Universidade Federal de São Carlos

Processo FAPESP #2019/24866-3

Algoritmos de Aproximação para Problemas de Corte em Grafos

Relatório Final

Orientador: Prof. Dr. Mário César San Felice
Bolsista: Esther Calderan Hoffmann

Julho de 2020 a Dezembro de 2020
São Carlos - SP, Brasil

Sumário

1 Introdução

Otimização combinatória é um ramo da ciência da computação que estuda problemas em que se busca maximizar ou minimizar uma determinada função. Como exemplo, considere o Problema do Corte Mínimo em grafos, cujo objetivo é desconectar um determinado par de vértices removendo um conjunto de arestas de menor custo.

A importância prática do problema pode ser exemplificada em sua origem, sendo inicialmente proposto durante a Guerra Fria. Relatos deste período mostram que a Força Aérea dos Estados Unidos da América coletava informações sobre a rede ferroviária soviética para determinar como os recursos eram enviados da União Soviética para a Europa. Em particular, estavam interessados em compreender qual era o número mínimo de lugares no sistema ferroviário que poderiam bombardear para impedir completa e precisamente o transporte entre tais regiões. Este cenário pode ser traduzido para o Problema do Corte Mínimo, ou seja, cortar o número mínimo de trilhos de trem de forma que nada possa ser entregue à Europa [11]. Atualmente, no entanto, existem aplicações muito mais pacíficas, como nas áreas de fluxo em redes, computação distribuída e segmentação de imagens.

O Problema do Corte Mínimo possui algoritmos conhecidos que podem resolvê-lo em tempo polinomial. Entretanto, neste projeto estamos particularmente interessados em generalizações NP-difíceis de problemas de corte. Uma delas é o Multiway Cut Problem, no qual há um grafo ponderado e k vértices distintos que desejamos separar. Mais especificamente, o objetivo é encontrar um conjunto de arestas com custo mínimo cuja remoção desconecta todos os k vértices.

Também temos interesse no Multicut Problem, uma generalização do anterior, no qual dado um grafo ponderado e um conjunto de k pares de vértices distintos, deseja-se encontrar um conjunto de arestas com custo mínimo cuja remoção desconecta cada um dos pares.

Ambos os problemas são importantes em diversos cenários de projetos de redes. Um exemplo comum é em computação distribuída, onde cada vértice representa um objeto, e uma aresta e de custo c_e representa a quantidade de comunicação entre os objetos. Os objetos precisam ser particionados para residir em k máquinas diferentes, com o objeto s_i precisando residir na i-ésima máquina. O objetivo é particionar os objetos nas k máquinas de forma que a comunicação entre elas seja minimizada.

Outro uso é na identificação de vulnerabilidade em redes, sendo que estas podem ser dos mais variados tipos: rede rodoviária, aérea, de tubulação, de computadores, entre outras. Assim como no cenário da Guerra Fria, o Multicut e o Multiway Cut podem ser usados para identificar gargalos por onde passam os caminhos que conectam determinados vértices de um grafo. Note que, para $k = 2$ no Multiway Cut e para um único par de vértices no Multicut, temos o Problema do Corte Mínimo.

Podemos também aplicá-los em problemas de agrupamento de dados. Como exemplo, cortes mínimos em grafos têm sido amplamente usados na área de visão computacional para segmentação de imagens [1, 9]. Neste cenário, pixels são representados por vértices, e o custo de uma aresta (u, v) corresponde à similaridade entre os pixels u e v.

Como dito anteriormente, tanto o problema do Multiway Cut quanto do Multicut são NP-difíceis, isto é, não podem ser resolvidos de forma ótima em tempo polinomial

a menos que P=NP. Por este motivo, estamos interessados em estudar algoritmos de aproximação, que são projetados para encontrar, em tempo polinomial, uma solução com alguma garantia de qualidade em relação à solução ótima.

Como exemplo, Dahlhaus et al. [3] obtiveram uma 2-aproximação para o Multiway Cut Problem utilizando, como sub-rotina, um algoritmo para o Problema do Corte Mínimo. Călinescu, Karloff e Rabani [2] propuseram um algoritmo de arredondamento probabilístico com razão de aproximação $\frac{3}{2}$ para o mesmo problema. Karger et al. [10] melhoraram a análise deste, obtendo tanto limitantes inferiores quanto limitantes superiores mais justos. Garg, Vazirani e Yannakakis [6] obtiveram uma $4(\ln k + 1)$-aproximação para o Multicut Problem, sendo k o número de pares da entrada, usando arredondamento probabilístico de programação linear em conjunto com a técnica de crescimento de regiões. Além disso, os mesmo autores desenvolveram um algoritmo primal-dual para o Multicut Problem em árvores [5].

Outro problema que temos interesse, e que é bastante relevante na prática, é o Balanced Cut Problem, no qual dado um grafo ponderado, o objetivo é pagar o menor custo de arestas para dividir um grafo em dois componentes conexos, S e $V - S$, de forma que ambas as partes possuam um determinado número mínimo de vértices. Encontrar cortes balanceados aparece, como exemplo, em esquemas de divisão e conquista. Nesse cenário, o corte é encontrado, algum problema em grafo é resolvido em S e $V - S$, e a solução é o resultado de combinar as duas soluções através das arestas entre eles. Se o custo do corte é baixo, a etapa de combinar é mais simples. Além disso, se S e $V - S$ possuem aproximadamente o mesmo tamanho, então o algoritmo pode ser aplicado recursivamente em cada lado, e a profundidade da recursão será $O(\log n)$. Para este problema, Even et al. [4] obtiveram um algoritmo de pseudo-aproximação utilizando a técnica de crescimento de regiões.

Desenvolvido Anteriormente. Na primeira etapa do projeto estudou-se o Problema do Corte Mínimo, sua relação com o Problema de Fluxo Máximo, e um algoritmo polinomial para o mesmo, sendo este um problema central no estudo de corte em grafos. Em seguida, foram analisados dois algoritmos de aproximação para o Multiway Cut [3, 2], um para o Multicut [6], e suas respectivas demonstrações de garantia de qualidade da solução. Por fim, foi iniciada a implementação do primeiro algoritmo estudado.

Organização do Relatório. Neste relatório, são apresentadas as atividades realizadas na segunda etapa do projeto. A Seção 2 contém a definição dos problemas estudados. Na Seção 3, são brevemente descritos os algoritmos vistos anteriormente, e é apresentado um algoritmo de pseudo-aproximação para o Problema do Balanced Cut, acompanhado de sua demonstração de garantia de qualidade da solução. A Seção 4 descreve a metaheurística BRKGA e como ela será aplicada para resolver o Problema do Multiway Cut. A Seção 5 descreve as implementações desenvolvidas e os testes realizados. Na Seção 6 encontra-se o cronograma atualizado do projeto. Por fim, a Seção 7 traz a conclusão.

2 Definições dos Problemas

No Problema do Multiway Cut, há um grafo $G = (V, E)$, com custo $c_e \geq 0$ para todas as arestas $e \in E$ e são dados k vértices distintos $s_1, \ldots, s_k$, também conhecidos como terminais. O objetivo é encontrar um conjunto F de arestas com custo mínimo cuja remoção desconecta todos os k vértices, i.e., nenhum par de vértices s_i e s_j, para $i \neq j$, pode estar no mesmo componente conexo de $(V, E - F)$.

No Problema do Multicut, é dado um grafo $G = (V, E)$ com custo $c_e \geq 0$ para todo $e \in E$ e um conjunto de k pares de vértices distintos (s_1, t_1), $\ldots$, (s_k, t_k). Deseja-se encontrar um conjunto F de arestas com custo mínimo cuja remoção desconecta todos os pares, i.e., para qualquer i, $1 \leq i \leq k$, não há um caminho de s_i até t_i em $(V, E - F)$.

No Problema do Balanced Cut, é dado um grafo $G = (V, E)$ com custo $c_e \geq 0$ para todo $e \in E$, e um parâmetro $b \in (0, 1/2]$. Dizemos que um conjunto de vértices S é um *corte $b-$balanceado* se $\lfloor bn \rfloor \leq |S| \leq \lceil (1-b)n \rceil$, onde $n = |V|$. Portanto, b indica quão desbalanceado pode ser um corte que separa o grafo em dois conjuntos de vértices. O objetivo é encontrar tal corte S que minimize o custo das arestas com exatamente um vértice em S. O caso em que $b = 1/2$ é chamado de *problema da bissecção mínima*.

Os três problemas descritos são NP-difíceis, ou seja, não podem ser solucionados em tempo polinomial, a não ser que P = NP. Sendo assim, se for encontrado um algoritmo que o faça, todos os problemas em NP poderiam também ser solucionados em tempo polinomial. Apesar dessa dificuldade de obter a solução em tempo polinomial, muitos problemas NP-difíceis possuem algoritmos de aproximação.

Definição 2.1 (Algoritmo de aproximação)**.** *Seja OPT(I) o valor ótimo para uma instância I de um problema de otimização. Um algoritmo é dito uma α-aproximação para esse problema se, para qualquer instância I, ele produz uma solução de valor $ALG(I)$ em tempo polinomial tal que*

$$ALG(I) \leq \alpha OPT(I) \ .$$

O Problema do Multiway Cut é um problema APX-difícil, significando que há uma constante $\delta > 1$ tal que é NP-difícil aproximar sua solução dentro de uma razão menor que δ. Como o Multiway Cut é um caso particular do Multicut, este último também é APX-difícil.

3 Algoritmos para Problemas de Corte em Grafos

Nesta seção, são apresentados quatro algoritmos de aproximação para problemas de corte em grafos: o algoritmo baseado no corte mínimo para o Multiway Cut, por Dahlhaus et al. [3], o algoritmo de arredondamento probabilístico de programação linear para o Multiway Cut, por Călinescu, Karloff e Rabani [2], o algoritmo de arredondamento probabilístico com crescimento de regiões para o Multicut, por Garg, Vazirani e Yannakakis [6], e o algoritmo de arredondamento probabilístico com crescimento de regiões para o Balanced Cut, por Even et al. [4].

Foi utilizado como principal material de apoio para o estudo desses algoritmos o livro "The Design of Approximation Algorithms", de Williamson e Shmoys [13]. Os três primeiros algoritmos mencionados foram descritos detalhadamente no relatório anterior,

acompanhados das demonstrações para suas razões de aproximação. Por isso, esta seção tratará brevemente desses algoritmos, focando maior atenção ao ainda não detalhado.

3.1 Estudados anteriormente

Algoritmo para o Multiway Cut Baseado no Corte-Mínimo. O algoritmo baseado no problema o corte mínimo, proposto por Dahlhaus et al. [3], é uma 2-aproximação para o problema do multiway cut. Considere qualquer solução válida F. Dado este conjunto de arestas $F \subseteq E$ que separa os terminais, chamaremos de C_i o conjunto de vértices alcançáveis em $(V, E - F)$, por cada terminal s_i. Definimos $F_i = \delta(C_i)$, onde $\delta(S)$ é o conjunto de todas as arestas que possuem apenas um vértice em S. Observe que cada F_i é um corte separando s_i dos demais terminais. Chamaremos F_i de um corte de isolamento. O algoritmo irá computar um corte de isolamento mínimo F_i entre s_i e $s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_k$, para cada $0 \leq i \leq k$. Para isto, basta transformar nosso grafo em um grafo direcionado, adicionar um vértice sorvedouro T e arestas de custo infinito indo dos vértices s_j ao T, para todo $j \neq i$, e então computar um $s_i - T$ corte mínimo, que será nosso C_i. Ao adicionarmos o sorvedouro T com tais arestas, o fluxo máximo de s_i à T será limitado apenas pela capacidade das arestas originais do grafo. Portanto, ao usar um algoritmo para o cálculo do $s_i - T$ corte mínimo, encontraremos o corte mínimo de isolamento de s_i. Após calcularmos F_i para $i = 1, \ldots, k$, teremos como solução $\cup_{i=1}^{k} F_i$. Entretanto, note que a união dos primeiros $k - 1$ cortes de isolamento é também uma solução viável para o problema. Na verdade, qualquer F_i pode ser removido da solução, que ela continuará viável, pois para todo $j \neq i$, F_j separa os vértices s_j e s_i. Como deseja-se a solução menos custosa, o algoritmo removerá, ao final, o F_i de maior valor.

Arredondamento Probabilístico de Programação Linear para o Multiway Cut. O algoritmo de arredondamento probabilístico de programação linear, proposto por Călinescu, Karloff e Rabani [2], é uma $\frac{3}{2}$-aproximação para o problema do multiway cut. Uma forma de interpretar o problema do multiway cut pode ser encontrar uma partição ótima dos vértices V nos conjuntos C_i, tal que $s_i \in C_i$ para todo i, e tal que o custo de $F = \cup_{i=1}^{k} \delta(C_i)$ seja minimizado. Para isto, é utilizada uma modelagem de programação linear inteira para o problema, tendo como variáveis de decisão x_u^i, para todo $u \in V$ e $i = 1, \ldots\ k$, que assume valor 1 caso o vértice u esteja no conjunto C_i, ou 0 caso contrário. A partir da solução da relaxação deste programa linear, podendo ser fracionários os valores indicando a qual conjunto C_i um vértice qualquer pertence, utiliza-se um algoritmo de aproximação baseado no arredondamento probabilístico desta solução para encontrar o corte que desejamos. Para isso, é gerada uma permutação aleatória π de $\{1, \ldots, k\}$, e escolhido um $r \in (0, 1)$ uniformemente ao aleatório. Em seguida, itera-se do primeiro até o penúltimo elemento de π. Em cada iteração i, são associados à $C_{\pi(i)}$ todo vértice u tal que $\frac{1}{2}|| \ s_i \ - \ x_u \ ||_1 \leq r$, com exceção dos vértices já associados à outro conjunto anteriormente. Ao final das $k-1$ iterações, os vértices restantes são associados ao conjunto C_k, e em seguida calcula-se o custo da solução, ou seja, o custo de F, com $F = \cup_{i=1}^{k} \delta(C_i)$.

Arredondamento Probabilístico de Programação Linear com Crescimento de Regiões para o Multicut. O algoritmo de arredondamento probabilístico com cresci-

mento de regiões, por Garg, Vazirani e Yannakakis [6], é uma $(4\ln(k+1))$-aproximação para o problema do multicut. Primeiro, é construída uma formulação do problema como programa linear inteiro, onde x_e é a variável de decisão binária, que assume valor 1 caso a aresta e esteja no corte e 0 caso contrário. A partir da solução fracionária ótima x do programa linear fruto da relaxação, iremos novamente utilizar um algoritmo de arredondamento de x para definir o conjunto F das arestas do corte. Além disso, será útil imaginar o grafo como uma rede de canos. O comprimento dos canos, ou arestas, será denotado por x_e, e sua área transversal por c_e. Assim, $x_e c_e$ corresponde ao volume da aresta e. Faremos $d_x(u,v)$ denotar o comprimento do menor caminho de u a v. Chamaremos de $B_x(s_i,r) = \{v \in V : d_x(s_i,v) \leq r\}$ a bola de raio r ao redor do vértice s_i, e de $V_x(s_i,r)$ o volume dos canos dentro de uma distância r de s_i, mais o termo extra $\frac{V^*}{k}$; isto é

$$V_x(s_i,r) = \frac{V^*}{k} + \sum_{e=(u,v):\ u,v\ \in\ B_x(s_i,r)} c_e x_e + \sum_{e=(u,v):\ u\ \in\ B_x(s_i,r),\ v\ \notin\ B_x(s_i,r)} c_e(r - d_x(s_i,u))$$

O algoritmo inicializa F vazio, e em seguida realiza k iterações. Em cada iteração i, se s_i e t_i estão conectados em $(V, E-F)$, é encontrado um raio $r < \frac{1}{2}$ ao redor de s_i tal que $c(\delta(B_x(s_i,r))) \leq (2\ \ln(k+1))\ V_x(s_i,r)$. Em seguida, adiciona-se em F as arestas de $\delta(B_x(s_i,r))$, e remove-se $B_x(s_i,r)$ e arestas incidentes do grafo.

3.2 Algoritmo para o Problema do Balanced Cut

Nesta subseção abordaremos o algoritmo de Even et al. [4] para o Balanced Cut. Este resultado é uma pseudo-aproximação que utiliza crescimento de regiões, com estratégia semelhante à estudada para o Multicut na etapa anterior do projeto.

O algoritmo encontrará um corte $b-$balanceado cujo custo está a um fator de $O(\log n)$ do custo do corte ótimo $b'-$balanceado, para $b' \neq b$. Em particular, sendo OPT(b) o valor do corte $b-$balanceado de custo mínimo, o algoritmo descrito encontrará um corte $1/3-$balanceado de valor no máximo $O(\log n)$ OPT(1/2). Nota-se primeiramente que OPT(1/3) $\leq$ OPT(1/2), já que qualquer corte $1/2-$balanceado também é um corte $1/3-$balanceado. Além disso, OPT(1/2) pode ser substancialmente maior que OPT(1/3). Afinal, como ilustrado na Figura 1, se o grafo consistir em uma clique em $2n/3$ vértices conectados por uma única aresta à outra clique em $n/3$ vértices, e todas as arestas tenham peso 1, OPT(1/3) = 1, enquanto OPT(1/2) = $\Omega(n^2)$. Assim, o algoritmo não é realmente um algoritmo de aproximação, já que compara o custo da solução encontrada pelo algoritmo com o ótimo de um problema cujo custo pode ser muito maior.

Primeiro, vamos assumir que o programa linear seguinte é uma relaxação do programa linear do problema da bisseção mínima. Dado um grafo G, seja $\mathcal{P}_{uv}$ o conjunto de todos os caminhos conectando u e v em G. Considere o seguinte programa linear

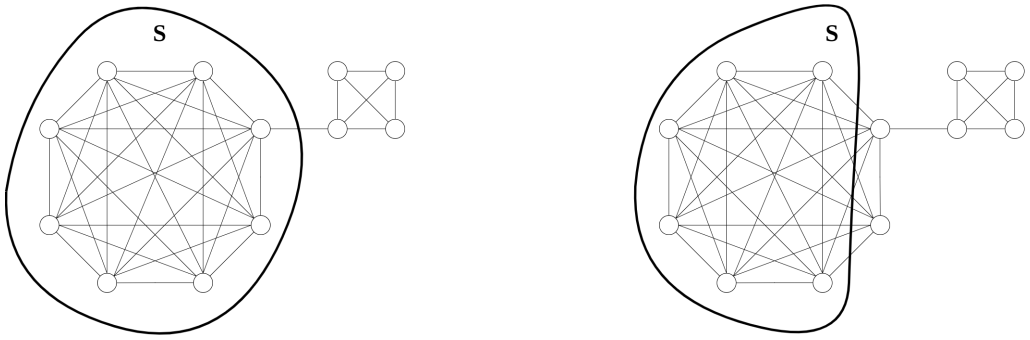

Figura 1: Exemplo ilustrando como OPT(1/2) pode ser muito maior que OPT(1/3).

$$\text{minimizar} \quad \sum_{e \in E} c_e x_e \tag{1}$$

$$\text{sujeito à} \quad d_{uv} \leq \sum_{e \in P} x_e, \qquad \forall u, v \in V, \forall P \in \mathcal{P}_{uv} \tag{2}$$

$$\sum_{v \in S} d_{uv} \geq \left(\frac{2}{3} - \frac{1}{2}\right) n, \qquad \forall S \subseteq V : |S| \geq \left\lceil \frac{2}{3} n \right\rceil + 1, \forall u \in S \tag{3}$$

$$d_{uv} \geq 0, \qquad \forall u, v \in V \tag{4}$$

$$x_e \geq 0, \qquad \forall e \in E \tag{5}$$

no qual x_e é a variável que indica se a aresta e está no corte, i.e., no programa não relaxado recebe 1 se a aresta estiver no corte, e 0 caso contrário. Ademais, d_{uv} denota a distância entre u e v usando x_e como comprimento das arestas.

Lema 3.1. *O programa linear descrito em* (1) – (5) *é uma relaxação do problema da bissecção mínima.*

Demonstração. Dada uma bissecção S, construímos uma solução $(\bar{d}, \bar{x})$ para o programa linear, ao definir $\bar{x_e} = 1$ se $e \in \delta(S)$ e $\bar{x_e} = 0$ caso contrário. Define-se $\bar{d}_{uv} = 1$ caso $u \in S$, $v \notin S$ e $\bar{d}_{uv} = 0$ caso contrário. Será demonstrado que esta é uma solução viável do programa linear, e isto é suficiente para provar o lema.

Temos que o primeiro conjunto de restrições é obedecido, visto que qualquer caminho P de $u \in S$ à $v \notin S$ precisa usar pelo menos uma aresta $e \in \delta(S)$. Agora, considere qualquer conjunto S' tal que $|S'| \geq \lceil \frac{2}{3} n \rceil + 1$.

Note que $\left\lfloor \frac{n}{2} \right\rfloor \leq |S| \leq \left\lceil \frac{n}{2} \right\rceil, \quad |S'| \geq \left\lceil \frac{2}{3} n \right\rceil + 1$

Portanto $|S' - S| \geq |S'| - |S| \geq \left\lceil \frac{2}{3} n \right\rceil + 1 - \left\lceil \frac{1}{2} n \right\rceil \geq \left(\frac{2}{3} - \frac{1}{2}\right) n$

Também temos que $|S' \cap S| \geq \left(\frac{2}{3} - \frac{1}{2}\right) n$, pois o tamanho da interseção mínima entre S' e S é pelo menos $|S'| - (n - |S|)$, como ilustrado na Figura 2.

Portanto $|S' \cap S| \geq |S'| - (n - |S|) \geq \left(\left\lceil \frac{2}{3} n \right\rceil + 1\right) - \left(n - \left\lfloor \frac{1}{2} n \right\rfloor\right) \geq \left(\frac{2}{3} - \frac{1}{2}\right) n$

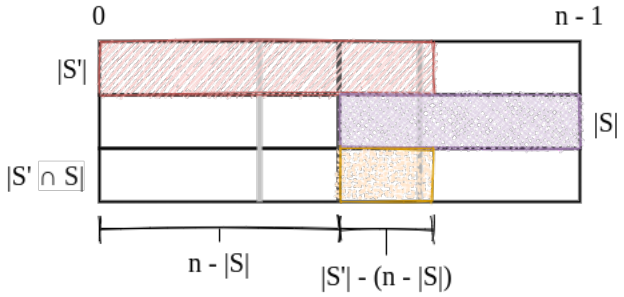

Figura 2: Exemplo ilustrando como o tamanho da interseção mínima entre S e S' deve ser pelo menos $|S'| - (n - |S|)$. Os retângulos de tamanho n representam a quantidade de vértices no grafo. As linhas verticais cinza claro dividem os retângulos em três partes iguais, e a linha cinza escuro, em duas.

Vamos chamar $S' - S$ e $S' \cap S$ as duas partes de S'. Uma parte está contida em S, e a outra não contém elementos de S. Assim, para u e v em partes diferentes, $d_{uv} = 1$. Escolha qualquer $u \in S'$. Como u está em uma das duas partes de S', há pelo menos $\left(\frac{2}{3} - \frac{1}{2}\right) n$ vértices v na outra parte de S' que não contém u.

Logo, $\sum_{v \in S'} d_{uv} \geq \left(\frac{2}{3} - \frac{1}{2}\right) n$.

□

Como mencionado anteriormente, a abordagem aqui utilizada segue os mesmos passos da análise do algoritmo visto para o Multicut. Nela, fez-se uso da analogia que arestas são canos de comprimento x_e e área transversal c_e. A função $d_x(u, v)$ denota o comprimento do menor $u - v$ caminho utilizando os custos x_e, e definimos $V^* = \sum_{e \in E} c_e x_e$ como o volume total dos canos. A solução será construída escolhendo bolas ao redor de vértices. Então, chamaremos de $B_x(u, r) = \{v \in V : d_x(u, v) \leq r\}$ a bola de raio r ao redor do vértice u. Assim, para um raio r dado, seja $c(\delta(B_x(u, r)))$ o custo das arestas em $\delta(B_x(u, r))$. Finalmente, definimos também $V_x(u, r)$ como o volume dos canos dentro de $B_x(u, r)$, mais o termo extra $\frac{V^*}{n}$; isto é

$$V_x(u, r) = \frac{V^*}{n} + \sum_{e=(v,w):\ v,w \in B_x(u,r)} c_e x_e + \sum_{e=(v,w):\ v \in B_x(u,r),\ w \notin B_x(u,r)} c_e (r - d_x(u, v))$$

Corolário 3.1. *Dado o comprimento x_e das arestas $e \in E$, e um vértice u, pode-se encontrar em tempo polinomial um raio $r \in [a, b)$ tal que*

$$c(\delta(B_x(u, r))) \ \leq \ \frac{1}{b - a} \ \ln\left(\frac{V_x(u, b)}{V_x(u, a)}\right) V_x(u, r)$$

Demonstração. Para simplificação, escreveremos $V(r) = V_x(u, r)$, e $c(r) = c(\delta(B_x(u, r)))$. Nossa prova irá mostrar que para r escolhido aleatoria e uniformemente em $[a, b)$, o valor esperado de $\frac{c(r)}{V(r)}$ não é mais que $\frac{1}{b-a} \ln\left(\frac{V(b)}{V(a)}\right)$, implicando que para algum valor de r, temos que $c(r) \leq \frac{1}{b-a} \ln\left(\frac{V(b)}{V(a)}\right)$. Em seguida mostraremos como encontrar este r deterministicamente.

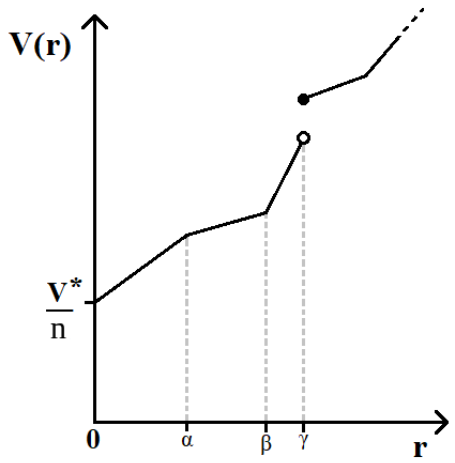

Figura 3: Crescimento de $V(r)$ em relação à r. Nos pontos α e β, $r = d_x(u, v)$ para algum vértice v. Em γ, $r = d_x(u, v) = d_x(u, w)$, tal que (v, w) é uma aresta do grafo.

Para computar a esperança, precisamos das seguintes observações. Note que a função $V(r)$ possui crescimento semelhante ao ilustrado na Figura 3. Note também que a função não é diferenciável nos pontos em que $r = d_x(u, v)$, para $v \in V$. Ademais, podemos perceber que nos intervalos contínuos e deriváveis da função, $V(r)$ possui crescimento linear. Isso é verdade pois em tais regiões são sempre as mesmas arestas que estão entrando na bola $B_x(u, r)$, conforme r aumenta. Além disso, pode-se verificar que a taxa de crescimento desses seguimentos é $c(r)$, e portanto $\frac{\partial V}{\partial r} = c(r)$.

Para continuar a prova do Corolário 3.1, iremos ordenar e renomear os vértices em $B_x(u, r)$ de acordo com suas distâncias de u. Seja $r_j = d_x(u, v_j)$ tal que $a = r_0 \leq r_1 \leq \cdots \leq r_{l-1} \leq b$. Então todos os vértices serão renomeados para $v_0, v_1, v_2, \ldots, v_{l-1}$. Faremos $r_l = b$, e r_j^- um valor infinitesimalmente menor que r_j.

Além disso, precisamos lembrar que a esperança de uma função $f(x)$ para uma variável aleatória $x \in [a, c]$ é

$$E[f(x)] = \int_a^c f(x)\ g(x)\ dx$$

onde $g(x)$ é a função de densidade de probabilidade de x. Como temos uma variável r uniformemente aleatória, $g(r) = \frac{1}{b-a}$. Lembre também que para $a < b < c$

$$\int_a^c f(x)\ dx = \int_a^b f(x)\ dx + \int_b^c f(x)\ dx$$

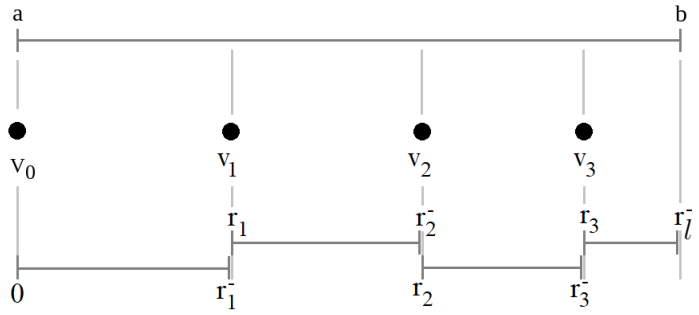

Figura 4: Representação dos intervalos que usaremos para calcular a esperança de $\frac{c(r)}{V(r)}$ para $r \in [a, b)$, de forma que todos os pontos sejam diferenciáveis.

Para calcular a esperança de $\frac{c(r)}{V(r)}$ para $r \in [a, b)$ escolhido de forma aleatória e uniforme, iremos segmentar o intervalo, calculando as integrais de cada intervalo $[r_j, r_{j+1}^-]$, para $j = 0, \ldots, l-1$, como indicado na Figura 4. Como nossa função de densidade é $\frac{1}{b-a}$, temos

$$\sum_{j=0}^{l-1} \int_{r_j}^{r_{j+1}^-} \frac{c(r)}{V(r)} \cdot \frac{1}{b-a}\, dr$$

Logo

$$E\left[\frac{c(r)}{V(r)}\right] = \frac{1}{b-a} \sum_{j=0}^{l-1} \int_{r_j}^{r_{j+1}^-} \frac{c(r)}{V(r)}\, dr = \frac{1}{b-a} \sum_{j=0}^{l-1} \int_{r_j}^{r_{j+1}^-} \frac{1}{V(r)} \cdot \frac{\partial V}{\partial r}\, dr$$

$$= \frac{1}{b-a} \sum_{j=0}^{l-1} [\ln\ V(r)]_{r_j}^{r_{j+1}^-} = \frac{1}{b-a} \sum_{j=0}^{l-1} [\ln\ V(r_{j+1}^-) - \ln\ V(r_j)]$$

$$\leq \frac{1}{b-a} \sum_{j=0}^{l-1} [\ln\ V(r_{j+1}) - \ln\ V(r_j)] = \frac{1}{b-a} (\ln\ V(b) - \ln\ V(a)) = \frac{1}{b-a} \ln\left(\frac{V(b)}{V(a)}\right)$$

Concluímos então que o valor esperado para $\frac{c(r)}{V(r)}$, quando $r \in [a, b)$, é limitado por

$$E\left[\frac{c(r)}{V(r)}\right] \leq \frac{1}{b-a} \ln\left(\frac{V(b)}{V(a)}\right)$$

Então deve ser o caso que existe um $r \in [a, b)$ tal que $c(r) \leq \frac{1}{b-a} \ln\left(\frac{V(b)}{V(a)}\right) V(r)$.

Podemos encontrar este r de forma determinística. Observe que para $r \in [r_j, r_{j+1}^-]$, $c(r)$ se mantém constante (como $B_x(u, r)$ não é alterado), e $V(r)$ é crescente. Assim, precisamos checar a inequação do Corolário 3.1 apenas em r_{j+1}^-, para $j = 0, \ldots, l-1$, pois o valor de $V(r)$ será o maior para tal intervalo. A inequação deve ser verdade para algum valor de j. □

Lema 3.2. *Dada uma solução válida (d, x) para o programa linear, é possível encontrar em tempo polinomial um raio $r < 1/12$ tal que*

$$c(\delta(B_x(u, r))) \leq (12\ \ln(n+1))\ V_x(u, r)$$

Demonstração. Aplicamos o Corolário 3.1 para o intervalo $[0, 1/12)$

$$c(\delta(B_x(u,r))) \leq \frac{1}{b-a}\ \ln\left(\frac{V(b)}{V(a)}\right)\ V(r) = \frac{1}{1/12}\ \ln\left(\frac{V_x(u,1/12)}{V_x(u,0)}\right)\ V_x(u,r)$$
$$\leq 12\ \ln\left(\frac{V^* + V^*/n}{V^*/n}\right)\ V_x(u,r) = 12\ \ln(n+1)\ V_x(u,r)$$

□

O lema visto será importante no algoritmo para que possamos analisar sua razão de aproximação, por possibilitar a comparação do seu custo ($\sum_{i=1}^{k} c(\delta(B_x(u,r))))$ com o volume total do grafo, $V^* \leq \text{OPT}$. Além disso, para que as soluções do nosso algoritmo sejam válidas, é necessário que o raio escolhido seja sempre $r < \frac{1}{1/12}$.

Algorithm 1 Algoritmo para o Corte 1/3-Balanceado

Input: A solução fracionária ótima (d, x) da relaxação do programa linear
Output: O conjunto de vértices S do corte, e o conjunto F das arestas que separam o corte.

1: $S \leftarrow \emptyset,\ \ F \leftarrow \emptyset$
2: **while** $|S| < \lfloor \frac{1}{3}n \rfloor$ **do**
3: Escolha $u, v \notin S$ tal que $d_x(u,v) \geq 1/6$
4: Escolha um raio $r < 1/12$ ao redor de u como no Lema 3.2
5: Escolha um raio $r' < 1/12$ ao redor de v como no Lema 3.2
6: **if** $|B_x(u,r)| \leq |B_x(v,r')|$ **then**
7: $S \leftarrow S \cup B_x(u,r)$
8: $F \leftarrow F \cup \delta(B_x(u,r))$
9: Remove $B_x(u,r)$ e arestas incidentes do grafo
10: **else**
11: $S \leftarrow S \cup B_x(v,r')$
12: $F \leftarrow F \cup \delta(B_x(v,r'))$
13: Remove $B_x(v,r')$ e arestas incidentes do grafo
14: **end if**
15: **end while**

O Algoritmo 1 encontra uma solução para o problema do corte 1/3−balanceado cujo custo é no máximo $(24\ \ln(n+1))\text{OPT}(1/2)$. É necessário, contudo, demonstrar que o mesmo está correto, i.e., realmente devolve um corte com o balanceamento desejado.

Lema 3.3. *Para qualquer iteração do algoritmo em que $|S| < \lfloor 1/3n \rfloor$, existe $u, v \notin S$ tal que $d_x(u,v) \geq 1/6$*

Demonstração. Considere $S' = V - S$. Então $|S'| \geq \lceil 2/3n \rceil + 1$. Pela restrição do programa linear, para qualquer $u \in S'$,

$$\sum_{v \in S'} d_{uv} \geq \frac{1}{6}\ n$$

Como S' possui no máximo n vértices, deve ser verdade que existe algum $v \in S'$ tal que $d_{uv} \geq 1/6$. □

Lema 3.4. *O algoritmo devolve S tal que $\lfloor 1/3\ n \rfloor \leq |S| \leq \lceil 2/3\ n \rceil$*

Demonstração. Pela construção do algoritmo, $|S| \geq \lfloor 1/3\ n \rfloor$. Então apenas precisamos mostrar que $|S| \leq \lceil 2/3\ n \rceil$. Seja $\hat{S}$ o conjunto dos vértices em S no começo da iteração antes do algoritmo terminar. Então $|\hat{S}| < \lfloor 1/3\ n \rfloor$ e adicionaremos a menor entre as bolas ao redor de u e v em $\hat{S}$ para obter a solução final S. Como consideramos $B_x(u, r)$ e $B_x(v, r')$ para $r < 1/12$, $r' < 1/12$, e $d_x(u, v) \leq 1/6$, deve ser o caso que as duas bolas são disjuntas. Visto que escolhemos a melhor bola entre as duas, isso implica que o tamanho da bola adicionada em $\hat{S}$ não possui mais que a metade dos vértices restantes, i.e., $(1/2)(n - |\hat{S}|)$. Assim,

$$|S| \leq |\hat{S}| + \frac{1}{2}\ (n - |\hat{S}|) = \frac{|\hat{S}|}{2} + \frac{n}{2}$$

Como $|\hat{S}| < \lfloor 1/3\ n \rfloor$,

$$|S| \leq \frac{1}{2}\ n + \frac{1}{2}\ \left\lfloor \frac{1}{3} n \right\rfloor = \left\lceil \frac{2}{3}\ n \right\rceil$$

□

Teorema 3.1. *O Algoritmo 1 devolve um corte $1/3-$balanceado de custo não maior que $(24 \ln(n+1)\mathrm{OPT}(1/2)$*

Demonstração. Chamaremos de V_i o volume das arestas removidas na iteração i do algoritmo, e F_i as arestas adicionadas em F nesta mesma iteração. Temos que $V_i \geq V_x(u, r) - (V^*/n)$, visto que V_i contém o volume total das arestas do corte, enquanto V_x contém apenas a parte que está dentro da bola B_x. Pela escolha de r no Lema 3.2, temos que

$$c(F_i) \leq 12\ \ln(n+1)\ V_x(u, r) \leq 12\ \ln(n+1) \left(V_i + \frac{V^*}{n} \right)$$

Além disso, observe que o volume de cada aresta pertence a no máximo um V_i. Logo, para l igual ao número de iterações,

$$\sum_{i=1}^{l} V_i \leq V^*$$

Então temos que

$$c(\delta(S)) \leq \sum_{e \in F} c_e = \sum_{i=1}^{l} \sum_{e \in F_i} c_e$$

$$\leq \sum_{i=1}^{l} 12\ \ln(n+1)\ \left(V_i + \frac{V^*}{n} \right) = 12\ \ln(n+1) \sum_{i=1}^{l} \left(V_i + \frac{V^*}{n} \right)$$

Como $\sum_{i=1}^{l} V_i \leq V^*$ e $\sum_{i=1}^{l} \frac{V^*}{n} \leq V^*$,

$$c(\delta(S)) \leq 24\ \ln(n+1)\ V^* \leq 24\ \ln(n+1)\ \mathrm{OPT}_{LP} \leq 24\ \ln(n+1)\ \mathrm{OPT}(1/2)$$

□

É importante mencionar que o algoritmo e as provas demostradas podem ser generalizados para fornecer um corte $b-$balanceado de custo máximo $O(\frac{1}{b'-b}\ \log n)\mathrm{OPT}(b')$, para $b \leq 1/3$ e $b < b'$.

4 Meta-heurística BRKGA

Com o objetivo de complementar a etapa experimental do projeto com diferentes abordagens, foi estudado o Biased Random-Key Genetic Algorithm (BRKGA), proposto por Gonçalves e Resende [7]. Esta, é uma meta-heurística de busca geral baseada em algoritmos genéticos, onde uma população de indivíduos evolui através do princípio Darwiniano de sobrevivência do mais *apto*.

Utiliza-se uma relação entre uma solução de determinado problema de otimização combinatória, e um indivíduo em uma população. Cada indivíduo j de uma população é representado por um *cromossomo* Q^j, que codifica uma solução. A Figura 5 ilustra esta estrutura. Cada cromossomo consiste em um vetor de m alelos. Cada alelo é uma chave aleatória uniformemente escolhida no intervalo $[0, 1]$. Chamamos de *decodificador* um algoritmo que traduz um cromossomo Q^j em uma solução R^j. A *adaptabilidade* é uma função que avaliará a solução R^j.

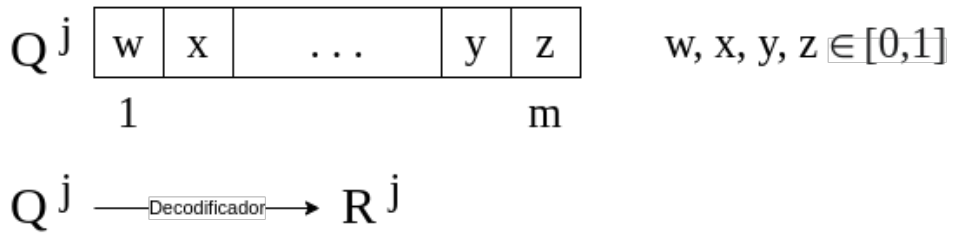

Figura 5: Representação do cromossomo no BRKGA.

O BRKGA inicializa a população com p cromossomos gerados aleatoriamente, e essa população evolui ao longo de g gerações. Para $i = 1, \ldots, g-1$, a população é atualizada da geração i para $i+1$ de acordo com os seguintes passos:

1. Para cada indivíduo j na geração i, o cromossomo Q^j é decodificado em R^j.
2. Avalia-se a adaptabilidade de cada solução R^j para cada indivíduo j na geração i.
3. São copiados os melhores p_e indivíduos (conjunto de elite) de i para $i+1$.
4. São adicionados p_m cromossomos gerados aleatoriamente (mutantes) em $i+1$.
5. Produz-se $p - (p_e + p_m)$ novos cromossomos para $i+1$ usando cruzamentos.

Os cruzamentos geram novos indivíduos ao misturar alelos de dois cromossomos. Ambos os pais são da geração atual i, sendo apenas 1 do conjunto de elite. A probabilidade de um alelo ser escolhido do pai elite é ρ_e. Após g gerações, o BRKGA retorna o cromossomo do melhor indivíduo Q^*, e sua solução decodificada R^*.

Para sua aplicação no problema do Multiway Cut, o algoritmo será utilizado de forma proposta por Hokama et al. [8]. Assim, considerando uma instância I do Multiway Cut, e um indivíduo j do BRKGA, definiremos os seguintes termos.

Cromossomo. Cada alelo do cromossomo j, de chave $Q^j(e)$, é associado à uma aresta $e \in E$ do grafo. Portanto, o cromossomo Q^j é um vetor de tamanho $|E|$.

Decodificador. Para cada aresta $e \in E$ e cromossomo j, a chave de alelo $Q^j(e)$ é usada como perturbação para o custo da aresta e correspondente. Chamaremos de $c^j(e)$, ou custo perturbado de e, o resultado da função de perturbação, que modifica $c(e)$ utilizando $Q^j(e)$. Em seguida, o cromossomo Q^j é decodificado em uma solução R^j utilizando um algoritmo Ap que resolve o problema do Multiway Cut, executando-o com a nova instância I^j cujas arestas possuem os custos perturbados. A ideia é que os custos perturbados permitam ao algoritmo Ap buscar por soluções possivelmente boas que não eram consideradas quando o mesmo trabalhava com os custos originais.

Adaptabilidade. A adaptabilidade de um indivíduo j é o custo da solução R^j decodificada de Q^j avaliada com os custos originais, i.e., $\sum_{e \in R^j} c(e)$. É importante notar que os custos perturbados são passados para Ap encontrar R^j, mas não são considerados para avaliar o custo real de R^j.

5 Resultados Experimentais

Toda a etapa de implementação do projeto foi realizada debruçando-se sobre o Problema do Multiway Cut. Inicialmente, foram implementadas variantes do primeiro algoritmo estudado, e analisados seus resultados. Em seguida, testou-se o potencial da meta-heurística BRKGA para a resolução deste mesmo problema, tendo seus resultados comparados com os algoritmos anteriores. A tabela completa com os resultados de todos os testes, bem como todo o código desenvolvido, podem ser encontrados no GitHub[1]. Os experimentos foram realizados em um computador com processador Intel Core i7, operando sob frequência de 2.7GHz com 16GB de RAM, e os códigos foram implementados em C++. Foram utilizadas 78 instâncias para teste, retiradas do pacote OR-Library[2].

5.1 Algoritmos Iniciais

Como exposto no relatório parcial, foi inicialmente implementado o primeiro algoritmo estudado, baseado no Problema do Corte Mínimo. Além disso, ainda na etapa anterior, foram desenvolvidas duas versões alternativas, a fim de avaliar empiricamente a qualidade

[1] https://github.com/estherhoffmann/Approx_Algorithms
[2] http://people.brunel.ac.uk/~mastjjb/jeb/info.html

de suas soluções. A seguir, encontra-se uma breve explicação do funcionamento de cada um desses algoritmos, com o objetivo de auxiliar a compreensão do trabalho experimental desenvolvido nesta etapa final do projeto.

Algoritmo original baseado no Problema do Corte Mínimo. Considere S como o conjunto de vértices terminais. Nesta abordagem, itera-se por cada terminal $u \in S$, calculando um *corte de isolamento* entre u e os demais terminais, i.e., um corte de custo mínimo que separa u e v, para todo $v \in S \setminus \{u\}$. Ao final das iterações, une-se os $k-1$ cortes de isolamento menos custosos, sendo k o número de terminais, obtendo um conjunto final de arestas F que separa cada vértice terminal em um diferente componente conexo do grafo $G = (V, E - F)$. Para isso, é utilizado um sorvedouro artificial T. No início de cada iteração, são criados arcos com peso infinito que conectam diretamente cada $v \in S \setminus \{u\}$ com o sorvedouro T. Em seguida, é calculado um corte mínimo entre o terminal u da iteração atual e T. Após, os arcos de peso infinito são excluídos, limpando o grafo para a iteração seguinte.

Algoritmo Alternativo 1. A primeira modificação que desejou-se testar empiricamente foi não recolocar ao final de cada iteração o arco de peso infinito (v, T), sendo v o terminal considerado na última iteração. Assim, determinando que os terminais das iterações seguintes não precisam ser separados de v, visto que já possuímos um corte mínimo que separa v de $S \setminus \{u\}$. A ideia é que os custos dos cortes das iterações subsequentes pode ser menor.

Algoritmo Alternativo 2. Na segunda modificação implementada, a cada iteração são removidas do grafo todas as arestas do corte mínimo encontrado, ocultando para as iterações seguintes os vértices que foram isolados. Ao final, temos o grafo resultante já segmentado em diferentes componentes conexos, com os terminais separados. Trata-se de uma variante melhorada da alternativa anterior

Foram realizados testes preliminares comparando o desempenho do algoritmo original com os dois alternativos. Na Figura 6 observa-se a razão (em porcentagem) dos resultados dos algoritmos alternativos, em relação aos resultados do algoritmo original implementado. Tabela 1 expõe mais informações sobre o resultado mostrado na Figura 6. Diante destes resultados, resolveu-se focar apenas no algoritmo original e no segundo algoritmo alternativo, a partir de agora chamado apenas de Algoritmo Alternativo.

Após tais análises, foi considerado que o Alg. Alternativo poderia melhorar se fosse alterada a ordem em que são calculados os cortes mínimos. Afinal, enquanto o algoritmo original calcula todos os cortes e descarta o mais custoso, o Alg. Alternativo apenas calcula os $k - 1$ primeiros cortes, na ordem numérica dos terminais. Com a finalidade de encontrar melhores resultados ainda se baseando nesses algoritmos, e aproveitando-se da simplicidade e rapidez dos mesmos, foram desenvolvidas outras três diferentes versões.

Algoritmo Multiway Mix. A ideia central é tornar o Alternativo pelo menos tão bom quanto o Original, deixando para o final o corte mínimo mais custoso, para que este não seja calculado. Para isto, primeiro é executado o algoritmo original, descobrindo os

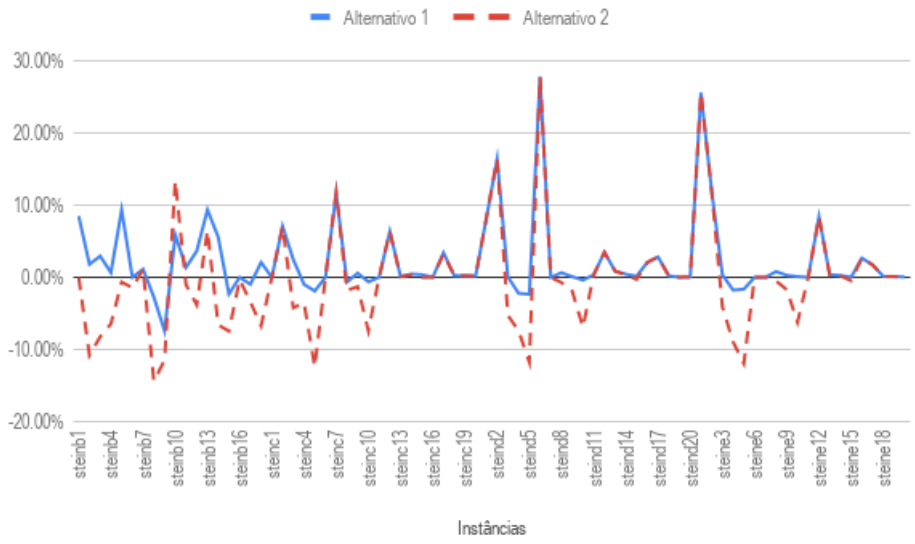

Figura 6: Razão dos resultados dos algoritmos alternativos em relação ao algoritmo original.

Medida	Alternativo 1	Alternativo 2
Média	2.25%	-0.23%
Desvio Padrão	5.46%	7.21%
Valor Mínimo	-7.43%	-14.38%
Valor Máximo	27.78%	27.78%
Média do Tempo de Execução	0.88s	0.68s

Tabela 1: Resultados da razão dos algoritmos alternativos em relação ao algoritmo original, e a média do tempo de execução por instância de cada algoritmo.

valores de cada um dos cortes de isolamento. Em seguida, coloca-se no final da lista o terminal com maior custo para ser separado. Finalmente, é executado o Alg. Alternativo, utilizando esta lista modificada dos terminais. Assim, os $k-1$ cortes que serão encontrados e colocados na solução final não inclui o corte de isolamento mais custoso, como no algoritmo original.

Algoritmo Multiway Sort. Muito parecido com o Multiway Mix, este algoritmo utiliza os custos dos cortes de isolamento encontrados pelo algoritmo original para reordenar completamente a lista de terminais que será utilizada pelo Alternativo, colocando os terminais em ordem crescente de custo do corte para separá-lo. Pode-se notar que novamente o corte mais custoso será deixado por último, não sendo calculado para a solução final.

Algoritmo Alternativo Aleatorizado. Seguindo a mesma ideia de reordenar a lista de terminais, esta versão aleatorizada executa o Alternativo x vezes, sendo x um parâmetro

definido pelo usuário. Para cada execução é gerada uma permutação aleatória da lista de terminais para ser utilizada. Ao final, o algoritmo devolve a melhor solução dentre as encontradas. Para os testes, utilizamos $x = 20$.

Apesar de simples, os algoritmos apresentaram bons resultados. Como esperado, as versões que desconsideram o corte de isolamento mais custoso encontram ao menos uma solução tão boa quanto a do Alg. Original. Além disso, foi obtida uma razão em relação ao Original melhor que -10% em várias instâncias. A Figura 7 expõe graficamente este resultado. A Tabela 2 fornece mais informações sobre a razão de cada um dos algoritmos, bem como seu tempo de execução médio.

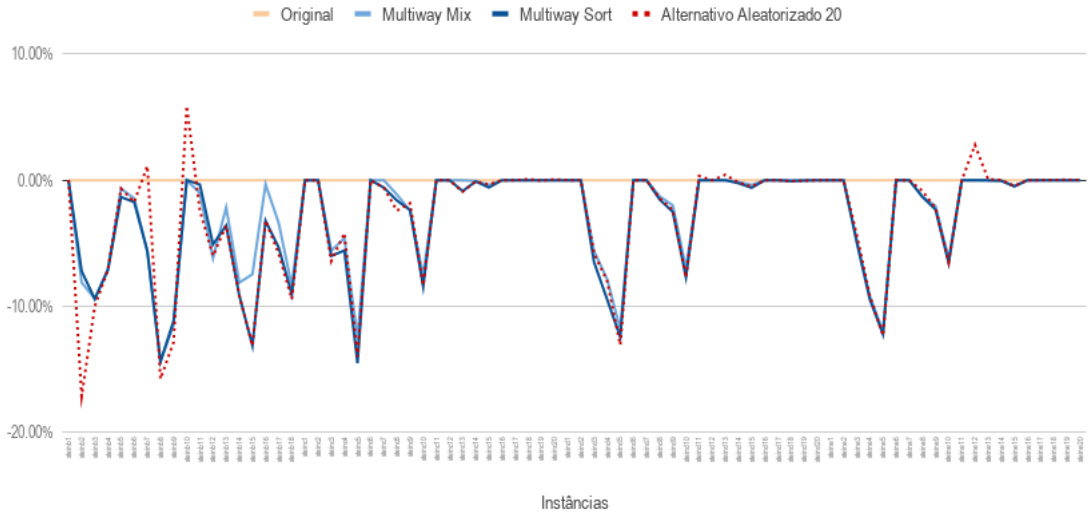

Figura 7: Razão dos resultados dos algoritmos Multiway Mix, Multiway Sort, e Alternativo Aleatorizado em relação ao algoritmo original.

Medida	Mix	Sort	Aleatorizado 20
Média	-2.58%	-2.89%	-2.83%
Desvio Padrão	3.88%	4.18%	4.69%
Valor Mínimo	-14.38%	-14.51%	-17.12%
Valor Máximo	0.00%	0.00%	5.88%
Média do Tempo de Execução	1.60s	1.46s	11.57s

Tabela 2: Resultados da razão dos algoritmos Multiway Mix, Multiway Sort, e Alternativo Aleatorizado em relação ao algoritmo original, e a média do tempo de execução por instância de cada algoritmo.

5.2 Aplicação da meta-heurística BRKGA

Desejando testar novas abordagens, foi aplicada a heurística BRKGA, descrita na Seção 4, para o problema do Multiway Cut. Utilizou-se a interface de programação em C++ desenvolvida por Toso e Resende [12]. Primeiramente, desenvolveu-se uma versão simples, com decodificador básico sem qualquer tomada de decisão esperta. Tal algoritmo é descrito a seguir.

BRKGA Simples. Neste, foi proposto um decodificador mais simples que o descrito na Seção 4. Cada alelo do cromossomo representa um vértice do grafo que não seja um terminal, e seu decodificador apenas calcula em qual partição está o vértice na solução final. Os terminais são associados de forma ordenada a cada uma das partições, i.e., o terminal com menor rótulo é associado à primeira partição, e o com maior, à última. De modo geral, sendo k o número de terminais, se o alelo possuir valor entre $[i/k, (i+1)/k)$, seu vértice pertencerá à parte do i-ésimo terminal, com $i = 0, \ldots, k-1$. Para a adaptabilidade, é retornado o somatório do custo das arestas cujos extremos estão em partes diferentes.

Por ser um algoritmo que depende da aleatoriedade de seus cromossomos, o resultado das soluções variam muito. Isto pode ser observado na Figura 8 e na Tabela 3, ao final desta seção. Mesmo obtendo soluções melhores que o Alg. Original para algumas instâncias, sua média da razão é 1354.42%. Assim, construiu-se um decodificador mais robusto, descrito abaixo.

BRKGA com Multiway Original. Como mencionado anteriormente, os alelos do cromossomo representam perturbações nos custos originais das arestas, e é utilizado um algoritmo polinomial para encontrar o resultado utilizando os custos perturbados. Utilizou-se a seguinte função de perturbação:

$$c'(e) = \frac{2^{\frac{N}{2}}\ c(r)}{2^{N\ Q^j(e)}}$$

onde $c(e)$ é o custo original da aresta e, $c'(e)$ é o custo perturbado da mesma, N é um número real maior que 0 que determina a intensidade da perturbação, e $Q^j(e)$ é o alelo da aresta e no cromossomo j. Para $Q^j(e) > 0.5$, a aresta r recebe um desconto, para $Q^j(e) < 0.5$, um aumento. Escolheu-se esta função de perturbação visando a distorção simples e proporcional dos custos. Note que

$$\text{se } Q^j(e) = 0, \text{ então } c'(e) = 2^{N/2}\ c(e)$$

$$\text{se } Q^j(e) = 0.5, \text{ então } c'(e) = c(e)$$

$$\text{se } Q^j(e) = 1, \text{ então } c'(e) = \frac{c(e)}{2^{N/2}}$$

Nos testes realizados, foi definido $N = 4$. Assim, o decodificador calcula os custos perturbados de todas as arestas do grafo, utilizando o cromossomo recebido, e utiliza o algoritmo original do Multiway Cut para encontrar a solução do problema com as arestas perturbadas. Em seguida, é calculado e devolvido o custo desta solução encontrada,

porém utilizando os custos originais do grafo. Optou-se por utilizar o Alg. Original no decodificador por ser um algoritmo de aproximação e de rápida execução comparado às variantes Mix ou Sort. Tal fato é importante, visto que o tempo de execução da heurística BRKGA cresce bastante para grafos grandes ou com muitos terminais.

BRKGA com Multiway Original e População Inicial. É executado uma vez o Multiway Sort e vinte vezes o Alg. Aleatorizado. Deste último guarda-se apenas as soluções com custos diferentes. Tais soluções, assim como a encontrada pelo Sort, são convertidas em cromossomos. Para isto, é atribuído 0.9 aos alelos das arestas do corte, forçando um grande desconto nas mesmas, e 0.1 aos alelos restantes. O objetivo é induzir o algoritmo do BRKGA a preferir escolher cortes próximos das soluções que já consideramos boas. Assim, são introduzidos na população inicial do BRKGA esses cromossomos criados, bem como um cromossomo neutro, preenchido com 0.5, garantindo que o algoritmo também considere a solução do problema com os custos originais.

Como o objetivo ao utilizar o BRKGA é buscar por soluções ainda melhores que as anteriores, para a análise de seus resultados calculou-se, para cada instância, o *Melhor Encontrado* entre os algoritmos da Subseção 5.1, i.e., a solução menos custosa para cada uma das instâncias, utilizando apenas os algoritmos apresentados na Subseção 5.1. Assim, nas Figuras 8 e 9, e na Tabela 3, pode-se verificar a variação percentual de seus resultados em relação ao *Melhor Encontrado*.

Análises da Var. Percentual	BRKGA Simples	BRKGA com Original	BRKGA com Pop. Inicial
Média	1358.42%	0.40%	-0.36%
Desvio Padrão	3260.57%	2.13%	0.56%
Valor Mínimo	-2.37%	-2.37%	-2.37%
Valor Máximo	16738.12%	16.28%	0.21%
Média dos Tempos de Execução	3029s	3249s	4254s

Tabela 3: Resultados da variação percentual dos algoritmos BRKGA em relação ao Melhor Encontrado, e a respectiva média do tempo de execução por instância.

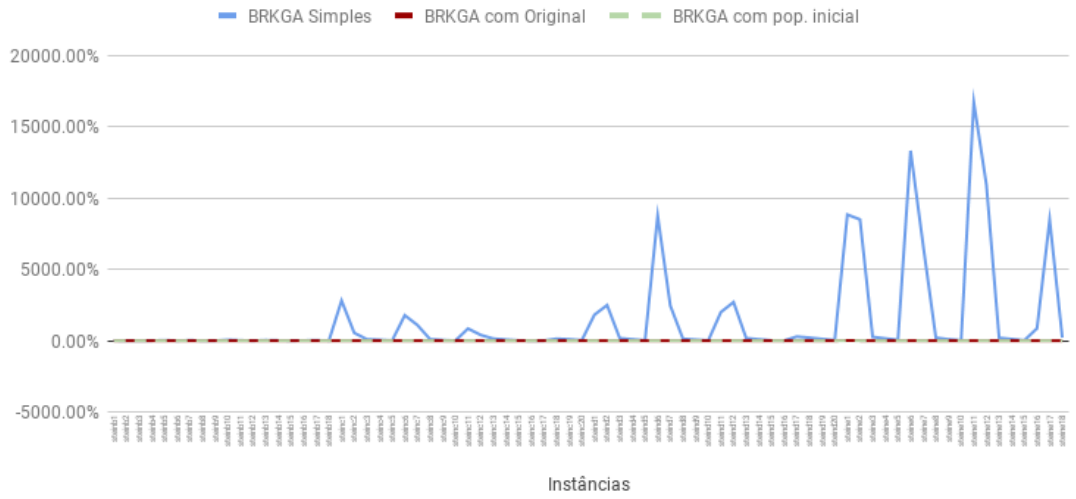

Figura 8: Variação percentual dos resultados dos algoritmos BRKGA em relação ao Melhor Encontrado.

É importante mencionar que, embora o tempo médio de execução dos algoritmos com o BRKGA serem altos, em instâncias pequenas sua execução demora apenas alguns segundos. Além disso, apesar do tempo crescer bastante conforme a instância aumenta, ele é diretamente relacionado com a quantidade de terminais no grafo.

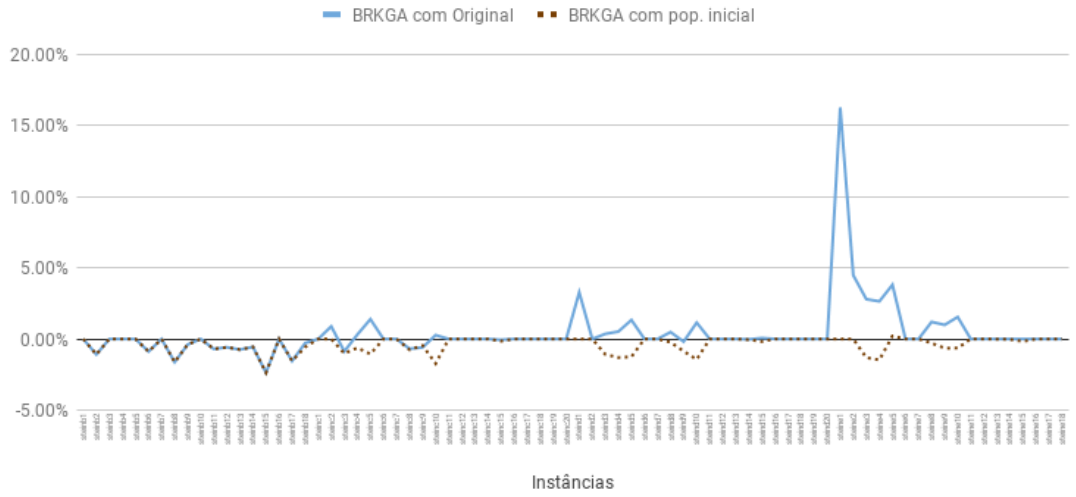

Figura 9: Variação percentual dos resultados dos algoritmos BRKGA com Multiway Original, e BRKGA com Original e População Inicial, em relação ao Melhor Encontrado.

6 Cronograma

A Tabela 4 contém o cronograma atualizado do projeto.

Atividades	Meses											
	1	2	3	4	5	6	7	8	9	10	11	12
Bases teóricas	•											
Multiway Cut e Multicut		•	•	•	•	•						
Balanced Cut							•	•				
Implementação e testes							•	•	•	•	•	•
Relatório Técnico	•	•	•	•	•	•	•	•	•	•	•	•

Tabela 4: Cronograma das atividades.

O primeiro semestre foi destinado ao estudo de bases teóricas, incluindo um algoritmo para o Problema do Fluxo Máximo/Corte Mínimo, e de algoritmos de aproximação para o Multiway Cut e Multicut. Também teve início nesse período a fase de implementação, sendo escolhido o algoritmo para o Multiway Cut baseado no Problema do Corte Mínimo.

No segundo semestre optou-se por focar na implementação, desenvolvimento e testes de algoritmos, em especial, variantes buscando melhores resultados. Assim, foram desenvolvidos três algoritmos utilizando aqueles implementados na etapa anterior. Além disso, estudou-se a meta-heurística BRKGA, e aplicou-a de formas diferentes para resolver o Problema do Multiway Cut. Foram realizados testes de todas as implementações, incluindo as desenvolvidas no primeiro semestre. Ademais, deu-se prosseguimento na parte teórica do projeto, estudando o algoritmo de pseudo-aproximação para o Problema do Balanced Cut, bem como sua garantia de qualidade. Decidiu-se não estudar o resultado para o Problema do Sparsest Cut, presente no cronograma apresentado no relatório parcial, pela falta de tempo e complexidade de sua teoria.

7 Conclusão

Inicialmente foi estudada a relação entre os Problemas do Corte Mínimo e do Fluxo Máximo, bem como um algoritmo para os mesmos. Em seguida, estudou-se algoritmos de aproximação para duas generalizações do Corte Mínimo, o Multiway Cut e o Multicut, e para o Balanced Cut, além de suas respectivas análises de garantia. Ademais, desenvolveu-se um código em C++ do primeiro algoritmo de aproximação estudado, e cinco variantes do mesmo. Seus resultados experimentais foram muito satisfatórios, executando rapidamente mesmo para instâncias grandes. Depois, estudou-se a meta-heurística BRKGA, aplicando-a de três formas diferentes em busca de melhores resultados. O BRKGA Simples, dependendo totalmente da aleatoriedade, obteve resultados ruins. O BRKGA utilizando o algoritmo Original em seu decodificador apresentou resultados satisfatórios. As melhores soluções, como esperado, foram encontradas pelo que utiliza o algoritmo Original em seu decodificador e possui indivíduos inseridos em sua população inicial. Contudo, o tempo de execução desses três algoritmos cresceu consideravelmente rápido, tornando-se inviável em situações de urgência para grafos grandes ou com muitos terminais. Uma implementação não realizada neste projeto, mas que seria interessante em etapas futuras, é a execução de um algoritmo exato para o Multiway Cut, a fim de comparar seus resultados com os expostos neste relatório.

Muito conhecimento foi adquirido sobre os problemas e algoritmos estudados e, em especial, sobre conteúdos básicos, fundamentais nas análises de garantia: lógica matemática, probabilidade e cálculo. Com a implementação, nota-se também grande melhoria em lógica de programação, na habilidade de resolver problemas de forma algorítmica, e maior familiaridade com a linguagem e bibliotecas utilizadas. Este projeto me introduziu na área de otimização combinatória, e cada vez mais pude compreender e analisar algoritmos complexos, além das vantagens e desvantagens de algoritmos de aproximação e heurísticas. Assim, considero a conclusão deste projeto bem sucedida.

Referências

[1] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 489–495 vol.1, 1999.

[2] 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.

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

[4] Guy Even, Joseph (Seffi) Naor, Satish Rao, and Baruch Schieber. Fast approximate graph partitioning algorithms. *SIAM Journal on Computing*, 28(6):2187–2214, 1999.

[5] N. Garg, V. V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.

[6] Naveen. Garg, Vijay V. Vazirani, and Mihalis. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25(2):235–251, 1996.

[7] José Fernando Gonçalves and Mauricio GC Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525, 2011.

[8] Pedro HDB Hokama, Mário C San Felice, Evandro C Bracht, and Fábio L Usberti. Evolutionary framework for two-stage stochastic resource allocation problems. *arXiv preprint arXiv:1903.01885*, 2018.

[9] Jörg Hendrik Kappes, Markus Speth, Björn Andres, Gerhard Reinelt, and Christoph Schn. Globally optimal image partitioning by multicuts. In Yuri Boykov, Fredrik Kahl, Victor Lempitsky, and Frank R. Schmidt, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 31–44, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[10] David R. Karger, Philip Klein, Cliff Stein, Mikkel Thorup, and Neal E. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. *Mathematics of Operations Research*, 29(3):436–461, 2004.

[11] Alexander Schrijver. On the history of the transportation and maximum flow problems. *Mathematical Programming*, 91(3):437–445, 2002.

[12] Rodrigo F Toso and Mauricio GC Resende. A c++ application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, 30(1):81–93, 2015.

[13] D.P. Williamson and D.B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, New York, NY, USA, 2011.