entre v_i e v_j é denotado por $v_i \cdot v_j$.

$$\begin{aligned} \text{Minimizar ou maximizar} \quad & \sum_{i,j} c_{ij} (v_i \cdot v_j) & \\ \text{sujeito a} \quad & \sum_{i,j} a_{ijk} (v_i \cdot v_j) = b_k, & \forall k, \\ & v_i \in \mathbb{R}^n, & i = 1, \ldots, n. \end{aligned}$$

O método *simplex* [16] é um exemplo de algoritmo capaz de resolver programação linear que, apesar de demandar tempo exponencial no pior caso, possui um desempenho prático

satisfatório na maior parte das instâncias. Os métodos de *pontos interiores* [15] são algoritmos polinomiais que também podem ser aplicados, tanto para programação linear como para programação vetorial, neste último, com soluções que apresentam um fator de erro que pode ser desprezado para esse problema [20]. Um desses métodos é o *método dos elipsoides*, esse melhor abordado no trabalho de Grötschel, Lovász e Schrijver [10]. Melhores explicações sobre programação vetorial se encontram no livro escrito por Williamson e Shmoys [20].

Existe também outra variação dos programas apresentados conhecida como *programação matemática inteira*. Essa variação requer que todos os valores das componentes do vetor variável sejam inteiros. Neste caso, deve-se usar outros métodos para se obter a solução, pois demandam tempo exponencial no pior caso. De fato, o problema de solucionar programação matemática inteira é um problema de otimização NP-difícil [8].

3.3 Espaços métricos

Um *espaço métrico* é uma dupla (V, d), sendo V um conjunto de elementos e d uma função sobre os elementos em V. A função d é denominada *função de distância* e atribui um valor real positivo a todos os subconjuntos de dois elementos $\{i, j\}$ possíveis de serem formados com os elementos em V. Alguns algoritmos apresentados utilizam o conceito de um tipo mais específico de função de distância: as *métricas* [14].

Definição 2. *Uma* métrica d *é uma função* $d : V \times V \to \mathbb{R}_+$ *tal que:*

- $d(i, j) = 0$ *se e somente se* $i = j$*;*
- $d(i, j) = d(j, i)$ *para todos* i *e* $j \in V$*;*
- $d(i, j) \leq d(i, k) + d(k, j)$ *para todos* i, j *e* $k \in V$*.*

Uma métrica define as distâncias entre cada par de vértices em V. As métricas são simétricas e será utilizada a notação d_{ij} no lugar de $d(i, j)$ e de $d(j, i)$.

Os espaços métricos podem ser classificados em dois tipos distintos: os *espaços métricos finitos* e os *espaços métricos infinitos*. A principal diferença entre eles está na cardinalidade do conjunto V da dupla (V, d). Nos **espaços métricos finitos**, a função de distância d pode ser especificada por um conjunto de $\binom{n}{2}$ valores reais positivos dispostos em uma matriz de dimensão $n \times n$, como no exemplo mostrado na Tabela 1.

Cada célula da matriz que define a função de distância d faz referência a um par de elementos pertencentes a V. Pelo fato da função de distância ser simétrica, os elementos localizados abaixo da diagonal principal da matriz podem ser desconsiderados. A diagonal

d	v_1	v_2	v_3	v_4
v_1	0	1	2	6
v_2		0	3	7
v_3			0	4
v_4				0

Tabela 1. Função de distância d de um espaço métrico finito

principal é formada unicamente por elementos zerados, pois diz respeito às distâncias entre um elemento e ele mesmo.

Para **espaços métricos infinitos**, o conjunto de vértices V de (V, d) representa os infinitos pontos existentes em um espaço vetorial. Normalmente, este conjunto é definido por um conjunto $\mathbb{R}^k$, sendo k um valor inteiro maior ou igual a 1. A função de distância d normalmente é definida com alguma *norma de Minkowski* ℓ_p, sendo $1 \leq p \leq +\infty$. Uma norma é uma função que associa um vetor do conjunto V a um valor real não negativo denominado *comprimento*. Uma norma de Minkowski define a função de distância pela equação 4, onde $v \in V$ e $\|v\|_p$ denota o comprimento de v na norma ℓ_p. A distância entre dois vértices v_1 e $v_2 \in V$ é calculada como $\|v_1 - v_2\|_p$.

$$\|v\|_p = \left(\sum_{i=1}^{k} |v_i|^p \right)^{1/p} \tag{4}$$

Dois casos particulares da norma de Minkowski são o ℓ_1, também conhecido como distância de Manhattan e o ℓ_2, também conhecido como distância Euclidiana. Denotamos ℓ_p^n como sendo o espaço métrico infinito $(\mathbb{R}^n, \ell_p)$.

4 Imersões métricas

A Definição 3 formaliza os conceitos de *função de imersão* e de imersão *isométrica*.

Definição 3. *Dados dois espaços métricos (V, d) e (V', d'), imersão é uma função $f : V \to V'$. Uma imersão é isométrica se para todo v_1 e $v_2 \in V$, $d(v_1, v_2) = d'(f(v_1), f(v_2))$.*

De forma geral, uma *função de imersão* converte um vértice pertencente a um espaço métrico (V, d) em um vértice correspondente a outro espaço métrico (V', d'). Uma imersão é dita *isométrica* se não existe diferença entre os valores de distância dos dois espaços métricos após a imersão. Um espaço métrico (V, d) é *ℓ_p-imersivo* se existe uma função de imersão

isométrica $f : V \to \mathbb{R}^m$, sendo $m > 0$, do espaço métrico (V, d) em outro espaço métrico definido pela norma ℓ_p, ou seja, $d(u, v) = \|f(u) - f(v)\|_p$, sendo u e $v \in V$.

As distâncias podem ser tanto aumentadas como diminuídas conforme uma razão após uma imersão. Dados dois espaços métricos (V, d) e (V', d') e o mapeamento $f : V \to V'$, a *contração* de f é o fator máximo que as distâncias são diminuídas. Em contrapartida, a *expansão* de f é o fator máximo que as distâncias são aumentadas. Os cálculos do fator de contração β e do fator de expansão γ são mostrados nas equações 5 e 6, respectivamente.

$$\beta = \max_{v_1, v_2 \in V} \frac{d(v_1, v_2)}{d'(f(v_1), f(v_2))} \tag{5}$$

$$\gamma = \max_{v_1, v_2 \in V} \frac{d'(f(v_1), f(v_2))}{d(v_1, v_2)} \tag{6}$$

Apesar desses dois fatores medirem o fator máximo que as distâncias aumentam e diminuem após uma imersão, o real fator que determina o quanto um espaço métrico diferencia de outro é a *distorção*. O **fator de distorção** α é determinado pelo produto dos fatores de contração e expansão, ou seja, $\alpha = \beta \cdot \gamma$. O fator de distorção é invariante sobre a operação de escala das distâncias, ou seja, não há mudança no fator de distorção caso todas as distâncias de um espaço métrico sejam escaladas por um mesmo fator.

4.1 Tipos de imersões

Espaços métricos também podem ser definidos por árvores. Williamson e Shmoys [20] conceituam *métricas em árvores* como espaços métricos cujas distâncias são iguais ao tamanho do (único) menor caminho em uma árvore. A Figura 3(a) mostra graficamente uma métrica em árvore (V, d) e o mesmo espaço métrico visualizado hierarquicamente (b). Toda função de distância d modelada por uma árvore sobre o conjunto de vértices V é também uma métrica, pois para todos os vértices u, v e $w \in V$, $d_{uv} \leq d_{uw} + d_{wv}$ é verdadeiro e todas as distâncias d_{uv} são iguais a d_{vu}.

Métricas em árvores são interessantes de serem estudadas, pois podem ser visualizadas de forma hierárquica e possuem propriedades que podem ser utilizadas para criar algoritmos de aproximação. Além disso, existem problemas NP-difíceis em grafos que se tornam polinomiais se restritos às árvores. Um bom exemplo é apresentado na solução projetada por Awerbuch e Azar [3] para o problema NP-difícil *buy-at-bulk network*.

Não existem **imersões determinísticas** isométricas de espaços métricos finitos gerais em métricas em árvores [17]. Isso porque imersões de espaços métricos finitos definidos por ciclos em métricas em árvores, por exemplo, têm um fator de distorção de $\Omega(n)$, sendo n o

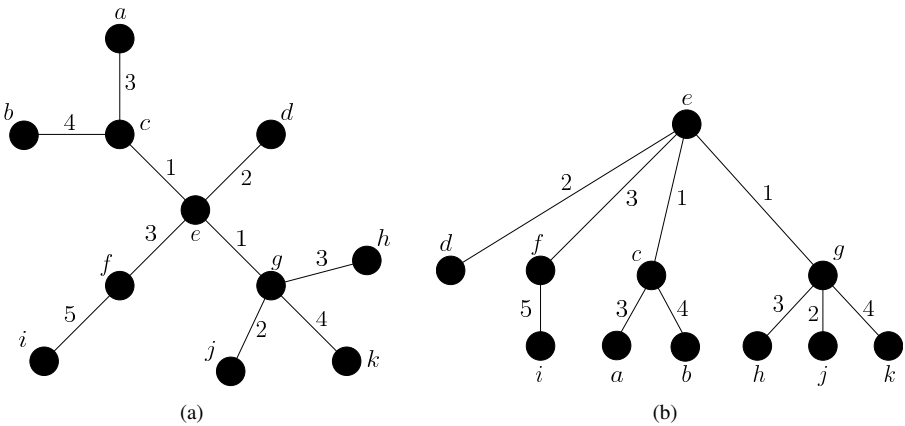


Figura 3. Métrica em árvore

número de vértices. O Teorema 1 foi provado por Rabinovich e Raz [17].

Teorema 1. *Um espaço métrico definido por um ciclo de comprimento n não pode ser imerso em outro espaço métrico definido por uma árvore com distorção menor do que $\Omega(n)$.*

Outra alternativa é projetar métodos aleatorizados de **imersão probabilística** e estudar o caso esperado. Dessa forma, é realizada uma análise probabilística sobre a distribuição de todas as árvores possíveis de serem geradas após a imersão de um espaço métrico finito geral em uma métrica em árvore. Pelo fato dessa análise ser probabilística, o valor de distorção α não é um valor determinístico, mas sim, um valor esperado, o que, de certa forma, evita o limitante inferior de pior caso de $\Omega(n)$ da contrapartida determinística.

4.2 Métricas em árvores e a norma ℓ_1

Fakcharoenphol, Rao e Talwar apresentaram um algoritmo de imersão de espaços métricos finitos gerais em métricas em árvores [7]. Esse procedimento é capaz de criar em tempo polinomial uma métrica em árvore que possui distorção esperada igual a $O(\log n)$.

A imersão em métricas em árvores é feita por um algoritmo que subdivide o espaço métrico que engloba todos os vértices contidos em V distanciados pela métrica d em outros subespaços menores. Cada subespaço encontrado é novamente dividido em outros subespaços. Essas divisões são realizadas até o momento em que cada subespaço contenha somente um único vértice de V. Ao final, tem-se uma árvore de subespaços que define em cada nível quais são os vértices pertencentes a V mais próximos entre si.

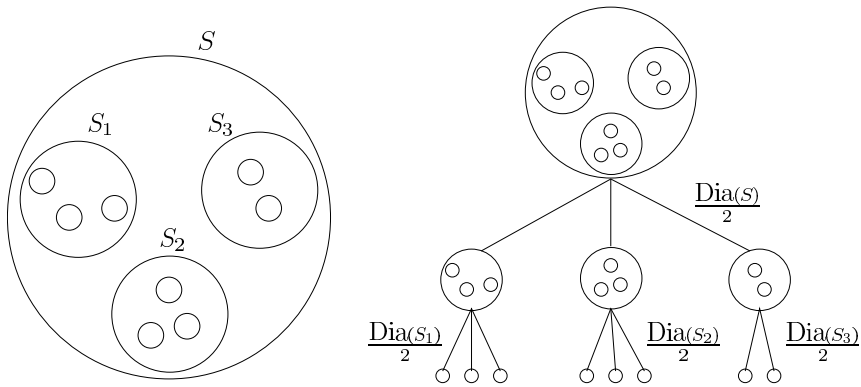


Figura 4. Decomposição hierárquica de cortes
[7]

Os autores referenciam esse processo de corte do conjunto de vértices baseado nas distâncias entre eles como *decomposição hierárquica de cortes*. A Figura 4 ilustra graficamente a explicação dada, onde um espaço métrico que define um conjunto S é subdividido em outros espaços métricos que definem os subconjuntos S_1, S_2 e S_3. Na mesma figura, $\frac{Dia(S)}{2}$ representa o valor numérico do raio da esfera que delimita o espaço métrico S. Uma melhor explanação sobre o algoritmo aleatorizado de imersão, bem como a prova do Teorema 2, podem ser encontradas no trabalho de Viertel [19].

Teorema 2. *Dado um espaço métrico finito (V, d), sendo $d_{uv} \geq 1$ para todo u e $v \in V$ e $u \neq v$, existe um algoritmo aleatorizado polinomial que gera uma métrica em árvore (V', T), sendo $V \subseteq V'$, de forma que $d_{uv} \leq T_{uv}$ e $\mathsf{E}[T_{uv}] \leq O(\log n) \cdot d_{uv}$.*

Um outro procedimento existente é a imersão isométrica de uma métrica em árvore (V, T) em espaços métricos definidos pela norma ℓ_1 que executa em tempo polinomial. O teorema a seguir mostra uma imersão isométrica de qualquer métrica em árvore no espaço métrico ℓ_1^{n-1}.

Teorema 3. *Toda métrica em árvore (V, T) é ℓ_1-imersível.*

A ideia da demonstração do Teorema 3 é a seguinte. Seja f uma função de imersão da métrica em árvore (V, T) no espaço métrico definido pela norma ℓ_1. Seja r o vértice raiz da árvore T e $\xi_{ur}(e)$ uma função que é igual a 1 se a aresta e da árvore T pertence ao caminho entre u e $r \in V$ na árvore T, ou igual a 0 caso contrário. Como T é uma árvore, então a quantidade de arestas em T é igual a quantidade de elementos em V menos 1, ou seja, $n - 1$.

Seja T_e o termo que representa o valor de peso da aresta e, a função $f : V \to \mathbb{R}^{n-1}$ é um mapeamento de forma que:

$$f(u) = (T_{e_1} \cdot \xi_{ur}(e_1), \ldots, T_{e_{n-1}} \cdot \xi_{ur}(e_{n-1})).$$

A imersão f associa um elemento $u \in V$ a um vetor de $n-1$ posições localizado no espaço métrico ℓ_1^{n-1}. Esse procedimento de imersão é executando em tempo polinomial e descreve uma imersão isométrica, sendo que a prova do Teorema 3 pode ser encontrada no trabalho de Viertel [19]. Essa imersão é importante no projeto de algoritmos de aproximação cuja entrada se comporta como espaços métricos. Um procedimento de imersão de espaços métricos finitos gerais em espaços métricos definidos pela norma ℓ_1 pode ser obtido com a aplicação de uma imersão de um espaço métrico finito geral (V, d) em uma métrica em árvore (V', T), seguido de uma imersão isométrica no espaço métrico definido pela norma ℓ_1.

5 Algoritmos de aproximação

Muitos problemas NP-difíceis podem ser modelados com programação matemática inteira. Apesar dela ser, em tese, capaz de encontrar a solução ótima, ela ainda não possui algoritmo que a soluciona em tempo polinomial. Porém, muitos autores apresentam técnicas que tomam decisões baseadas na resposta de um programa matemático cujas variáveis não necessariamente devem ser inteiras e que possui algoritmo polinomial que o soluciona. Nas seções a seguir, são apresentadas três soluções que fazem uso de tal abordagem, um algoritmo apresentado por Goemans e Williamson [9], outro por Călinescu, Karloff e Rabani [4] e, por último, um apresentado por Aumann e Rabani [2].

5.1 Corte máximo

Nesta seção, veremos uma forma de utilizar programação vetorial no desenvolvimento de um algoritmo de aproximação aleatorizado capaz de resolver o problema do corte máximo. Esse algoritmo foi apresentado por Goemans e Williamson [9] e possui o valor esperado de garantia de aproximação igual a 0,878. O programa não linear inteiro a seguir modela o problema do corte máximo, onde as variáveis y_i são iguais a -1 se o vértice $i \in U$ ou são iguais a $+1$ se o vértice $i \in W$.

$$\begin{array}{rll} \text{Maximizar} & \frac{1}{2} \sum\limits_{\{i,j\} \in E} w_{ij}(1 - y_i y_j) & \\ \text{sujeito a} & y_i \in \{-1, +1\}, & i = 1, \ldots, n. \end{array} \tag{7}$$

Lema 1. *O programa 7 modela o problema do corte máximo.*

Demonstração. Como $U = \{i : y_i = -1\}$ e $W = \{i : y_i = +1\}$, então quando uma aresta $\{i, j\}$ pertence ao conjunto de corte, o valor de $y_i y_j$ é igual a -1. Quando uma aresta não pertence ao conjunto de corte, então $y_i y_j$ é igual a $+1$. Na função objetivo, quando $\{i, j\}$ pertence ao conjunto de corte, o peso da aresta é multiplicado por 2 e acumulado com a mesma multiplicação realizada também com as outras arestas que fazem parte do conjunto de corte. Como todas as arestas que estão no conjunto de corte têm o peso multiplicado por 2, então o resultado do somatório é duas vezes a soma dos pesos das arestas que estão no conjunto de corte. A multiplicação por $1/2$ ao final resulta no valor exato do corte. □

O programa não linear inteiro 7 pode ser reescrito como um programa vetorial que admite valores contínuos como resposta. Esse programa vetorial é obtido substituindo o termo $y_i y_j$ pelo termo $v_i \cdot v_j$ na função objetivo e substituindo o termo $y_i \in \{-1, +1\}$ pelo termo $v_i \cdot v_i = 1$ no único conjunto de restrições. Exemplificando, um termo v_i é um vetor que pode ser calculado por $v_i = (y_i, 0, \ldots, 0)$ e $v_i \cdot v_j$ corresponde ao produto escalar entre os vetores v_i e v_j. É possível notar que, para o exemplo dado, $v_i \cdot v_i = 1$, sendo y_i igual a 1 ou igual a -1, e que $v_i \cdot v_j = y_i y_j$, possibilitando as duas substituições. Para finalizar, é adicionado um novo conjunto de restrições que permitem que as variáveis assumam valores pertencentes ao conjunto dos números reais, resultando, assim, no programa vetorial 8.

$$\begin{array}{rll} \text{Maximizar} & \frac{1}{2} \sum\limits_{\{i,j\} \in E} w_{ij}(1 - v_i \cdot v_j) & \\ \text{sujeito a} & v_i \cdot v_i = 1, & i = 1, \ldots, n, \\ & v_i \in \mathbb{R}^n, & i = 1, \ldots, n. \end{array} \tag{8}$$

O programa vetorial 8 é considerado uma relaxação do programa não linear inteiro 7, pois admite valores fracionários de coordenadas para os vetores variáveis v_i. Pelo fato do programa 8 ser modelado por variáveis que assumem valores fracionários, uma solução ótima para esse programa possivelmente não é uma solução factível para o problema original. O programa 8 é chamado de *programa vetorial relaxado* e pode ser solucionado em tempo polinomial. Para isso, ele pode ser reescrito como um programa semidefinido e submetido como entrada de um algoritmo que implementa o método dos elipsoides. Melhores explicações sobre a equivalência entre programas vetoriais e programas semidefinidos podem ser encontradas no trabalho de Viertel [19] e sobre o método dos elipsoides podem ser encontradas no trabalho de Grötschel, Lovász e Schrijver [10].

Imagine todos os n vetores unitários (são unitários porque $v_i \cdot v_i = 1$) encontrados na solução do programa 8 com suas origens posicionadas no centro de uma esfera. Essa esfera

é partida ao meio por um hiperplano posicionado exatamente sobre o seu centro e com o vetor normal igual a um vetor r. Os vetores que estão internos em uma partição da esfera são inclusos em U e os vetores resultantes, que estão internos na outra partição, são inclusos em W. A Figura 5 ilustra a esfera partida ao meio por um hiperplano aleatório.

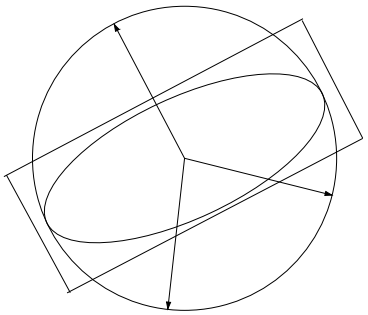


Figura 5. Esfera partida ao meio por um hiperplano aleatório
[20]

É possível obter um algoritmo de aproximação que soluciona o problema do corte máximo realizando um processamento sobre a solução do programa 8. Inicialmente o programa vetorial é solucionado, obtendo-se os vetores v_i^*. Logo após, é gerado um vetor $r = (r_1, \ldots, r_n)$ de forma que cada componente seja adquirida aleatoriamente de uma distribuição normal com média 0 e variância 1. Após o vetor r ter sido gerado, é realizada a normalização dele fazendo com que o seu módulo seja igual a 1. Finalmente a solução é obtida da seguinte maneira: se $v_i^* \cdot r \geq 0$ então $i \in U$ ou $i \in W$ caso contrário. Segue a definição do algoritmo aleatorizado [19].

Teorema 4. *O Algoritmo 1 é um algoritmo aleatorizado 0,878-aproximado para o problema do corte máximo.*

A prova do Teorema 4 pode ser encontrada no trabalho de Viertel [19]. Erdős apresentou um algoritmo aleatorizado para o mesmo problema que devolve uma solução com valor esperado igual a pelo menos metade da soma dos pesos das arestas [6]. Sahni e Gonzalez apresentaram um algoritmo determinístico com garantia de aproximação igual a $\frac{1}{2}$ [18]. Apesar da garantia de aproximação de ambos os algoritmos serem consideráveis, Goemans e Williamson desconfiam que não existe algoritmo com garantia de aproximação superior, a não ser que $\mathsf{P} = \mathsf{NP}$ [9].

Seja v^* uma solução ótima do programa vetorial relaxado 8;
para $i \leftarrow 1$ **até** n **faça**
    Faça r_i um valor aleatório amostrado de uma distribuição normal com média 0 e variância 1;
fim
$r = r/\|r\|$;
$U \leftarrow \emptyset$;
$W \leftarrow \emptyset$;
para $i \leftarrow 1$ **até** n **faça**
    se $v_i^* \cdot r \geq 0$ **então**
        $U \leftarrow U \cup \{i\}$;
    fim
    senão
        $W \leftarrow W \cup \{i\}$;
    fim
fim
retorna U e W;

Algoritmo 1: Algoritmo de aproximação para o problema do corte máximo

5.2 Corte multisseparador mínimo

Nesta seção, discutiremos um algoritmo aleatorizado $\frac{3}{2}$-aproximado que soluciona o problema do corte multisseparador mínimo usando a relaxação de um programa linear inteiro. Tal algoritmo foi apresentado por Călinescu, Karloff e Rabani [4]. Sendo esse um problema NP-difícil, como comprovado por Dahlhaus, Johnson, Papadimitriou, Seymour e Yannakakis [5], o simples fato de se utilizar um algoritmo que soluciona programação linear inteira torna a solução difícil computacionalmente. Isto aponta a necessidade de uma abordagem diferenciada para tratar o problema.

No programa matemático inteiro que modela esse problema, existem k variáveis diferentes $x_u^1, \ldots, x_u^k$ para cada vértice $u \in V$. Uma variável x_u^i é igual a 1 se e somente se o vértice u for atribuído ao subconjunto C_i na solução e 0 caso contrário. No mesmo programa existem também k variáveis $z_e^1, \ldots, z_e^k$ para cada aresta $e \in E$. Uma variável z_e^i é igual a 1 se e somente se a aresta e faz parte do conjunto de corte F e 0 caso contrário. Cada aresta pertencente ao conjunto de corte está obrigatoriamente em algum subconjunto C_i e em algum subconjunto C_j, sendo $i \neq j$. Assim sendo, a função objetivo do programa matemático inteiro é minimizar $\frac{1}{2} \sum_{e \in E} c_e \sum_{i=1}^{k} z_e^i$.

Cada vértice $u \in V$ precisa estar atribuído a exatamente um subconjunto C_i. Para

que isso seja garantido, cria-se o primeiro conjunto de restrições $\sum_{i=1}^{k} x_u^i = 1$, para todo $u \in V$. Para garantir que a variável z_e^i seja igual a 1 somente quando a aresta e fizer parte do corte, adiciona-se outro conjunto de restrições $z_e^i \geq \left|x_u^i - x_v^i\right|$, para toda aresta $e = \{u, v\} \in E$. Assim, a aresta e pertence ao conjunto de corte se u e v pertencerem a subconjuntos diferentes. Finalizando, são adicionados um conjunto de restrições que assegura que um vértice selecionado s_i pertença ao subconjunto C_i, $x_{s_i}^i = 1$, e outro que assegura que os valores das variáveis x_u^i sejam iguais a 0 ou 1, $x_u^i \in \{0, 1\}$. Segue o programa linear inteiro que modela o problema.

$$
\begin{array}{lll}
\text{Minimizar} & \frac{1}{2} \sum\limits_{e \in E} c_e \sum\limits_{i=1}^{k} z_e^i & \\
\text{sujeito a} & \sum\limits_{i=1}^{k} x_u^i = 1, & \forall u \in V, \\
& z_e^i \geq x_u^i - x_v^i, & \forall e = \{u, v\} \in E, \\
& z_e^i \geq x_v^i - x_u^i, & \forall e = \{u, v\} \in E, \\
& x_{s_i}^i = 1, & i = 1, \ldots, k, \\
& x_u^i \in \{0, 1\}, & \forall u \in V, i = 1, \ldots, k.
\end{array}
$$

O programa linear apresentado tem grande relação com o cálculo da distância na norma ℓ_1. Como a função objetivo busca minimizar z_e^i com valores não negativos, então pode-se assumir que $z_e^i = \left|x_u^i - x_v^i\right|$. Dada a equação 4, aplicada para o cálculo da distância na norma de ℓ_1, a função objetivo pode ser reescrita como $\frac{1}{2} \sum_{\{u,v\} \in E} c_{uv} \left\|x_u - x_v\right\|_1$. Com isso, os conjuntos de restrições $z_e^i \geq x_u^i - x_v^i$ e $z_e^i \geq x_v^i - x_u^i$ ficam implícitos na própria função objetivo.

O relaxamento do programa linear inteiro utiliza o conceito de n-simplexo. Um n-simplexo é um conjunto infinito de pontos que determina um politopo regular de $n + 1$ dimensões. A Definição 4 [20] formaliza esse conceito.

Definição 4. *Um n-simplexo é um subconjunto de pontos do $\mathbb{R}^{n+1}$ dados por:*

$$
\Delta_n = \left\{ (x^1, \ldots, x^{n+1}) \in \mathbb{R}^{n+1} \,\middle|\, \sum_{i=1}^{n+1} x^i = 1 \text{ e } x^i \geq 0 \text{ para todo } i \right\}.
$$

O politopo definido por um n-simplexo possui um total de $n + 1$ vértices (pontos extremais) definidos de tal forma que $e_1 = (1, 0, \ldots, 0)$, ..., $e_{n+1} = (0, 0, \ldots, 1)$.

Fazendo uso da Definição 4, o relaxamento do programa linear inteiro se inicia com a reescrita do conjunto de restrições $\sum_{i=1}^{k} x_u^i = 1$ como $x_u \in \Delta_{k-1}$. O conjunto de restrições

$x^i_{s_i} = 1$ é reescrito como $x_{s_i} = e_i$, fazendo com que cada vértice selecionado s_i seja posicionado em um dos pontos extremais do politopo formado pelo $(k-1)$-simplexo. Finalmente, o conjunto de restrições $x^i_u \in \{0,1\}$ é representado pela relaxação $x^i_u \geq 0$. Segue o programa linear relaxado que modela o problema.

$$\begin{array}{lll} \text{Minimizar} & \frac{1}{2} \sum\limits_{\{u,v\} \in E} c_{uv} \left\| x_u - x_v \right\|_1 & \\ \text{sujeito a} & x_u \in \Delta_{k-1}, & \forall u \in V, \\ & x_{s_i} = e_i, & i = 1, \ldots, k, \\ & x^i_u \geq 0, & \forall u \in V, i = 1, \ldots, k. \end{array} \tag{9}$$

Esse programa linear relaxado faz uso da norma ℓ_1 e encontra vetores que possuem suas caudas posicionadas na origem do sistema de coordenadas e o seu ponto final sobre o $(k-1)$-simplexo. Dessa forma, os vetores retornados como resposta do programa linear também podem ser visualizados como pontos sobre o $(k-1)$-simplexo. A Figura 6 ilustra a representação dos vetores por pontos sobre um 2-simplexo.

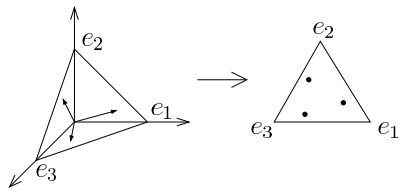


Figura 6. Vetores representados por pontos sobre um 2-simplexo

A ideia geral para encontrar um corte baseado na solução do programa linear relaxado consiste em atribuir ao subconjunto C_i os vértices representados pelos pontos mais próximos do ponto extremal que representa o vértice s_i no $(k-1)$-simplexo. Seja $B(s_i, r)$ o conjunto de vértices representados pelos pontos x_u internos a uma bola de raio r na norma ℓ_1 posicionada sobre o vértice e_i no $(k-1)$-simplexo. A bola B é formalmente definida como:

$$B(s_i, r) = \left\{ u \in V : \frac{1}{2} \left\| e_i - x_u \right\|_1 \leq r \right\}.$$

A sequência de execução do algoritmo aleatorizado começa com a determinação do valor do raio r, que é um valor aleatório amostrado de uma distribuição uniforme que varia entre 0 e 1. Logo após, determina-se aleatoriamente uma permutação π referente aos índices dos k subconjuntos. Para cada índice $\pi(i)$, com i iniciando em 1 e finalizando em $k -$

1, o algoritmo atribui a $C_{\pi(i)}$ todos os vértices representados pelos pontos pertencentes a $B(s_{\pi(i)}, r)$ que ainda não foram atribuídos a nenhum subconjunto anteriormente. Ao final, para que todos os vértices em V sejam atribuídos a um subconjunto, os vértices restantes, que não foram atribuídos a algum subconjunto, são atribuídos a $C_{\pi(k)}$. Segue o algoritmo que soluciona o problema do corte multisseparador mínimo, sendo $\delta(C_i)$ o conjunto de arestas que possui somente um vértice em C_i . O Algoritmo 2 e o Teorema 5 são melhor explicados no trabalho de Viertel [19] e também apresentados por Williamson e Shmoys [20].

Seja x uma solução do programa linear relaxado 9;
para $i \leftarrow 1$ **até** k **faça** $C_i \leftarrow \emptyset$;
Escolha $r \in (0, 1)$ aleatoriamente de uma distribuição uniforme;
Escolha uma permutação aleatória π de $\{1, \ldots, k\}$;
$X \leftarrow \emptyset$;
para $i \leftarrow 1$ **até** $k - 1$ **faça**
    $C_{\pi(i)} \leftarrow B(s_{\pi(i)}, r) - X$;
    $X \leftarrow X \cup C_{\pi(i)}$;
fim
$C_{\pi(k)} \leftarrow V - X$;
retorna $\bigcup_{i=1}^{k} \delta(C_i)$

Algoritmo 2: Algoritmo aleatorizado para o corte multisseparador mínimo

Teorema 5. *O Algoritmo 2 é um algoritmo aleatorizado $\frac{3}{2}$-aproximado para o problema do corte multisseparador mínimo.*

Karger, Klein, Stein, Thorup e Young [12] apresentaram um algoritmo que possui uma garantia de aproximação melhor em comparação ao algoritmo que utiliza programação linear apresentado por Călinescu, Karloff e Rabani [4]. Foi provado nesse trabalho que, para um corte multisseparador mínimo com 3 subconjuntos disjuntos, o algoritmo possui garantia de aproximação igual a $\frac{12}{11}$. Karger, Klein, Stein, Thorup e Young acreditam que essa garantia de aproximação é a melhor possível para algoritmos que fazem uso da resposta do programa linear relaxado 9.

5.3 Corte mais disperso

A métrica em árvore é também uma ferramenta poderosa que pode ser utilizada na elaboração de algoritmos de aproximação. Será apresentado nesta seção um algoritmo aleatorizado de aproximação que soluciona o problema do corte mais disperso com uso de imersão de espaços métricos finitos gerais em métricas em árvores.

O problema do corte mais disperso pode ser modelado pelo programa linear inteiro 10. A variável x_e é igual 1 se a aresta e pertence ao conjunto de corte ou é igual a 0 caso contrário. A variável y_i é igual 1 se o par (s_i, t_i) é separado pelo corte ou é igual a 0 caso contrário. A função objetivo corresponde à razão da soma das capacidades das arestas pertencentes ao conjunto de corte $\sum_{e \in E} c_e x_e$ pela soma das demandas separadas por ele $\sum_{i=1}^{k} d_i y_i$.

$$\begin{aligned} \text{Minimizar} \quad & \frac{\sum\limits_{e \in E} c_e x_e}{\sum\limits_{i=1}^{k} d_i y_i} \\ \text{sujeito a} \quad & \sum_{e \in P} x_e \geq y_i, \quad \forall P \in \mathcal{P}_i, 1 \leq i \leq k, \\ & x_e \in \{0,1\}, \quad \forall e \in E, \\ & y_i \in \{0,1\}, \quad i = 1, \ldots, k. \end{aligned} \tag{10}$$

O conjunto P contém as arestas que formam um caminho possível para enviar a demanda i no grafo, partindo do vértice s_i com destino ao vértice t_i. O conjunto $\mathcal{P}_i$ contém os conjuntos que formam todos os caminhos possíveis de interligar s_i a t_i no mesmo grafo. Um ponto importante sobre o programa linear 10 é que, mesmo se ele for relaxado, ainda existirá um número exponencial de restrições que representam cada um dos caminhos possíveis de serem gerados para todas as demandas. Aumann e Rabani [2] criaram o programa linear relaxado 11, que possui um número polinomial de restrições e é uma relaxação do programa 10 seguido de uma imersão no espaço ℓ_∞^k. Uma melhor explicação sobre os programas lineares e sobre a imersão no espaço métrico ℓ_∞^k pode ser encontrada no trabalho citado.

$$\begin{aligned} \text{Minimizar} \quad & \sum_{e \in E} c_e x_e \\ \text{sujeito a} \quad & \sum_{i=1}^{k} d_i y_i = 1, \\ & x_e \geq p_i^u - p_i^v, \quad \forall e = \{u, v\} \in E, i = 1, \ldots, k, \\ & x_e \geq p_i^v - p_i^u, \quad \forall e = \{u, v\} \in E, i = 1, \ldots, k, \\ & y_i \leq p_i^{t_i} - p_i^{s_i} \quad i = 1, \ldots, k, \\ & p_i^{t_i} \geq 0, \quad i = 1, \ldots, k, \\ & p_i^{s_i} \leq 0, \quad i = 1, \ldots, k. \end{aligned} \tag{11}$$

A variável x descreve uma métrica, onde x_e é igual a 1 se a aresta e pertence ao conjunto de corte ou é igual a 0 caso contrário. Porém, como o programa linear é uma relaxação, então os valores x_e não são obrigatoriamente iguais a 0 ou 1. Nesse caso, têm-se um espaço métrico finito geral (V, x) definido pela métrica x sobre os vértices em V.

Aumann e Rabani apresentaram um algoritmo polinomial que encontra o melhor corte caso o problema seja modelado por um espaço métrico definido pela norma ℓ_1. Como esse não é necessariamente o caso, para que o problema do corte mais disperso seja solucionado, basta realizar uma imersão do espaço métrico finito geral (V, x) em um espaço métrico finito que seja ℓ_1-imersível e submeter o espaço métrico resultante ao algoritmo apresentado pelos autores. Como o algoritmo encontra a resposta ótima, o fator de aproximação α será simplesmente igual a distorção da imersão de (V, x) em um espaço métrico finito ℓ_1-imersível. O Teorema 6 mostra a existência de tal imersão [19].

Teorema 6. *Existe um procedimento aleatorizado polinomial que realiza a imersão* $f : V \to \mathbb{R}^m$ *de um espaço métrico finito* (V, d) *no espaço métrico definido pela norma* ℓ_1 *com distorção esperada igual a* $O(\log n)$*, ou seja,* $\mathsf{E}[\|f(u) - f(v)\|_1] \leq O(\log n) \cdot d_{uv}$ *e* $d_{uv} \leq \|f(u) - f(v)\|_1$ *sendo* u *e* $v \in V$.

Aumann e Rabani apresentam uma imersão de (V, x) em um espaço métrico finito ℓ_1-imersível com distorção igual a $O(\log k)$, sendo k o número de demandas [2]. A solução mostrada no presente trabalho foi apresentada por Viertel [19]. Esta última imersão é realizada por duas imersões sequenciais, aquela realizada no Teorema 2 seguida da imersão isométrica realizada no Teorema 3. Mais explicações sobre o Teorema 6 e sobre as imersões citadas se encontram no trabalho de Viertel [19]. A Figura 7 ilustra a imersão apresentada por Aumann e Rabani [2] e a imersão por um caminho alternativo, apresentada por Viertel [19].

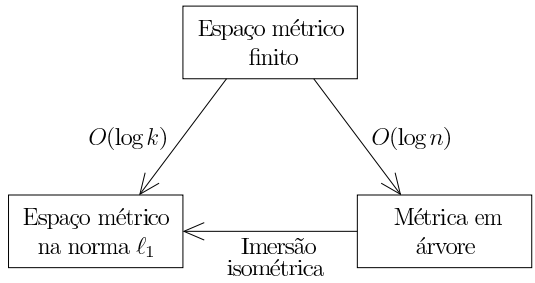


Figura 7. Imersões realizadas para solucionar o problema do corte mais disperso

A imersão do espaço métrico finito geral (V, x) no espaço métrico finito (p, ℓ_1), sendo p o conjunto que contém os pontos que representam cada vértice em V, não é o suficiente para solucionar o problema do corte mais disperso. Na verdade, apenas se obtém um espaço métrico finito conhecido como *métrica de corte*. Tal métrica pode ser decomposta em uma soma de pesos determinados pelos vários cortes possíveis de serem realizados no conjunto V. O algoritmo que soluciona esse problema realiza a decomposição do espaço métrico (p, ℓ_1).

Estando (V, x) imerso em (p, ℓ_1), todo o vértice de V tem um ponto que o representa em p. Sendo m o número de dimensões do espaço métrico após a imersão, então é possível decompor o espaço métrico finito (p, ℓ_1) em m dimensões. A disposição dos pontos em p no espaço métrico faz com que em cada dimensão m exista uma sequência distinta dos pontos em p, partindo daquele com o menor valor de coordenada com destino ao de maior valor. São gerados consecutivos cortes S com a realização de uma varredura em cada dimensão na norma ℓ_1. O algoritmo devolve como solução o subconjunto S que tem o menor valor de $\rho(S)$. A prova do Teorema 7 pode ser encontrada no trabalho de Viertel [19].

Teorema 7. *O algoritmo que realiza a decomposição de uma métrica de corte é um algoritmo aleatorizado polinomial com garantia de aproximação igual a $O(\log n)$ para o problema do corte mais disperso.*

Apesar dessa solução possuir garantia de aproximação igual a $O(\log n)$, Aumann e Rabani [2] apresentaram um algoritmo com garantia de aproximação igual a $O(\log k)$. Arora, Rao e Vazirani publicaram um algoritmo que soluciona o problema do corte mais disperso com uma garantia de aproximação melhorada de $O(\sqrt{\log n} \log \log n)$ [1].

6 Conclusão

O presente trabalho apresenta algumas abordagens adotadas para solucionar três problemas NP-difíceis de cortes em grafos. Algumas técnicas utilizadas podem ser listadas como importantes no desenvolvimento de algoritmos.

Modelar problemas por meio de programação matemática se mostrou presente em todos os casos. Foram realizados relaxamentos de programas matemáticos inteiros, gerando, assim, programas similares que adotam soluções contínuas. Esse método se mostrou eficaz na solução dos problemas. Em contrapartida, faz-se necessário um tratamento especial da resposta do programa relaxado.

Os algoritmos apresentados fazem uso da aleatoriedade para tomar decisões. Sendo o algoritmo aleatório, a única forma de apresentar uma garantia de aproximação é por meio do valor esperado da resposta devolvida. Dessa forma, existem soluções devolvidas que os seus valores vão além da garantia de aproximação. No entanto, a "média" das respostas está mais próxima do valor esperado de garantia de aproximação.

Outra ferramenta que mostrou a sua importância no projeto de algoritmos de aproximação foi a métrica. Bastou realizar uma imersão no espaço métrico definido pela norma ℓ_1 para que pudesse ser obtida uma resposta aproximada para o problema do corte mais disperso. O estudo de imersões é muito importante no projeto de algoritmos de aproximação e o algoritmo apresentado na seção 5.3 foi apenas um exemplo de aplicação. Outro exemplo, já

citado no presente trabalho, é o algoritmo que soluciona o problema *buy-at-bulk network* por meio de uma imersão de espaços métricos finitos gerais em métricas em árvores [3]. Uma importante descoberta na área de algoritmos é o resultado apresentado por Arora, Lee e Naor [1], que faz uso de imersões métricas para solucionar o problema do corte mais disperso.

7 Agradecimentos

Esse estudo teve o apoio financeiro do governo federal por meio do Programa de Apoio a Planos de Reestruturação e Expansão das Universidades Federais (Reuni).

Referências

[1] Sanjeev Arora, James R. Lee, and Assaf Naor. Euclidean distortion and the sparsest cut. In *In Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 553–562. ACM Press, 2005.

[2] Yonatan Aumann and Yuval Rabani. An O(log k) approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1):291–301, February 1998.

[3] Baruch Awerbuch and Yossi Azar. Buy-at-bulk network design. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 542–547, 1997.

[4] Gruia Călinescu, Howard Karloff, and Yuval Rabani. An improved approximation algorithm for MULTIWAY CUT. *Journal of Computer and System Sciences*, 60(3):564 – 574, 2000.

[5] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4):864–894, August 1994.

[6] P. Erdős. Gráfok páros körüljárású részgráfjairól (On bipartite subgraphs of graphs, em húngaro). *Matematikai Lapok*, 18:283–288, 1967.

[7] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, November 2004.

[8] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[9] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, November 1995.

[10] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[11] Karin Hogstedt, Doug Kimelman, V T Rajan, Tova Roth, and Mark Wegman. Graph cutting algorithms for distributed applications partitioning. *SIGMETRICS Perform. Eval. Rev.*, 28(4):27–29, March 2001.

[12] David R. Karger, Philip Klein, Cliff Stein, Mikkel Thorup, and Neal E. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, pages 668–678, New York, NY, USA, 1999. ACM.

[13] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[14] E.L. Lima. *Espaços métricos*. Projeto Euclides. Instituto de Matemática Pura e Aplicada, CNPq, 1977.

[15] Y. Nesterov and A. Nemirovskii. *Interior Point Polynomial Algorithms in Convex Programming*. SIAM studies in applied and numerical mathematics: Society for Industrial and Applied Mathematics. Society for Industrial and Applied Mathematics, 1994.

[16] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[17] Yuri Rabinovich and Ran Raz. Lower bounds on the distortion of embedding finite metric spaces in graphs. *Discrete & Computational Geometry*, 19(1):79–94, 1998.

[18] Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, July 1976.

[19] Santiago Viertel. Programação matemática e imersões métricas para aproximações em problemas de corte. Master's thesis, Universidade Federal do Paraná, May 2014.

[20] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2011.