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

# Algoritmos de Aproximação para Problemas de Clustering

**Aluno:** João Guilherme Alves Santos
**Orientadora:** Cristina Gomes Fernandes

MAC0499 - Trabalho de Formatura Supervisionado

São Paulo
2025

# Agradecimentos

O presente trabalho foi realizado com apoio da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), Brasil. Processo nº 2023/16197-0.

# Conteúdo

**1 Introdução** **4**

**2 $k$-Centros** **8**

2.1 Algoritmo guloso . . . . . 10

2.2 Método do gargalo . . . . . 12

**3 Localização de instalações** **15**

3.1 Algoritmos baseados em programação linear . . . . . 16

3.1.1 Método primal-dual . . . . . 18

3.1.2 Arredondamento do programa linear . . . . . 22

3.1.3 Arredondamento probabilístico . . . . . 25

3.2 Busca local . . . . . 27

3.3 Algoritmo guloso . . . . . 34

3.4 Inaproximabilidade . . . . . 41

**4 $k$-Medianas** **44**

4.1 Busca local . . . . . 45

4.2 Algoritmos baseados em programação linear . . . . . 50

4.2.1 Relaxação Lagrangeana . . . . . 51

4.3 Inaproximabilidade . . . . . 59

**5 Algoritmo de Li e Svensson** **63**

5.1 Obtenção de soluções a partir de soluções aditivas . . . . . 63

5.1.1 Instâncias esparsas . . . . . 64

5.1.2 Soluções para instâncias esparsas a partir de pseudo-soluções . . 66

5.1.3 Combinando os algoritmos . . . . . 72

5.2 Uma aproximação aditiva . . . . . 74

**6 Conclusão** **83**

# 1 Introdução

Problemas de otimização têm o objetivo de encontrar um ponto ótimo de uma função definida sobre um certo domínio. Especificamente, os problemas de otimização combinatória têm domínio finito. Muitos desses problemas são *NP*-difíceis. Para problemas *NP*-difíceis, não existem algoritmos eficientes que encontrem uma solução ótima para toda instância de tais problemas a menos que $P = NP$.

Nesse contexto, algoritmos de aproximação surgiram. A ideia é abrir mão de encontrar soluções ótimas para encontrar, eficientemente, uma solução cujo valor garante uma relação pré-estabelecida com o valor ótimo.

Clustering refere-se a uma classe de problemas de otimização cujo objetivo é agrupar objetos de maneira que objetos no mesmo cluster apresentem mais semelhanças quando comparados a objetos em clusters diferentes. Tais semelhanças serão definidas pelo problema em questão. Neste projeto, iremos estudar, sob o ponto de vista de algoritmos de aproximação, três problemas de clustering *NP*-difíceis: localização de instalações, $k$-medianas e $k$-centros.

Localização de instalações é um problema que visa determinar a melhor localização para instalações, como fábricas ou depósitos, com base no custo de abertura das instalações e custos de transporte. Além disso, pode ser modelado para outras aplicações como problemas de posicionamento de caches em um computador ou problemas de projeto de redes. Existem várias versões do problema de localização de instalações, a mais simples delas é a versão sem capacidades em que as instalações não têm limitações para suprir os clientes.

No problema de localização de instalações sem capacidades, temos um grafo $(F, D)$-bipartido completo em que $D$ é o conjunto de clientes a serem atendidos e $F$ o conjunto de instalações que podem ser abertas. Para cada cliente $j \in D$ e cada instalação $i \in F$, há um custo $c_{ij}$ para a aresta $ij$ em associar o cliente $j$ à instalação $i$. Além disso, existe um custo de abertura $f_i$ para cada instalação $i \in F$. Seja $F' \subseteq F$, uma associação referente a $F'$ é uma função $\sigma : D \to F'$. Definimos o custo para $F'$ e $\sigma$ como $\text{custo}(F', \sigma) \coloneqq \sum_{i \in F'} f_i + \sum_{j \in D} c_{\sigma_j j}$. Além disso, definimos $\text{custo}(F') \coloneqq \min_\sigma \text{custo}(F', \sigma)$. Claramente, o $\sigma^*$ que minimiza isso é $\sigma^*(j) = \min_{i \in F'} c_{ij}$, para todo $j \in D$. Assim, o objetivo do nosso problema é encontrar um subconjunto $F' \subseteq F$ que minimize $\text{custo}(F')$.

O problema das $k$-medianas é muito parecido com o problema de localização de instalações. A diferença aqui é que não temos custo para a abertura de instalações, mas podemos abrir no máximo $k$ delas. Assim como no localização de instalações, no

problema das $k$-medianas temos um grafo $(F, D)$-bipartido completo em que $D$ é o conjunto de clientes a serem atendidos e $F$ o conjunto de instalações que podem ser abertas. Para cada cliente $j \in D$ e cada instalação $i \in F$, há um custo $c_{ij}$ para a aresta $ij$ em associar o cliente $j$ à instalação $i$. Temos também um inteiro $k$ que representa a quantidade de instalações que podem ser abertas. Então, queremos encontrar um conjunto $F' \subseteq F$ de tamanho $k$ que minimize a soma das distâncias entre cada cliente e a instalação aberta mais próxima a ele, ou seja, minimizar custo$(F') := \sum_{j \in D} \min_{i \in F'} c_{ij}$.

No problema dos $k$-centros não existe diferença entre instalações que podem ser abertas e clientes. Temos cidades e escolheremos $k$ delas para construir instalações. Assim, temos um grafo $G = (V, E)$ completo em que $V$ são cidades e temos um custo $c_e$, para cada $e \in E$, para associar as cidades que são extremos de uma aresta $e$. Temos também um inteiro $k$ que representa a quantidade de cidades em que uma instalação será aberta. Cada cidade será associada a uma cidade com uma instalação aberta com menor custo de associação entre elas. O objetivo do nosso problema é minimizar o maior custo de associação entre uma cidade qualquer e a cidade a qual ela está associada, ou seja, encontrar $F' \subseteq V$ com $|F| \leq k$ que minimize $\max_{j \in V} \min_{i \in F'} c_{ij}$.

Iremos nos restringir às versões métricas de cada um desses problemas. Nesse caso, as funções de custo representam distâncias e satisfazem a desigualdade triangular. A desigualdade triangular se comporta de maneira diferente entre os problemas devido às diferentes definições das funções de custo. Para uma instância $(G, c)$ do problema dos $k$-centros, dadas cidades $i, j, k \in V(G)$ vale que $c_{ij} \leq c_{ik} + c_{kj}$. Para uma instância $(F, D, c, .)$ do problema das $k$-medianas ou de localização de instalações, para $j, \ell \in D$ e $i, h \in F$ vale que $c_{ij} \leq c_{i\ell} + c_{h\ell} + c_{h\ell}$.

Diversos métodos podem ser utilizados para aproximar o problema de localização de instalações métrico. Charikar e Guha desenvolveram um algoritmo com razão de aproximação 2.414 utilizando o método de busca local [4]. Esse problema também pode ser modelado como um problema de programação linear inteira e, por isso, técnicas envolvendo programação linear podem ser aplicadas a ele. Por exemplo, há algoritmos que fazem o arredondamento de soluções da relaxação do programa linear inteiro para obter uma solução aproximada do problema. Alguns destes algoritmos atingem boas razões de aproximação, por exemplo, chegando a 1.677 [2]. Entretanto, a melhor aproximação encontrada utiliza vários métodos, incluindo o conhecido método primal-dual, e garante uma razão de aproximação 1.488 [20]. Essa não é muito distante do melhor que se poderia encontrar, uma vez que Guha e Khuller mostraram que não existe algoritmo polinomial para esse problema com razão de aproximação melhor que 1.463 [9], a menos que $P = NP$.

Dentre os três problemas apresentados, o problema das $k$-medianas métrico é o que tem a maior folga entre o melhor resultado de inaproximabilidade e a razão do melhor algoritmo de aproximação conhecido. Jain, Mahdian e Saberi [15] provaram que não existe algoritmo polinomial com razão de aproximação $1 + \frac{2}{e}$ para o problema das $k$-medianas, assumindo que $P \neq NP$, enquanto a melhor aproximação encontrada tem razão $2.675 + \epsilon$ [3].

Hsu e Nemhauser [13] mostraram que não existe algoritmo polinomial com razão de aproximação menor que 2 para o problema dos $k$-centros métrico, assumindo que $P \neq NP$. Neste caso, temos algoritmos de aproximação que apresentam o melhor desempenho possível: utilizando o método do gargalo, Gonzalez [8] e independentemente Hochbaum e Shmoys [11] desenvolveram um algoritmo polinomial com razão de aproximação igual a 2.

A Seção 2 trata do problema dos $k$-centros. Começamos com uma prova de que o problema é $NP$-difícil, assim como uma prova de que não existe algoritmo de aproximação para a versão que não assume função de custo métrica, a menos que $P = NP$. Além disso, também está descrito o melhor resultado de inaproximabilidade, desenvolvido por Hsu e Nemhauser [13], que demonstra a impossibilidade de obter uma $(2 - \epsilon)$-aproximação para esse problema, a menos que $P = NP$. Nela também encontram-se duas 2-aproximações para esse problema: um algoritmo guloso desenvolvido por Gonzalez [8] e um algoritmo desenvolvido por Hochbaum e Shmoys que utiliza o método do gargalo [12].

Na Seção 3 estão presentes algoritmos de aproximação com diversas abordagens para o problema de localização de instalações, várias dessas estão ligadas à programação linear. Primeiramente, falamos um pouco sobre o problema e mostramos que ele é $NP$-difícil. Logo após, é descrita uma 3-aproximação que utiliza o método primal-dual desenvolvida por Jain e Vazirani [16]. Posteriormente, são apresentados dois algoritmos que fazem arredondamentos de uma solução relaxada de um programa inteiro que modela o problema: uma 4-aproximação que faz arredondamento determinístico e uma 3-aproximação que faz arredondamento probabilístico, ambas desenvolvidas por Chudak e Shmoys [5]. Além dessas, é apresentado um algoritmo guloso que é equivalente a uma 2-aproximação que utiliza o método dual fitting desenvolvida por Jain, Mahdian, Markakis, Saberi e Vazirani [14], bem como uma $(1 + \sqrt{2} + \epsilon)$-aproximação baseada em técnicas de busca local desenvolvida por Gupta e Tangwongsan [10]. É destacado o resultado de Guha e Khuller [9], que demonstra a inexistência de uma 1.46-aproximação para esse problema, a menos que $P = NP$.

Na Seção 4 destacamos o problema das $k$-medianas. Assim como no problema de

localização de instalações, começamos a seção falando um pouco sobre o problema e mostrando que ele é *NP*-difícil. Após isso, descrevemos dois algoritmos que utilizam o método de busca local. Nos algoritmos de busca local para o problema das $k$-medianas começamos com uma solução viável e realizamos operações que trocam instalações abertas por instalações fechadas. No primeiro deles, são permitidas apenas trocas unitárias, ou seja, uma instalação aberta é fechada e uma instalação fechada é aberta, e com isso é possível atingir uma 5-aproximação. O segundo algoritmo é uma generalização do primeiro e é parametrizado por um número $p$. São permitidas trocas de até $p$ instalações simultaneamente. Esse algoritmo é uma $(3 + \frac{2}{p})$-aproximação. Aumentando o valor de $p$, é possível fazer o termo $\frac{2}{p}$ ser tão pequeno quanto quisermos e, com isso, conseguimos uma $(3 + \epsilon)$-aproximação, mas isso reflete no tempo de execução do algoritmo. Ambos os algoritmos foram desenvolvidos por Arya, Garg, Khandekar, Meyerson, Munagala e Pandit [1]. Posteriormente, falamos sobre um algoritmo primal-dual que utiliza a técnica de relaxação Lagrangeana. Esta é uma técnica sofisticada que permite substituir restrições por uma penalidade na função objetivo para soluções que não respeitam tais restrições. Além disso, esse algoritmo utiliza um arredondamento probabilístico para transformar a solução relaxada numa solução do problema original e mostramos a desaleatorização desse algoritmo. Esse algoritmo foi desenvolvido por Jain e Vazirani [16] e é uma 4-aproximação para o problema das $k$-medianas. Na Seção 5, apresentamos o algoritmo descrito no artigo "Approximating $k$-Median via Pseudo-Approximation" de Li e Svensson [21]. Esse algoritmo foi fundamental para a descoberta dos algoritmos de aproximação mais recentes encontrados, uma vez que ele conseguiu não se limitar ao *gap* de integralidade, permitindo encontrar resultados melhores para um problema que estava há uma década sem novas descobertas. Em resumo, eles provisoriamente permitem abrir mais do que $k$ instalações e transformam isso em uma solução viável sem perder muito no valor da solução. Esse algoritmo é uma $(1 + \sqrt{3} + \epsilon)$-aproximação. É destacado o resultado de Jain, Mahdian, Markakis, Saberi e Vazirani [14], que demonstra a inexistência de uma 1.736-aproximação para esse problema, a menos que $P = NP$.

Com exceção do algoritmo de busca local que troca várias instalações simultaneamente e o algoritmo que utiliza pseudo-aproximação, ambos para o problema das $k$-medianas, todos os outros algoritmos foram estudados pelos livros "*Approximation Algorithms*" de Vijay Vazirani (V2001) [22] e "*The Design of Approximation Algorithms*" de David Williamson e David Shmoys (WS2011) [23].

# 2 $k$-Centros

No problema dos $k$-centros, dado um grafo $G = (V, E)$ completo com função de pesos $c$ nas arestas e um inteiro $k$, queremos encontrar um conjunto $S \subseteq V$ com tamanho no máximo $k$ que minimize a maior distância entre um vértice qualquer e esse conjunto, ou seja, que minimize $\text{raio}(S) \coloneqq \max_{u \in V \setminus S} c(u, S)$, em que $c(u, S) \coloneqq \min_{v \in S} c_{uv}$.

Seja $I = (G = (V, E), c, k)$ uma instância do problema dos $k$-centros e $S \subseteq V$ com $|S| \leq k$ uma solução viável para $I$. Vamos definir alguns termos que facilitarão as explicações seguintes. Os vértices de $S$ serão chamados de *centros de cluster*. Os vértices de $V$ serão particionados em $k$ conjuntos chamados *clusters* e cada um deles terá exatamente um centro de cluster. Um vértice estará no mesmo cluster que um centro de cluster mais próximo a ele. Cada cluster terá um *raio* que é o maior custo, de acordo com $c$, entre o seu centro e um vértice qualquer dele. Denotamos por $\text{raio}(S)$ o maior raio de um cluster induzido por $S$. Assim, o problema dos $k$-centros consiste em encontrar um conjunto $S$ de $k$ centros tal que $\text{raio}(S)$ é mínimo.

Antes de falarmos sobre algoritmos de aproximação para o problema dos $k$-centros, vamos mostrar que o problema é *NP*-difícil. Esse resultado é encontrado a partir de uma pequena alteração na prova do Teorema 5.7 do livro V2001. Para isso, vamos definir o problema do $k$-conjunto dominante.

**Definição 2.1.** *Seja $G = (V, E)$ um grafo. Um conjunto $D \subseteq V$ é* dominante *se, para todo vértice $u \in V \setminus D$, existe um vértice $v \in D$ tal que $uv \in E$.*

**Problema 2.2** ($k$-conjunto dominante)**.** *Dado um grafo $G$ e um inteiro $k$, decidir se $G$ tem um conjunto dominante $D$ tal que $|D| \leq k$.*

Esse problema é *NP*-completo, sendo o problema GT2 do famoso livro de Garey e Johnson [7]. Usaremos este problema para mostrar que o problema dos $k$-centros é *NP*-difícil.

**Teorema 2.3.** *O problema dos $k$-centros para instâncias métricas é NP-difícil.*

*Demonstração.* Vamos provar o teorema fazendo uma redução do problema do $k$-conjunto dominante para o problema dos $k$-centros métrico. Seja $I = (G, k)$ uma instância do problema do $k$-conjunto dominante com grafo $G = (V, E)$. Vamos criar uma instância $I' = (G', c, k)$ do problema dos $k$-centros a partir da instância $I$. A

instância $I'$ tem como grafo o grafo completo $G' = (V, E')$ e, para todo $e \in E'$,

$$c_e = \begin{cases} 1, \text{ se } e \in E \\ 2, \text{ caso contrário.} \end{cases}$$

Note que $c$ satisfaz a desigualdade triangular e pode ser obtida de $I$ em tempo polinomial. Precisamos mostrar que a resposta para $I$ é sim se e somente se $\text{opt}(I') = 1$.

Primeiramente, vamos mostrar que se a resposta para $I$ é sim, então $\text{opt}(I') = 1$. Se a resposta para $I$ é sim, então $G$ contém um $k$-conjunto dominante. Vamos chamar tal conjunto de $D$. É evidente que $\text{raio}(D) = 1$, uma vez que para todo $u \in V \setminus D$ existe $v \in D$ tal que $uv \in E$ e, portanto, $c_{uv} = 1$. Desse modo,

$$\text{raio}(D) = \max_{u \notin D} c(u, D) = \max_{u \notin D} 1 = 1.$$

Como toda aresta de $E'$ tem custo pelo menos 1, então $\text{opt}(I') = 1$.

Agora, vamos mostrar que se $\text{opt}(I') = 1$, então a resposta para $I$ é sim. Seja $S \subseteq V$ com $|S| \leq k$ tal que $\text{raio}(S) = 1$. Como $\text{raio}(S) = 1$, então para todo $u \notin S$ existe $v \in S$ tal que $c_{uv} = 1$ e, portanto, $uv \in E$. Desse modo, é evidente que $S$ é um $k$-conjunto dominante de $G$. Assim, a resposta para $I$ é sim.

Portanto, conseguimos resolver em tempo polinomial o problema do $k$-conjunto dominante, o que implicaria que $P = NP$. □

O resultado acima pode ser adaptado para dar um resultado mais forte de inaproximabilidade para a versão geral do problema dos $k$-centros, não restrita a métricas.

**Teorema 2.4.** *Seja $\alpha(n)$ uma função computável em tempo polinomial em $n$ com $\alpha(n) \geq 1$ para todo $n$. Não existe $\alpha(n)$-aproximação para a versão geral do problema dos $k$-centros, onde $n$ é o número de vértices do grafo da instância, a menos que $P = NP$.*

*Demonstração.* Suponha que exista um algoritmo polinomial $A$ que é uma $\alpha(n)$-aproximação para o problema dos $k$-centros e seja $I = (G, k)$ uma instância do problema do $k$-conjunto dominante em que $G = (V, E)$ é um grafo com $n$ vértices. Vamos criar uma instância $I' = (G', c, k)$ do problema dos $k$-centros a partir da instância $I$. A instância $I'$ tem como grafo o grafo completo $G' = (V, E')$ e, para cada $e \in E'$,

$$c_e = \begin{cases} 1, \text{ se } e \in E \\ \alpha(n) + 1, \text{ caso contrário.} \end{cases}$$

Como $\alpha(n)$ é uma função computável em tempo polinomial em $n$, essa instância pode ser construída a partir de $I$ em tempo polinomial. Se $\alpha(n) = 1$, então $c$ obedece a desigualdade triangular e $A$ é um algoritmo polinomial e exato, o que, pelo Teorema 2.3, é absurdo. Então, suponha $\alpha(n) > 1$.

O algoritmo $A$ aplicado à instância $I'$ encontra uma solução $S$ de tamanho $k$. Como $c_e = 1$ ou $c_e = \alpha(n) + 1$ para todo $e \in E'$, então raio$(S) = 1$ ou raio$(S) = \alpha(n) + 1$. Se raio$(S) = 1$ então todos os vértices estão ligados ao centro do seu cluster por uma aresta de $G$, logo $S$ é um conjunto dominante em $G$. Se raio$(S) = \alpha(n) + 1$, então opt$(I') \geq \frac{\alpha(n)+1}{\alpha(n)} > 1$. Assim, não existe solução $S'$ de $I'$ tal que raio$(S') = 1$ e não existe $k$-conjunto dominante em $G$.

Portanto, conseguimos resolver o problema do $k$-conjunto dominante em tempo polinomial o que, assumindo que $P \neq NP$, é um absurdo. □

Fica, então, explícita a impossibilidade de encontrarmos algoritmos de aproximação para a versão geral do problema dos $k$-centros, a menos que $P = NP$.

Agora que justificamos o estudo de algoritmos de aproximação para a versão métrica desse problema vamos mostrar um limitante inferior para a razão de aproximação desses algoritmos. Esse teorema é de autoria do Hsu e Nemhauser [13] e é o Teorema 5.7 do livro V2001.

**Teorema 2.5.** *Seja $\varepsilon \in (0, 1]$. Não existe $(2 - \varepsilon)$-aproximação para a versão métrica do problema dos $k$-centros, a menos que $P = NP$.*

A prova do Teorema 2.4 essencialmente serve também para esse teorema. A única diferença é que aqui assumimos a existência de uma $(2-\varepsilon)$-aproximação para o problema dos $k$-centros e utilizamos custo 2 para as arestas que não estão no grafo da instância do problema do $k$-conjunto dominante. Assim, chegamos no mesmo absurdo.

Vimos então que não existe aproximação para o caso geral do problema dos $k$-centros e que a melhor razão de aproximação possível para o $k$-centros métrico é 2, a menos que $P = NP$. Assim, começaremos com um algoritmo simples para o problema dos $k$-centros métrico, mas que garante a melhor razão de aproximação.

## 2.1 Algoritmo guloso

Nessa seção vamos falar sobre o algoritmo guloso para o problema dos $k$-centros. Esse algoritmo foi desenvolvido pelo González [8] e foi estudado na Seção 2.2 do livro WS2011.

A ideia desse algoritmo guloso se concentrará em, a cada iteração, escolher o vértice mais distante do conjunto de centros de clusters escolhidos até aquele momento para entrar no conjunto. Começaremos com um vértice arbitrário e iremos continuar inserindo novos centros de clusters até que tenhamos $k$ deles.

**Algoritmo 1** GULOSO-GONZÁLEZ$(G, c, k)$

1: Escolha arbitrariamente $u \in V(G)$.
2: $S \leftarrow \{u\}$.
3: **Enquanto** $|S| < k$ **faça**
4: $\quad v \leftarrow \arg\max_{j \in V(G)} c(j, S)$
5: $\quad S \leftarrow S \cup \{v\}$
6: **Devolva** $S$.

**Teorema 2.6.** *O algoritmo* GULOSO-GONZÁLEZ$(G, c, k)$ *é uma* 2*-aproximação do problema dos* $k$*-centros métrico.*

*Demonstração.* É evidente que o algoritmo roda em tempo polinomial.

Seja $S^*$ uma resposta ótima de $I = (G, c, k)$ e $S$ o conjunto devolvido pelo algoritmo GULOSO-GONZÁLEZ para a instância $I$. Vamos mostrar que $\text{raio}(S) \leq 2\,\text{raio}(S^*)$. Seja $v \in S^*$ e $u_1, u_2 \in V \setminus S$ dois vértices tais que $v = \arg\min_{i \in S} c_{iu_1} = \arg\min_{i \in S} c_{iu_2}$, ou seja, dois vértices no mesmo cluster que $v$. Vale que

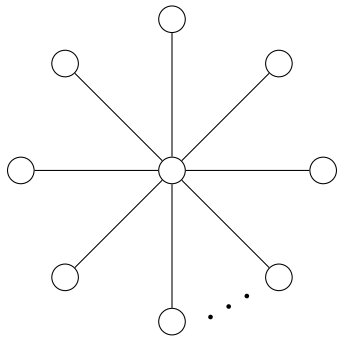
$$c_{u_1u_2} \leq c_{u_1v} + c_{vu_2} = c(u_1, S) + c(u_2, S) \leq 2\,\text{raio}(S^*),$$

em que a primeira desigualdade vale pela desigualdade triangular e a última desigualdade vale pela definição de $\text{raio}(S^*)$. Assim, a distância entre quaisquer dois vértices de um mesmo cluster é no máximo duas vezes o raio daquele cluster que por sua vez é no máximo $\text{raio}(S^*)$. Assim, se há exatamente um vértice de $S$ em cada cluster de $S^*$, então todos os vértices estão ligados com uma aresta de custo no máximo $2\,\text{opt}(I)$ a um vértice de $S$. Então $\text{raio}(S) \leq 2\,\text{opt}(I)$. Senão, seja $S_{i-1} \coloneqq \{u_0, u_1, \ldots, u_{i-1}\}$ o conjunto $S$ ao final da iteração $i-1$ do laço **Enquanto** da linha 3 e suponha que cada $u_j$ está em um cluster diferente de $S^*$ para $j = 1, \ldots, i-1$. Seja $u_i$ o vértice escolhido na iteração $i$ e suponha que ele está no mesmo cluster em $S^*$ que um vértice $u_j$ para algum $j < i$. Note que $\text{raio}(S_{i-1}) = c(u_i, S_{i-1}) \leq c_{u_iu_j}$, em que a primeira igualdade vale pela escolha feita na linha 4 e a segunda igualdade vale pela definição de $c(u_i, S_{i-1})$. Como $u_i$ e $u_j$ estão no mesmo cluster em $S^*$, vale que $c_{u_iu_j} \leq 2\,\text{raio}(S^*)$. Desse modo, concluímos que $\text{raio}(S) \leq \text{raio}(S_{i-1}) \leq 2\,\text{opt}(I)$.

□

Vamos mostrar que essa análise é justa, ou seja, existe uma instância $I = (G, c, k)$ em que o algoritmo devolve uma solução $S$ tal que $\text{raio}(S) = 2\,\text{opt}(I)$.

Seja $G$ um grafo com pelo menos $k+2$ vértices em que as arestas têm custo 1 ou 2. O grafo induzido pelas arestas de custo 1 é uma estrela, como mostrado na figura abaixo.



Claramente, o raio de uma resposta ótima dessa instância é 1 e inclui o vértice do centro da estrela como um dos centros de cluster. Note que se, no algoritmo guloso, o vértice escolhido arbitrariamente for algum dos vértices da ponta dessa estrela o vértice do centro nunca será escolhido, uma vez que ele nunca maximizará a função $c(j, S)$ na linha 4 do algoritmo. Assim, serão escolhidos apenas vértices da ponta da estrela e como temos pelo menos $k + 1$ delas, sempre existirá um vértice ligado ao centro do seu cluster com uma aresta de custo 2.

## 2.2 Método do gargalo

Nessa seção vamos apresentar o algoritmo que utiliza o método do gargalo para o problema dos $k$-centros métrico. Esse algoritmo foi desenvolvido por Hochbaum e Shmoys [12] e foi estudado no capítulo 5 do livro V2001.

Problemas de gargalo são problemas definidos em grafos com pesos nas arestas tais que a função objetivo é o peso de uma aresta. Note que esse é o caso do problema dos $k$-centros.

Algumas definições serão necessárias antes de mostrarmos o algoritmo.

**Definição 2.7.** *Seja $G = (V, E)$ um grafo. Um conjunto $S \subseteq V$ é um conjunto* independente *se não existe $uv \in E$ tal que $u, v \in S$.*

Seja $I = (G, c, k)$ uma instância do problema dos $k$-centros. Podemos supor que as arestas do grafo estão ordenadas de modo não decrescente pelo seu custo, ou seja, $E = \{e_1, e_2, \ldots, e_{|E|}\}$ com $c_{e_i} \leq c_{e_{i+1}}$ para todo $i = 1, \ldots, |E| - 1$. Seja

$E_i \coloneqq \{e_1, e_2, \ldots, e_i\}$ e $G_i \coloneqq (V, E_i)$. Seja também $i^*$ o menor $i$ tal que $G_i$ tem um $k$-conjunto dominante. Como $G$ é completo, $i^*$ existe. Claramente $c_{e_{i^*}} = \text{opt}(I)$, porém não conseguimos encontrar $i^*$ eficientemente, uma vez que não é possível saber se um grafo tem um $k$-conjunto dominante em tempo polinomial, a menos que $P = NP$. Vamos usar um conjunto independente maximal para aproximar uma resposta.

**Lema 2.8.** *Seja $G = (V, E)$ um grafo. Um conjunto independente maximal em $G$ é também um conjunto dominante.*

*Demonstração.* Seja $G = (V, E)$ um grafo e $S$ um conjunto independente maximal em $G$. Suponha, por absurdo, que $S$ não é um conjunto dominante. Então, existe vértice $u \in V \setminus S$ que não é vizinho de nenhum dos vértices de $S$. Portanto, $S \cup \{u\}$ é também um conjunto independente e $S \subset \{S \cup \{u\}\}$, uma contradição, pois $S$ é maximal. □

Então, se encontrarmos um conjunto independente maximal de tamanho $k$ em $G$ teremos um conjunto dominante de mesmo tamanho. No entanto, não conseguimos garantir que iremos encontrar esse conjunto em $G$ e, por isso, vamos definir e usar o chamado quadrado de $G$.

**Definição 2.9.** *Seja $G = (V, E)$ um grafo. Denotamos por $G^2 = (V, E^2)$ o* quadrado *de $G$ em que $E^2 = E \cup \{uv : u \text{ e } v \text{ têm vizinhos em comum em } G\}$.*

Dada a definição vamos enunciar e provar um lema que nos ajudará no algoritmo.

**Lema 2.10.** *Seja $G$ um grafo e $k$ um inteiro positivo. Se $G$ contém um $k$-conjunto dominante então todo conjunto independente em $G^2$ tem tamanho no máximo $k$.*

*Demonstração.* Seja $S \subseteq V(G)$ um conjunto independente em $G^2$ e $D \subseteq V(G)$ um $k$-conjunto dominante em $G$. Vamos mostrar que $|S| \leq |D|$. Seja $u \in D$ e seja $N(u) \coloneqq \{v \in V(G) : uv \in E(G)\}$ o conjunto dos vizinhos de $u$ em $G$. Note que $u$ e $N(u)$ formam uma estrela em $G^2$, uma vez que todos os vértices em $N(u)$ têm $u$ como vizinho em comum. Desse modo, para cada $u \in D$ no máximo um vértice de $u \cup N(u)$ pode estar em $S$. Como $D$ é um conjunto dominante, todos os vértices de $G$ estão em $D$ ou na vizinhança de algum vértice de $D$. Assim, $|S| \leq |D| \leq k$.

□

Agora, temos todas as definições e lemas que serão necessários para o algoritmo.

**Algoritmo 2** GARGALO-HS$(G, c, k)$

1: $i \leftarrow 0$
2: $M_0 \leftarrow V(G)$
3: **Enquanto** $|M_i| > k$ **faça**
4: $\quad i \leftarrow i + 1$
5: $\quad$ Seja $M_i$ um conjunto independente maximal em $G_i^2$
6: **Devolva** $M_i$

**Teorema 2.11.** *O algoritmo* GARGALO-HS *é uma 2-aproximação do problema dos k-centros métrico.*

*Demonstração.* Primeiro vamos mostrar que o algoritmo é polinomial.

Como $G_{i^*}$ tem um $k$-conjunto dominante, então o laço vai iterar no máximo $i^* \leq |E|$ vezes, pois pelo Lema 2.10 qualquer conjunto independente encontrado em $G_{i^*}^2$ terá tamanho no máximo $k$. Também é fácil mostrar que é possível encontrar um conjunto independente maximal em tempo polinomial. Um algoritmo simples começa com um conjunto $A = \{u\}$ sendo $u$ um vértice arbitrário e, a cada iteração, coloca em $A$ um vértice que não é adjacente a nenhum vértice de $A$ até não ser mais possível. Além disso, também conseguimos construir o grafo $G_i^2$ em tempo polinomial. Começaremos $E_i^2$ como uma cópia de $E_i$ e, para cada tripla de vértice $(u, v, w)$ caso já não exista uma aresta $uw \in E_i$, vamos inseri-la em $E_i^2$ se $v$ for um vizinho comum de $u$ e $w$ em $E_i$. Como temos no máximo $|V|^3$ triplas de vértices e todas as operações que serão feitas tomam tempo polinomial, então podemos construir $G_i^2$ em tempo polinomial.

Agora, vamos mostrar que o algoritmo é uma 2-aproximação.

Para um grafo $H$ com peso nas arestas, definimos $\max(H)$ como o maior peso de uma aresta. Seja $i'$ o valor de $i$ ao final do algoritmo e $M_{i'}$ a solução devolvida por ele. Como $M_{i'}$ é um conjunto independente maximal de tamanho no máximo $k$, então pelo Lema 2.8 ele é um $k$-conjunto dominante. Como o grafo induzido $G_{i'}[M_{i'}]$ é um subgrafo de $G_{i'}^2$ então $\max(G_{i'}[M_{i'}]) \leq \max(G_{i'}^2)$. Pela desigualdade triangular, é fácil notar que $\max(G_{i'}^2) \leq 2 \max(G_{i'})$. Assim,

$$\max(G_{i'}[M_{i'}]) \leq \max(G_{i'}^2) \leq 2 \max(G_{i'}) \leq 2 \max(G_{i^*}) = 2\text{opt}(I).$$

□

# 3 Localização de instalações

Nessa seção falaremos sobre a versão métrica do problema de localização de instalações. O problema de localização de instalações métrico consiste em, dado um conjunto de instalações $F$, um conjunto de clientes $D$, uma métrica $c : F \times D \to \mathbb{R}$ e uma função de custo de abertura $f : F \to \mathbb{R}$, encontrar $S \subseteq F$ que minimize $\text{custo}(S) \coloneqq \sum_{i \in S} f_i + \sum_{j \in D} \min_{i \in S} c_{ij}$.

Antes de falarmos sobre algoritmos de aproximação para o problema da localização de instalações métrico, vamos mostrar que, assumindo $P \neq NP$, não existe algoritmo polinomial que resolva nosso problema, ou seja, vamos mostrar que nosso problema é $NP$-difícil. Para isso, vamos definir o problema da cobertura por vértices.

**Problema 3.1.** *(Cobertura por vértices) Dado um grafo $G$ e um inteiro $k$, decidir se $G$ tem uma cobertura por vértices de tamanho $k$.*

Esse problema é $NP$-completo, sendo um dos famosos 21 problemas do Karp [17]. Disso deriva-se o seguinte.

**Teorema 3.2.** *O problema de localização de instalações métrico é $NP$-difícil.*

*Demonstração.* Seja $I = (G, k)$ uma instância do problema da cobertura por vértices. Tome $F \coloneqq V(G)$ e $D \coloneqq E(G)$. Considere também $f_i \coloneqq 1$ para todo $i \in V$ e $c_{ij} \coloneqq 1$ se $i \in V(G)$ é extremo de $j \in E(G)$ e $c_{ij} \coloneqq 3$ caso contrário. Note que $c$ é uma métrica. Assim, construímos uma instância $I' = (F, D, c, f)$ do problema de localização de instalações métrico a partir de uma instância do problema da cobertura por vértices. Precisamos mostrar que a resposta para $I$ é sim se e somente se $\text{opt}(I') \leq |D| + k$.

Vamos mostrar que se a resposta para $I$ é sim, então $\text{opt}(I') \leq |D| + k$. Como a resposta para $I$ é sim, então existe uma cobertura por vértices $S$ tal que $|S| \leq k$. É fácil notar que a solução de $I'$ em que abrimos as instalações referentes a $S$ tem custo no máximo $|D| + k$, uma vez que para cada cliente $j$ existe $i \in S$ tal que $i$ é extremo de $j$ e, consequentemente, $c_{ij} = 1$ e o custo de abertura das instalações é no máximo $k$ uma vez que cada instalação tem custo de abertura 1. Portanto, $\text{opt}(I') \leq |D| + |k|$.

Agora vamos mostrar que se $\text{opt}(I') \leq |D| + k$, então a resposta para $I$ é sim. Seja $X$ a resposta ótima da instância $I'$. Note que não existe $j \in D$ tal que $c(j, X) = 3$. Caso contrário, conseguimos diminuir o custo de $X$ abrindo algum $i \in F$, tal que $c_{ij} = 1$. É evidente que $i$ existe, uma vez que para todo $j \in D$ existem exatamente duas instalações que satisfazem $c_{ij} = 1$, que são os extremos da aresta $j$. Como, para todo $j \in D$, $c(j, X) = 1$, e, como $X$ é ótimo, então $X$ é uma cobertura de vértices

mínima de $G$. Assim, $\text{opt}(I') - |D| \leq k$ é o tamanho de uma cobertura mínima de vértices de $G$.

□

## 3.1 Algoritmos baseados em programação linear

Nessa seção vamos mostrar algoritmos para o problema de localização de instalações que utilizam métodos baseados em programação linear.

Vamos modelar o problema de localização de instalações como um programa linear inteiro. Vamos relaxar esse programa e encontrar o seu dual.

Para uma instância $(F, D, c, f)$ do problema de localização de instalações, o programa inteiro terá dois tipos de variáveis. Uma variável $y_i$ para cada $i \in F$ que terá valor 1 se a instalação $i$ foi aberta e 0, caso contrário, e uma variável $x_{ij}$, para cada $i \in F$ e $j \in D$, que terá valor 1 se o cliente $j$ estiver associado a instalação $i$ e 0, caso contrário.

Assim, uma instância $(F, D, c, f)$ do problema de localização de instalações pode ser modelada como o seguinte programa linear inteiro:

$$\begin{aligned}
\text{Minimizar} \quad & \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in D} c_{ij} x_{ij} & & \\
\text{sujeito a} \quad & \sum_{i \in F} x_{ij} \geq 1, & \forall j \in D & \quad (1) \\
& x_{ij} \leq y_i, & \forall i \in F, j \in D & \quad (2) \\
& x_{ij} \in \{0, 1\}, & \forall i \in F, j \in D & \quad (3) \\
& y_i \in \{0, 1\}, & \forall i \in F. & \quad (4)
\end{aligned}$$

A restrição (1) garante que todo cliente esteja associado a alguma instalação, a restrição (2) garante que todo cliente esteja associado apenas a instalações abertas.

Para a relaxação desse programa permitiremos que as variáveis em $x$ e em $y$ adotem quaisquer valores não negativos. Portanto, a relaxação do programa inteiro do problema de localização de instalações resulta no seguinte programa.

$$\text{Minimizar} \quad \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in D} c_{ij} x_{ij} \tag{PL}$$

$$\text{sujeito a} \quad \sum_{i \in F} x_{ij} \geq 1, \qquad \forall j \in D \tag{P2}$$

$$y_i - x_{ij} \geq 0, \qquad \forall i \in F, j \in D \tag{P3}$$

$$x_{ij} \geq 0, \qquad \forall i \in F, j \in D \tag{P4}$$

$$y_i \geq 0, \qquad \forall i \in F. \tag{P5}$$

O dual do programa linear acima consiste no seguinte programa.

$$\text{Maximizar} \quad \sum_{j \in D} v_j \tag{PD}$$

$$\text{sujeito a} \quad \sum_{j \in D} w_{ij} \leq f_i, \qquad \forall i \in F \tag{D2}$$

$$v_j - w_{ij} \leq c_{ij}, \qquad \forall i \in F, j \in D \tag{D3}$$

$$w_{ij} \geq 0, \qquad \forall i \in F, j \in D \tag{D4}$$

$$v_j \geq 0, \qquad \forall j \in D. \tag{D5}$$

Sabemos que cada variável de um destes programas está associada a uma restrição do outro programa. Especificamente, a variável $x_{ij}$ está associada à desigualdade (D3), a variável $y_i$ está associado à desigualdade (D2), a variável $w_{ij}$ está associada à desigualdade (P3) e a variável $v_j$ está associado a desigualdade (P2).

Note que toda solução viável do programa linear inteiro é solução viável do programa linear relaxado. Desse modo, a resposta ótima do programa linear relaxado tem valor objetivo no máximo o valor objetivo da resposta ótima do programa linear inteiro.

Vamos aqui interpretar o programa dual. Chamaremos as variáveis $v$ de orçamento e as variáveis $w$ de contribuição. Dizemos que um cliente $j$ *contribui* para uma instalação $i$ se $w_{ij} > 0$. Uma instalação recebe contribuições dos clientes para pagar pela sua abertura. Uma vez que as contribuições são suficientes para a sua abertura, a instalação não precisa receber mais contribuições. Isso está explicito na restrição (D2).

O orçamento de um cliente é no máximo o custo de sua associação a uma instalação e sua contribuição para a abertura dela. Isso está explicito na restrição (D3).

Para entender como as variáveis primais e duais se relacionam, vamos analisar as condições de folgas complementares. Vamos supor a existência de uma solução ótima

inteira $(x^*, y^*)$ para o primal e seja $(v^*, w^*)$ uma solução ótima para o dual. Em uma solução inteira do primal, as variáveis $x$ e $y$ satisfazem as restrições do programa inteiro. Como ambas são soluções ótimas, valem as folgas complementares. A partir dos pares variável-restrição correspondentes já vistos, as condições de folgas complementares estabelecem que

(i) para todo $i \in F$ e $j \in D$, se $x^*_{ij} > 0$ então $v^*_j - w^*_{ij} = c_{ij}$;

(ii) para todo $i \in F$, se $y^*_i > 0$ então $\sum_{j \in D} w^*_{ij} = f_i$;

(iii) para todo $i \in F$ e $j \in D$, se $w^*_{ij} > 0$ então $y_i = x_{ij}$;

(iv) para todo $j \in D$ se $v^*_j > 0$ então $\sum_{i \in F} x^*_{ij} = 1$.

Pela condição (i), se um cliente $j$ está associado a uma instalação $i$ então o orçamento do cliente $j$ é exatamente o custo dele se associar a $i$ mais a sua contribuição para a abertura de $i$. Então podemos interpretar $v_j$ como o valor que o cliente paga para a instalação a qual ele estará associado. Pela condição (ii), cada instalação aberta precisa ter contribuições suficientes para pagar pela sua abertura. Pela condição (iii), temos que um cliente que contribui para uma instalação aberta está associado a ela. Juntando as condições (ii) e (iii), temos que as contribuições recebidas pelas instalações abertas são vindas apenas de clientes associados a elas e são suficientes para pagar pela sua abertura. Pela condição (iv), um cliente que tem orçamento não nulo está associado a exatamente uma instalação.

### 3.1.1 Método primal-dual

Nessa seção vamos apresentar o algoritmo primal-dual para o problema de localização de instalações métrico. Esse algoritmo foi estudado na Seção 7.6 do livro WS2011 e foi desenvolvido pelo Jain e pelo Vazirani [16]. Vamos utilizar os programas lineares (PL) e (PD).

Chamamos uma solução $(v, w)$ para (PD) de *maximal* se não existe solução viável $(v', w')$ tal que:

(i) $v_j \leq v'_j$ para todo cliente $j$;

(ii) $w_{ij} \leq w'_{ij}$ para todo cliente $j$ e instalação $i$;

(iii) $v_j < v'_j$ para algum cliente $j$;

ou seja, não conseguimos encontrar uma solução viável com valor objetivo maior apenas aumentando os valores das variáveis.

Vamos utilizar a seguinte definição ao longo da explicação do método.

**Definição 3.3.** *Seja* $(v, w)$ *uma solução viável de* (PD). *Denotamos por* $N(j) \coloneqq \{i \in F : v_j \geq c_{ij}\}$ *a* vizinhança *de um cliente* $j$ *e* $N(i) \coloneqq \{j \in D : v_j \geq c_{ij}\}$ *a* vizinhança *de uma instalação* $i$.

**Teorema 3.4.** *Seja* $(v, w)$ *uma solução maximal de* (PD) *e* $T \coloneqq \{i \in F : \sum_{j \in D} w_{ij} = f_i\}$ *o conjunto de instalações que receberam contruibuições suficientes para serem abertas. Então, todo cliente está na vizinhança de uma instalação em* $T$.

*Demonstração.* Vamos provar por absurdo. Seja $(v, w)$ uma solução maximal e suponha a existência de um cliente $j$ que não está na vizinhança de nenhuma instalação em $T$. Como $(v, w)$ é maximal então existe $i \in F$ tal que a desigualdade (D3) é justa para $i$ e $j$, caso contrário conseguiríamos aumentar $v_j$. Como $j$ não tem vizinho em $T$, é evidente que $i \notin T$, uma vez que, para $i \in T$, vale que $v_j < c_{ij} \leq c_{ij} + w_{ij}$. Portanto, vale que $v_j = c_{ij} + w_{ij}$ para algum $i \in F \setminus T$. Como $i \notin T$, então $\sum_{j \in D} w_{ij} < f_i$. Assim, conseguimos aumentar o valor de $w_{ij}$ sem violar (D2) e, após isso, aumentar $v_j$ no mesmo valor sem violar (D3). □

Como cada cliente está na vizinhança de uma instalação que pertence a $T$, abrir todas as instalações em $T$ seria suficiente, mas um cliente poderia contribuir para mais que uma instalação de $T$. Assim, possivelmente estaríamos contando o orçamento de um cliente na contribuição para a abertura de mais de uma instalação, o que interferiria na comparação do custo da solução com o valor objetivo de (PD) e, consequentemente, com o valor da solução ótima. Para evitar este problema, podemos escolher um conjunto $T' \subseteq T$ tal que cada cliente contribua para no máximo uma instalação de $T'$ e vamos garantir que um cliente que não esteja na vizinhança de nenhuma instalação de $T'$ esteja próximo a alguma delas.

Nosso algoritmo contará com as seguintes invariantes.

- $(v, w)$ é uma solução viável de (PD);
- $T$ é um conjunto de instalações que têm contribuições suficientes para serem abertas; e
- $S$ é o conjunto de clientes que não têm nenhum vizinho em $T$.

**Algoritmo 3** PrimalDual-JV$(F, D, c, f)$

1: $v_j \leftarrow 0$ para todo $j \in D$
2: $w_{ij} \leftarrow 0$ para todo $i \in F$ e $j \in D$
3: $S \leftarrow D$
4: $T \leftarrow \emptyset$
5: **Enquanto** $S \neq \emptyset$ **faça**
6: $\quad \theta_1 \leftarrow \min\{c_{ij} - v_j : j \in D \text{ e } i \notin N(j)\}$
7: $\quad \theta_2 \leftarrow \min\{(f_i - \sum_{j \in N(i)} w_{ij})/|N(i)| : i \in F \text{ tal que } N(i) \neq \emptyset\}$
8: $\quad \theta \leftarrow \min\{\theta_1, \theta_2\}$
9: $\quad v_j \leftarrow v_j + \theta$ para todo $j \in D$
10: $\quad w_{ij} \leftarrow w_{ij} + \theta$ para todo $j \in D$ e $i \in N(j)$
11: $\quad$ **Se** $v_j = c_{hj}$ para algum $j \in S$ e $h \in T$ **então**
12: $\quad\quad S \leftarrow S \setminus \{j\}$
13: $\quad$ **Se** $\sum_{j \in N(i)} w_{ij} = f_i$ para algum $i \notin T$ **então**
14: $\quad\quad T \leftarrow T \cup \{i\}$
15: $\quad\quad S \leftarrow S \setminus N(i)$
16: $T' \leftarrow \emptyset$
17: **Enquanto** $T \neq \emptyset$ **faça**
18: $\quad$ Escolha $i \in T$ arbitrariamente
19: $\quad T' \leftarrow T' \cup \{i\}$
20: $\quad T \leftarrow T \setminus \{h \in T : \text{existe } k \in D \text{ com } w_{ik} > 0 \text{ e } w_{hk} > 0\}$
21: **Devolva** $T'$

No algoritmo PrimalDual-JV, claramente as coordenadas das variáveis $v$ e $w$ crescerão de maneira uniforme, pois em cada iteração o mesmo valor $\theta$ será somado a elas. Precisamos então garantir que a nossa solução seja sempre viável. Pela escolha de $\theta_1$ na linha 6 do algoritmo, garantimos que a desigualdade (D3) não seja violada para $i$ e $j$ em que $i$ é uma instalação que não está na vizinhança do cliente $j$. Quando a instalação $i$ está na vizinhança de $j$, aumentamos $v_j$ e $w_{ij}$ do mesmo valor $\theta$ e isso nunca violará a desigualdade (D3). Pela escolha de $\theta_2$ na linha 7, garantimos que a desigualdade (D2) não seja violada para nenhuma instalação. Assim, $(v, w)$ é uma solução viável de (PD) durante todo o algoritmo. Para cada cliente $j$ e cada instalação $i$, primeiro aumentaremos apenas as variáveis $v_j$ até que $j$ esteja na vizinhança de $i$. Note que nesse momento a desigualdade (D3) se torna justa para $i$ e $j$, pois ainda não aumentamos a variável $w_{ij}$. Uma vez que isso acontece, aumentamos uniformemente $v_j$ e $w_{ij}$, assim a desigualdade (D3) continuará justa. Desse modo, para $j \in D$ e $i \in N(j)$, vale que

$$w_{ij} = v_j - c_{ij}. \tag{5}$$

Note também que ao final do algoritmo $(v, w)$ é uma solução maximal de (PD). Isso acontece pois para todo cliente $j$ existe uma instalação $i$ tal que a desigualdade (D3) é justa para $i$ e $j$ e, além disso, a desigualdade (D2) é justa para $i$. Desse modo, não conseguimos encontrar uma solução viável com valor objetivo maior apenas aumentando as variáveis, pois $w_{ij}$ não pode ser aumentado e é ele quem está impedindo o aumento de $v_j$.

Para provar a razão de aproximação vamos precisar primeiro de um lema.

**Lema 3.5.** *Seja $T'$ a solução devolvida e $(v, w)$ solução de* (PD) *produzida por* PrimalDual-JV. *Se $j \in D$ não está na vizinhança de nenhuma instalação de $T'$, então existe uma instalação $i \in T'$ tal que $c_{ij} \leq 3v_j$.*

*Demonstração.* Seja $h \in T$ a instalação responsável pela remoção de $j$ de $S$, conforme a linha 12 ou 16 do algoritmo. Claramente, $j$ pertence a vizinhança de $h$. Sabemos que $h$ não pertence a $T'$ uma vez que, por hipótese, $j$ não tem vizinhos em $T'$. Como $h$ foi retirada de $T$, conforme a linha 21 do algoritmo, existe $i \in T'$ e $k \in D$ tal que $k$ contribui para $i$ e para $h$. Pela desigualdade triangular,

$$c_{ij} \leq c_{hj} + c_{hk} + c_{ik}$$

como $j \in N(h)$ e $k \in N(h) \cap N(i)$ vale que $c_{hj} \leq v_j$ e $c_{hk} + c_{ik} \leq 2v_k$. Como $k$ contribui para $h$, então $k$ já estava na vizinhança de $h$ no momento que $h$ foi retirado de $T$. Assim, $k$ saiu de $S$ no mesmo momento ou em um momento anterior a $h$ sair de $T$. Como $h$ foi responsável pela retirada de $j$ de $S$, então $j$ não foi retirado de $S$ antes de $k$. Como as variáveis crescem de maneira uniforme, então $v_k \leq v_j$. Portanto, $c_{ij} \leq 3v_j$. □

Agora, podemos mostrar o teorema abaixo.

**Teorema 3.6.** *O algoritmo* PrimalDual-JV *é uma 3-aproximação para o problema de localização de instalações métrico.*

*Demonstração.* O algoritmo roda em tempo polinomial, uma vez que para cada iteração do laço **Enquanto** da linha 5 pelo menos uma restrição de (PD) se torna justa. Desse modo, o laço executa $O(|F| \cdot |D|)$ iterações. As outras linhas são claramente polinomiais.

Para cada cliente que contribui para uma instalação de $T'$, vamos associá-lo a essa instalação. Como cada cliente contribui para no máximo uma instalação de $T'$, então essa associação é única. Para clientes que estão na vizinhança de instalações de $T'$, mas não contribuem para nenhuma delas, vamos associá-los a qualquer instalação de $T'$ na

sua vizinhança. Seja $A(i) \subseteq N(i)$ os clientes vizinhos associados a instalação $i \in T'$. Então o custo de abertura das facilidades em $T'$ mais o custo de associar os clientes vizinhos é

$$\sum_{i \in T'} (f_i + \sum_{j \in A(i)} c_{ij}) = \sum_{i \in T'} \sum_{j \in A(i)} (w_{ij} + c_{ij}) = \sum_{i \in T'} \sum_{j \in A(i)} v_j$$

em que a primeira igualdade vale pela definição de $T$ e a segunda igualdade vale por (5) e também pois $w_{ij} > 0$ implica $j \in A(i)$. Claramente o somatório não repete clientes, uma vez que cada cliente está associado a apenas uma instalação.

Para um cliente $j$ que não está na vizinhança de nenhuma instalação de $T'$, podemos utilizar o Lema 3.5. Seja $Z$ o conjunto de clientes que não têm vizinhos em $T'$. Temos

$$\sum_{j \in Z} c(j, T') \leq 3 \sum_{j \in Z} v_j.$$

Juntando os limitantes encontrados para os clientes que têm vizinhos em $T'$ e os que não têm, encontramos

$$\sum_{i \in T'} (f_i + \sum_{j \in A(i)} c_{ij}) + \sum_{j \in Z} c(j, T') \leq \sum_{i \in T'} \sum_{j \in A(i)} v_j + 3 \sum_{j \in Z} v_j \leq 3 \sum_{j \in D} v_j \leq 3 \operatorname{opt}(I)$$

em que a última desigualdade vale pela dualidade fraca. □

### 3.1.2 Arredondamento do programa linear

Nessa seção vamos apresentar o algoritmo que faz arredondamento determinístico de uma solução relaxada do programa inteiro que modela o problema localização de instalações. Esse algoritmo foi estudado na Seção 4.5 do livro WS2011 e foi desenvolvido por Chudak e Shmoys [5]. Vamos utilizar os programas lineares (PL) e (PD).

Começaremos com algumas definições que serão necessárias para o algoritmo.

**Definição 3.7.** *Seja* $(x, y)$ *uma solução de* (PL). *Denotamos por* $N(j) \coloneqq \{i \in F : x_{ij} > 0\}$ *a* vizinhança *de um cliente* $j$. *Além disso, denotamos por* $N^2(j) \coloneqq \{\ell \in D : N(j) \cap N(\ell) \neq \emptyset\}$ *o conjunto de clientes que compartilham vizinhos com* $j$.

A ideia do algoritmo é que escolheremos de maneira quase gulosa clientes e instalações que contribuem pouco no valor objetivo de (PD).

Para provar a razão de aproximação de ArredDet-CS vamos precisar de uma sequência de lemas. No primeiro deles, vamos limitar o custo de associação dos clientes

**Algoritmo 4** ArredDet-CS$(F, D, c, f)$

1: Sejam $(x^*, y^*)$ e $(v^*, y^*)$ soluções ótimas para (PL) e (PD).
2: $X \leftarrow \emptyset$
3: $S \leftarrow D$
4: **Enquanto** $S \neq \emptyset$ **faça**
5: Escolha $j \in S$ que minimize $v_j^*$
6: Escolha $i \in N(j)$ que minimize $f_i$
7: $X \leftarrow X \cup \{i\}$
8: $S \leftarrow S \setminus N^2(j)$
9: **Devolva** $X$

escolhidos na linha 5 do algoritmo.

**Lema 3.8.** *Sejam $(x^*, y^*)$ e $(v^*, w^*)$ soluções ótimas de* (PL) *e* (PD)*, respectivamente, e $j$ um cliente qualquer. Para todo $i \in N(j)$, vale que $c_{ij} \leq v_j^*$.*

*Demonstração.* Como $i \in N(j)$, então $x_{ij}^* > 0$. Assim, pelas folgas complementares, a desigualdade (D3) do dual correspondente a variável $x_{ij}^*$ vale por igualdade, então $v_j^* - w_{ij}^* = c_{ij}$. Como $w_{ij}^* \geq 0$, então $c_{ij} \leq v_j^*$. □

Com esse lema, conseguimos provar o segundo lema que limita o custo dos clientes em $N^2(j)$ que são removidos de $S$ na linha 8 do algoritmo.

**Lema 3.9.** *Sejam $(x^*, y^*)$ e $(v^*, w^*)$ soluções ótimas de* (PL) *e* (PD)*, respectivamente, e $j$ um cliente qualquer. Para todo $i \in N(j)$ e $\ell \in N^2(j)$ tal que $v_j^* \leq v_\ell^*$, vale que $c_{i\ell} \leq 3v_\ell^*$.*

*Demonstração.* Seja $h \in N(j) \cap N(\ell)$. Pela desigualdade triangular, vale que $c_{i\ell} \leq c_{ij} + c_{hj} + c_{h\ell}$. Pelo Lema 3.8, vale que $c_{ij} \leq v_j^*$, $c_{hj} \leq v_j^*$ e $c_{h\ell} \leq v_\ell^*$, uma vez que $i, h \in N(j)$ e $h \in N(\ell)$. Assim,

$$c_{i\ell} \leq c_{ij} + c_{hj} + c_{h\ell} \leq 2v_j^* + v_\ell^* \leq 3v_\ell^*.$$

□

Assim, resta apenas limitar o custo de abertura das instalações que escolhemos na linha 6 do algoritmo.

**Lema 3.10.** *Sejam $(x^*, y^*)$ e $(v^*, w^*)$ soluções ótimas de* (PL) *e* (PD)*, respectivamente, e $j$ um cliente qualquer. Para $i$ que minimiza $f_i$ em $N(j)$, vale que*

$$f_i \leq \sum_{h \in N(j)} f_h y_h^*.$$

*Demonstração.* É evidente que

$$f_i \leq f_i \sum_{h \in N(j)} x^*_{hj} = \sum_{h \in N(j)} f_i x^*_{hj} \leq \sum_{h \in N(j)} f_h y^*_h,$$

onde a primeira desigualdade vale pela restrição (P2) e pela definição de $N(j)$ e a última desigualdade vale pela restrição (P3) e por $i$ minimizar $f_i$ em $N(j)$. □

Com isso, conseguimos provar a razão de aproximação de ArredDet-CS.

**Teorema 3.11.** *O algoritmo* ArredDet-CS *é uma* 4*-aproximação para o problema de localização de instalações métrico.*

*Demonstração.* Primeiro, vamos mostrar que o algoritmo é polinomial. Sabemos que é possível encontrar uma solução para (PL) e (PD) em tempo polinomial utilizando o método dos elipsoides [18]. Sabemos que o laço da linha 5 irá executar no máximo $|D|$ iterações, uma vez que sempre retiramos pelo menos um elemento de $S$. Além disso, as linhas $6 - 12$ são claramente polinomiais.

Agora, vamos mostrar que o algoritmo é uma 4-aproximação. Vamos denotar por $S_k$ o conjunto $S$ no início da iteração $k$ do laço **Enquanto** da linha 4. Além disso, vamos denotar por $j_k$ o cliente escolhido na linha 5 nessa iteração e $i_k$ a instalação escolhida na linha 6 dessa iteração. Note que

$$\begin{aligned}
\text{custo}(X) &\leq \sum_k \left( f_{i_k} + \sum_{\ell \in N^2(j_k) \cap S_k} c(i_k, \ell) \right) \\
&= \sum_k f_{i_k} + \sum_k \sum_{\ell \in N^2(j_k) \cap S_k} c(i_k, \ell) \\
&\leq \sum_k \sum_{h \in N(j_k)} f_h y^*_h + \sum_k \sum_{\ell \in N^2(j_k) \cap S_k} c(i_k, \ell) \\
&\leq \sum_{i \in F} f_i y^*_i + \sum_k \sum_{\ell \in N^2(j_k) \cap S_k} 3 v^*_\ell \\
&\leq \sum_{i \in F} f_i y^*_i + 3 \sum_{j \in D} v^*_j \\
&\leq \text{opt}(I) + 3\,\text{opt}(I) = 4\,\text{opt}(I),
\end{aligned}$$

em que a segunda desigualdade vale pelo Lema 3.10, a terceira desigualdade vale pelo Lema 3.9 e a quinta desigualdade vale pela dualidade fraca. Note que não existe repetição de termo no somatório $\sum_k \sum_{h \in N(j_k)}$, pois sempre que escolhemos um cliente

na linha 5 do algoritmo, retiramos todos os clientes que tem instalações em comum em suas vizinhanças. Assim, vale a quarta desigualdade.

□

### 3.1.3 Arredondamento probabilístico

Nessa seção vamos apresentar o algoritmo que faz arredondamento probabilístico de uma solução ótima de (PL) e (PD). Esse algoritmo foi estudado na seção 5.8 do livro WS2011 e foi desenvolvido pelo Chudak e Shmoys [5]. Vamos utilizar os programas lineares (PL) e (PD).

Relembre a Definição 3.7. Além disso, para uma solução ótima $(x^*, y^*)$ de (PL), denotamos $C_j^* \coloneqq \sum_{i \in F} x_{ij}^* c_{ij}$ para todo cliente $j \in D$.

**Algoritmo 5** ArredProb-CS$(F, D, c, f)$

1: Sejam $(x^*, y^*)$ e $(v^*, y^*)$ soluções ótimas de (PL) e (PD), respectivamente.
2: $X \leftarrow \emptyset$
3: $S \leftarrow D$
4: **Enquanto** $S \neq \emptyset$ **faça**
5: Escolha $j \in S$ que minimize $v_j^* + C_j^*$
6: Escolha uma instalação em $N(j)$ aleatoriamente com probabilidade $x_{ij}^*$ para a instalação $i$.
7: $X \leftarrow X \cup \{i\}$
8: $S \leftarrow S \setminus N^2(j)$
9: **Devolva** $X$

**Teorema 3.12.** *O algoritmo* ArredProb-CS *é uma 3-aproximação para o problema de localização de instalações métrico.*

*Demonstração.* É evidente que o algoritmo toma tempo polinomial.

Seja $j_k$ o cliente escolhido na linha 5 de uma iteração $k$ qualquer do algoritmo e $S_k$ o conjunto $S$ no início da iteração $k$. Note que, para um mesmo par de soluções ótimas de (PL) e (PD), a escolha de $j_k$ para uma iteração $k$ qualquer é determinística e, portanto, $N^2(j_k) \cap S_k$ é sempre igual para uma iteração $k$. Portanto, a parte probabilística do algoritmo se da apenas na escolha de uma instalação $i \in N(j)$. Seja $i_k$ a variável aleatória que indica a instalação em $N(j_k)$ que será escolhida na linha 6 do algoritmo. Assim, temos que $P(i_k = i) = x_{ij_k}^*$ para todo $i \in N(j_k)$. Desse modo, o custo esperado de abertura para $i_k$ é

$$\mathbb{E}[f_{i_k}] = \sum_{i \in N(j_k)} f_i P(i_k = i) = \sum_{i \in N(j_k)} f_i x^*_{ij_k} \leq \sum_{i \in N(j_k)} f_i y^*_i$$

onde a desigualdade vale pela restrição P3. Seja $c_{i_k j}$ a variável aleatória que representa o custo de transporte do cliente $j \in N^2(j_k) \cap S_k$. Para $j_k$ temos que

$$\mathbb{E}[c_{i_k j_k}] = \sum_{i \in N(j_k)} c_{ij_k} P(i_k = i) = \sum_{i \in N(j_k)} c_{ij_k} x^*_{ij_k} = C^*_{j_k}.$$

Seja $\ell$ um cliente em $N^2(j_k) \cap S_k$ diferente de $j_k$ e $h \in N(j_k)$ tal que $x^*_{h\ell} > 0$. Note que, pela definição de $N^2(j_k)$, $h$ existe. Pela desigualdade triangular vale que $c_{i\ell} \leq c_{ij_k} + c_{hj_k} + c_{hj}$ e, portanto,

$$\begin{aligned}
\mathbb{E}[c_{i_k \ell}] = \sum_{i \in N(j_k)} c_{i\ell} P(i_k = i) &= \sum_{i \in N(j_k)} c_{i\ell} x^*_{ij_k} \\
&\leq \sum_{i \in N(j_k)} (c_{ij_k} + c_{hj} + c_{h\ell}) x^*_{ij_k} \\
&= c_{hj} + c_{h\ell} + C^*_{j_k} \\
&\leq v^*_\ell + v^*_{j_k} + C^*_{j_k} \\
&\leq 2v^*_\ell + C^*_\ell
\end{aligned}$$

onde a segunda desigualdade vale pelo Lema 3.5 e a terceira vale pela escolha de $j_k$.

Então fica evidente que o valor esperado da nossa solução é

$$\begin{aligned}
\sum_k \big(\mathbb{E}[f_{i_k}] + \sum_{j \in N^2(j_k) \cap S_k} \mathbb{E}[c_{i_k j}]\big) &\leq \sum_k \big( \sum_{i \in N(j_k)} f_i y^*_i + \sum_{j \in N^2(j_k) \cap S_k} (2v^*_j + C^*_j)\big) \\
&= \sum_k \sum_{i \in N(j_k)} f_i y^*_i + \sum_k \sum_{j \in N^2(j_k) \cap S_k} (2v^*_j + C^*_j) \\
&\leq \sum_{i \in F} f_i y^*_i + \sum_{j \in D} (2v^*_j + C^*_j) \\
&= \sum_{i \in F} f_i y^*_i + \sum_{i \in F, j \in D} c_{ij} x^*_{ij} + 2 \sum_{j \in D} v_j \\
&\leq 3\text{opt}(I)
\end{aligned}$$

onde a segunda igualdade vale pois para $k_1 < k_2$, vale que $N(j_{k_1}) \cap N(j_{k_2}) = \emptyset$, caso contrário $j_{k_2}$ estaria em $N^2(j_{k_1})$ e, portanto, não estaria em $S_{k_2}$. □

## 3.2 Busca local

Nessa seção falaremos sobre o algoritmo de busca local para o problema de localização de instalações métrico. Esse algoritmo foi estudado na Seção 9.1 do livro WS2011 e foi proposto primeiramente por Kuen e Hamburger [19]. Charikar e Guha [4] provaram a razão de aproximação igual a 3 e introduziram a ideia de reescala, porém a análise a ser apresentada foi feita por Gupta e Tangwongsan [10].

Numa instância $(F, D, c, f)$ do problema de localização de instalações, o custo de transporte está definido apenas para um cliente e uma instalação. Definimos o custo de transporte entre duas instalações como o menor custo de transporte entre essas duas instalações passando por um cliente qualquer. Igualmente, definimos o custo de transporte entre dois clientes como o menor custo de transporte entre esses dois clientes passando por uma instalação qualquer. Então a desigualdade triangular valerá da seguinte forma: para todo $i, j, \ell \in F \cup D$,

$$c_{ij} \leq c_{i\ell} + c_{\ell j}.$$

Um algoritmo de busca local começa com uma solução viável para o problema e checa se alguma alteração local melhora o custo da solução atual. Caso essa melhora ocorra, essa alteração é feita. Esse processo se repete até que não exista alteração local que melhore o custo da solução corrente. A solução resultante é chamada de *localmente ótima*. O tempo de execução de uma implementação dessa ideia e a qualidade da solução obtida dependem da definição adotada de alteração local. Quanto mais abrangente essa definição, melhor a solução, porém em geral mais lento será o algoritmo. Quanto mais restrita a definição, mais rápido o algoritmo, porém pior em geral a qualidade da solução final. Usualmente adotam-se restrições mínimas que garantam a polinomialidade do algoritmo.

O algoritmo que descreveremos contará com três operações: abrir instalações fechadas, fechar instalações abertas ou trocar uma instalação aberta por uma fechada. A solução inicial terá todas as instalações abertas. Para garantir a polinomialidade, uma operação só será feita se diminuir o custo da solução atual em uma razão de $1 - \delta$, para um $\delta > 0$. Como essa solução não é necessariamente localmente ótima, iremos chamá-la de solução *quase localmente ótima*. Ao final, mostraremos que, dado um $\epsilon > 0$, o algoritmo é uma $3(1 + \epsilon)$-aproximação para o problema de localização de instalações métrico, onde o $\delta$ será escolhido em função do $\epsilon$ e o consumo de tempo do algoritmo será afetado pelo valor de $\delta$.

**Algoritmo 6** BuscaLocal$_\epsilon$-KHCGGT$(F, D, c, f)$

$\delta \leftarrow \frac{\epsilon}{(1+\epsilon)2|F|}$
$S' \leftarrow F$
**repita**
    $S \leftarrow S'$
    **Se** existe $i \in S$ tal que custo$(S \setminus \{i\}) \leq (1-\delta)$ custo$(S)$ **então**
        $S' \leftarrow S \setminus \{i\}$
    **Se** existe $i' \in F \setminus S$ tal que custo$(S \cup \{i'\}) \leq (1-\delta)$ custo$(S)$ **então**
        $S' \leftarrow S \cup \{i'\}$
    **Se** existem $i \in S$ e $i' \in F \setminus S$ tal que
                custo$(S \setminus \{i\} \cup \{i'\}) \leq (1-\delta)$ custo$(S)$ **então**
        $S' \leftarrow S \setminus \{i\} \cup \{i'\}$
**até que** $S = S'$
**Devolva** $S$

Vamos mostrar o Lema 3.13 e o Lema 3.15 que serão fundamentais para chegar na razão de aproximação do algoritmo. A partir daqui utilizaremos uma instância $I = (F, D, c, f)$. Seja $S \subseteq F$ solução devolvida pelo algoritmo BuscaLocal$_\epsilon$-KHCGGT$(F, D, c, f)$. Seja $\sigma : D \to S$ com $\sigma(j) \coloneqq \arg\min_{i \in S} c_{ij}$ função de associação dos clientes para a instalação mais próxima em $S$. Sejam $A \coloneqq \sum_{i \in S} f_i$ e $T \coloneqq \sum_{j \in D} \min_{i \in S} c_{ij}$ custos de abertura das instalações e de associação dos clientes na solução $S$, respectivamente. Seja $S^* \subseteq F$ tal que custo$(S^*) = \text{opt}(I)$ uma solução ótima para $I$. Seja $\sigma^* : D \to S^*$ com $\sigma^*(j) \coloneqq \arg\min_{i \in S^*} c_{ij}$ função de associação dos clientes para a instalação mais próxima em $S^*$. Sejam também $A^* \coloneqq \sum_{i \in S^*} f_i$ e $T^* \coloneqq \sum_{j \in D} \min_{i \in S^*} c_{ij}$ custos de abertura das instalações e de associação dos clientes na solução $S^*$, respectivamente. Seja $m \coloneqq |F|$.

**Lema 3.13.** *Sejam $S$ e $S^*$ as soluções como definidas. Então, vale que*

$$T - A^* - T^* \leq m\delta(A + T).$$

*Demonstração.* Para todo $i^* \in S^* \setminus S$, como a solução $S$ é quase localmente ótima, se abrirmos a instalação $i^*$ e associarmos a ela, em $\sigma$, os clientes que estão associados a ela em $\sigma^*$, o custo da solução não diminui em mais que uma fração $1 - \delta$. Então,

$$A + f_{i^*} + T + \sum_{j:\sigma^*(j)=i^*} (c_{i^*j} - c_{\sigma(j)j}) > (1-\delta)(A+T)$$

que equivale a

$$f_{i^*} + \sum_{j:\sigma^*(j)=i^*} (c_{i^*j} - c_{\sigma(j)j}) > -\delta(A+T).$$

Vamos agora mostrar que essa desigualdade também é válida para todo $i^* \in S^* \cap S$. Como os clientes sempre estão ligados a uma instalação aberta mais próxima a eles, ao trocar os clientes que estão associados a $i^*$ em $\sigma^*$ para $i^*$ em $\sigma$, o custo de transporte não pode diminuir. Assim,

$$f_{i^*} + \sum_{j:\sigma^*(j)=i^*} (c_{i^*j} - c_{\sigma(j)j}) \;\geq\; \sum_{j:\sigma^*(j)=i^*} (c_{i^*j} - c_{\sigma(j)j}) \;\geq\; 0 \;\geq\; -\delta(A+T).$$

Note que esses dois casos cobrem todas as instalações presentes em $S^*$. Assim, somando as desigualdades para todas essas instalações, vale que

$$\begin{aligned} m\delta(A+T) &\geq -\sum_{i^* \in S^*} \Big( f_{i^*} + \sum_{j:\sigma^*(j)=i^*} (c_{i^*j} - c_{\sigma(j)j}) \Big) \\ &= -(A^* + \sum_{j \in D} c_{\sigma^*(j)j} - \sum_{j \in D} c_{\sigma(j)j}) \\ &= -(A^* + T^* - T) = T - A^* - T^*. \end{aligned}$$

□

Para o Lema 3.15, precisaremos de uma função e de um lema que ajudará a limitar o custo da redistribuição de clientes. Vamos definir a função $\phi : S^* \to S$ como $\phi(i^*) \coloneqq \arg\min_{i \in S} c_{i^*i}$, ou seja, uma instalação $i$ em $S$ mais próxima à $i^*$ em $S^*$. Então, teremos o seguinte lema.

**Lema 3.14.** *Seja $j$ um cliente tal que $\sigma(j) = i$, $\sigma^*(j) = i^*$, $\phi(i^*) = i'$ e $i \neq i'$. Então,*

$$c_{i'j} - c_{ij} \leq 2c_{i^*j}.$$

*Demonstração.* Pela desigualdade triangular, temos que

$$c_{i'j} \leq c_{i'i^*} + c_{i^*j}.$$

Além disso, pela definição de $i'$, temos que $c_{i'i^*} \leq c_{ii^*}$. Assim,

$$c_{i'j} \leq c_{ii^*} + c_{i^*j}.$$

Novamente, pela desigualdade triangular, vale que $c_{ii^*} \leq c_{ij} + c_{i^*j}$. Então,

$$c_{i'j} \leq c_{ij} + 2c_{i^*j}.$$

Equivalentemente $c_{i'j} - c_{ij} \leq 2c_{i^*j}$. □

**Lema 3.15.** *Sejam $S$ e $S^*$ soluções como já definidas. Então,*

$$A - A^* - 2T^* \leq m\delta(A + T).$$

*Demonstração.* Uma instalação $i \in S$ é *segura* se não existe $i^* \in S^*$ tal que $\phi(i^*) = i$.

Seja $i \in S$ uma instalação segura. Como $S$ é uma solução quase localmente ótima, então se fecharmos $i$ e redistribuirmos cada cliente $j$ que estava associado à $i$ para $\phi(\sigma^*(j))$ não melhoramos o custo da solução em uma razão de $1 - \delta$. Então,

$$A - f_i + T + \sum_{j:\sigma(j)=i} (c_{\phi(\sigma^*(j))j} - c_{\sigma(j)j}) > (1 - \delta)(A + T),$$

o que equivale a

$$-f_i + \sum_{j:\sigma(j)=i} (c_{\phi(\sigma^*(j))j} - c_{\sigma(j)j}) > -\delta(A + T).$$

Como $i$ é uma instalação segura, o Lema 3.14 vale para todos os clientes que estão associados a $i$ em $\sigma$. Então

$$-f_i + \sum_{j:\sigma(j)=i} 2c_{\sigma^*(j)j} > -\delta(A + T). \tag{9}$$

Seja $i \in S$ uma instalação não segura. Definimos $R(i) \coloneqq \{i^* \in S^* : \phi(i^*) = i\}$ como o conjunto de instalações de $S^*$ para as quais $i$ é a instalação mais próxima em $S$. Seja $i' \coloneqq \arg\min_{i^* \in R(i)} c_{i^*i}$ a instalação de $R(i)$ mais próxima à $i$. Para cada $i^* \in R(i) \setminus \{i'\}$, abrir $i^*$ e associar a $i^*$ os clientes que estão associados a $i$ em $\sigma$ e a $i^*$ em $\sigma^*$ não pode melhorar a solução em uma razão de $1 - \delta$. Assim

$$A + f_{i^*} + T + \sum_{j:\sigma(j)=i \text{ e } \sigma^*(j)=i^*} (c_{i^*j} - c_{ij}) > (1 - \delta)(A + T),$$

o que equivale a

$$f_{i^*} + \sum_{j:\sigma(j)=i \text{ e } \sigma^*(j)=i^*} (c_{i^*j} - c_{ij}) > -\delta(A + T). \tag{10}$$

Agora vamos ver o que acontece se abrirmos $i'$, fecharmos $i$ e associarmos cada cliente $j$ associado a $i$ em $\sigma$ para $\phi(\sigma^*(j))$ se $\sigma^*(j) \notin R(i)$ e a $i'$ caso contrário. Disso, deduzimos que

$$-f_i + f_{i'} + \sum_{j:\sigma(j)=i \text{ e } \sigma^*(j)\notin R(i)} (c_{\phi(\sigma^*(j))j} - c_{ij}) + \sum_{j:\sigma(j)=i \text{ e } \sigma^*(j)\in R(i)} (c_{i'j} - c_{ij}) > -\delta(A+T).$$

No somatório dos clientes $j$ em que $\sigma^*(j) \notin R(i)$, podemos utilizar o Lema 3.14 e obtemos

$$-f_i + f_{i'} + \sum_{j:\sigma(j)=i \text{ e } \sigma^*(j)\notin R} 2c_{\sigma^*(j)j} + \sum_{j:\sigma(j)=i \text{ e } \sigma^*(j)\in R(i)} (c_{i'j} - c_{ij}) > -\delta(A+T).$$

Juntando essa desigualdade com (10) para todas as instalações em $R(i) \setminus \{i'\}$, temos

$$\begin{aligned} -f_i + \sum_{i^*\in R(i)} f_{i^*} + \sum_{j:\sigma(j)=i \text{ e } \sigma^*(j)\notin R(i)} 2c_{\sigma^*(j)j} + \sum_{j:\sigma(j)=i \text{ e } \sigma^*(j)\in R(i)} (c_{i'j} - c_{ij}) \\ + \sum_{j:\sigma(j)=i \text{ e } \sigma^*(j)\in R(i)\setminus\{i'\}} (c_{\sigma^*(j)j} - c_{ij}) > -|R(i)|\delta(A+T). \end{aligned}$$

Vamos reduzir essa expressão. Seja $j$ tal que $\sigma(j) = i$ e $\sigma^*(j) \in R(i)$. Se $\sigma^*(j) = i'$, então $j$ só aparece no terceiro somatório e certamente $c_{i'j} - c_{ij} \leq 2c_{i'j}$. Se $\sigma^*(j) \neq i'$, então os termos referentes a $j$ no somatório são $c_{i'j} + c_{\sigma^*(j)j} - 2\,c_{ij}$. Assim, temos que

$$c_{i'j} + c_{\sigma^*(j)j} - 2\,c_{ij} \leq c_{\sigma^*(j)j} + c_{i'i} - c_{ij} \leq c_{\sigma^*(j)j} + c_{\sigma^*(j)i} - c_{ij} \leq 2c_{\sigma^*(j)j}$$

onde a primeira desigualdade vale pois $c_{i'j} \leq c_{i'i} + c_{ij}$, a segunda desigualdade vale pela escolha de $i'$ e a terceira desigualdade vale pois $c_{\sigma^*(j)i} \leq c_{\sigma^*(j)j} + c_{ij}$.

Então, para qualquer $j$ temos que o custo referente a $j$ na soma é no máximo $2c_{\sigma^*(j)j}$. Assim,

$$\sum_{i^*\in R(i)} f_{i^*} - f_i + \sum_{j:\sigma(j)=i} 2c_{\sigma^*(j)j} > -|R(i)|\delta(A+T).$$

Vamos juntar essa última desigualdade para todas as instalações não seguras e (9) para todas as instalações seguras. Chamando de $P$ o conjunto das instalações seguras, temos que

$$\sum_{i\in P}\Big(-f_i + \sum_{j:\sigma(j)=i} 2c_{\sigma^*(j)j}\Big) + \sum_{i\in S\setminus P}\Big(\sum_{i^*\in R(i)} f_{i^*} - f_i + \sum_{j:\sigma(j)=i} 2c_{\sigma^*(j)j}\Big) > -m\delta(A+T).$$

É fácil notar que estamos subtraindo o custo de abertura de todas as instalações de $S$ e somando $2c_{\sigma^*(j)j}$ para todo cliente $j \in D$. Notemos também que estamos somando o custo de abertura de todas as instalações de $S^*$ exatamente uma vez, uma vez que toda instalação de $S^*$ pertence a exatamente um conjunto $R(i)$. Portanto,

$$\begin{aligned} m\delta(A+T) &\geq -(\sum_{i^* \in S^*} f_{i^*} + 2\sum_{j \in D} c_{\sigma^*(j)j} - \sum_{i \in S} f_i) \\ &= -(A^* + 2T^* - A) = A - A^* - 2T^*. \end{aligned}$$

□

Agora que temos essas duas desigualdades, conseguimos mostrar o seguinte.

**Teorema 3.16.** *O algoritmo* BuscaLocal$_\epsilon$-KHCGGT *é uma* $3(1+\epsilon)$*-aproximação para o problema de localização de instalações métrico.*

*Demonstração.* Primeiro, vamos mostrar que o algoritmo executa em tempo polinomial. Claramente, todas as operações podem ser feitas em tempo polinomial, precisamos apenas mostrar que o algoritmo sempre executa um número polinomial de operações.

Uma instância do problema de localização de instalações consiste em inteiros $n$ e $m$ que designam o número de instalações e clientes, uma matriz $C$ com $nm$ elementos e uma matriz $F$ com $n$ elementos. Se $C_{\max}$ é o maior valor absoluto de um elemento de $C$ e $F_{\max}$ é o maior valor absoluto de um elemento de $F$, então o tamanho da instância é $O(mn \log C_{\max} + n \log F_{\max})$. Podemos assumir sem perda de generalidade que todos os custos são inteiros. Seja $M$ o valor da solução inicial, note que $M \leq mC_{\max} + nF_{\max}$. Note que se $\rho$ é um inteiro tal que $(1-\delta)^\rho M < 1$, então o algoritmo não fará mais que $\rho$ iterações. Como $1 + x < e^x$ para todo $x \neq 0$, sabemos que $(1-\delta)^{\frac{1}{\delta}} < \frac{1}{e}$. Quando elevamos tudo por $\ln M$ temos $(1-\delta)^{\frac{1}{\delta}\ln M} < \frac{1}{M}$ e, portanto, vale que $(1-\delta)^{\frac{1}{\delta}\ln M} M < 1$. Então o número de iterações do laço da linha 3 do algoritmo é no máximo

$$\frac{1}{\delta} \ln M \leq \frac{1}{\delta} \ln(mC_{\max} + nF_{\max}).$$

Assim, concluímos que o número de iterações é polinomial no tamanho da instância.

Agora, vamos mostrar a razão de aproximação. Somando as desigualdades encontradas nos Lemas 3.13 e 3.15, temos que

$$A + T - 2A^* - 3T^* \leq 2m\delta(A+T)$$

e, assim, podemos concluir que

$$A + T \leq \frac{2A^* + 3T^*}{1 - 2m\delta} \leq \frac{3}{1 - 2m\delta} \text{ opt}(I) = (1 + \epsilon)3 \text{ opt}(I)$$

em que a última igualdade vale uma vez que $\delta = \frac{\epsilon}{(1+\epsilon)2m}$. □

Durante a prova da razão do algoritmo, utilizamos que $(2A^* + 3T^*) \leq 3\text{opt}(I)$. Vamos utilizar essa folga para melhorar o algoritmo. A partir da instância recebida, vamos criar uma nova instância $(F, D, c, \frac{f}{\mu})$ dividindo o custo de abertura das instalações por uma constante $\mu \leq 1$. Ao final, vamos encontrar um valor para $\mu$ que irá diminuir a razão de aproximação do algoritmo o máximo possível.

Seja $\bar{S}$ a solução devolvida pelo algoritmo BuscaLocal$_\epsilon$-KHCGGT utilizando a instância $(F, D, c, \frac{f}{\mu})$ como entrada. Seja $\bar{A} \coloneqq \sum_{i \in \bar{S}} \frac{f_i}{\mu}$ e $\bar{T} \coloneqq \sum_{j \in D} \min_{i \in \bar{S}} c_{ij}$. Note que existe uma solução para essa nova instância com custos $\frac{A^*}{\mu}$ e $T^*$. Note também que em todos os nossos lemas não utilizamos que a solução comparada era ótima, então os lemas valem também quando nossa solução é comparada com a solução anterior. Assim, vale que

$$\overline{T} - \frac{A^*}{\mu} - T^* \leq m\delta(\bar{A} + \overline{T})$$

e

$$\bar{A} - \frac{A^*}{\mu} - 2T^* \leq m\delta(\bar{A} + \overline{T}).$$

Se multiplicarmos os custos de abertura das instalações escolhidas por $\mu$ temos o custo de uma solução viável para a instância original. Seja $A \coloneqq \mu\bar{A}$ e $T \coloneqq \bar{T}$ os custos dessa solução. Vale

$$T - \frac{A^*}{\mu} - T^* \leq m\delta(\frac{A}{\mu} + T)$$

e

$$\frac{A}{\mu} - \frac{A^*}{\mu} - 2T^* \leq m\delta(\frac{A}{\mu} + T),$$

como $\mu \leq 1$ então $m\delta(\frac{A}{\mu} + T) \leq m\delta\frac{1}{\mu}(A + T)$ e valendo

$$T - \frac{A^*}{\mu} - T^* \leq m\delta\frac{1}{\mu}(A + T)$$

e

$$\frac{A}{\mu} - \frac{A^*}{\mu} - 2T^* \leq m\delta\frac{1}{\mu}(A + T).$$

Somando a primeira desigualdade com a segunda multiplicada por $\mu$ temos

$$A + T - A^*(1 + \frac{1}{\mu}) - T^*(1 + 2\mu) \leq (1 + \frac{1}{\mu})m\delta(A + T),$$

o que é equivalente a

$$(A + T)(1 - (1 + \frac{1}{\mu})m\delta) \leq A^*(1 + \frac{1}{\mu}) + T^*(1 + 2\mu).$$

Note que $(1 + \frac{1}{\mu})$ decresce e $1 + 2\mu$ cresce quando $\mu$ cresce. Então, o menor valor do máximo deles dois será quando eles forem iguais. Igualando eles, encontraremos $\mu = \frac{1}{\sqrt{2}}$ e ambos serão iguais a $1 + \sqrt{2}$. Assim temos

$$(A + T)(1 - (1 + \frac{1}{\mu})m\delta) \leq A^*(1 + \frac{1}{\mu}) + T^*(1 + 2\mu) \leq (1 + \sqrt{2})\text{opt}(I)$$

e, assim,

$$(A + T) \leq \frac{(1 + \sqrt{2})}{(1 - (1 + \sqrt{2})m\delta)}\text{opt}(I).$$

Analogamente ao que foi feito na análise da razão de aproximação $3(1 + \epsilon)$, escolhendo $\delta = \frac{\epsilon}{2m(1+\sqrt{2})}$, deduzimos que $(A + T) \leq (1 + \sqrt{2} + \epsilon)\text{opt}(I)$. Assim, se fizermos a reescala dos custos de abertura das facilidades antes de executar o algoritmo de busca local temos um novo algoritmo que é uma $(1 + \sqrt{2} + \epsilon)$-aproximação para o problema de localização de instalações métrico.

## 3.3 Algoritmo guloso

Nessa seção vamos apresentar o algoritmo guloso que é equivalente a uma 2-aproximação que utiliza o método de *dual fitting*. Esse algoritmo foi estudado pela Seção 9.4 do livro WS2011 e foi desenvolvido por Jain, Mahdian, Markakis, Saberi e Vazirani [14].

O algoritmo guloso para o problema da localização de instalações consiste em, a cada iteração, abrir uma instalação fechada e associá-la a um conjunto de clientes ainda não associados, garantindo um baixo aumento no custo total. Isso se repete até que todos os clientes estejam associados a uma instalação aberta. Então, seja $X$ o conjunto de instalações abertas até o momento e $S$ o conjunto de clientes ainda não associados a uma instalação em $X$. Queremos escolher $i \in F \setminus X$ e $\emptyset \neq Y \subseteq S$ que minimize

$$\frac{f_i + \sum_{j \in Y} c_{ij}}{|Y|}.$$

Para contemplar a possibilidade que, em alguma iteração do algoritmo, a melhor escolha seja associar mais alguns clientes a alguma instalação já aberta, no momento em que uma instalação é aberta, o seu custo de abertura é alterado para 0 e permite-se que ela seja escolhida novamente no futuro. Também há um ajuste no critério de escolha que permite que clientes mudem sua escolha de instalação aberta à medida que mais instalações são abertas. Desse modo, conseguimos melhorar ainda mais o custo da solução produzida por esse algoritmo. Defina $(a)_+ \coloneqq \max\{0, a\}$. Assim, teremos o seguinte algoritmo guloso para o problema da localização de instalações.

**Algoritmo 7** GULOSO-JMMSV$(F, D, c, f)$

1: $S \leftarrow D$
2: $X \leftarrow \emptyset$
3: **Enquanto** $S \neq \emptyset$ **faça**
4: Escolha $i \in F$ e $\emptyset \neq Y \subseteq S$ que minimize $\frac{f_i + \sum_{j \in Y} c_{ij} - \sum_{j \notin C} (c(j,X) - c_{ij})_+}{|Y|}$
5: $f_i \leftarrow 0$
6: $S \leftarrow S \setminus Y$
7: $X \leftarrow X \cup \{i\}$
8: **Devolva** $X$

Para a análise do algoritmo guloso, iremos apresentar um algoritmo que utiliza o método dual fitting, mostrar uma razão de aproximação para ele e mostrar que eles são equivalentes.

Relembre os programas lineares (PL) e (PD). Ao longo do algoritmo vamos construir variáveis $\alpha_j$ semelhantes às variáveis $v_j$ do programa (PD). Denote por $N(i) \coloneqq \{j \in d : \alpha_j \geq c_{ij}\}$.

A intuição do algoritmo DUALFITTING-JMMSV é parecida com a ideia do algoritmo PRIMALDUAL-JV da Seção 3.1.1. As variáveis $\alpha_j$ tem interpretações semelhantes às variáveis $v_j$ do programa linear (PD). A linha 6 do algoritmo está relacionada com a restrição (D3) e a linha 7 está relacionada com a restrição (D2). Assim como no algoritmo primal dual, os clientes também vão contribuir para a abertura de uma instalação fechada. Um cliente $j$ que está em $S$ contribui $(\alpha_j - c_{ij})_+$ para a abertura de $i$, enquanto um cliente que não está em $S$ contribui $(c(j, X) - c_{ij})_+$ para a abertura de $i$. A ideia é que o cliente consegue contribuir mais para uma instalação caso diminua o seu custo de conexão.

Note que os valores de $\alpha$ crescem de maneira uniforme para os clientes que estão em $S$. Assim, para quaisquer clientes $\ell$ e $j$ tais que $\alpha_\ell < \alpha_j$, vale que $\ell$ foi removido do conjunto $S$ em uma iteração anterior àquela em que $j$ foi removido de $S$.

**Algoritmo 8** DualFitting-JMMSV$(F, D, c, f)$

$\alpha_j \leftarrow 0$ para todo $j \in D$
$S \leftarrow D$
$X \leftarrow \emptyset$
$f' \leftarrow 2f$
**Enquanto** $S \neq \emptyset$ **faça**
    $\theta_1 \leftarrow \min\{c_{ij} - \alpha_j : j \in S, i \in X\}$
    $\theta_2 \leftarrow \min\{(f'_i - \sum_{j \in S}(\alpha_j - c_{ij})_+ - \sum_{j \notin S}(c(j, X) - c_{ij})_+)/|S| : i \in F \setminus X\}$
    $\theta \leftarrow \min\{\theta_1, \theta_2\}$
    $\alpha_j \leftarrow \alpha_j + \theta$ para todo $j \in S$
    **Se** $\alpha_j = c_{ij}$ para algum $j \in S$ e $i \in X$ **então**
        $S \leftarrow S \setminus \{j\}$
    **Se** $\sum_{j \in S}(\alpha_j - c_{ij})_+ + \sum_{j \notin S}(c(j, X) - c_{ij})_+ = f'_i$ para algum $i \in F \setminus X$ **então**
        $S \leftarrow S \setminus N(i)$
        $X \leftarrow X \cup \{i\}$
**Devolva** $X$

Vamos mostrar que, ao final do algoritmo, custo$(X) \leq \sum_{j \in D} \alpha_j$ e que conseguimos obter variáveis $w$ tais que $(\frac{\alpha}{2}, w)$ é solução viável para (PD). Assim, deduzimos que o algoritmo é uma 2-aproximação.

Apresentaremos dois lemas principais, os Lemas 3.20 e 3.21, donde segue a razão de aproximação do algoritmo DualFitting-JMMSV. Para o primeiro desses lemas, iremos, primeiramente, provar outros três lemas.

**Lema 3.17.** *Seja $k$ a iteração em que o cliente $j$ é removido de $S$ e seja $\ell$ um cliente tal que $\alpha_\ell \leq \alpha_j$. Então, a oferta do cliente $\ell$ para uma facilidade $i$ no início da iteração $k$ é pelo menos $\alpha_j - c_{ij} - 2c_{i\ell}$.*

*Demonstração.* Se $\alpha_\ell = \alpha_j$, então $\ell$ é removido de $S$ na iteração $k$. Portanto, no começo da iteração $k$, $\ell$ contribui $(\alpha_\ell - c_{i\ell})_+ = (\alpha_j - c_{i\ell})_+ \geq \alpha_j - c_{ij} - 2c_{i\ell}$ para a abertura de $i$, onde a desigualdade vale pois $c_{ij} \geq 0$ e $c_{i\ell} \geq 0$.

Se $\alpha_\ell < \alpha_j$, então $\ell$ foi removido de $S$ antes da iteração $k$. Seja $h$ a instalação de $X$ que minimiza $c_{h\ell}$. Então, $\ell$ contribui $(c_{h\ell} - c_{i\ell})_+$ para a abertura de $i$ nesse momento. Pela desigualdade triangular, $c_{hj} \leq c_{h\ell} + c_{i\ell} + c_{ij}$. Note que $\alpha_j \leq c_{hj}$, caso contrário $j$ estaria na vizinhança de uma instalação de $X$ antes da iteração $k$ e, portanto, teria sido removido de $S$. Então $\alpha_j \leq c_{h\ell} + c_{i\ell} + c_{ij}$. Portanto,

$$(c_{h\ell} - c_{i\ell})_+ \geq c_{h\ell} - c_{i\ell} \geq \alpha_j - c_{ij} - 2c_{i\ell}.$$

□

**Lema 3.18.** *Seja $A \subseteq D$ um conjunto qualquer de clientes. Podemos assumir que $A = \{1, \ldots, |A|\}$ onde $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_{|A|}$. Então, para $i \in F$ e $j \in A$, vale que*

$$\sum_{\ell=1}^{j-1}(\alpha_j - c_{ij} - 2c_{i\ell}) + \sum_{\ell=j}^{|A|}(\alpha_j - c_{i\ell}) \leq f'_i.$$

*Demonstração.* Sabemos que as contribuições recebidas por $i$ para a sua abertura sempre serão no máximo $f'_i$. Assim, é suficiente mostrar que o lado esquerdo da desigualdade é no máximo a soma das contribuições recebidas por $i$ em alguma iteração. Seja $k$ a iteração em que $j$ se conecta a uma instalação pela primeira vez. Pelo Lema 3.17, vale que a oferta recebida por $i$ na iteração $k$ por cada cliente $\ell$ tal que $\alpha_\ell \leq \alpha_j$ é no máximo $\alpha_j - c_{ij} - 2c_{i\ell}$. Sabemos que um cliente $\ell$ tal que $\alpha_\ell \geq \alpha_j$, no início da iteração $k$, ainda não está associado a uma instalação e, portanto, oferta a $i$ exatamente $(\alpha_j - c_{i\ell})_+$ que é pelo menos $\alpha_j - c_{i\ell}$. Portanto, $\sum_{\ell=1}^{j-1}(\alpha_j - c_{ij} - 2c_{i\ell}) + \sum_{\ell=j}^{|A|}(\alpha_j - c_{i\ell}) \leq f'_i$. □

**Lema 3.19.** *Seja $A \subseteq D$ um conjunto qualquer de clientes. Podemos assumir que $A = \{1, \ldots, |A|\}$ onde $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_{|A|}$. Então, para $i \in F$, vale que*

$$\sum_{j \in A} \alpha_j - 2c_{ij} \leq f'_i.$$

*Demonstração.* Seja $p \coloneqq |A|$. Usando o Lema 3.18 para todo $j \in A$, temos que

$$\begin{aligned}
pf'_i &\geq \sum_{j=1}^{p}\Big(\sum_{k=1}^{j-1}(\alpha_j - c_{ij} - 2c_{ik}) + \sum_{k=j}^{p}(\alpha_j - c_{ik})\Big) \\
&= p\sum_{j\in A}\alpha_j - \sum_{k=1}^{p}(k-1)c_{ik} - p\sum_{k=1}^{p}c_{ik} - \sum_{k=1}^{p}(p-k)c_{ik} \\
&= p\sum_{j\in A}\alpha_j - \sum_{k=1}^{p}(k-1+p+p-k)c_{ik} \\
&= p\sum_{j\in A}\alpha_j - (2p-1)\sum_{k=1}^{p}c_{ik} \\
&\geq p\sum_{j\in A}\alpha_j - 2p\sum_{k=1}^{p}c_{ik} = p\sum_{j\in A}(\alpha_j - 2c_{ij}).
\end{aligned}$$

Então, temos que $\sum_{j\in A}\alpha_j - 2c_{ij} \leq f'_i$. □

Assim, temos os lemas necessários para provar o primeiro dos dois lemas fundamen-

tais para a prova da razão de aproximação do algoritmo DualFitting-JMMSV.

**Lema 3.20.** *Seja $\alpha$ o vetor com as variáveis produzidas pelo algoritmo* DualFitting-JMMSV*, seja também $v_j \coloneqq \alpha_j/2$ e $w_{ij} \coloneqq (v_j - c_{ij})_+$ para todos os clientes $j$ e instalações $i$. Então $(v, w)$ é solução viável para* (PD)*.*

*Demonstração.* É evidente que $v_j \geq 0$ para todo $j \in D$ e que $w_{ij} \geq 0$ para todo $i \in F$ e $j \in D$. É evidente também que $v_{ij} - w_{ij} \leq c_{ij}$ para todo $i \in F$ e $j \in D$. Pelo Lema 3.19, para uma instalação $i \in F$ e $A \coloneqq \{j \in D : w_{ij} > 0\}$, temos que

$$2f_i = f'_i \geq \sum_{j \in A} (\alpha_j - 2c_{ij}) = \sum_{j \in A} (2v_j - 2c_{ij}) = 2 \sum_{j \in A} w_{ij}.$$

Assim, temos que $\sum_{j \in D} w_{ij} \leq f_i$. Portanto, $(v, w)$ é solução viável para (PD). □

Agora, o último lema que será necessário para mostrar a razão de aproximação do algoritmo DualFitting-JMMSV.

**Lema 3.21.** *Seja $\alpha$ a variável produzida e $X$ o conjunto de clientes escolhido pelo algoritmo* DualFitting-JMMSV*. Vale que*

$$\sum_{j \in D} \alpha_j = 2 \sum_{i \in X} f_i + \sum_{j \in D} c(j, X).$$

*Demonstração.* Seja $S_k$ e $X_k$ os conjuntos $S$ e $X$ no início da iteração $k$ do algoritmo DualFitting-JMMSV, respectivamente. Vamos provar que, para qualquer $k$, vale que $\sum_{j \in D \setminus S_k} \alpha_j = 2 \sum_{i \in X_k} f_i + \sum_{j \in D \setminus S_k} c(j, X)$. Como no final do algoritmo $S = \emptyset$, então $D \setminus S = D$ e o que vamos provar implica o lema.

Suponha, por absurdo, que a afirmação é falsa. Seja $k$ a primeira iteração em que a afirmação não vale. Se $k = 1$, então no começo da iteração $k$ vale que $D \setminus C_k = \emptyset$. Portanto, a afirmação vale. Então $k > 1$.

Se, na iteração $k - 1$, vale que $\alpha_j = c_{ij}$ para algum $j \in S_{k-1}$ e $i \in X_{k-1}$, então $S_k = S_{k-1} \setminus \{j\}$ e $X_k = X_{k-1}$. Portanto

$$\begin{aligned} \sum_{\ell \in D \setminus S_k} \alpha_\ell = \alpha_j + \sum_{\ell \in D \setminus S_{k-1}} \alpha_\ell &= \alpha_j + 2 \sum_{i \in X_{k-1}} f_i + \sum_{\ell \in D \setminus S_{k-1}} c(\ell, X_{k-1}) \\ &= 2 \sum_{i \in X_k} f_i + \sum_{\ell \in D \setminus S_k} c(\ell, X_k), \end{aligned}$$

em que a segunda igualdade vale, pois a afirmação vale para $S_{k-1}$ e a terceira vale pois $\alpha_j = c_{ij} = c(j, X)$, caso contrário $j$ não estaria em $S_{k-1}$.

Se, na iteração $k-1$, vale que $\sum_{j\in S_{k-1}}(\alpha_j - c_{ij})_+ + \sum_{j\notin S_{k-1}}(c(j,X) - c_{ij})_+ = f'_i$ para algum $i \in F \setminus X_{k-1}$, então $X_k = X_{k-1}\cup\{i\}$ e $S_k = S_{k-1}\setminus N(i)$. Consequentemente

$$\begin{aligned}\sum_{j\in D\setminus S_k}\alpha_j &= \sum_{j\in D\setminus S_{k-1}}\alpha_j + \sum_{j\in S_{k-1}\cap N(i)}\alpha_j \\ &= \sum_{j\in S_{k-1}\cap N(i)}\alpha_j + \sum_{j\in D\setminus S_{k-1}} c(j,X_{k-1}) + 2\sum_{h\in X_{k-1}} f_h.\end{aligned}$$

Defina $A \coloneqq \{j \in D\setminus S_{k-1} : c(j,X_{k-1}) \geq c_{ij}\}$. Vale que

$$\sum_{j\in S_{k-1}\cap N(i)}(\alpha_j - c_{ij}) + \sum_{j\in A}(c(j,X_{k-1}) - c_{ij}) = f'_i = 2f_i$$

e, portanto,

$$\sum_{j\in S_{k-1}\cap N(i)}\alpha_j = \sum_{j\in S_{k-1}\cap N(i)} c_{ij} - \sum_{j\in A}(c(j,X_{k-1}) - c_{ij}) + 2f_i.$$

Seja $B$ tal que $A\cap B = \emptyset$ e $A\cup B = D\setminus S_{k-1}$. Portanto,

$$\begin{aligned}\sum_{j\in D\setminus S_k}\alpha_j &= 2\sum_{h\in X_k} f_h + \sum_{j\in B} c(j,X_{k-1}) + \sum_{j\in A} c_{ij} + \sum_{j\in S_{k-1}\cap N(i)} c_{ij} \\ &= 2\sum_{h\in X_k} f_h + \sum_{j\in D\setminus S_k} c(j,X_k).\end{aligned}$$

Logo, todos os casos caem em uma contradição e a afirmação é verdadeira. □

Agora, podemos provar a razão de aproximação do algoritmo DualFitting-JMMSV.

**Teorema 3.22.** *O algoritmo* DualFitting_JMMSV *é uma 2-aproximação para o problema de localização de instalações métrico.*

*Demonstração.* Seja $X$ o conjunto de instalações devolvido pelo algoritmo DualFitting-JMMSV. O custo da solução em que abrimos as instalações em $X$ e conectamos cada cliente à instalação aberta mais próxima a ele é

$$\sum_{i\in X} f_i + \sum_{j\in D} c(j,X) \leq 2\sum_{i\in X} f_i + \sum_{j\in D} c(j,X) = \sum_{j\in D}\alpha_j = 2\sum_{j\in D}\frac{\alpha_j}{2} \leq 2\,\mathrm{opt}(I)$$

em que a primeira igualdade vale pelo Lema 3.21 e a última desigualdade vale pois

$\sum_{j \in D} \alpha_j/2$ é o valor objetivo da solução viável do dual construída como no Lema 3.20 e, portanto, a desigualdade vale pela dualidade fraca. □

Agora, vamos mostrar que o algoritmo Guloso-JMMSV e o algoritmo DualFitting-JMMSV são equivalentes. Para facilitar a prova, vamos supor que no algoritmo guloso o desempate é feito escolhendo o conjunto com menor tamanho e que no algoritmo de dual fitting quando a condicional da linha 10 é verdadeira apenas um cliente é retirado de $S$.

**Teorema 3.23.** *Os algoritmos* DualFitting_JMMSV *e* Guloso_JMMSV *são equivalentes.*

*Demonstração.* Vamos chamar de $(S_k^1, X_k^1)$ e $(S_k^2, X_k^2)$ os pares de conjuntos $S$ e $X$ no começo da iteração $k$ no algoritmo guloso e no algoritmo de dual fitting, respectivamente. Para mostrar que os algoritmos são equivalentes, é suficiente mostrar que $(S_k^1, X_k^2) = (S_k^2, X_k^2)$ para todo $k$.

Suponha, por absurdo, que a afirmação é falsa para algum $k$. Seja $k$ a primeira iteração tal que a afirmação não vale. Se $k = 1$, então $(S_k^1, X_k^1) = (D, \emptyset) = (S_k^2, X_k^2)$. Então, $k > 1$. Caso, na iteração $k - 1$, o algoritmo de dual fitting escolha abrir uma instalação $i$. Então, vale que

$$\sum_{j \in S_{k-1}^2} (\alpha_j - c_{ij})_+ + \sum_{j \notin S_{k-1}^2} (c(j, X_{k-1}^2) - c_{ij})_+ = 2f_i$$

e, portanto,

$$\sum_{j \in S_{k-1}^2 \cap N(i)} \alpha_j = 2f_i - \sum_{j \notin S_{k-1}^2} (c(j, X) - c_{ij})_+ + \sum_{j \in S_{k-1}^2 \cap N(i)} c_{ij}.$$

Note que, por construção, todos os clientes de $S_{k-1}^2 \cap N(i)$ têm o mesmo valor de $\alpha$ neste momento. Seja $j \in S_{k-1}^2 \cap N(i)$, vale que

$$|S_{k-1}^2 \cap N(i)|\alpha_j = \sum_{j \in S_{k-1}^2 \cap N(i)} \alpha_j = 2f_i - \sum_{j \notin S_{k-1}^2} (c(j, X) - c_{ij})_+ + \sum_{j \in S_{k-1}^2 \cap N(i)} c_{ij}.$$

Logo,

$$\alpha_j = \frac{2f_i - \sum_{j \notin S_{k-1}^2} (c(j, X) - c_{ij})_+ + \sum_{j \in S_{k-1}^2 \cap N(i)} c_{ij}}{|S_{k-1}^2 \cap N(i)|}.$$

Como as variáveis em $\alpha$ crescem uniformemente e como $(S_{k-1}^1, X_{k-1}^1) = (S_{k-1}^2, X_{k-1}^2)$, é fácil notar que o par $(i, S_{k-1}^2 \cap N(i))$ minimiza a função de escolha do algoritmo

guloso e, portanto, $(S_k^1, X_k^1) = (S_k^2, X_k^2)$. Então, na iteração $k-1$, o algoritmo de dual fitting escolhe apenas retirar um elemento de $S_{k-1}^2$. Seja $j$ o elemento que foi retirado e $i \in X_{k-1}^2$ a instalação tal que $\alpha_j = c_{ij}$. Como $i \in X_{k-1}^2$ e $X_{k-1}^1 = X_{k-1}^2$, então na iteração $k-1$ do algoritmo guloso, vale que $f_i = 0$. Note que a função de escolha do guloso aplicada ao par $(i, \{j\})$ tem valor $c_{ij}$, uma vez que a instalação $i$ já estava aberta no início da iteração $k-1$ e, portanto, não haverá melhora no custo de conexão dos clientes que já estavam ligados a alguma instalação aberta. Novamente, como as variáveis em $\alpha$ crescem uniformemente e como $(S_{k-1}^1, X_{k-1}^1) = (S_{k-1}^2, X_{k-1}^2)$, é fácil notar que o par $(i, \{j\})$ minimiza a função de escolha do algoritmo guloso e, portanto, $(S_k^1, X_k^1) = (S_k^2, X_k^2)$.

Todos os casos nos levam a uma contradição à escolha de $k$, então a afirmação é verdadeira. □

## 3.4 Inaproximabilidade

Nesta seção vamos mostrar um resultado que, sob certa hipótese, limita inferiormente a razão de aproximação para os algoritmos do problema de localização de instalações métrico. Esse resultado é o Corolário 16.16 que se encontra na Seção 16.2 do livro WS2011. Ele foi provado por Guha e Kuller [9].

Para esse resultado vamos precisar definir a versão de otimização do problema de cobertura de conjuntos.

**Problema 3.24.** *Dada um conjunto $E$ e uma família $\mathcal{S} = \{S_1, \ldots, S_m\}$ de subconjuntos de $E$, encontrar $C \subseteq \mathcal{S}$ tal que $\bigcup_{S \in C} S = E$ que minimize $|C|$.*

Esse problema é *NP*-difícil, uma vez que sua versão de decisão é um dos 21 problemas *NP*-completos de Karp [17]. Além disso, Feige [6] provou que não existe $(1-\epsilon)$-aproximação para esse problema, a menos que, para todo problema em $NP$, exista um algoritmo determinístico que resolva o problema em tempo $n^{O(\log \log n)}$. Assim como $P = NP$ essa também é uma conjectura conhecida na área de complexidade computacional que acreditam ser falsa.

Vamos mostrar que se existe uma $\alpha$-aproximação para o problema de localização de instalações com $\alpha < 1.463$ então existe um algoritmo que é uma $(c \ln n)$-aproximação para o problema da cobertura de conjuntos sem peso com $c < 1$.

Vamos apresentar um algoritmo de aproximação para o problema da cobertura de conjuntos que utiliza várias chamadas ao algoritmo que é uma $\alpha$-aproximação para o problema de localização de instalações métrico.

**Algoritmo 9** INAPROX-GK$(E, \{S_1, \ldots, S_m\})$

```
1: γ ← 0.463
2: Para i ← 1 até m faça
3:     Para todo j ∈ E faça
4:         Se j ∈ S_i então
5:             c_ij ← 1
6:         Senão
7:             c_ij ← 3
8: Para k ← 1 até m faça
9:     I_k ← ∅
10:    D ← E;    F ← [m]
11:    Enquanto D ≠ ∅ faça
12:        f_i ← γ|D|/k para todo i ∈ F
13:        F' ← α-APROX-LOC-INST(F, D, c, f)
14:        I_k ← I_k ∪ F'
15:        D' ← {j ∈ D : c(j, F') = 1}
16:        F ← F \ F';   D ← D \ D'
17: Devolva I_k que minimize |I_k|.
```

Seja $I = (E, \{S_1, \ldots, S_m\})$ uma instância do problema de cobertura de conjuntos. Criaremos uma sequência de instâncias para o problema de localização de instalações métrico. Para um $k$ fixo, o algoritmo cria uma sequência de instâncias métricas para o problema de localização de instalações. Todas elas terão a mesma função de custo $c$ tal que, para todo $i \in [m]$ e $j \in E$, $c_{ij} = 1$ se $j \in S_i$ e $c_{ij} = 3$, caso contrário. É fácil notar que essa função é métrica. A primeira instância da sequência terá o conjunto de instalações $F \coloneqq [m]$ e o conjunto de clientes $D \coloneqq E$. Além disso, toda instalação terá custo de abertura $f_i \coloneqq \gamma|D|/k$. A ideia é que usamos o algoritmo $\alpha$-APROX-LOC-INST para encontrar um conjunto $F' \subseteq F$ de instalações que serão abertas e existirá um conjunto $D' \subseteq D$ tal que, para todo $j \in D'$ existe $i \in F'$ tal que $j \in S_i$. Então, para construir a próxima instância, usaremos o conjunto de instalações $F \setminus F'$ e o conjunto de clientes $D \setminus D'$, que representam os subconjuntos de $E$ que ainda podem ser escolhidos e os elementos de $E$ que ainda não foram cobertos, respectivamente. Isso acontecerá sucessivamente até que todos os elementos de $E$ sejam cobertos, ou seja, até que $D = \emptyset$.

**Teorema 3.25.** *Se existe uma $\alpha$-aproximação para o problema da localização de instalações com $\alpha < 1.463$, então* INAPROX-GK *é uma $(c \ln n)$-aproximação para o problema de cobertura de conjuntos com $c < 1$ quando $n$ é suficiente grande.*

*Demonstração.* Seja $k^*$ o tamanho de uma cobertura de conjuntos ótima para a

instância $(E, \{S_1, \ldots, S_m\})$ do problema de cobertura de conjuntos. Suponha que estamos na iteração $k^*$ do laço **Para** e na iteração $\ell$ do laço **Enquanto** do algoritmo INAPROX-GK. Sejam $D_\ell$ e $F_\ell$ os conjuntos $D$ e $F$ no começo dessa iteração do laço **Enquanto**. Sejam $n_\ell \coloneqq |D_\ell|$ e $f^\ell \coloneqq \gamma n_\ell / k^*$. Note que existe uma solução para a instância $(D_\ell, F_\ell, c, f^\ell)$ do problema de localização de instalações com custo no máximo $f^\ell k^* + n_\ell$ em que abrimos todas as instalações que representam subconjuntos que estão em uma cobertura de conjuntos ótima. Seja $F'_\ell$ o conjunto das instalações devolvido pela chamada de $\alpha$-APROX-LOC-INST na linha 13 e $D'_\ell$ o conjunto de clientes encontrado na linha 15. Sejam $\beta_\ell \coloneqq |F'_\ell|/k^*$ e $p_\ell \coloneqq |D'_\ell|/n_\ell$. É fácil notar que o custo da solução $F'_\ell$ é

$$|F'_\ell| f^\ell + p_\ell n_\ell + 3(1-p_\ell)n_\ell = n_\ell(\beta_\ell \gamma + 3 - 2p_\ell) \leq \alpha(f^\ell k^* + n_\ell) = \alpha n_\ell(\gamma + 1)$$

e, portanto,

$$\frac{\beta_\ell + 3 - 2p_\ell}{\gamma + 1} \leq \alpha. \tag{16}$$

Seja $c$ uma constante entre 0 e 1 que vamos escolher depois. Suponha que $p_\ell \leq 1 - e^{-\frac{\beta_\ell}{c}}$ para algum $\ell$ e $0 < \gamma < 1$. Defina

$$f(\beta_\ell) \coloneqq \frac{\beta_\ell \gamma + 1 + 2e^{-\frac{\beta_\ell}{c}}}{\gamma + 1}.$$

Note que $f(\beta_\ell) \leq \frac{\beta_\ell + 3 - 2p_\ell}{1+\gamma} \leq \alpha$.

O valor de $\beta_\ell$ que minimiza $f$ é $c\ln(\frac{2}{\gamma c})$, uma vez que $f'(c\ln(\frac{2}{\gamma c})) = 0$ e $f''(\beta_\ell) > 0$ qualquer que seja o valor de $\beta_\ell$. Assim, temos que

$$f(c\ln\left(\frac{2}{\gamma c}\right)) = \frac{1}{1+\gamma}\left(\gamma c + \ln\left(\frac{2}{\gamma c}\right) + \gamma c + 1\right).$$

Os valores de $c$ e $\gamma$ que maximizam essa função são $\gamma = 0.463$ e $c$ próximo de 1, e teremos $1.463 \leq f(1.463) \leq \alpha < 1.463$, o que é um absurdo.

Agora, vamos mostrar que, se $p_\ell > 1 - e^{-\frac{\beta_\ell}{c}}$ para todo $\ell$, então o algoritmo INAPROX-GK com $\gamma = 0.463$ devolve uma $(c' \ln n)$-aproximação para o problema da cobertura de conjuntos com $c' < 1$. Vamos chamar de $r$ a quantidade de iterações que o laço **Enquanto** realiza. É evidente que $|I_{k^*}| = k^* \sum_{\ell=1}^{r} \beta_\ell$, logo a razão de aproximação desse algoritmo é no máximo $\sum_{\ell=1}^{r} \beta_\ell$. Para todo $\ell$, como vale que $p_\ell > 1 - e^{-\frac{\beta_\ell}{c}}$, então temos que $\beta_\ell < c\ln\left(\frac{1}{1-p_\ell}\right)$. Note que $n_{\ell+1} = n_\ell(1-p_\ell)$ e $n_r \geq 1$. Assim, vale que

$n_1 \prod_{\ell=1}^{r-1}(1-p_\ell) = n_r \geq 1$ e, consequentemente, $\ln\left(\prod_{\ell=1}^{r-1} \frac{1}{1-p_\ell}\right) \leq \ln n_1 = \ln n$. Assim,

$$\sum_{\ell=1}^{r} \beta_\ell = \sum_{\ell=1}^{r-1} \beta_\ell + \beta_r < c \sum_{\ell=1}^{r-1} \ln\left(\frac{1}{1-p\ell}\right) + \beta_r \leq c \ln n + \beta_r.$$

Como $r$ é a última iteração, vale que $p_r = 1$. Então pela desigualdade (16) temos que $\beta_r \leq \alpha(1 + \frac{1}{\gamma}) < 4\alpha$ uma vez que escolhemos $\gamma = 0.463$. Portanto,

$$\sum_{\ell=1}^{r} \beta_\ell < c \ln n + \beta_r < c \ln n + 4\alpha \leq c' \ln n,$$

em que a última desigualdade vale para algum $c' \in (c, 1)$, uma vez que $4\alpha$ é constante e, pelo enunciado do teorema, podemos assumir que $n$ é suficientemente grande.

Assim, temos uma aproximação para o problema de cobertura de conjuntos com razão de aproximação estritamente menor que $\ln n$ para $n$ suficientemente grande. □

Juntando esse resultado com o resultado do Feige, temos que a existência de uma $\alpha$-aproximação para o problema de localização de instalações métrico com $\alpha < 1.463$ implica a existência de um algoritmo determinístico que leva tempo $O(n^{O(\log \log n)})$ para todo problema *NP*-completo.

# 4 $k$-Medianas

Nessa seção falaremos sobre a versão métrica do problema das $k$-medianas. O problema das $k$-medianas métrico consiste em, dado um conjunto de instalações $F$, um conjunto de clientes $D$, uma métrica $c : F \times D \to \mathbb{R}$ e um inteiro $k$, encontrar $S \subseteq F$ com $|S| \leq k$ que minimize custo$(S) \coloneqq \sum_{j \in D} c(j, S)$, em que $c(j, S) \coloneqq \min_{i \in S} c_{ij}$.

Antes de falarmos sobre algoritmos de aproximação para o problema das $k$-medianas métrico, vamos mostrar que, assumindo que $P \neq NP$, não existe algoritmo polinomial que resolva esse problema. Vamos fazer uma redução do problema da cobertura de conjuntos na sua versão de decisão para o problema das $k$-medianas métrico.

**Problema 4.1.** *Dada um conjunto $E$, uma família $\mathcal{S} = \{S_1, \ldots, S_m\}$ de subconjuntos de $E$ e um inteiro $k$, decidir se existe um conjunto $C \subseteq \mathcal{S}$ tal que $\bigcup_{S \in C} S = E$ e $|C| \leq k$.*

Esse problema é *NP*-completo, sendo um dos famosos 21 problemas de Karp [17]. Disso, deriva-se o seguinte.

**Teorema 4.2.** *O problema das k-medianas métrico é NP-difícil.*

*Demonstração.* Seja $I = (E, \{S_1, \ldots, S_m\}, k)$ uma instância da versão de decisão do problema da cobertura de conjuntos. Vamos definir o conjunto de instalações $F \coloneqq [m]$, o conjunto de clientes $D \coloneqq E$ e a função de custo $c_{ij} \coloneqq 1$ se $j \in S_i$ e $c_{ij} \coloneqq 3$ caso contrário, para cada $i \in F$ e $j \in D$. Assim, temos uma instância $I' = (F, D, c, k)$ para o problema das $k$-medianas. É fácil notar que essa corresponde a uma instância métrica do problema. Então, vamos mostrar que a resposta para $I$ é sim se e somente se $\text{opt}(I') = |D|$. Antes, note que $\text{opt}(I') \geq |D|$, uma vez que $c_{ij} \geq 1$ para todo $i \in F$ e $j \in D$.

Vamos mostrar que se a resposta para $I$ é sim, então $\text{opt}(I') = |D|$. Como a resposta para $I$ é sim, então existem $k$ elementos de $\{S_1, \ldots, S_m\}$ tal que a união deles é igual a $E$. Vamos supor sem perda de generalidade que esses elementos são os $k$ primeiros. Seja $F' \coloneqq [k]$. A solução de $I'$ em que abrimos as instalações em $F'$ tem custo $|D|$, uma vez que para cada cliente $j$ temos que $j \in E = \bigcup_{i=1}^{k} S_i$. Então existe $i \in [k]$ tal que $j \in S_i$ e, consequentemente, $c_{ij} = 1$. Portanto, $\text{opt}(I') \leq |D|$. Como já sabemos que $\text{opt}(I') \geq |D|$, então $\text{opt}(I') = |D|$.

Agora vamos mostrar que se $\text{opt}(I') = |D|$, então a resposta para $I$ é sim. Como $\text{opt}(I') = |D|$, então existe um conjunto $F' \subseteq F$ de tamanho $k$ tal que para todo cliente $j$ existe uma instalação $i \in F'$ tal que $c_{ij} = 1$ e, consequentemente, $j \in S_i$. Como isso vale para todo cliente $j \in D$, então a união dos conjuntos referentes às instalações em $F'$ é uma cobertura de conjuntos de $E$ com tamanho $k$. Portanto, a resposta para $I$ é sim. □

## 4.1 Busca local

Nessa seção falaremos sobre o algoritmo de busca local com trocas unitárias para o problema das $k$-medianas. Esse algoritmo foi estudado na Seção 9.2 do livro WS2011 e foi desenvolvido por Arya, Garg, Khandekar, Meyerson, Munagala e Pandit [1].

Numa instância do problema das $k$-medianas o custo de transporte está definido apenas para um cliente e uma instalação. Definimos o custo de transporte entre duas instalações como o menor custo de transporte entre essas duas instalações passando por um cliente qualquer. Analogamente, definimos o custo de transporte entre dois clientes como o menor custo de transporte entre esses dois clientes passando por uma instalação qualquer. Então a desigualdade triangular valerá da seguinte forma: para todo $i, j, k \in F \cup D$,

$$c_{ij} \leq c_{ik} + c_{kj}.$$

Um algoritmo de busca local começa com uma solução viável para o problema e checa se alguma alteração local melhora o custo da solução atual. Caso essa melhora ocorra, essa alteração é feita. Esse processo se repete até que não exista alteração que melhore o custo da solução corrente. A solução resultante é chamada de *localmente ótima*. O tempo de execução de uma implementação dessa ideia e a qualidade da solução obtida dependem da definição adotada de alteração local. Quanto mais abrangente essa definição, melhor a solução, porém em geral mais lento será o algoritmo. Quanto mais restrita a definição, mais rápido o algoritmo, porém pior em geral a qualidade da solução final. Usualmente adotam-se restrições mínimas que garantam a polinomialidade do algoritmo.

O algoritmo que descreveremos começa com um conjunto arbitrário de $k$ instalações abertas e conta apenas com operações de troca em que fechamos uma instalação aberta e abrimos uma instalação fechada. Operações que exclusivamente abram ou fechem instalações não são considerados, uma vez que apenas abrir uma instalação faz com que nossa solução se torne inviável e é fácil notar que somente fechar alguma instalação não melhora o valor da solução. O algoritmo é parametrizado por uma constante $\epsilon > 0$ que afeta a sua razão de aproximação e o seu consumo de tempo. Para garantir a polinomialidade, uma operação de troca só será feita se diminuir o custo da solução atual em uma razão de $1 - \delta$, onde $\delta$ é escolhido em função de $\epsilon$ e $k$. Como a solução produzida não é necessariamente localmente ótima, iremos chamá-la de solução *quase localmente ótima*.

**Algoritmo 10** BuscaLocal$_\epsilon$-AGKMMP$(F, D, c, k)$

1: $\delta \leftarrow \frac{\epsilon}{(5+\epsilon)k}$
2: Escolha arbitrariamente $S' \subseteq F$ com $|S'| = k$
3: **repita**
4: $\quad S \leftarrow S'$
5: $\quad$ **Se** existem $i \in S$ e $i' \in F \setminus S$ tais que
6: $\qquad\qquad$ custo$(S \setminus \{i\} \cup \{i'\}) <$ custo$(S)$ **então**
7: $\qquad S' \leftarrow S \setminus \{i\} \cup \{i'\}$
8: **até que** $S = S'$
9: **Devolva** $S$

**Teorema 4.3.** *A solução devolvida por* BuscaLocal$_\epsilon$-AGKMMP *tendo a instância $I$ como entrada tem custo no máximo* $(5 + \epsilon)\,\mathrm{opt}(I)$.

*Demonstração.* Seja $S$ a solução devolvida pelo algoritmo e $S^* \subseteq F$ com $|S^*| = k$ uma

resposta ótima para $I$. Seja $\sigma : D \to S$ com $\sigma(j) \coloneqq \arg\min_{i \in S} c_{ij}$ e $\sigma^* : D \to S^*$ com $\sigma^*(j) \coloneqq \arg\min_{i \in S^*} c_{ij}$ funções que associam cada cliente à instalação de $S$ e $S^*$ mais próxima, respectivamente. Além disso, seja $\phi : S^* \to S$ com $\phi(i^*) \coloneqq \arg\min_{i \in S} c_{i^*i}$ função que mapeia cada instalação de $S^*$ para a instalação de $S$ mais próxima.

Para relacionar o custo de $S$ e o custo de $S^*$, iremos descrever um conjunto de $k$ trocas entre instalações de $S$ e instalações de $S^*$ que serão chamadas de trocas *cruciais*. Como $S$ é quase localmente ótima, nenhuma dessas trocas pode melhorar o custo da solução $S$ em uma razão melhor que $1 - \delta$. Isso vai nos permitir delimitar o custo de $S$ em termos de $\delta$ e do custo de $S^*$.

Vamos definir uma partição $\{Z, O, T\}$ das instalações de $S$ que será fundamental para descrevermos as trocas cruciais. Uma instalação $i \in S$ pertence a

i. $Z$ se $|\{i^* \in S^* : \phi(i^*) = i\}| = 0$;

ii. $O$ se $|\{i^* \in S^* : \phi(i^*) = i\}| = 1$;

iii. $T$ se $|\{i^* \in S^* : \phi(i^*) = i\}| > 1$.

Vamos também definir uma partição $\{O^*, T^*\}$ de $S^*$. Uma instalação $i^* \in S^*$ pertence a $O^*$ se $\phi(i^*) \in O$ e, caso contrário, pertence a $T^*$. Note que $|O| = |O^*|$. Consequentemente $|T^*| = |T \cup Z|$. Além disso, como cada instalação de $T$ é imagem de pelo menos duas instalações de $T^*$ em $\phi$, então $|T| \leq \frac{|T^*|}{2}$ e, portanto, $|Z| \geq \frac{|T^*|}{2}$.

Agora, vamos descrever as trocas cruciais. Para cada $i^* \in O^*$, consideramos uma troca entre o par $(\phi(i^*), i^*)$ que gera uma solução $S \setminus \{\phi(i^*)\} \cup \{i^*\}$. Além disso, vamos descrever uma coleção com $|T^*|$ trocas, em que cada uma delas retira da solução uma instalação de $Z$ e inclui na solução uma instalação de $T^*$. Essas trocas podem ser escolhidas arbitrariamente, contanto que cada instalação de $T^*$ apareça uma vez e cada instalação de $Z$ apareça no máximo duas vezes. Isso é possível, uma vez que $|T^*| \leq 2|Z|$.

Considere uma troca crucial entre a instalação $i \in S$ e a instalação $i^* \in S^*$. Seja $S' \coloneqq S \setminus \{i\} \cup \{i^*\}$. Vamos construir uma função de associação $\sigma' : D \to S'$. Para cada cliente $j$ tal que $\sigma^*(j) \neq i^*$ e $\sigma(j) \neq i$, tome $\sigma'(j) \coloneqq \sigma(j)$. Para cada cliente $j$ tal que $\sigma^*(j) = i^*$, tome $\sigma'(j) \coloneqq i^*$. Para cada cliente $j$ tal que $\sigma^*(j) \neq i^*$ e $\sigma(j) = i$, tome $\sigma'(j) \coloneqq \phi(\sigma^*(j))$. Para que $\sigma'(j) \in S'$, é essencial que nesse último caso valha que $\phi(\sigma^*(j)) \neq i$ então vamos mostrar que isso acontece. Suponha por absurdo que $\phi(\sigma^*(j)) = i$. Então, sabemos que $i$ não pertence a $Z$. Como nas trocas cruciais as instalações de $S$ que serão fechadas são sempre de $O$ ou de $Z$, então $i$ pertence a $O$. Logo, existe apenas uma instalação $h$ em $O^*$ tal que $\phi(h) = i$ que é $h = i^*$ e, portanto,

$\sigma^*(j) = i^*$, o que é uma contradição à hipótese desse caso. Assim, vale que

$$\begin{aligned}
-\delta \text{ custo}(S) &\leq \text{custo}(S') - \text{custo}(S) \\
&\leq \sum_{j \in D} c_{\sigma'(j)j} - \text{custo}(S) \\
&= \sum_{j:\sigma^*(j)=i^*} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) \; + \sum_{\substack{j:\sigma^*(j)\neq i^*, \\ \sigma(j)=i}} (c_{\phi(\sigma^*(j))j} - c_{\sigma(j)j}).
\end{aligned}$$

Primeiro, vamos encontrar um limitante superior para o segundo somatório. Pela desigualdade triangular, temos que $c_{\phi(\sigma^*(j))j} \leq c_{\phi(\sigma^*(j))\sigma^*(j)} + c_{\sigma^*(j)j}$. Além disso, por definição de $\phi$, vale que $c_{\phi(\sigma^*(j))\sigma^*(j)} \leq c_{\sigma(j)\sigma^*(j)}$. Aplicando a desigualdade triangular novamente, temos que $c_{\sigma(j)\sigma^*(j)} \leq c_{\sigma(j)j} + c_{\sigma^*(j)j}$. Juntando essas desigualdades, temos que $c_{\phi(\sigma^*(j))j} \leq c_{\sigma(j)j} + 2c_{\sigma^*(j)j}$ e, consequentemente, $c_{\phi(\sigma^*(j))j} - c_{\sigma(j)j} \leq 2c_{\sigma^*(j)j}$. Portanto, vale que

$$\begin{aligned}
-\delta \text{ custo}(S) &\leq \sum_{j:\sigma^*(j)=i^*} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) \; + \sum_{\substack{j:\sigma^*(j)\neq i^*, \\ \sigma(j)=i}} (c_{\phi(\sigma^*(j))j} - c_{\sigma(j)j}) \\
&\leq \sum_{j:\sigma^*(j)=i^*} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) \; + 2 \sum_{\substack{j:\sigma^*(j)\neq i^*, \\ \sigma(j)=i}} c_{\sigma^*(j)j}.
\end{aligned}$$

Como essa desigualdade vale para toda troca crucial, podemos somá-las. Vamos chamar de $i_\ell$ e $i^*_\ell$ a instalação de $S$ e de $S^*$ referentes à troca $\ell$, respectivamente. Observe que $\{i^*_1, \ldots, i^*_k\} = O^* \cup T^*$. Portanto,

$$\begin{aligned}
-k\delta C &\leq \sum_{\ell=1}^{k} \left( \sum_{j:\sigma^*(j)=i^*_\ell} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) \; + 2 \sum_{\substack{j:\sigma^*(j)\neq i^*_\ell, \\ \sigma(j)=i_\ell}} c_{\sigma^*(j)j} \right) \\
&= \sum_{j \in D} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) + 2 \sum_{\ell=1}^{k} \sum_{\substack{j:\sigma^*(j)\neq i^*_\ell, \\ \sigma(j)=i_\ell}} c_{\sigma^*(j)j} \\
&= C^* - C + 2 \sum_{\ell=1}^{k} \sum_{\substack{j:\sigma^*(j)\neq i^*_\ell, \\ \sigma(j)=i_\ell}} c_{\sigma^*(j)j}
\end{aligned}$$

em que $C \coloneqq \text{custo}(S)$ e $C^* \coloneqq \text{custo}(S^*)$. Considere o último somatório. É evidente

que

$$\sum_{\substack{j:\sigma^*(j)\neq i_\ell^*,\\ \sigma(j)=i_\ell}} c_{\sigma^*(j)j} \leq \sum_{j:\sigma(j)=i_\ell} c_{\sigma^*(j)j}.$$

Cada instalação de $S$ aparece em no máximo duas trocas crucias. Então

$$\sum_{\ell=1}^{k} \sum_{j:\sigma(j)=i_\ell} c_{\sigma^*(j)j} \leq 2 \sum_{i\in S} \sum_{j:\sigma(j)=i} c_{\sigma^*(j)j} = 2 \sum_{j\in D} c_{\sigma^*(j)j} = 2C^*.$$

Assim, vale que

$$-k\delta C \leq C^* - C + 2\sum_{\ell=1}^{k} \sum_{\substack{j:\sigma^*(j)\neq i_\ell^*,\\ \sigma(j)=i_\ell}} c_{\sigma^*(j)j} \leq 5C^* - C,$$

consequentemente,

$$C \leq \frac{5C^*}{1-k\delta} = (5+\epsilon)C^*$$

em que a igualdade vale uma vez que $\delta = \frac{\epsilon}{(5+\epsilon)k}$. □

Agora, é necessário mostrar que o algoritmo é polinomial.

**Teorema 4.4.** *O algoritmo* BuscaLocal$_\epsilon$-AGKMMP *toma tempo polinomial para executar.*

*Demonstração.* Uma instância do problema das $k$-medianas consiste em inteiros $n$ e $m$ que designam o número de instalações e clientes, uma matriz $C$ com $nm$ custos não-negativos e um inteiro $k$. Se $C_{\max}$ é o maior custo em $C$, então o tamanho da instância é $O(mn \log C_{\max})$.

Claramente, uma iteração do laço da linha 3 consome tempo polinomial. Precisamos apenas mostrar que o número de iterações do laço do linha 3 é polinomial no tamanho da instância. Podemos assumir sem perda de generalidade que todos os custos são inteiros. Seja $M$ o valor da solução inicial. Note que se $\rho$ é um inteiro tal que $(1-\delta)^\rho M < 1$, então o algoritmo não fará mais que $\rho$ iterações. Como $1 + x < e^x$ para todo $x \neq 0$, sabemos que $(1-\delta)^{\frac{1}{\delta}} < \frac{1}{e}$. Quando elevamos tudo a $\ln M$ temos $(1-\delta)^{\frac{1}{\delta}\ln M} < \frac{1}{M}$ e, portanto, vale que $(1-\delta)^{\frac{1}{\delta}\ln M} M < 1$. Então o número de iterações do laço da linha 3 do algoritmo é no máximo $\frac{1}{\delta}\ln M$. Como $M \leq mC_{\max}$, concluímos que o número de iterações é polinomial no tamanho da instância. □

Arya, Garg, Khandekar, Meyerson, Munagala e Pandit [1] também desenvolveram

um algoritmo de busca local para o problema das $k$-medianas métrico que permite trocas de múltiplas instalações simultaneamente. Esse algoritmo é parametrizado em um número $p$ e, no lugar de trocar uma instalação aberta por uma instalação fechada, ele troca um conjunto $A$ de instalações abertas por um conjunto $B$ de instalações fechadas, tais que $|A| = |B| \leq p$. Isso resulta em uma $(3 + \frac{2}{p})$-aproximação. O parâmetro $p$ está diretamente ligado ao tempo de execução do algoritmo. Tomando $p$ grande o suficiente, é possível desenvolver um algoritmo que é uma $(3 + \epsilon)$-aproximação para qualquer $\epsilon > 0$.

## 4.2 Algoritmos baseados em programação linear

Nessa seção vamos mostrar algoritmos para o problema das $k$-medianas que utilizam métodos baseados em programação linear.

Vamos modelar o problema das $k$-medianas como um programa linear inteiro. Vamos relaxar esse programa e encontrar o seu dual.

Para uma instância $(F, D, c, k)$ do problema das $k$-medianas, o programa inteiro terá dois tipos de variáveis. Uma variável $y_i$ para cada $i \in F$ que terá valor 1 se a instalação $i$ foi aberta e 0 caso contrário, e uma variável $x_{ij}$, para cada $i \in F$ e $j \in D$, que terá valor 1 se o cliente $j$ estiver associado a instalação $i$ e 0, caso contrário.

Assim, uma instância $(F, D, c, k)$ do problema das $k$-medianas pode ser modelada como o seguinte programa linear inteiro:

$$
\begin{aligned}
\text{Minimizar} \quad & \sum_{i \in F, j \in D} c_{ij} x_{ij} & \\
\text{sujeito a} \quad & \sum_{i \in F} x_{ij} \geq 1, & \forall j \in D, \\
& y_i - x_{ij} \geq 0, & \forall i \in F, j \in D, \\
& \sum_{i \in F} y_i \leq k, & \\
& x_{ij} \in \{0, 1\}, & \forall i \in F, j \in D, \\
& y_i \in \{0, 1\}, & \forall i \in F.
\end{aligned}
$$

Para a relaxação desse programa permitiremos que as variáveis em $x$ e em $y$ adotem quaisquer valores não negativos. Portanto, a relaxação do programa inteiro do problema

das $k$-medianas resulta no seguinte programa.

$$\begin{aligned}
\text{Minimizar} \quad & \sum_{i \in F, j \in D} c_{ij} x_{ij} && && \text{(PL)} \\
\text{sujeito a} \quad & \sum_{i \in F} x_{ij} \geq 1, && \forall j \in D, && \\
& y_i - x_{ij} \geq 0, && \forall i \in F, j \in D, && \\
& \sum_{i \in F} y_i \leq k, && && (20) \\
& x_{ij} \geq 0, && \forall i \in F, j \in D, && \\
& y_i \geq 0, && \forall i \in F. &&
\end{aligned}$$

O dual do programa linear acima consiste no seguinte programa.

$$\begin{aligned}
\text{Minimizar} \quad & \sum_{j \in D} v_j - zk && \\
\text{sujeito a} \quad & v_j - w_{ij} \leq c_{ij}, && \forall i \in F, j \in D, \\
& \sum_{j \in D} w_{ij} \leq z, && \forall i \in F, \\
& v_j \geq 0, && \forall j \in D, \\
& w_{ij} \geq 0, && \forall i \in F, j \in D, \\
& z \geq 0. &&
\end{aligned}$$

Note que existe uma grande semelhança entre os programas lineares derivados do problema das $k$-medianas e os derivados do problema de localização de instalações, que se encontram na Seção 3.1. Desse modo, as variáveis e as restrições desses programas lineares têm interpretações equivalentes.

### 4.2.1 Relaxação Lagrangeana

Nessa seção falaremos sobre o algoritmo que utiliza o método de relaxação Lagrangeana para o problema das $k$-medianas métrico. Esse algoritmo foi estudado na Seção 7.7 do livro WS2011 e foi desenvolvido por Jain e Vazirani [16].

A relaxação Lagrangeana é um modo de aumentar o conjunto de soluções viáveis de um programa linear retirando uma de suas restrições e penalizando, na função objetivo, soluções que não respeitem essa restrição. Para o programa linear que é uma relaxação da formulação por um programa linear inteiro do problema das $k$-medianas, iremos

retirar a restrição (20), que limita a quantidade de instalações abertas. Essa remoção resultará no seguinte programa.

$$\begin{aligned}
\text{Minimizar} \quad & \sum_{i\in F, j\in D} c_{ij}x_{ij} + \lambda\left(\sum_{i\in F}(y_i) - k\right) && && \text{(RL)}\\
\text{sujeito a} \quad & \sum_{i\in F} x_{ij} \geq 1, && \forall j \in D,\\
& y_i - x_{ij} \geq 0, && \forall i \in F, j \in D,\\
& x_{ij} \geq 0, && \forall i \in F, j \in D,\\
& y_i \geq 0, && \forall i \in F.
\end{aligned}$$

A constante $\lambda$ é chamada de *multiplicador de Lagrange*, e é o que nos deixará escolher o quão grande será o desconto no valor de soluções que infrinjam a restrição (20). Como queremos piorar o valor das solução que não respeitavam a restrição (20) e como nosso problema é de maximização, então vamos restringir $\lambda$ a valores não-negativos. O valor de uma solução ótima desse programa é no máximo o valor de uma solução ótima do programa original, uma vez que as soluções anteriores continuam viáveis e o seu custo não aumentou.

Como essa relaxação é um programa linear, podemos considerar o seu dual. O dual desse programa é o seguinte.

$$\begin{aligned}
\text{Minimizar} \quad & \sum_{j\in D} v_j - \lambda k && && \text{(DL)}\\
\text{sujeito a} \quad & v_j - w_{ij} \leq c_{ij}, && \forall i \in F, j \in D,\\
& \sum_{j\in D} w_{ij} \leq \lambda, && \forall i \in F,\\
& v_j \geq 0, && \forall j \in D,\\
& w_{ij} \geq 0, && \forall i \in F, j \in D.
\end{aligned}$$

As restrições desses programas são iguais às restrições do programa linear do problema de localização de instalações e do seu dual, respectivamente, da Seção 3.1, considerando que cada instalação tem custo de abertura $\lambda$. Desse modo, podemos utilizar algum algoritmo para o problema de localização de instalações para encontrar uma solução para o problema das $k$-medianas.

No algoritmo de dual fitting para o problema de localização de instalações, da Seção 3.3, encontramos um conjunto $X$ de instalações e valores para as variáveis $\alpha$

tais que, definindo $v_j \coloneqq \frac{\alpha_j}{2}$ e $w_{ij} \coloneqq \max(0, v_j - c_{ij})$ para toda instalação $i$ e cliente $j$, vale que $(v, w)$ é solução viável do dual e, pelo Teorema 3.21, vale que

$$\sum_{j \in D} c(j, X) + 2 \sum_{i \in X} f_i = \sum_{j \in D} \alpha_j = 2 \sum_{j \in D} v_j.$$

Substituindo $f_i$ por $\lambda$, deduzimos que

$$\sum_{j \in D} c(j, X) \leq 2 \left( \sum_{j \in D} v_j - \lambda |X| \right).$$

Se $|X| = k$, então a solução encontrada pelo algoritmo de dual fitting para o problema de localização de instalações também é solução para o problema das $k$-medianas com valor no máximo duas vezes o ótimo.

Note que o valor de $\lambda$ está diretamente relacionado com a quantidade de instalações que serão abertas pelo algoritmo de dual fitting para o problema de localização de instalações. Caso $\lambda = 0$, é normal esperar que muitas instalações sejam abertas e, caso $\lambda = \sum_{j \in D} \sum_{i \in F} c_{ij}$, o algoritmo abrirá apenas uma instalação. Então é natural buscar um método que garanta um $\lambda$ que faça o algoritmo abrir exatamente $k$ instalações. Infelizmente, um tal método não é conhecido. No entanto, podemos tentar encontrar um bom valor para $\lambda$ utilizando busca binária. Nessa busca binária, manteremos dois valores de $\lambda$: $\lambda_1$, que inicialmente terá valor 0, e $\lambda_2$, que inicialmente terá valor $\sum_{j \in D} \sum_{i \in F} c_{ij}$. Excluindo-se casos triviais, com esses valores de $\lambda$ o algoritmo retorna soluções $X_1$ e $X_2$ (respectivamente) com $|X_1| > k$ e $|X_2| < k$. Ao executar o algoritmo com o valor $\lambda \coloneqq \frac{1}{2}(\lambda_1 + \lambda_2)$, este retorna um conjunto de instalações $X$. Caso $|X| = k$, então encontramos um $\lambda$ que garante uma 2-aproximação. Caso contrário, atualizaremos o valor de $\lambda_1$ ou de $\lambda_2$: caso $|X| > k$ atualizaremos o valor de $\lambda_1$ e, caso $|X| < k$, atualizaremos o valor de $\lambda_2$.

Pararemos a busca binária se encontrarmos $\lambda$ que garanta a abertura de $k$ instalações ou se $\lambda_2 - \lambda_1 \leq \frac{\epsilon c_{\min}}{|F|}$, onde $\epsilon$ é um parâmetro da busca binária e $c_{\min}$ é o custo da aresta com menor custo não nulo. Vamos assumir aqui que $0 < c_{\min} \leq \text{opt}(I)$ e vamos mostrar depois que é possível encontrar uma solução ótima para o problema das $k$-medianas métrico em tempo polinomial quando $\text{opt}(I) = 0$.

A seguir formalizamos o algoritmo que representa a busca binária em $\lambda$.

O pseudocódigo para o algoritmo DualFitting-JMMSV está presente na Seção 3.3. Para cada solução $X$ devolvida por DualFitting-JMMSV também são produzidas variáveis $\alpha \in \mathbb{R}^D$ que serão utilizadas posteriormente. Note que Busca-

**Algoritmo 11** BuscaBinária$_\epsilon(F, D, c, k)$

$\lambda_1 \leftarrow 0$; $\lambda_2 \leftarrow \sum_{j \in D} \sum_{i \in F} c_{ij}$
$X_1 \leftarrow F$; $X_2 \leftarrow \emptyset$
**Enquanto** $\lambda_2 - \lambda_1 > \frac{\epsilon c_{\min}}{|F|}$ **faça**
    $\lambda \leftarrow \frac{\lambda_1 + \lambda_2}{2}$
    $f_i \leftarrow \lambda$ para cada $i \in F$
    $X \leftarrow$ DualFitting-JMMSV$(F, D, c, f)$
    **Se** $|X| = k$ **então**
        **Devolva** $(X, \emptyset)$
    **Senão Se** $|X| < k$ **então**
        $\lambda_2 \leftarrow \lambda$; $X_2 \leftarrow X$
    **Senão**
        $\lambda_1 \leftarrow \lambda$; $X_1 \leftarrow X$
**Devolva** $(X_1, X_2)$

Binária$_\epsilon$ consome tempo polinomial para ser executado, uma vez que faz $O(\log \frac{|F| \sum c_{ij}}{\epsilon c_{\min}})$ chamadas ao algoritmo DualFitting-JMMSV, que por sua vez é polinomial.

Agora, vamos provar o seguinte lema que nos permitirá limitar as instâncias às quais iremos tratar.

**Lema 4.5.** *Seja $I$ uma instância para o problema das $k$-medianas métrico. É possível decidir se* $\text{opt}(I) = 0$ *em tempo polinomial.*

*Demonstração.* Seja $I = (F, D, c, k)$ uma instância para o problema das $k$-medianas métrico. Seja $Z(i) \coloneqq \{j \in D : c_{ij} = 0\}$, ou seja, $Z(i)$ é o conjunto de clientes que a instalação $i$ pode suprir com custo 0. Então, para que $\text{opt}(I) = 0$, precisa existir $S \subseteq F$ tal que $|S| \leq k$ e $Z(S) \coloneqq \bigcup_{i \in S} Z(i) = D$.

Primeiro, vamos mostrar que, para $i_1, i_2 \in F$, se $Z(i_1) \cap Z(i_2) \neq \emptyset$, então $Z(i_1) = Z(i_2)$. Seja $j \in Z(i_1)$ e $\ell \in Z(i_1) \cap Z(i_2)$. Então $c_{i_1 j} = c_{i_1 \ell} = c_{i_1 \ell} = c_{i_2 \ell} = 0$ e, pela desigualdade triangular, vale que

$$c_{i_2 j} \leq c_{i_2 \ell} + c_{i_1 \ell} + c_{i_1 j} = 0.$$

Como todos os custos são não negativos, então $c_{i_2 j} = 0$ e portanto $j \in Z(i_2)$. Analogamente, conseguimos ver que para todo $j \in Z(i_2)$, vale que $j \in Z(i_1)$.

Seja $S \subseteq F$ construído escolhendo uma instalação para cada grupo de instalações com o mesmo $Z$, ignorando-se instalações com $Z$ vazio. É evidente que $S$ pode ser construído em tempo polinomial. Vamos mostrar que $\text{opt}(I) = 0$ se e somente se $|S| \leq k$ e $Z(S) = D$.

Vamos mostrar apenas que $\text{opt}(I) = 0$ implica que $|S| \leq k$ e $Z(S) = D$, uma vez que a volta é trivial. Seja $S^*$ tal que $\text{custo}(S^*) = 0$. Suponha sem perda de generalidade que $Z(i) \neq \emptyset$ para todo $i \in S^*$. Se $S^* = S$, então a afirmação é verdadeira. Suponha $S^* \neq S$. Como $Z(S^*) = D$ não é possível ter $S^* \subset S$. Para cada $i^* \in S^* \setminus S$, como $Z(i^*) \neq \emptyset$ e $i^* \notin S$, então existe $i \in S$ tal que $Z(i) = Z(i^*)$. Então, sejam $A = S \cap S^*$ e $B \coloneqq \{i \in S : \text{existe } i^* \in S^* \setminus S \text{ tal que } Z(i) = Z(i^*)\}$. Note que $Z(B) = Z(S^* \setminus S)$. Assim,

$$Z(S) \supseteq Z(A) \cup Z(B) = Z(S^* \cap S) \cup Z(S^* \setminus S) = Z(S^*) = D.$$

Como $Z(S) \subseteq D$, então $Z(S) = D$. Além disso, como $S$ não tem duas instalações com o mesmo $Z$, então $S = A \cup B$. Portanto, $|S| = |A| + |B| \leq |S^* \cap S| + |S^* \setminus S| = |S^*| \leq k$. □

A partir desse momento, vamos considerar que $I = (F, D, c, k)$ é uma instância com $\text{opt}(I) \neq 0$ tal que $\text{BuscaBinária}_\epsilon(F, D, c, k)$ termina com $\lambda_2 - \lambda_1 \leq \frac{\epsilon c_{\min}}{|F|}$ e $|X_1| > k > |X_2|$. Agora, iremos relacionar o custo das soluções $X_1$ e $X_2$ encontradas com o custo de $\text{opt}(I)$.

Considere as variáveis $\alpha^1$ e $\alpha^2$ produzidas por DualFitting-JMMSV associadas às soluções $X_1$ e $X_2$, respectivamente. Seja $v_j^\ell \coloneqq \frac{\alpha_j^\ell}{2}$ e $w_{ij}^\ell \coloneqq \max(0, v_j^\ell - c_{ij})$ para $i \in F$, $j \in D$ e $\ell = 1, 2$. Sejam $a \coloneqq \frac{k - |X_2|}{|X_1| - |X_2|}$ e $b \coloneqq \frac{|X_1| - k}{|X_1| - |X_2|}$. É evidente que $a + b = 1$ e $a, b > 0$. Note que $a|X_1| + b|X_2| = k$. Portanto, $k$ é combinação convexa de $|X_1|$ e $|X_2|$.

Defina $v \coloneqq av^1 + bv^2$ e $w \coloneqq aw^1 + bw^2$. Note que $(v, w)$ é solução viável do programa linear (DL) utilizando custos de abertura $\lambda_2$. Isso é verdade pois $(v^1, w^1)$ também é solução viável desse dual quando utilizamos custos de abertura $\lambda_2$ e, portanto, $(v, w)$ é combinação convexa de duas solução viáveis de (DL) e é um fato que o conjunto de soluções viáveis de um programa linear é convexo.

**Lema 4.6.** *Seja $(X_1, X_2)$ o par devolvido por* $\text{BuscaBinária}_\epsilon$ *com $|X_2| \neq 0$. Seja também $a, b > 0$ tal que $a + b = 1$ e $a|X_1| + b|X_2| = k$, então*

$$a\,\text{custo}(X_1) + b\,\text{custo}(X_2) \leq (1 + \epsilon)2\text{opt}(I).$$

*Demonstração.* Primeiro observemos que

$$\begin{aligned}
\text{custo}\,(X_1) &\leq 2\left(\sum_{j\in D} v_j^1 - \lambda_1|X_1|\right)\\
&= 2\left(\sum_{j\in D} v_j^1 - \lambda_2|X_1|\right) + 2\,(\lambda_2-\lambda_1)\,|X_1|\\
&\leq 2\left(\sum_{j\in D} v_j^1 - \lambda_2|X_1|\right) + 2\frac{\epsilon\, c_{\min}}{|F|}|X_1|\\
&\leq 2\left(\sum_{j\in D} v_j^1 - \lambda_2|X_1|\right) + 2\,\epsilon\,\text{opt}(I)
\end{aligned}$$

onde a primeira desigualdade vale pela garantia do algoritmo utilizado para encontrar a solução $X_1$, a segunda vale pelo limitante imposto para $\lambda_2 - \lambda_1$ ao final do algoritmo e a última vale, pois $c_{\min} \leq \text{opt}(I)$ e $|X_1| \leq |F|$. Utilizando adicionalmente a desigualdade análoga à primeira desigualdade só que para custo$(X_2)$, deduzimos que

$$\begin{aligned}
a\,\text{custo}(X_1) + b\,\text{custo}(X_2) &\leq 2a\left(\sum_{j\in D} v_j^1 - \lambda_2|X_1|\right) + 2a\,\epsilon\,\text{opt}(I) + 2b\left(\sum_{j\in D} v_j^2 - \lambda_2|X_2|\right)\\
&= 2\sum_{j\in D}\left(av_j^1 + bv_j^2\right) - 2\lambda_2\left(a|X_1| + b|X_2|\right) + 2a\,\epsilon\,\text{opt}(I)\\
&= 2\left(\sum_{j\in D} v_j\right) - 2\lambda_2 k + 2a\,\epsilon\,\text{opt}(I)\\
&= 2\left(\sum_{j\in D} v_j - \lambda_2 k\right) + 2a\,\epsilon\,\text{opt}(I)\\
&\leq 2(1+\epsilon)\text{opt}(I)
\end{aligned}$$

onde a última desigualdade vale pois $(v, w)$ é uma solução viável para o dual com custo de abertura de instalações $\lambda_2$ e porque $a \leq 1$. □

Agora, vamos mostrar o algoritmo que utiliza o método de relaxação Lagrangeana e é uma $4(1+\epsilon)$-aproximação para o problema das $k$-medianas.

**Lema 4.7.** *O algoritmo* RelLag-JV *é uma* $4(1+\epsilon)$*-aproximação para o problema das* $k$*-medianas.*

*Demonstração.* É fácil ver que o algoritmo é polinomial. As linhas 2 e 3 cobrem o

**Algoritmo 12** RelLag-JV$(F, D, c, k)$

1: $(X_1, X_2) \leftarrow$ BuscaBinária$_\epsilon(F, D, c, k)$
2: **Se** $X_2 = \emptyset$ **então**
3: **Devolva** $X_1$
4: $a \leftarrow \frac{k-|X_2|}{|X_1|-|X_2|}$; $b \leftarrow \frac{|X_1|-k}{|X_1|-|X_2|}$
5: **Se** $b \geq \frac{1}{2}$ **então**
6: **Devolva** $X_2$
7: $X \leftarrow \emptyset$
8: **Para** $i \in X_2$ **faça**
9: $X \leftarrow X \cup \{\arg\min_{i' \in X_1} c_{ii'}\}$
10: **Se** $|X| < |X_2|$ **então**
11: Seja $A \subseteq X_1 \setminus X$ tal que $|X| + |A| = |X_2|$ escolhido arbitrariamente
12: $X \leftarrow X \cup A$
13: Seja $S \subseteq X_1 \setminus X$ com $|S| = k - |X_2|$ escolhido aleatoriamente.
14: **Devolva** $X \cup S$

caso em que BuscaBinária$_\epsilon$ encontra $\lambda$ tal que o algoritmo de dual fitting para o problema de localização de instalações abre exatamente $k$ instalações (veja as linhas 7 e 8 de BuscaBinária$_\epsilon$). Assim, podemos nos limitar a falar de instâncias em que BuscaBinária$_\epsilon$ termina com $\lambda_2 - \lambda_1 \leq \frac{\epsilon c_{\min}}{|F|}$ e $|X_1| > k > |X_2|$.

As próximas linhas do algoritmo vão produzir uma solução viável para a instância $I = (F, D, c, k)$ do problema das $k$-medianas combinando as soluções $X_1$ e $X_2$.

O algoritmo então considera dois casos: um caso simples, em que $b \geq \frac{1}{2}$, e um caso um pouco mais complicado, em que $b < \frac{1}{2}$. Se $b \geq \frac{1}{2}$, então devolve $X_2$ como solução viável, uma vez que $|X_2| < k$. Usando $b \geq \frac{1}{2}$ e o Lema 4.6, temos que

$$\begin{aligned}\text{custo}(X_2) \leq 2b\,\text{custo}(X_2) &\leq 2\left(a\,\text{custo}(X_1) + b\,\text{custo}(X_2)\right) \\ &\leq 2(1+\epsilon)2\,\text{opt}(I) = 4(1+\epsilon)\,\text{opt}(I).\end{aligned}$$

Agora precisamos mostrar que esse limitante superior também vale para a solução produzida pelo algoritmo no caso em que $b < \frac{1}{2}$. Considerando $X$ inicializado nas linhas 7-9 do algoritmo, para cada instalação $i \in X_2$, a instalação $h \in X_1$ que minimiza $c_{ih}$ estará em $X$. É possível que instalações de $X_2$ tenham como instalação mais próxima em $X_1$ a mesma instalação. Em seguida, nas linhas 10-12, incluímos em $X$ instalações de $X_1$ arbitrariamente até que $|X| = |X_2|$. Após isso, na linha 13, escolhemos aleatoriamente um subconjunto de tamanho $k - |X_2|$ das $|X_1| - |X_2|$ instalações restantes de $X_1$ para incluir em $X$.

Considere o custo de associação esperado entre um cliente $j$ e uma instalação em $X$. Seja $i_1$ a instalação de $X_1$ mais próxima de $j$ e $i_2$ a instalação de $X_2$ mais próxima de $j$. Note que a probabilidade de $i_1$ ser aberta pela parte aleatória do algoritmo é $\frac{k-|X_2|}{|X_1|-|X_2|} = a$, caso ela não tenha sido aberta na primeira parte do algoritmo. Então, com probabilidade pelo menos $a$, o custo de associação de $j$ para a instalação mais próxima em $X$ é no máximo $c_{i_1 j}$. Com probabilidade no máximo $1-a = b$, a instalação $i_1$ não será aberta. Assim, no pior dos casos, iremos associar $j$ a uma instalação aberta no primeiro passo do algoritmo, em particular, a instalação de $X_1$ mais próxima à $i_2$. Seja $i$ a instalação de $X_1$ mais próxima a $i_2$. Então, pela desigualdade triangular, vale que

$$c_{ij} \leq c_{ii_2} + c_{i_2 j}.$$

Pela definição de $i$, sabemos que $c_{ii_2} \leq c_{i_1 i_2}$. Então $c_{ij} \leq c_{i_1 i_2} + c_{i_2 j}$. Novamente pela desigualdade triangular, vale que

$$c_{i_1 i_2} \leq c_{i_1 j} + c_{i_2 j}$$

e, consequentemente,

$$c_{ij} \leq c_{i_1 j} + c_{i_2 j} + c_{i_2 j} = c(j, X_1) + 2c(j, X_2).$$

Assim, o custo de associação esperado de $j$ para a instalação mais próxima em $X$ é

$$\mathbb{E}[c(j, X)] \leq ac\,(j, X_1) + b\,(c(j, X_1) + 2c(j, X_2)) = c(j, X_1) + 2bc(j, X_2).$$

Como $b < \frac{1}{2}$, então $a = 1 - b > \frac{1}{2}$ e

$$\mathbb{E}[c(j, X)] \leq 2\,(ac(j, X_1) + bc(j, X_2))\,.$$

Somando para todos os clientes $j$, temos

$$\begin{aligned}\sum_{j \in D} \mathbb{E}[c(j, X)] &\leq \sum_{j \in D} 2\,(ac(j, X_1) + bc(j, X_2)) = 2\left(a \sum_{j \in D} c(j, X_1) + b \sum_{j \in D} c(j, X_2)\right) \\ &= 2\,(a\,\text{custo}(X_1) + b\,\text{custo}(X_2)) \leq 2(1+\epsilon)2\text{opt}(I) = 4(1+\epsilon)\,\text{opt}(I)\end{aligned}$$

em que a última desigualdade vale pelo Lema 4.6. □

Assim, temos uma $4(1+\epsilon)$-aproximação aleatória para o problema das $k$-medianas

que utiliza o método de relaxação Lagrangeana. Vamos agora desaleatorizá-lo.

A desaleatorização utiliza o método das esperanças condicionais. Não precisamos tratar o caso em que $b \geq \frac{1}{2}$, uma vez que ele é determinístico. Então vamos tratar apenas do caso em que $b < \frac{1}{2}$.

No caso em que $b < \frac{1}{2}$, abrimos $|X_2|$ instalações de $X_1$ de maneira determinística. Após isso, escolhemos aleatoriamente $k - |X_2|$ instalações das $|X_1| - |X_2|$ instalações restantes de $X_1$. Considerando $X$ na linha 13 do algoritmo, seja $X_1' \coloneqq X_1 \setminus X$. Para uma escolha $S \subseteq X_1'$ com $|S| \leq k - |X_2|$, vamos definir $\mathbb{E}(S, X_1' - S)$ o custo esperado em que todas as facilidades em $S$ são abertas e outras $k - |X_2| - |S|$ instalações são escolhidas aleatoriamente de $X_1' - S$. Como cada instalação de $X_1' \setminus S$ tem a mesma chance de ser escolhida, temos que

$$\mathbb{E}(S, X_1' \setminus S) = \frac{1}{|X_1' - S|} \sum_{i \in X_1' - S} \mathbb{E}(S \cup \{i\}, X_1' \setminus (S \cup \{i\})).$$

Isso implica que existe um $i$ tal que $\mathbb{E}(S \cup \{i\}, X_1' \setminus (S \cup \{i\})) \leq \mathbb{E}(S, X_1' \setminus S)$. Assim, para desaleatorizar o algoritmo, trocaremos a linha 13 de RelLag-JV pelo seguinte conjunto de linhas.

---

1: $X_1' \leftarrow X_1 \setminus X$
2: $S \leftarrow \emptyset$
3: **Enquanto** $|S| < k - |X_2|$ **faça**
4: $\quad i \leftarrow \arg\min_{i \in S} \mathbb{E}(S \cup \{i\}, X_1' \setminus (S \cup \{i\}))$
5: $\quad S \leftarrow S \cup \{i\}$

---

## 4.3 Inaproximabilidade

Nesta seção vamos mostrar um resultado que, sob certa hipótese, limita inferiormente a razão de aproximação para os algoritmos do problema das $k$-medianas métrico. Esse resultado é o Teorema 16.17 que se encontra na Seção 16.2 do livro WS2011. Ele foi provado por Jain, Mahdian, Markakis, Saberi e Vazirani [14].

Relembre o Problema (3.24) que é a versão de otimização do problema de cobertura de conjuntos. Vamos mostrar que, se existe uma $\alpha$-aproximação $A$ para o problema das $k$-medianas métrico com $\alpha < 1.735$, então existe uma $(c \ln n)$-aproximação para o problema da cobertura de conjuntos para uma constante $c < 1$. Feige [6] mostrou que a existência desse segundo algoritmo implica na existência de um algoritmo determinístico que leva tempo $O(n^{O(\log \log n)})$ para resolver instâncias de tamanho $n$ de

qualquer problema em $NP$.

Vamos apresentar um algoritmo de aproximação para o problema da cobertura de conjuntos que utiliza várias chamadas ao algoritmo que é uma $\alpha$-aproximação para o problema das $k$-medianas métrico.

**Algoritmo 13** INAPROX-JMMSV$(E, \{S_1, \ldots, S_m\})$

**Para** $i \leftarrow 1$ até $m$ **faça**
    **Para** todo $j \in E$ **faça**
        **Se** $j \in S_i$ **então**
            $c_{ij} \leftarrow 1$
        **Senão**
            $c_{ij} \leftarrow 3$
**Para** $k \leftarrow 1$ até $m$ **faça**
    $I_k \leftarrow \emptyset$;  $D \leftarrow E$;  $F \leftarrow [m]$
    **Enquanto** $D \neq \emptyset$ **faça**
        $F' \leftarrow \alpha$-APROX-K-MED$(F, D, c, k)$
        $I_k \leftarrow I_k \cup F'$
        $D' \leftarrow \{j \in D : c(j, F') = 1\}$
        $F \leftarrow F \setminus F'$;  $D \leftarrow D \setminus D'$
**Devolva** $I_k$ que minimize $|I_k|$.

Seja $I = (E, \{S_1, \ldots, S_m\})$ uma instância do problema da cobertura de conjuntos. Para um $k$ fixo, o algoritmo cria uma sequência de instâncias métricas para o problema das $k$-medianas. Todas elas terão a mesma função de custo $c$ tal que, para todo $i \in [m]$ e $j \in E$, $c_{ij} = 1$ se $j \in S_i$ e $c_{ij} = 3$ caso contrário. É fácil notar que essa função é uma métrica. A primeira instância da sequência terá o conjunto de instalações $F := [m]$ e o conjunto de clientes $D := E$. A ideia é que usamos o algoritmo $\alpha$-APROX-K-MED para encontrar um conjunto $F' \subseteq F$ de instalações que serão abertas e existirá um conjunto $D' \subseteq D$ tal que, para todo $j \in D'$, existe $i \in F'$ tal que $j \in S_i$. Então, para construir a próxima instância, usaremos o conjunto de instalações $F \setminus F'$ e o conjunto de clientes $D \setminus D'$, que representam os subconjuntos de $E$ que ainda podem ser escolhidos e os elementos de $E$ que ainda não foram cobertos, respectivamente. Isso acontecerá sucessivamente até que todos os elementos de $E$ sejam cobertos, ou seja, até que $D = \emptyset$. Queremos escolher $k^*$ instalações em cada chamada do algoritmo do problema das $k$-medianas, em que $k^* = \text{opt}(I)$. Como é difícil saber o tamanho de uma cobertura de conjuntos ótima, vamos rodar essa rotina para todo $k \in [m]$, ou seja, para todos os candidatos para o tamanho $k^*$ de uma cobertura de conjuntos ótima para essa instância.

**Teorema 4.8.** *Se existe uma $\alpha$-aproximação para o problema das $k$-medianas com $\alpha < 1.735$, então* Inaprox-JMMSV *é uma $(c \ln n)$-aproximação para o problema de cobertura de conjuntos com $c < 1$ quando $n$ é suficiente grande.*

*Demonstração.* Seja $k^*$ o tamanho de uma cobertura de conjuntos ótima da instância $(E, \{S_1, \ldots, S_m\})$ do problema de cobertura de conjuntos. Suponha que estamos na iteração $k^*$ do laço **Para** e na iteração $\ell$ do laço **Enquanto** do algoritmo Inaprox-JMMSV. Seja $D_\ell$ e $F_\ell$ os conjuntos $D$ e $F$ no começo dessa iteração do laço **Enquanto**. Seja $n_\ell \coloneqq |D_\ell|$. Note que existe uma solução para a instância $(F_\ell, D_\ell, c, k^*)$ do problema das $k$-medianas com custo $n_\ell$ em que abrimos todas as instalações que representam subconjuntos que estão em uma mesma resposta ótima da cobertura de conjuntos. Seja $F'_\ell$ o conjunto das instalações devolvido pela chamada de $\alpha$-aprox-k-med na linha 10 e $D'_\ell$ o conjunto de clientes encontrado na linha 12. Denote $p_\ell \coloneqq |D'_\ell|/n_\ell$. É fácil notar que o custo da solução $F'_\ell$ é

$$p_\ell n_\ell + 3(1 - p_\ell)n_\ell \leq \alpha n_\ell$$

e, portanto,

$$3 - 2p_\ell \leq \alpha.$$

Seja $c$ uma constante entre 0 e 1 que vamos escolher depois. Suponha que $p_\ell \leq 1 - e^{-\frac{1}{c}}$ para algum $\ell$. Então

$$1 + 2e^{-\frac{1}{c}} \leq 3 - 2p_\ell < \alpha.$$

O valor de $c$ que maximiza o lado esquerdo da desigualdade é $c$ próximo de 1 e teremos $1.735 \leq \alpha < 1.735$ o que é um absurdo. Note que o lado esquerdo da desigualdade é quem nos traz o valor do resultado de inaproximabilidade para o problema das $k$-medianas.

Agora, vamos mostrar que, se $p_\ell > 1 - e^{-\frac{1}{c}}$ para todo $\ell$, então Inaprox-JMMSV devolve uma $(c' \ln n)$-aproximação para o problema da cobertura de conjuntos com $c' < 1$. Vamos chamar de $r$ a quantidade de iterações que o laço **Enquanto** realiza. É evidente que $|I_{k^*}| \leq k^* r$, logo a razão de aproximação desse algoritmo é no máximo $r$. Para todo $\ell$ vale que $p_\ell > 1 - e^{-\frac{1}{c}}$, então temos que $1 < c \ln\left(\frac{1}{1-p_\ell}\right)$. Note que $n_{\ell+1} = n_\ell(1 - p_\ell)$ e $n_r \geq 1$. Assim, vale que $n_1 \prod_{\ell=1}^{r-1}(1 - p_\ell) = n_r \geq 1$ e, consequentemente,

$\ln\left(\prod_{\ell=1}^{r-1}\frac{1}{1-p_\ell}\right) \leq \ln n_1 = \ln n$. Assim,

$$r = (r-1)+1 < c\sum_{\ell=1}^{r-1}\ln\left(\frac{1}{1-p_\ell}\right)+1 \leq c\ln n+1 \leq c'\ln n,$$

em que a última desigualdade vale para algum $c' \in (c,1)$, uma vez que, pelo enunciado do teorema, podemos assumir que $n$ é suficientemente grande.

Assim, temos uma aproximação para o problema de cobertura de conjuntos com razão de aproximação estritamente menor que $\ln n$ para $n$ suficientemente grande. □

Juntando esse resultado com o resultado do Feige, temos que a existência de uma $\alpha$-aproximação para o problema das $k$-medianas métrico com $\alpha < 1.735$ implica a existência de um algoritmo determinístico que leva tempo $O(n^{O(\log\log n)})$ para todo problema $NP$-completo.

# 5 Algoritmo de Li e Svensson

Nessa seção vamos falar sobre o algoritmo que é uma $(1+\sqrt{3}+\epsilon)$-aproximação para o problema das $k$-medianas métrico, desenvolvido por Li e Svensson [21].

O método do algoritmo é dividido em duas componentes principais de interesses independentes. Na primeira, é mostrado que, para obter uma $(\alpha+\epsilon)$-aproximação para o problema das $k$-medianas métrico, é suficiente obter um algoritmo polinomial $A$ tal que, dado uma instância $I = (F, D, c, k)$ do problema das $k$-medianas métrico, encontra um conjunto $S \subseteq F$ com $|S| = k + O(1)$ em que custo$(S) \leq \alpha\,\mathrm{opt}(I)$. A segunda componente complementa a primeira, encontrando tal algoritmo $A$ com $\alpha = 1+\sqrt{3}+\epsilon$.

Primeiramente, vamos definir e relembrar definições que serão necessárias. Seja $I = (F, D, c, k)$ uma instância do problema das $k$-medianas métrico. Uma *pseudo-solução* de $I$ é um conjunto $S \subseteq F$. Dizemos que uma pseudo-solução $S$ é uma *solução $d$-aditiva* se $|S| \leq k + d$. Se $S$ é uma solução 0-aditiva, então $S$ é uma solução. Falaremos que um algoritmo que produz uma solução $d$-aditiva com custo no máximo $\alpha\,\mathrm{opt}(I)$ é uma $\alpha$-aproximação $d$-aditiva para o problema das $k$-medianas. Em vários momentos precisaremos falar sobre clientes ou instalações que estão próximos, por isso, definimos, para qualquer $p \in F \cup D$ e $r \geq 0$, os conjuntos $\mathrm{FBall}(p, r) \coloneqq \{i \in F : c(p, i) < r\}$ e $\mathrm{DBall}(p, r) \coloneqq \{j \in D : c(p, j) < r\}$. Além disso, utilizaremos o algoritmo Busca-Binária$_\epsilon$ e o Lema 4.6 da Seção 4.2.1.

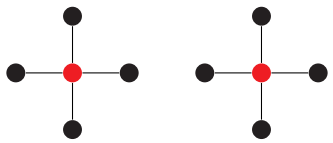
## 5.1 Obtenção de soluções a partir de soluções aditivas

Essa seção foca na primeira componente do método. Queremos provar o seguinte teorema.

**Teorema 5.1.** *Seja $A$ uma $\alpha$-aproximação $d$-aditiva para o problema das $k$-medianas métrico, para algum $\alpha > 0$. Então, para todo $\epsilon > 0$, existe uma $(\alpha+\epsilon)$-aproximação para o problema com tempo de execução $O(n^{O(d/\epsilon)})$ vezes o tempo de execução de $A$.*

Existem instâncias para as quais abrir instalações adicionais pode diminuir muito o valor da solução. Podemos observar isso na instância a seguir em que as bolas pretas representam os clientes e as bolas vermelhas representam as instalações. As arestas desenhadas têm custo 0 e as outras arestas entre clientes e instalações têm custo $M$,

sendo $M$ um número positivo qualquer.



Para $k = 1$, podemos notar que uma pseudo-solução $T$ com $|T| = 2$ tem custo 0, enquanto o custo da solução ótima é $4M$. Por isso, vamos fazer um pré-processamento na instância para evitar que abrir um número constante de instalações a mais não aumente muito o custo da pseudo-solução encontrada. Para isso vamos definir o conceito de instâncias esparsas.

**Definição 5.2.** *Para $\beta > 0$, uma instância $I = (F, D, c, k)$ do problema das $k$-medianas métrico é* $\beta$-esparsa *se, para cada instalação $i \in F$, vale que*

$$(1 - \xi)\, c(i, S^*)\, |\mathrm{DBall}(i, \xi c(i, S^*))| \leq \beta, \tag{25}$$

*em que $\xi \coloneqq 1/3$ e $S^*$ é uma solução ótima para a instância $I$. Também dizemos que uma instalação $i$ é* $\beta$-densa *se ela viola* (25).

Veremos que qualquer solução $d$-aditiva para uma instância esparsa pode ser transformada em uma solução sem aumentar muito o seu custo. Além disso, também veremos que é possível transformar qualquer instância em uma instância esparsa sem perder informações que julgaremos importantes sobre a instância original.

Então, a ideia final é que receberemos uma instância $I$ do problema das $k$-medianas métrico e criaremos uma instância $I'$ $\mathrm{opt}(I)/t$-esparsa a partir de $I$, para algum inteiro $t$. Em seguida, utilizaremos o algoritmo $A$ que é uma $\alpha$-aproximação $d$-aditiva em $I'$ e, com a solução $d$-aditiva devolvida, construiremos uma solução para a instância $I$ com custo no máximo $(\alpha + \epsilon)\,\mathrm{opt}(I)$. O valor de $t$ vai depender do valor de $\epsilon > 0$ escolhido e do valor de $d$.

### 5.1.1 Instâncias esparsas

A ideia do algoritmo que constrói instâncias esparsas é devolver várias instâncias do problema das $k$-medianas construídas retirando instalações da instância recebida. De todas essas instâncias devolvidas, o método garante que alguma delas carregue informações importantes da instância original.

**Algoritmo 14** ESPARSA$(F, D, c, k, t)$

$\mathcal{I} \leftarrow \emptyset$
**Para** toda sequência de pares de instalações $(i_1, i_1^*), (i_2, i_2^*), \ldots, (i_{t'}, i_{t'}^*)$ com $t' \leq t$ **faça**
    $F' \leftarrow F \setminus \bigcup_{z=1}^{t'} \text{FBall}(i_z, c(i_z, i_z^*))$
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{(F', D, c, k)\}$
**Devolva** $\mathcal{I}$

**Lema 5.3.** *Dada uma instância $I = (F, D, c, k)$ do problema das $k$-medianas métrico com $n = |F|$ e um inteiro positivo $t$, o Algoritmo* ESPARSA *devolve em tempo $n^{O(t)}$ um conjunto de instâncias do problema das $k$-medianas métrico em que pelo menos uma, digamos $I' = (F' \subseteq F, D, c, k)$, satisfaz*

1. *uma solução ótima de $I$ também é uma solução ótima de $I'$; e*
2. *$I'$ é $\text{opt}(I)/t$-esparsa.*

*Demonstração.* É fácil notar que o algoritmo seleciona $n^{O(t)}$ sequências de pares de instalações e pode ser implementado para executar em tempo $n^{O(t)}$. Considere uma sequência maximal $(i_1, i_1^*), (i_2, i_2^*), \ldots, (i_\ell, i_\ell^*)$ tal que, para todo $b = 1, \ldots, \ell$, vale

- $i_b \in F \setminus \bigcup_{z=1}^{b-1} \text{FBall}(i_z, c(i_z, i'_z))$ é uma instalação $\text{opt}(I)/t$-densa; e
- $i_b^*$ é a instalação em $S^*$ mais próxima à $i_b$, sendo $S^*$ uma solução ótima de $I$.

Seja $I' \coloneqq (F', D, c, k)$ com $F' \coloneqq F \setminus \bigcup_{z=1}^{\ell} \text{FBall}(i_z, c(i_z, i_z^*))$. É fácil notar que uma solução ótima de $I$ também é solução ótima de $I'$, uma vez que $S^* \subseteq F'$. Também é fácil notar que $I'$ é uma instância $\text{opt}(I)/t$-esparsa, caso contrário a sequência não seria maximal.

Agora, precisamos mostrar que a sequência $(i_1, i_1^*), (i_2, i_2^*), \ldots, (i_\ell, i_\ell^*)$ está entre as sequências escolhidas por ESPARSA, ou seja, mostrar que $\ell \leq t$. Seja $B_z \coloneqq \text{DBall}(i_z, \xi c(i_z, i_z^*))$ e $w$ tal que $1 \leq z < w \leq \ell$. Vamos mostrar que $B_z \cap B_w = \emptyset$. Note que

$$c(i_w, i_w^*) \leq c(i_w, i_z^*) \leq c(i_w, i_z) + c(i_z, i_z^*) \leq 2c(i_w, i_z), \tag{26}$$

em que a primeira desigualdade vale uma vez que $i_w^*$ é uma instalação de $S^*$ mais próxima à $i_w$, a segunda desigualdade vale pela desigualdade triangular e a terceira vale pois $i_w \notin \text{FBall}(i_z, c(i_z, i_z^*))$. Seja $j$ um cliente em $B_z$. Vale que

$$c(i_z, i_w) \leq c(i_z, j) + c(i_w, j) < \xi c(i_z, i_z^*) + c(i_w, j), \tag{27}$$

em que a primeira desigualdade vale pela desigualdade triangular e a segunda vale pela definição de $B_z$. Assim,

$$c(i_w, j) > c(i_z, i_w) - \xi c(i_z, i_z^*) \geq c(i_z, i_w) - \xi c(i_w, i_z) = 2\xi c(i_z, i_w) \geq \xi c(i_w, i_w^*),$$

em que a primeira desigualdade vale por (27), a segunda desigualdade vale pois $i_w \notin \text{FBall}(i_z, c(i_z, i_z^*))$ e a última desigualdade vale por (26). Portanto, $j \notin B_w$. Agora que mostramos que as bolas de clientes não se intersectam, vamos mostrar que isso implica que $\ell \leq t$. Seja $h^*$ uma instalação de $S^*$ mais próxima a $j$. Vale que

$$\begin{aligned} c(j, h^*) &\geq c(i_z, h^*) - c(i_z, j) > c(i_z, h^*) - \xi c(i_z, i_z^*) \\ &\geq c(i_z, i_z^*) - \xi c(i_z, i_z^*) = (1 - \xi) c(i_z, i_z^*), \end{aligned}$$

em que a primeira desigualdade vale pela desigualdade triangular, a segunda vale pois $j \in B_z$ e a terceira vale pela definição de $i_z^*$. Como isso vale para qualquer $1 \leq z \leq \ell$ e qualquer $j \in B_z$, temos que

$$\text{opt}(I) = \sum_{j \in D} c(j, S^*) \geq \sum_{z \in [\ell]} \sum_{j \in B_z} c(j, S^*) > \sum_{z \in [\ell]} (1 - \xi) c(i_z, i_z^*) |B_z|.$$

Como $i_z$ é $\text{opt}(I)/t$-densa, então

$$\text{opt}(I) > \sum_{z \in [\ell]} (1 - \xi) c(i_z, i_z^*) |B_z| \geq \sum_{z \in [\ell]} \frac{\text{opt}(I)}{t} = \text{opt}(I) \cdot \frac{\ell}{t}.$$

Assim, concluímos que $\ell < t$. □

Logo, conseguimos transformar uma instância $I$ qualquer em uma instância $I'$ $\text{opt}(I)/t$-esparsa em tempo $n^{O(t)}$, para qualquer $t$ inteiro, mantendo o seu valor ótimo.

### 5.1.2 Soluções para instâncias esparsas a partir de pseudo-soluções

O seguinte algoritmo será responsável por, a partir de uma solução $d$-aditiva $\mathcal{T}$ para uma instância $\beta$-esparsa, encontrar uma solução para essa instância com custo não muito maior do que o custo de $\mathcal{T}$. O algoritmo será dividido em duas partes. Na primeira delas, removeremos instalações de $\mathcal{T}$ que não aumentem muito o custo da solução. Caso seja possível remover instalações assim até que o tamanho de $\mathcal{T}$ seja menor ou igual a $k$, então já obtemos uma tal solução. A segunda parte só será executada se não for possível remover instalações a ponto de transformar $\mathcal{T}$ em uma solução. Nisto, iremos enumerar

vários conjuntos de $k$ instalações de modo que um deles garantidamente estará próximo de uma solução ótima.

O algoritmo recebe uma instância $I$ $\beta$-esparsa, uma solução $\mathcal{T}$ $d$-aditiva, um inteiro $t$ e um número $\delta \in (0, 1/8)$, e devolve uma solução $S$ para $I$ tal que $\text{custo}(S) \leq \max\left\{\text{custo}(\mathcal{T}) + dB, \frac{1+3\delta}{1-3\delta}\text{opt}(I)\right\}$, onde $B = 2 \cdot \frac{\beta+\text{custo}(\mathcal{T})/t}{\delta\xi}$.

**Algoritmo 15** Pseudo-para-Sol-Esp$(I = (F, D, c, k), \mathcal{T}, t, \delta)$

1: $\mathcal{T}' \leftarrow \mathcal{T}$; $\quad B \leftarrow 2 \cdot \frac{\beta+\text{custo}(\mathcal{T})/t}{\delta\xi}$
2: **Enquanto** $|\mathcal{T}'| > k$ **e** existe $i \in \mathcal{T}'$ tal que $\text{custo}(\mathcal{T}' \setminus \{i\}) \leq \text{custo}(\mathcal{T}') + B$ **faça**
3: $\quad \mathcal{T}' \leftarrow \mathcal{T}' \setminus \{i\}$
4: **Se** $|\mathcal{T}'| \leq k$ **então**
5: $\quad$ **Devolva** $S \leftarrow \mathcal{T}'$
6: **Para** todo $Q \subseteq \mathcal{T}'$ e $V \subseteq F$ tal que $|Q| + |V| = k$ e $|V| < t$ **faça**
7: $\quad S_{Q,V} \leftarrow V$
8: $\quad$ **Para** cada $i \in Q$ **faça**
9: $\quad\quad L_i \leftarrow c(i, \mathcal{T}' \setminus \{i\})$
10: $\quad\quad$ Seja $f_i$ uma instalação em $\text{FBall}(i, \delta L_i)$ que minimiza

$$\sum_{j \in \text{DBall}(i, L_i/3)} \min\{c(f_i, j), c(j, V)\}$$

11: $\quad\quad S_{Q,V} \leftarrow S_{Q,V} \cup \{f_i\}$
12: **Devolva** $S \leftarrow \arg\min_{S_{Q,V}} \text{custo}(S_{Q,V})$

**Lema 5.4.** *Para uma instância $\beta$-esparsa $I = (F, D, c, k)$, uma solução $d$-aditiva $\mathcal{T}$, um inteiro $t \geq \frac{2d}{\delta\xi}$ e um número $\delta \in (0, 1/8)$,* Pseudo-Para-Sol-Esp *encontra em tempo $t\, n^{O(t)}$, onde $n = |F|$, um conjunto $S \subseteq F$ tal que*

- $|S| \leq k$; *e*
- $\text{custo}(S) \leq \max\left\{\text{custo}(\mathcal{T}) + dB, \frac{1+3\delta}{1-3\delta}\text{opt}(I)\right\}$, *onde* $B = 2 \cdot \frac{\beta+\text{custo}(\mathcal{T})/t}{\delta\xi}$.

A prova do Lema 5.4 será dividida em vários lemas. O primeiro deles irá provar a delimitação no tempo de execução do algoritmo.

**Lema 5.5.** *Para uma instância $\beta$-esparsa $I = (F, D, c, k)$, uma solução $d$-aditiva $\mathcal{T}$, um inteiro $t \geq \frac{2d}{\delta\xi}$ e um número $\delta \in (0, 1/8)$,* Pseudo-Para-Sol-Esp *leva tempo $t\, n^{O(t)}$ para executar, onde $n = |F|$.*

*Demonstração.* É evidente que o **Enquanto** da linha 2 executa no máximo $d$ iterações, uma vez que $\mathcal{T}$ é uma solução $d$-aditiva e em cada iteração uma instalação é removida

de $\mathcal{T}'$. Então, suponha que, após remover instalações cuja remoção não aumenta muito o custo de $\mathcal{T}$, obtemos um conjunto $\mathcal{T}' \subseteq \mathcal{T}$ em que $|\mathcal{T}'| > k$. Assim, o laço **Para** da linha 6 é executado. O número de iterações desse laço é igual ao número de pares de conjuntos $Q \subseteq \mathcal{T}'$ e $V \subseteq F$ com $|Q| + |V| = k$ e $|V| < t$. A quantidade desses pares é

$$\begin{aligned}
\sum_{\ell=1}^{t-1} \binom{|\mathcal{T}'|}{k-\ell}\binom{|F|}{\ell} &\leq \sum_{\ell=0}^{t-1} \binom{k+d}{k-\ell}\binom{|F|}{\ell} \\
&= \sum_{\ell=0}^{t-1} \binom{k+d}{d+\ell}\binom{|F|}{\ell} \\
&\leq \sum_{\ell=0}^{t-1} (k+d)^{(d+\ell)} n^{\ell} \\
&\leq \sum_{\ell=0}^{t-1} n^{(d+2\ell)} \\
&\leq \sum_{\ell=0}^{t-1} n^{3t} = t\, n^{3t}
\end{aligned}$$

em que a primeira e a terceira desigualdades valem uma vez que $|\mathcal{T}'| \leq k + d \leq n$. Assim, temos no máximo $t\, n^{3t} = t\, n^{O(t)}$ iterações do laço da linha 6. □

Agora precisamos mostrar que o conjunto $S \subseteq F$ devolvido por Pseudo-para-Sol-Esp satisfaz o que é proposto pelo Lema 5.4. Vamos separar no caso em que $S$ é devolvido na linha 5 e em que $S$ é devolvido na linha 12. Caso $S$ seja devolvido na linha 5 é evidente que $|S| \leq k$ e que custo$(S) \leq$ custo$(\mathcal{T}) + dB$. Caso $S$ seja devolvido na linha 12 também é evidente que $|S| \leq k$, mas não é evidente que o custo de $S$ satisfaz o que é proposto. Como chegamos no laço **Para** da linha 6, então temos um conjunto $\mathcal{T}' \subseteq \mathcal{T}$ com $|\mathcal{T}'| > k$ e que custo$(\mathcal{T}' \setminus \{i\}) >$ custo$(\mathcal{T}') + B$, para todo $i \in \mathcal{T}'$. Assim, iremos mostrar conjuntos $Q_0 \subseteq \mathcal{T}'$ e $V_0 \subseteq F$ com $|Q_0| + |V_0| = k$ e $|V_0| < t$ tais que custo$(S_{Q_0,V_0}) \leq \frac{1+3\delta}{1-3\delta}$opt$(I)$. Para mostrar esses conjuntos usaremos o seguinte conceito.

**Definição 5.6.** *Para toda instalação $i \in \mathcal{T}'$, seja $L_i := c(i, \mathcal{T}' \setminus \{i\})$ e $\ell_i := c(i, S^*)$, em que $S^*$ é uma solução ótima de $I$. Dizemos que uma instalação $i \in \mathcal{T}'$ é* determinada *se $\ell_i < \delta L_i$.*

Vamos então definir os conjuntos $Q_0$ e $V_0$. O conjunto $Q_0$ é formado pelas instalações determinadas de $\mathcal{T}'$. Seja $f_i^*$ uma instalação de $S^*$ mais próxima de $i$, para cada $i \in Q_0$. Definimos, então, $V_0 := S^* \setminus \{f_i^* : i \in Q_0\}$. A intuição dessa escolha é que a solução $S_{Q_0,V_0}$ é muito próxima de $S^*$, uma vez que as instalações de $Q_0$ têm instalações de

$S^*$ em uma vizinhança muito próxima e as instalações de $V_0$ são instalações de $S^*$. Precisamos provar que os conjuntos $Q_0$ e $V_0$ são selecionados pelo algoritmo, ou seja, que $|Q_0| + |V_0| = k$ e $|V_0| < t$.

**Lema 5.7.** *Sejam $Q_0$ e $V_0$ os conjuntos definidos no parágrafo anterior. Vale que $|Q_0| + |V_0| = k$.*

*Demonstração.* Note que

$$|\{f_i^* : i \in Q_0\}| + |V_0| = |S^*| = k,$$

em que $f_i^*$ é uma instalação de $S^*$ mais próxima à $i$ e estamos assumindo sem perda de generalidade que $|S^*| = k$. Mostraremos que $|Q_0| = |\{f_i^* : i \in Q_0\}|$. Para isso, é suficiente mostrar que, para duas instalações $i, i' \in Q_0$ distintas, vale que $f_i^* \neq f_{i'}^*$. Suponha, por absurdo, que $f_i^* = f_{i'}^*$. Note que $c(i, i') \geq \max\{L_i, L_{i'}\}$. Como ambas instalações são determinadas, então

$$c(i, i') \geq \max\{L_i, L_{i'}\} > \frac{1}{\delta} \max\{\ell_i, \ell_{i'}\} > 8 \max\{\ell_i, \ell_{i'}\}.$$

Além disso, pela desigualdade triangular, vale que

$$c(i, i') \leq c(i, f_i^*) + c(i', f_i^*) = c(i, f_i^*) + c(i', f_{i'}^*) = \ell_i + \ell_{i'} \leq 2 \max\{\ell_i, \ell_{i'}\}.$$

Assim, temos que

$$8 \max\{\ell_i, \ell_{i'}\} < c(i, i') \leq 2 \max\{\ell_i, \ell_{i'}\}$$

o que é um absurdo, uma vez que todos os custos são não negativos e $i \neq i'$. Então, $f_i^* \neq f_{i'}^*$ para todo $i, i' \in Q_0$ distintos e, consequentemente, $|\{f_i^* : i \in Q_0\}| = |Q_0|$. Logo, $|V_0| + |Q_0| = k$. □

Para garantir que os conjuntos serão escolhidos pelo algoritmo em algum momento, precisamos mostrar que $|V_0| < t$.

**Lema 5.8.** *Sejam $Q_0$ e $V_0$ os conjuntos definidos no parágrafo anterior ao Lema 5.7. Vale que $|V_0| < t$.*

*Demonstração.* Como $|V_0| = k - |Q_0|$ e as instalações de $\mathcal{T}'$ podem ser particionadas em determinadas e não determinadas, então vale que $|V_0| = k - (|\mathcal{T}'| - |U_0|)$ em que $U_0$ é o conjunto das instalações não determinadas de $\mathcal{T}'$. Como $|\mathcal{T}'| > k$, então $|V_0| < |U_0|$. Vamos mostrar que $|U_0| \leq t$. Suponha, por absurdo, que $|U_0| > t$. Seja

$\mathcal{C}_i \coloneqq \{j \in D : i = \arg\min_{i' \in \mathcal{T}'} c_{i'j}\}$ para todo $i \in \mathcal{T}'$, ou seja, o conjunto de clientes tais que a instalação de $\mathcal{T}'$ mais próxima a eles é $i$. Seja também $C_i \coloneqq \sum_{j \in \mathcal{C}_i} c_{ij}$. Então, custo$(\mathcal{T}') = \sum_{i \in \mathcal{T}'} C_i$. Seja $i$ uma instalação de $U_0$ com menor $C_i$. É evidente que $C_i \leq$ custo$(\mathcal{T}')/t$, uma vez que $|U_0| > t$. Seja $i'$ uma instalação de $\mathcal{T}' \setminus \{i\}$ mais próxima à $i$. Vamos analisar o custo resultante ao remover $i$ de $\mathcal{T}'$ e conectar os clientes em $\mathcal{C}_i$ à $i'$. Vamos analisar particionando os clientes de $\mathcal{C}_i$ em duas partes.

- $\mathcal{C}_i \cap \mathrm{DBall}(i, \delta\xi L_i)$. Como $i$ não é $\beta$-densa, vale que $(1-\xi)\ell_i|\mathrm{DBall}(i, \xi\ell_i)| \leq \beta$. Além disso, como $i$ não é determinada, então $\delta L_i \leq \ell_i$ e, consequentemente, $\mathrm{DBall}(i, \xi\delta L_i) \subseteq \mathrm{DBall}(i, \xi\ell_i)$. Assim,

  $$(1-\xi)\delta L_i|\mathrm{DBall}(i, \xi\delta L_i)| \leq (1-\xi)\ell_i|\mathrm{DBall}(i, \xi\ell_i)| \leq \beta.$$

  Multiplicando por $(1+\delta\xi)$ e dividindo por $\delta(1-\xi)$, temos que

  $$(1+\delta\xi)L_i|\mathrm{DBall}(i, \xi\delta L_i)| \leq \frac{1+\delta\xi}{\delta(1-\xi)}\beta \leq \frac{\beta}{\delta\xi}$$

  em que a última desigualdade vale pois $1 + \delta\xi \leq 2$ e $\xi = \frac{1}{3}$. Seja $j \in \mathcal{C}_i \cap \mathrm{DBall}(i, \delta\xi L_i)$. Pela desigualdade triangular vale que

  $$c(i', j) \leq c(i, j) + c(i, i') = c(i, j) + L_i < \delta\xi L_i + L_i = (1+\delta\xi)L_i.$$

  Portanto, a soma do custo de associar todos os clientes em $\mathcal{C}_i \cap \mathrm{DBall}(i, \delta\xi L_i)$ à instalação $i'$ é no máximo $|\mathcal{C}_i \cap \mathrm{DBall}(i, \delta\xi L_i)|(1+\delta\xi)L_i \leq \frac{\beta}{\delta\xi}$.

- $\mathcal{C}_i \setminus \mathrm{DBall}(i, \delta\xi L_i)$. Seja $j \in \mathcal{C}_i \setminus \mathrm{DBall}(i, \delta\xi L_i)$. Como, pela desigualdade triangular, $c(i', j) \leq c(i, j) + c(i, i') \leq c(i, j) + L_i$ e, pela escolha de $j$, $c(i, j) \geq \delta\xi L_i$, então

  $$\frac{c(i', j) - c(i, j)}{c(i, j)} \leq \frac{L_i}{\delta\xi L_i} = \frac{1}{\delta\xi}.$$

  Logo, conectar um cliente em $\mathcal{C}_i \setminus \mathrm{DBall}(i, \delta\xi L_i)$ à $i'$ aumenta o custo em uma razão de no máximo $\frac{1}{\delta\xi}$. Assim, associar todos os clientes em $\mathcal{C}_i \setminus \mathrm{DBall}(i, \delta\xi L_i)$ à $i'$ aumenta o custo em no máximo $\frac{C_i}{\delta\xi}$, que pela escolha de $i$ é no máximo $\frac{\text{custo}(\mathcal{T}')}{\delta\xi t}$.

Assim, juntando o aumento do custo de cada uma das partes, temos que

$$\begin{aligned}
\text{custo}(\mathcal{T}' \setminus \{i\}) &\leq \text{custo}(\mathcal{T}') + \frac{\beta + \text{custo}(\mathcal{T}')/t}{\delta\xi} \\
&< \text{custo}(\mathcal{T}') + \frac{\beta + (\text{custo}(\mathcal{T}) + dB)/t}{\delta\xi} \\
&= \text{custo}(\mathcal{T}') + \frac{\beta + \text{custo}(\mathcal{T})/t}{\delta\xi} + \frac{dB}{t\delta\xi} \\
&= \text{custo}(\mathcal{T}') + \frac{B}{2} + \frac{dB}{\delta\xi}\frac{1}{t} \\
&\leq \text{custo}(\mathcal{T}') + \frac{B}{2} + \frac{dB}{\delta\xi}\frac{\delta\xi}{2d} \leq \text{custo}(\mathcal{T}') + B.
\end{aligned}$$

Isso é um absurdo, pois conseguiríamos retirar $i$ de $\mathcal{T}'$ nas linhas 2-3 do algoritmo. Portanto, $|U_0| \leq t$ e, consequentemente, $|V_0| < t$. □

Para completar a prova do Lema 5.4, vamos mostrar o seguinte lema.

**Lema 5.9.** *Sejam $Q_0$ e $V_0$ os conjuntos definidos no parágrafo anterior ao Lema 5.7. Vale que*

$$\textit{custo}(S_{Q_0,V_0}) \leq \frac{1+3\delta}{1-3\delta}\,\text{opt}(I).$$

*Demonstração.* Seja $j$ um cliente em $\text{DBall}(i, L_i/3)$ e, seja $h$ uma instalação em $\text{FBall}(i, \delta L_i)$. Vale que

$$c(h,j) \leq c(i,j) + c(h,i) < \frac{L_i}{3} + \delta L_i = (\delta + \frac{1}{3})L_i.$$

Seja $i' \in Q_0 \setminus \{i\}$ e $h' \in \text{FBall}(i', \delta L_{i'})$. Vale que

$$\begin{aligned}
c(h',j) &\geq c(i,h') - c(i,j) \geq c(i,i') - c(h',i') - c(i,j) \\
&> c(i,i') - \delta L_{i'} - \frac{L_i}{3} \geq c(i,i') - \delta c(i,i') - \frac{c(i,i')}{3} \\
&= \left(\frac{2}{3} - \delta\right) c(i,i') \geq \left(\frac{2}{3} - \delta\right) L_i,
\end{aligned}$$

em que as duas primeira desigualdades valem pela desigualdade triangular e as outras valem pela definição de $L_i$ e de $L_{i'}$. Concluímos que

$$c(h,j) < \left(\frac{1}{3} + \delta\right) L_i < \left(\frac{2}{3} - \delta\right) L_i \leq c(h',j)$$

em que a segunda desigualdade vale pois $\delta < \frac{1}{8}$. Como, para todo $i \in Q_0$, vale que $f_i^* \in \text{FBall}(i, \delta L_i)$, então, para todo $i' \in Q_0 \setminus \{i\}$, temos que $c(f_i^*, j) < c(f_{i'}^*, j)$, em que $f_i^*$ é a instalação de $S^*$ mais próxima à $i$. Então para um cliente $j \in \text{DBall}(i, L_i/3)$ a instalação de $S^*$ mais próxima a ele será $f_i^*$ ou alguma instalação de $V_0$. Assim, para $j \in \bigcup_{i \in Q_0} \text{DBall}(i, L_i/3)$, vale que $c(j, S_{Q_0,V_0}) \leq c(j, S^*)$. Seja $j$ um cliente que não está em $\bigcup_{i \in Q_0} \text{DBall}(i, L_i/3)$. Caso a instalação de $S^*$ mais próxima a $j$ seja uma instalação em $V_0$, então $c(j, S_{Q_0,V_0}) \leq c(j, V_0) = c(j, S^*)$, uma vez que $V_0 \subseteq S_{Q_0,V_0}$. Caso contrário, então a instalação de $S^*$ mais próxima a $j$ é $f_i^*$ para algum $i \in Q_0$, uma vez que $V_0 = S^* \setminus \{f_i^* : i \in Q_0\}$ . Como $i \in Q_0$ é, por definição, determinado, então $f_i^* \in \text{FBall}(i, \delta L_i)$. Pela desigualdade triangular, temos que $c(i, j) \leq c(i, f_i^*) + c(f_i^*, j)$ e $c(j, f_i^*) \geq c(i, j) - c(i, f_i^*) > c(i, j) - \delta L_i$. Seja $f_i \in \text{FBall}(i, \delta L_i)$ a instalação escolhida na linha 10 do algoritmo para a instalação $i \in Q_0$. Pela desigualdade triangular, vale que $c(j, f_i) \leq c(i, j) + c(i, f_i) < c(i, j) + \delta L_i$. Assim,

$$\frac{c(f_i, j)}{c(f_i^*, j)} \leq \frac{c(i, j) + \delta L_i}{c(i, j) - \delta L_i} \leq \frac{L_i/3 + \delta L_i}{Li/3 - \delta L_i} = \frac{1 + 3\delta}{1 - 3\delta}.$$

É fácil ver que a segunda desigualdade vale uma vez que $c(i, j) \geq L_i/3$. Seja $S$ a solução devolvida na linha 12 e seja $\text{DB} \coloneqq \bigcup_{i \in Q_0} \text{DBall}(i, L_i/3)$. Portanto, vale que

$$\begin{aligned}
\text{custo}(S) \leq \text{custo}(S_{Q_0,V_0}) &= \sum_{j \in D} c(j, S_{Q_0,V_0}) \\
&= \sum_{j \in \text{DB}} c(j, S_{Q_0,V_0}) + \sum_{j \notin \text{DB}} c(j, S_{Q_0,V_0}) \\
&\leq \sum_{j \in \text{DB}} c(j, S^*) + \frac{1 + 3\delta}{1 - 3\delta} \sum_{j \notin \text{DB}} c(j, S^*) \\
&\leq \frac{1 + 3\delta}{1 - 3\delta} \text{opt}(I),
\end{aligned}$$

em que a última desigualdade vale uma vez que $\frac{1+3\delta}{1-3\delta} \geq 1$. □

Assim, está provado o Lema 5.4.

### 5.1.3 Combinando os algoritmos

Agora, utilizaremos os algoritmos mostrados nas Seções 5.1.1 e 5.1.2 para provar o Teorema 5.1. Suponha que temos um algoritmo $A$ que é uma $\alpha$-aproximação $d$-aditiva para o problema das $k$-medianas métrico. O seguinte algoritmo, parametrizado em $\epsilon$,

é uma $(\alpha+\epsilon)$-aproximação para o problema das $k$-medianas métrico com tempo de execução $O(n^{O(d/\epsilon)})$ vezes o tempo de execução de $A$.

**Algoritmo 16** PSEUDO-PARA-SOL$_\epsilon(F, D, c, k)$

1: Escolha o maior $\delta \in (0, 1/8)$ tal que $\frac{1+3\delta}{1-3\delta} \leq \alpha$
2: $t \leftarrow \frac{4}{\epsilon} \cdot \frac{\alpha d}{\xi\delta}$
3: $\mathcal{I} \leftarrow$ ESPARSA$((F, D, c, k), t)$
4: **Para** cada instância $I' \in \mathcal{I}$ **faça**
5: $\quad \mathcal{T}_{I'} \leftarrow A(I')$
6: $\quad S_{I'} \leftarrow$ PSEUDO-PARA-SOL-ESP$(I', \mathcal{T}_{I'}, t, \delta)$
7: $S \leftarrow \arg\min_{S_{I'}} \text{custo}(S_{I'})$
8: **Devolva** $S$

É fácil perceber que o tempo de execução desse algoritmo é $O(n^{O(d/\epsilon)})$ vezes o tempo de execução de $A$, uma vez que temos $n^{O(t)} = n^{O(d/\epsilon)}$ instâncias em $\mathcal{I}$ e, para cada uma delas, executamos o algoritmo $A$ uma vez. Além disso, executamos PSEUDO-PARA-SOL-ESP para cada solução devolvida por $A$. Cada uma dessas execuções toma tempo $O(t\, n^{O(t)}) = O(\frac{d}{\epsilon}\, n^{O(d/\epsilon)})$, o que totaliza $n^{O(d/\epsilon)} \cdot O(\frac{d}{\epsilon}\, n^{O(d/\epsilon)}) = O(\frac{d}{\epsilon}\, n^{O(d/\epsilon)})$.

Agora mostraremos que PSEUDO-PARA-SOL$_\epsilon$ é uma $(\alpha+\epsilon)$-aproximação para o problema das $k$-medianas métrico. Seja $I = (F, D, c, k)$. É evidente que o conjunto $S$ devolvido é uma solução viável para a instância $I$ do problema das $k$-medianas métrico. Precisamos mostrar que $\text{custo}(S) \leq (\alpha+\epsilon)\,\text{opt}(I)$. Seja $I'$ a instância que satisfaz as propriedades do Lema 5.3. Como $A$ é uma $\alpha$-aproximação $d$-aditiva para o problema das $k$-medianas métrico, vale que $\text{custo}(\mathcal{T}_{I'}) \leq \alpha\,\text{opt}(I') = \alpha\,\text{opt}(I)$. Como vale que $I'$ é uma instância $\text{opt}(I)/t$-esparsa, $\delta \in (0, 1/8)$ e $t = \lceil \frac{4}{\epsilon} \cdot \frac{\alpha d}{\xi\delta} \rceil \geq \frac{2d}{\delta\xi}$, então, pelo Lema 5.4, vale que

$$\begin{aligned}
\text{custo}(S) &\leq \text{custo}(S_{I'}) \\
&\leq \max\left\{\text{custo}(\mathcal{T}_{I'}) + d \cdot 2\frac{\text{opt}(I) + \text{custo}(\mathcal{T}_{I'})}{t\xi\delta}, \frac{1+3\delta}{1-3\delta}\text{opt}(I)\right\} \\
&\leq \max\left\{\alpha\,\text{opt}(I) + \text{opt}(I)\frac{4\alpha d}{t\xi\delta}, \alpha\,\text{opt}(I)\right\} \\
&\leq \max\left\{(\alpha+\epsilon)\,\text{opt}(I), \alpha\,\text{opt}(I)\right\} \leq (\alpha+\epsilon)\,\text{opt}(I).
\end{aligned}$$

Assim, o Teorema 5.1 está provado.

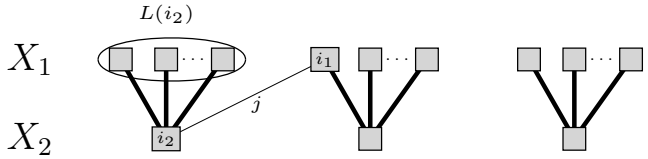
## 5.2 Uma aproximação aditiva

Nessa seção vamos apresentar um algoritmo que é uma $(1+\sqrt{3}+\epsilon)$ aproximação $O(1/\epsilon)$-aditiva desenvolvida por Li e Svensson para o problema das $k$-medianas métrico. Esse algoritmo faz o papel do algoritmo $A$ presente na linha 5 do **Algoritmo 16**.

Vamos relembrar o algoritmo BuscaBinária$_\epsilon$ (Algoritmo 11), que recebe uma instância $I = (F, D, c, k)$ do problema das $k$-medianas métrico e devolve dois conjuntos $X_1, X_2 \subseteq F$. Se $|X_2| = 0$, então $X_1$ é uma solução com custo no máximo duas vezes $\text{opt}(I)$. Se $|X_2| \neq 0$, então $|X_1| > k > |X_2|$ e, pelo Lema 4.6, para $a \coloneqq \frac{k-|X_2|}{|X_1|-|X_2|}$ e $b \coloneqq \frac{|X_1|-k}{|X_1|-|X_2|}$, vale que $a\,\text{custo}(X_1) + b\,\text{custo}(X_2) \leq 2(1+\epsilon)\,\text{opt}(I)$.

Sejam $(x_1, y_1)$ e $(x_2, y_2)$ soluções (não necessariamente viáveis) do programa linear relaxado (PL) que representam $X_1$ e $X_2$, respectivamente. Chamamos de solução *bi-point* a solução $(ax_1 + bx_2, ay_1 + by_2)$, ou seja, a solução que é combinação convexa de $(x_1, y_1)$ e $(x_2, y_2)$ com coeficientes $a$ e $b$. É fácil mostrar que essa solução é realmente viável para o programa linear relaxado. É evidente que o valor objetivo dessa solução é igual a $a\,\text{custo}(X_1) + b\,\text{custo}(X_2)$ que por sua vez é no máximo $2(1+\epsilon)\text{opt}(I)$. Vamos utilizar essa solução para construir uma solução inteira $O(1/\epsilon)$-aditiva.

É conveniente pensarmos na solução bi-point como um grafo $(X_1, X_2)$-bipartido em que para cada cliente $j$ temos uma aresta que liga $i_1(j)$ e $i_2(j)$, em que $i_1(j)$ é uma instalação de $X_1$ mais próxima a $j$ e $i_2(j)$ é uma instalação de $X_2$ mais próxima a $j$. Seja $\phi : X_1 \to X_2$ tal que $\phi(i_1) \coloneqq \arg\min_{i_2 \in X_2} c(i_1, i_2)$, ou seja, a função que mapeia cada instalação em $X_1$ a uma instalação em $X_2$ mais próxima a ela. Vamos particionar os elementos de $X_1$ com base na função $\phi$. Para cada $i_2 \in X_2$, criamos uma parte $L(i_2) \coloneqq \{i_1 \in X_1 : \phi(i_1) = i_2\}$. Vamos nos referir a $L(i_2)$ como as folhas de uma estrela com centro $i_2$.




A figura acima é uma representação do grafo bipartido associado à solução bi-point. As arestas grossas são os clientes que formam as arestas da estrela. A aresta fina representa um cliente $j$ tal que $i_1(j) \notin L(i_2(j))$.

**Algoritmo 17** PSEUDO$_\epsilon(I = (F, D, c, k))$

1: $\epsilon' \leftarrow \frac{\epsilon}{3(1+\sqrt{3})}$
2: $(X_1, X_2) \leftarrow$ BUSCABINÁRIA$_{\epsilon'}(F, D, c, k)$
3: **Se** $|X_2| = 0$ **então**
4: **Devolva** $X_1$
5: $a \leftarrow \frac{k-|X_2|}{|X_1|-|X_2|}$; $b \leftarrow \frac{|X_1|-k}{|X_1|-|X_2|}$
6: **Se** $b > \frac{2}{1+\sqrt{3}}$ **então**
7: **Devolva** $X_2$
8: **Se** $b \leq \frac{\sqrt{3}-1}{4}$ **então**
9: **Devolva** B-PEQUENO$(I, X_1, X_2, a, b)$
10: **Senão**
11: **Devolva** B-MÉDIO$_\epsilon(I, X_1, X_2, a, b)$

As funções B-PEQUENO e B-MÉDIO serão definidas posteriormente junto com as suas explicações.

Seja $X$ a solução devolvida por PSEUDO$_\epsilon$ utilizando a instância $I = (F, D, c, k)$ como entrada. Vamos mostrar que custo$(X) \leq (1 + \sqrt{3} + \epsilon)$opt$(I)$ e que $|X| \leq k + O(1/\epsilon)$.

Se $X$ foi devolvido na linha 3, então custo$(X) \leq 2\,$opt$(I)$ e $|X| = k$. Caso $X$ não tenha sido devolvido na linha 3, o algoritmo segue caminhos diferentes dependendo do valor de $b$. Vamos então descrever as funções não definidas e explicar o algoritmo para cada um dos intervalos delimitados para o valor de $b$.

Note que a função BUSCABINÁRIA foi utilizada com parâmetro $\epsilon' = \frac{\epsilon}{3(1+\sqrt{3})}$. Assim, vale que

$$a\,\text{custo}(X_1) + b\,\text{custo}(X_2) \leq 2(1+\epsilon')\text{opt}(I) = 2(1 + \frac{\epsilon}{3(1+\sqrt{3})})\text{opt}(I).$$

**Caso** $b > \frac{2}{1+\sqrt{3}}$ :

Neste caso, devolvemos $X$ na linha 6 do algoritmo e $X = X_2$. Portanto, $|X| = |X_2| < k$. Além disso,

$$\begin{aligned}
\text{custo}(X_2) &\leq \frac{1}{b}(a\,\text{custo}(X_1) + b\,\text{custo}(X_2)) \\
&\leq \frac{2}{b}(1 + \frac{\epsilon}{3(1+\sqrt{3})})\,\text{opt}(I) \\
&< (1+\sqrt{3})(1 + \frac{\epsilon}{1+\sqrt{3}})\text{opt}(I) = (1 + \sqrt{3} + \epsilon)\text{opt}(I).
\end{aligned}$$

**Caso** $b \leq \frac{\sqrt{3}-1}{4}$ :

Neste caso, executamos a função B-PEQUENO, onde a escolha aleatória da linha 16 é feita de acordo com a distribuição uniforme.

**Algoritmo 18** B-PEQUENO$(I, X_1, X_2, a, b)$

1: $X \leftarrow X_2$
2: $X_2' \leftarrow X_2$
3: **Para** cada $i \in X_2'$ tal que $|L(i)| = 1$ **faça**
4: $\quad X \leftarrow X \cup \{L(i)\} \setminus \{i\}$
5: $\quad X_2' \leftarrow X_2' \setminus \{i\}$
6: cap $\leftarrow k - |X|$
7: **Enquanto** $X_2' \neq \emptyset$ **faça**
8: $\quad$ **Se** cap $\leq 0$ **então**
9: $\quad\quad$ **Devolva** $X$
10: $\quad i \leftarrow \arg\min_{i \in X_2'} \sum_{j \in \delta(L(i))} \frac{c(j,X_1)+c(j,X_2)}{|L(i)|-1}$
11: $\quad X_2' \leftarrow X_2' \setminus \{i\}$
12: $\quad$ **Se** $|L(i)| - 1 \leq$ cap **então**
13: $\quad\quad X \leftarrow X \cup L(i) \setminus \{i\}$
14: $\quad\quad$ cap $\leftarrow$ cap $- (|L(i)| - 1)$
15: $\quad$ **Senão**
16: $\quad\quad$ Seja $R \subseteq L(i)$ com $|R| = \left\lceil \frac{\text{cap}}{|L(i)|-1} \cdot |L(i)| \right\rceil$ escolhido aleatoriamente
17: $\quad\quad X \leftarrow X \cup R$
18: $\quad\quad$ cap $\leftarrow$ cap $- (|L(i)| - 1)$
19: **Devolva** $X$

O conjunto $\delta(L(i))$ representa todos os clientes $j$ tais que $i_1(j) \in L(i)$. Durante o algoritmo, consideramos que $X$ é o conjunto de instalações abertas. O algoritmo tem como invariante a seguinte propriedade: se a instalação $i \in X_2$ estiver fechada, ou seja, $i \notin X$, então todas as instalações em $L(i)$ estão abertas, ou seja, $L(i) \subseteq X$. Note que isso é verdade, uma vez que inicializamos $X$ como $X_2$, e sempre que tiramos uma instalação $i \in X_2$ de $X$ colocamos em $X$ todas as instalações de $L(i)$.

A ideia quando o valor de $b$ é pequeno é que o tamanho de $X_1$ é próximo de $k$, logo podemos abrir muitas instalações de $X_1$. Vamos começar com $X_2$ como solução inicial e vamos escolher, quase gulosamente, estrelas para trocar o seu centro pelas suas folhas.

Seja $j$ um cliente qualquer e $i \in X_2$ tal que $i_1(j) \in L(i)$. O custo de conexão do

cliente $j$ é $c(j, X_1)$ caso $i_1(j)$ esteja aberta e no máximo

$$\begin{aligned} c(i,j) &\leq c(i, i_1(j)) + c(i_1(j), j) \\ &\leq c(i_1(j), i_2(j)) + c(i_1(j), j) \\ &\leq 2\, c(i_1(j), j) + c(i_2(j), j) = 2\, c(j, X_1) + c(j, X_2) \end{aligned}$$

caso contrário, em que a primeira e a terceira desigualdade valem pela desigualdade triangular e a segunda desigualdade vale pela definição de $L(i)$. Assim, toda vez que escolhemos uma estrela cujo centro $i$ será fechado e cujas folhas serão abertas, diminuímos o limitante superior do custo da solução em $\sum_{j \in \delta(L(i))}(c(j, X_1) + c(j, X_2))$. Assim, podemos formular um programa linear para escolher para quais estrelas fecharemos o centro e abriremos as folhas, a fim de maximizar o quanto podemos diminuir esse limitante superior a partir da solução inicial $X_2$.

$$\begin{aligned} \text{Maximize} \quad & \sum_{i \in X_2} \sum_{j \in \delta(L(i))} (c(j, X_1) + c(j, X_2)) x_i \\ \text{sujeito a} \quad & \sum_{i \in X_2} x_i (|L(i)| - 1) \leq k - |X_2| \\ & 0 \leq x_i \leq 1, \quad \forall\, i \in X_2. \end{aligned}$$

Intuitivamente, $x_i$ tem valor 1 se fechamos $i$ e abrimos todas as instalações em $L(i)$, e 0 se mantemos $i$ aberta. Note que $x_i \coloneqq a$ para todo $i \in X_2$ é uma solução viável, uma vez que $\sum_{i \in X_2} a(|L(i)| - 1) = a(|X_1| - |X_2|) = k - |X_2|$ e $0 \leq a \leq 1$. Assim, uma solução ótima desse programa tem custo pelo menos $a(\text{custo}(X_1) + \text{custo}(X_2))$.

Note que esse programa linear é equivalente ao programa linear do problema da mochila fracionária em que cada instalação $i \in X_2$ é um objeto com valor $\sum_{j \in \delta(L(i))} c(j, X_1) + c(j, X_2)$ e peso $|L(i)| - 1$, e a capacidade da mochila é $k - |X_2|$. Portanto, existe uma solução ótima para ele com no máximo uma variável fracionária. Nas linhas $3-5$ tratamos as estrelas com apenas uma folha, trocamos o seu centro pela sua folha, uma vez que elas são objetos sem custos. Nas linhas $7-18$ de B-PEQUENO, seguindo a estratégia gulosa, construímos, implicitamente, uma solução ótima desse tipo. Suponha que estamos na iteração para a instalação $i$ do laço **Enquanto** da linha 7. Se a condicional da linha 8 é satisfeita, então temos $x_i = 0$ para $i$ e para todas as instalações que sobraram em $X_2'$. Caso a condicional da linha 12 seja satisfeita então $x_i = 1$. Caso nenhuma das duas condicionais sejam satisfeitas, então $x_i$ é uma variável fracionária. Note que só teremos uma variável fracionária, uma vez que se entramos no **Senão** da

linha 15, atualizamos cap para um valor negativo, uma vez que cap $< |L(i)| - 1$ e, após isso, a condição da linha 8 será satisfeita e o algoritmo devolve $X$. Evidentemente, se $x_i = 1$, então fechamos $i$ e abrimos todas as instalações em $L(i)$. Se $x_i = 0$, então mantemos $i$ aberta. Se $x_i$ é fracionária, então mantemos $i$ aberta e abrimos $\lceil x_i|L(i)|\rceil$ instalações de $L(i)$ aleatoriamente. Assim, abrimos instalações de modo que a redução no custo esperado da solução inicial $X_2$ é pelo menos o valor da solução ótima desse programa linear. Portanto,

$$\begin{aligned}\text{custo}(X) &\leq 2\,\text{custo}(X_1) + \text{custo}(X_2) - a(\text{custo}(X_1) + \text{custo}(X_2)) \\ &= (1+b)\text{custo}(X_1) + b\,\text{custo}(X_2).\end{aligned}$$

Além disso, $|X| \leq k+2$, onde o termo aditivo 2 vem da estrela com valor de $x$ fracionário. Podemos assumir que $\text{custo}(X_1) \leq \text{custo}(X_2)$, então

$$\begin{aligned}\text{custo}(X) &\leq (1+b)\text{custo}(X_1) + b\,\text{custo}(X_2) \\ &\leq (1+2b)(a\,\text{custo}(X_1) + b\,\text{custo}(X_2)) \\ &\leq \frac{1+\sqrt{3}}{2}(a\,\text{custo}(X_1) + b\,\text{custo}(X_2)) \\ &\leq (1+\sqrt{3})(1+\frac{\epsilon}{3(1+\sqrt{3})})\text{opt}(I) \\ &< (1+\sqrt{3}+\epsilon)\,\text{opt}(I),\end{aligned}$$

em que a terceira desigualdade vale uma vez que $b \leq \frac{\sqrt{3}-1}{4}$.

**Caso** $\frac{\sqrt{3}-1}{4} < b \leq \frac{2}{1+\sqrt{3}}$ :

No caso em que $b$ tem valor no intervalo intermediário, o algoritmo B-MÉDIO abaixo é executado. Novamente as escolhas aleatórias são feitas de acordo com a distribuição uniforme e os diferentes sorteios são independentes um do outro.

A ideia é que queremos um arredondamento probabilístico em que abrimos uma instalação de $X_1$ com probabilidade aproximadamente $a$ e uma instalação de $X_2$ com probabilidade aproximadamente $b$ enquanto mantemos a seguinte propriedade: se a instalação $i \in X_2$ estiver fechada, então todas as instalações em $L(i)$ estão abertas.

O algoritmo classifica as estrelas em pequenas e grandes. Dizemos que uma estrela com centro $i \in X_2$ é *grande* se $|L(i)| \geq 2/(ab\eta)$ e *pequena*, caso contrário. O conjunto $B$ na linha 2 contém o centro das estrelas grandes. Além disso, na linha 3, o algoritmo particiona as estrelas pequenas em $\lceil 2/(ab\eta)\rceil$ partes, juntando estrelas com o mesmo

**Algoritmo 19** B-MÉDIO$_\epsilon(I, X_1, X_2, a, b)$

1: $\eta \leftarrow \frac{\epsilon}{3(1+\sqrt{3})}$
2: $B \leftarrow \{i \in X_2 : |L(i)| \geq \frac{2}{ab\eta}\}$
3: $\mathcal{U}_h \leftarrow \{i \in X_2 : |L(i)| = h\}$ para todo $h = 0, 1, \ldots, \left\lceil \frac{2}{ab\eta} \right\rceil - 1$
4: $X \leftarrow \emptyset$
5: **Para** cada $i \in B$ **faça**
6: $\quad X \leftarrow X \cup \{i\}$
7: $\quad$ Seja $R \subseteq L(i)$ com $|R| = \lfloor a(|L(i)| - 1) \rfloor$ escolhido aleatoriamente
8: $\quad X \leftarrow X \cup R$
9: **Para** cada $h = 0, 1, \ldots, \left\lceil \frac{2}{ab\eta} \right\rceil - 1$ **faça**
10: $\quad$ Seja $R \subseteq \mathcal{U}_h$ com $|R| = \lceil b|\mathcal{U}_h| \rceil + 1$ escolhido aleatoriamente
11: $\quad X \leftarrow X \cup R$
12: $\quad$ **Para** cada $i \in \mathcal{U}_h \setminus R$ **faça**
13: $\quad\quad X \leftarrow X \cup L(i)$
14: $\quad \ell \leftarrow ah|\mathcal{U}_h| - \sum_{i \in \mathcal{U}_h \setminus R} |L(i)|$
15: $\quad$ Seja $p$ um número entre 0 e 1 escolhido aleatoriamente
16: $\quad$ **Se** $p \leq \lceil \ell \rceil - \ell$ **então**
17: $\quad\quad$ Seja $Z \subseteq \cup_{i \in R} L(i)$ com $|Z| = \lfloor \ell \rfloor$ escolhido aleatoriamente
18: $\quad$ **Senão**
19: $\quad\quad$ Seja $Z \subseteq \cup_{i \in R} L(i)$ com $|Z| = \lceil \ell \rceil$ escolhido aleatoriamente
20: $\quad X \leftarrow X \cup Z$
21: **Devolva** $X$

tamanho.

Executamos as linhas $5 - 8$ para as estrelas grandes. Para cada estrela grande com centro $i$, abrimos $i$ e $\lfloor b(|L(i)| - 1) \rfloor$ instalações em $L(i)$ aleatoriamente.

Para as estrelas pequenas, executamos as linhas $9 - 20$. Seja $\mathcal{U}_h$ o grupo de estrelas pequenas de tamanho $h$. Abrimos o centro de $\lceil b|\mathcal{U}_h| \rceil + 1$ estrelas escolhidas aleatoriamente de $\mathcal{U}_h$ e abrimos todas as folhas das estrelas restantes de $\mathcal{U}_h$. Além disso, sendo $\ell$ o número de folhas abertas até o momento subtraído de $ah|\mathcal{U}_h|$, abrimos, das $\lceil b|\mathcal{U}_h| \rceil + 1$ estrelas escolhidas aleatoriamente, $\lfloor \ell \rfloor$ ou $\lceil \ell \rceil$ folhas escolhidas aleatoriamente, com probabilidade $\lceil \ell \rceil - \ell$ e $\ell - \lfloor \ell \rfloor$ respectivamente.

Note que, para uma estrela grande, o algoritmo sempre abre o centro da estrela e quase uma fração $a$ das suas folhas. Para um grupo $\mathcal{U}_h$ de estrelas pequenas, ele abre o centro de uma estrela com probabilidade pelo menos $b$ ou todas as suas folhas. Além disso, abrimos mais algumas folhas para que o número esperado de folhas abertas seja uma fração $a$ do número de folhas de $\mathcal{U}_h$.

Temos que mostrar que, para $X$ devolvido por B-MÉDIO, o tamanho de $X$ não é

muito maior que $k$ e que podemos limitar o custo de $X$ superiormente.

**Lema 5.10.** *Seja $X$ devolvido por* B-MÉDIO. *Vale que $|X| \leq k + 3\lceil 2/(ab\eta)\rceil$.*

*Demonstração.* Lembre que $a|X_1| + b|X_2| = k$. Portanto,

$$\sum_{i \in X_2} (a|L(i)| + b) = k. \tag{34}$$

Seja $i \in X_2$ o centro de uma estrela grande. Para essa estrela, o algoritmo abre $1 + \lfloor a(|L(i)| - 1)\rfloor \leq 1 + a(|L(i)| - 1) = b + a|L(i)|$ instalações.

Agora considere um grupo $\mathcal{U}_h$ de pequenas estrelas e seja $m \coloneqq |\mathcal{U}_h|$. Para esse grupo, o algoritmo abre $\lceil bm\rceil + 1 \leq bm + 2$ instalações em $X_2$ nas linhas 10 e 11. Além disso abre no máximo

$$(m - \lceil bm\rceil - 1)h + \lceil ahm - (m - \lceil bm\rceil - 1)h\rceil \leq ahm + 1$$

instalações de $X_1$, em que o primeiro termo da soma vem das linhas 12 e 13 e o segundo termo da soma vem das linhas $14 - 19$. Note que $(m - \lceil bm\rceil - 1)h = \sum_{i \in \mathcal{U}_h \setminus R} |L(i)|$, uma vez que $|L(i)| = h$ para todas essas estrelas. Assim, o total de instalações abertas para cada grupo $\mathcal{U}_h$ de estrelas pequenas é no máximo $m(b + ah) + 3$. Assim,

$$\begin{aligned} |X| &\leq \sum_{i \in B} b + a|L(i)| + \sum_{h=0}^{\lceil 2/(ab\eta)\rceil} (|\mathcal{U}_h|(b + ah) + 3) \\ &= \sum_{i \in X_2} (b + a|L(i)|) + 3\lceil 2/(ab\eta)\rceil \\ &= k + 3\lceil 2/(ab\eta)\rceil, \end{aligned}$$

em que a primeira igualdade vale pela definição de $\mathcal{U}_h$ e a segunda desigualdade vale por (34). □

Agora precisamos limitar o custo de $X$.

**Lema 5.11.** *Seja $X$ o conjunto devolvido por* B-MÉDIO. *O custo esperado de $X$ é no máximo*

$$(1 + \eta)(b\,\text{custo}(X_2) + a(1 + 2b)\text{custo}(X_1)).$$

*Demonstração.* Vamos fixar um cliente $j$ e definir $i_1 \coloneqq i_1(j)$ e $i_2 \coloneqq i_2(j)$. Seja $i \in X_2$ tal que $i_1 \in L(i)$. Note que $c(i_1, i) \leq c(i_1, i_2) \leq c(i_1, j) + c(i_2, j)$, pela definição de $i$ e pela desigualdade triangular. Assim, $c(i, j) \leq c(i, i_1) + c(i_1, j) \leq 2c(i_1, j) + c(i_2, j)$.

Se $i_1$ for aberto, ou seja, se $i_1 \in X$, então o custo de conexão de $j$ é $c(i_1, j)$. Se $i_1$ estiver fechado e $i_2$ for aberto, ou seja, se $i_1 \notin X$ mas $i_2 \in X$, então o custo de conexão de $j$ é $c(i_2, j)$. Se ambas estiverem fechadas, então o custo de conexão de $j$ é $c(i, j) \leq 2c(i_1, j) + c(i_2, j)$. Vamos, por abuso de notação, indicar por $i$ o evento em que a instalação $i$ está aberta e $\bar{i}$ o evento em que a instalação $i$ está fechada. Assim, o custo de conexão esperado para $j$ é no máximo

$$\Pr[i_1] \cdot c(i_1, j) + \Pr[\bar{i_1} i_2] c(i_2, j) + \Pr[\bar{i_1}\bar{i_2}](2c(i_1, j) + c(i_2, j))$$

que, substituindo $\Pr[\bar{i_1}\bar{i_2}]$ por $1 - \Pr[i_1] - \Pr[\bar{i_1} i_2]$, é igual a

$$(2 - \Pr[i_1] - 2\Pr[\bar{i_1} i_2])c(i_1, j) + (1 - \Pr[i_1])c(i_2, j). \tag{35}$$

Vamos limitar essa expressão analisando essas probabilidades separadamente.

Vamos começar analisando o valor de $\Pr[\bar{i_1} i_2]$. Se $i_1 \in L(i_2)$, ou seja, $i_2 = i$, então $i_2$ sempre estará aberto se $i_1$ estiver fechado, assim $\Pr[\bar{i_1} i_2] = \Pr[\bar{i_1}] = 1 - \Pr[i_1]$. O mesmo vale se a estrela com centro $i_2$ for uma estrela grande, uma vez que sempre abrimos os centros das estrelas grandes.

Vamos considerar então o caso em que a estrela com centro $i_2$ é uma estrela pequena do grupo $\mathcal{U}_h$ com $m \coloneqq |\mathcal{U}_h|$ e $i_2 \neq i$. Note que se a estrela com centro $i$ for uma estrela grande ou uma estrela pequena com tamanho diferente de $h$, então os eventos $i_2$ e $\bar{i}_1$ são independentes. Nesse caso, vale que

$$\Pr[\bar{i_1} i_2] = \Pr[i_2] \cdot (1 - \Pr[i_1]) = \frac{\lceil bm \rceil + 1}{m} \cdot (1 - \Pr[i_1]) > b(1 - \Pr[i_1]).$$

Resta apenas o caso em que a estrela com centro $i$ é uma estrela do grupo $\mathcal{U}_h$. Assim,

$$\begin{aligned}
\Pr[\bar{i_1} i_2] &= \Pr[i_2 | \bar{i_1}] \cdot (1 - \Pr[i_1]) \\
&= \frac{\lceil bm \rceil}{m - 1} \cdot (1 - \Pr[i_1]) \\
&\geq \frac{\lceil bm \rceil}{m} \cdot (1 - \Pr[i_1]) \geq b(1 - \Pr[i_1]),
\end{aligned}$$

em que a segunda igualdade vale uma vez que se $i_1$ está fechada, então $i$ está aberta sobrando escolher $\lceil bm \rceil$ estrelas dentre $m - 1$ estrelas para abrir seus centros. Portanto, $\Pr[\bar{i_1} i_2]$ é sempre pelo menos $b(1 - \Pr[i_1])$. Substituindo esse valor em (35), temos que

o custo de conexão esperado de $j$ é no máximo

$$(2a + (2b-1)\Pr[i_1])c(i_1, j) + (1 - \Pr[i_1])c(i_2, j). \tag{36}$$

Agora analisaremos o valor de $\Pr[i_1]$. Se $i_1$ é folha de uma estrela grande, então

$$\begin{aligned}\Pr[i_1] &= \frac{\lfloor a(|L(i)|-1)\rfloor}{|L(i)|} = \frac{\lfloor a|L(i)| - a\rfloor}{|L(i)|} \\ &\geq \frac{a|L(i)| - a - 1}{|L(i)|} = a - \frac{a+1}{|L(i)|} \\ &\geq a - \frac{2}{|L(i)|} \geq a(1 - b\eta)\end{aligned}$$

em que a última desigualdade vale pela definição de estrelas grandes.

Se $i_1$ é uma folha de uma estrela pequena do grupo $\mathcal{U}_h$ com $m \coloneqq |\mathcal{U}_h|$, então a quantidade esperada de folhas abertas em $\mathcal{U}_h$ é exatamente $amh$. Assim, $\Pr[i_1] = a$. Desse modo, temos que $a(1 - b\eta) \leq \Pr[i_1] \leq a$. Como $(1+\eta)\cdot(1-b\eta) \geq 1$, podemos atualizar o limitante superior para o custo esperado de $j$ em (36) para

$$(1+\eta)((2a + (2b-1)a)c(i_1, j) + (1-a)c(i_2, j))$$

que é igual a

$$(1+\eta)(a(1+2b)c(i_1, j) + bc(i_2, j)).$$

Somando esse limitante superior para todos os clientes, temos o custo esperado para a solução $X$. □

Completamos a análise balanceando a solução obtida com a solução trivial $X_2$.

**Lema 5.12.** *Seja $d_1 \coloneqq \text{custo}(X_1)$ e $d_2 \coloneqq \text{custo}(X_2)$. Vale que*

$$\min\{d_2, bd_2 + a(1+2b)d_1\} \leq \frac{1+\sqrt{3}}{2}(ad_1 + bd_2).$$

*Demonstração.* Mudaremos $d_1$ e $d_2$ simultaneamente para que o valor $ad_1 + bd_2$ não se altere. Escolhemos diminuir um entre $d_1$ e $d_2$ e aumentar o outro, a fim de aumentar o lado esquerdo da desigualdade. Essa operação pode ser aplicada até que uma das três condições sejam verdadeiras:

1. $d_1 = 0$;
2. $d_2 = 0$;

3. $d_2 = bd_2 + a(1+2b)d_1$.

Para os dois primeiros casos é evidente que a desigualdade é verdadeira. No terceiro caso, temos que $d_2 = (1+2b)d_1$. Assim $\frac{d_2}{ad_1+bd_2} = \frac{1+2b}{b(1+2b)+1-b} = \frac{1+2b}{1+2b^2} \leq \frac{1+\sqrt{3}}{2}$, em que o último valor é obtido quando $b = \frac{\sqrt{3}-1}{2}$. Como aumentamos os valores do lado esquerdo da desigualdade, então a desigualdade também é válida para os valores iniciais. □

Agora, vamos limitar o custo da solução devolvida por B-MÉDIO.

**Lema 5.13.** *Seja $X$ devolvido por* B-MÉDIO$_\epsilon$. *O custo esperado de $X$ é no máximo* $(1+\sqrt{3}+\epsilon)\text{opt}(I)$.

*Demonstração.* Seja $d_1 \coloneqq \text{custo}(X_1)$ e $d_2 \coloneqq \text{custo}(X_2)$. Suponha que $d_1 > b\,d_2 + a(1+2b)d_1$. Portanto, pelo Lema 5.11 o custo esperado de $X$ é no máximo

$$\begin{aligned}
(1+\eta)(b\,d_2 + a(1+2b)d_1) &\leq (1+\eta)\frac{1+\sqrt{3}}{2}(ad_1+bd_2) \\
&\leq (1+\eta)(1+\sqrt{3})(1+\epsilon')\text{opt}(I) \\
&= (1+\sqrt{3})(1+\eta)^2\text{opt}(I) \\
&= (1+\sqrt{3}+(2\eta+\eta^2)(1+\sqrt{3}))\text{opt}(I) \\
&< (1+\sqrt{3}+3\eta(1+\sqrt{3}))\text{opt}(I) \\
&= (1+\sqrt{3}+\epsilon)\text{opt}(I).
\end{aligned}$$

em que a primeira desigualdade vale pelo Lema 5.12. □

# 6 Conclusão

Notamos que os algoritmos de aproximação para os problemas de clustering abrangem métodos de diferentes naturezas.

O problema dos $k$-centros métrico, por ter uma função objetivo mais comportada, consegue alcançar sua melhor razão de aproximação com algoritmos não muito complexos, como é o caso do algoritmo guloso.

O problema de localização de instalações métrico beneficia-se de muitos métodos já conhecidos, entre eles os métodos de busca local, guloso e os baseados em programação linear. O método com a melhor razão de aproximação que estudamos foi o guloso, que é equivalente ao método de dual fitting — este último baseado em programação linear. Outros algoritmos, que combinam métodos apresentados aqui, conseguem atingir

melhores razões de aproximação. O melhor resultado conhecido foi encontrado por Shi Li [20] e corresponde a uma 1.488-aproximação. Em contrapartida, é *NP*-difícil aproximar esse problema com razão de aproximação melhor que 1.463. Nota-se que a diferença entre a melhor razão de aproximação possível e a melhor encontrada não é tão grande.

Por outro lado, as razões de aproximação do melhor algoritmo possível para o problema das $k$-medianas métrico e do melhor algoritmo conhecido ainda estão muito distantes. O algoritmo com a melhor razão de aproximação conhecida é uma 2.675-aproximação, desenvolvida por Byrka *et al.* [3], que utiliza diretamente as ideias introduzidas por Li e Svensson [21]. Observamos que, embora os problemas das $k$-medianas e de localização de instalações sejam parecidos, a restrição de abrir no máximo $k$ instalações aumenta substancialmente a dificuldade do problema. Ainda assim, ambos os problemas têm muitos algoritmos que compartilham os mesmos métodos.

# Referências


[1] Vijay Arya, Naveen Garg, Rohit Khandekar, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for $k$-median and facility location problems. *SIAM Journal on Computing*, 33, 01 2003.

[2] Jaroslaw Byrka and Karen Aardal. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM Journal on Computing*, 39(6):2212–2231, 2010.

[3] Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for $k$-median and positive correlation in budgeted optimization. *ACM Trans. Algorithms*, 13(2), mar 2017.

[4] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for facility location problems. *SIAM Journal on Computing*, 34(4):803–824, 2005.

[5] Fabián A. Chudak and David B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Compututing*, 33:1–25, 2003.

[6] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, jul 1998.

[7] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, first edition, 1979.

[8] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[9] Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.

[10] Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554, 2008.

[11] Dorit Hochbaum and David Shmoys. A best possible heuristic for the $k$-center problem. *Mathematics of Operations Research (MOR)*, 10:180–184, 05 1985.

[12] Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *J. ACM*, 33(3):533–550, 1986.

[13] Wen-Lian Hsu and George L. Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1(3):209–215, 1979.

[14] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *J. ACM*, 50(6):795–824, November 2003.

[15] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, STOC'02, page 731–740, New York, NY, USA, 2002. Association for Computing Machinery.

[16] Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, mar 2001.

[17] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[18] Leonid G. Khachiyan. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk SSSR*, 1979.

[19] Alfred A. Kuehn and Michael J. Hamburger. A heuristic program for locating warehouses. *Management Science*, 9(4):643–666, 1963.

[20] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013. 38th International Colloquium on Automata, Languages and Programming (ICALP 2011).

[21] Shi Li and Ola Svensson. Approximating $k$-median via pseudo-approximation. *SIAM Journal on Computing*, 45(2):530–547, 2016.

[22] Vijay V. Vazirani. *Approximation Algorithms.* Springer, 2001.

[23] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms.* Cambridge University Press, 2011.