Problema dos k-centros e variantes

Samuel Plaça de Paula

Texto apresentado
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Mestrado em Ciência da Computação
Orientadora: Prof.ª Dr.ª Cristina Gomes Fernandes

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CAPES.

São Paulo, setembro de 2015

Problema dos k-centros e variantes

Esta é a versão original da dissertação elaborada pelo
candidato (Samuel Plaça de Paula), tal como
submetida à Comissão Julgadora.

Resumo

Samuel Plaça de Paula. **Problema dos k-centros e variantes**. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2014.

Problemas de clustering surgem nas mais variadas áreas do conhecimento, como biologia, economia ou linguística. Sua forma geral consiste no seguinte: dado um conjunto de elementos comparáveis entre si por algum tipo de distância ou semelhança (correlação), particionar o conjunto de maneira que cada partição agrupe elementos semelhantes uns aos outros. Geralmente, o problema de determinar clusters, ou aglomerações, representa algum tipo de otimização logística ou a identificação de categorias. Existem diversas maneiras de expressar esses objetivos como problemas de otimização combinatória. Muitas das formulações mais úteis, no entanto, são NP-difíceis. Isso significa que não é possível resolver esses problemas de maneira eficiente a menos que P = NP. Uma das maneiras de lidar com isso é o projeto de algoritmos de aproximação. Um algoritmo de aproximação para um problema de otimização é um algoritmo eficiente que encontra soluções viáveis cujo valor está a um determinado fator do valor ótimo. Esse fator é chamado de garantia de desempenho. Neste trabalho, investigamos uma série de problemas de clustering NP-difíceis. Começamos estudando o problema dos k-centros, também conhecido como k-*center*, um problema clássico para o qual Gonzalez apresentou em 1985 um algoritmo de aproximação com a melhor garantia de desempenho possível sob a hipótese de que P $\neq$ NP. Exploramos, em seguida, resultados disponíveis na literatura para diversas generalizações dos k-centros. Para a maioria desses problemas, ainda há espaço para melhorar os resultados conhecidos, seja na garantia de desempenho dos algoritmos ou em melhores resultados de impossibilidade de aproximação (resultados de inaproximabilidade).

Palavras-chave: problema dos k-centros, clustering, algoritmos de aproximação.

Sumário

Capítulo 1

Introdução e preliminares

Há uma quantidade imensa de problemas práticos que podem ser modelados como problemas de *otimização combinatória*. Por exemplo, encontrar um conjunto mínimo de pedaços suspeitos de código de forma que, com os pedaços tomados, seja possível determinar "assinaturas" de todo um conjunto dos vírus de computador conhecidos. Ou ainda, determinar onde devem ser abertos centros de distribuição de um certo produto de modo que nenhuma loja que o vende esteja muito distante do centro de distribuição mais próximo, ou descobrir como fazer uma partição de um conjunto de objetos em categorias de modo que estas representem objetos com características semelhantes.

No entanto, ao tentar utilizar um computador para resolver os problemas de otimização combinatória derivados, nos deparamos com um obstáculo. Muitos deles são computacionalmente difíceis, no sentido de que não podem ser resolvidos por algoritmos eficientes (isto é, de tempo polinomial no tamanho da descrição do problema), a menos que P = NP. Como em geral precisamos obter respostas em tempo razoável, uma maneira de enfrentar esse obstáculo é desenvolver *algoritmos de aproximação*.

Algoritmos de aproximação são algoritmos que, para um problema de otimização, calculam eficientemente soluções aproximadamente ótimas. Ou seja, para qualquer instância I do problema a que se destina, um algoritmo de aproximação devolve em tempo polinomial uma solução cujo valor dista do valor ótimo para I de no máximo um determinado fator, chamado garantia de desempenho.

Neste trabalho, descreveremos alguns resultados em algoritmos de aproximação para problemas de *clustering*. Nosso foco são o problema dos k-centros (*k-center*) e algumas de suas numerosas variantes.

Há diversos tipos de problemas de clustering. Sua forma geral consiste no seguinte: dado um conjunto de pontos que podemos comparar através de uma função distância ou correlação, encontrar um agrupamento desses pontos de modo que a quantidade de grupos usados e a disposição dos pontos nos grupos sejam úteis para o problema em questão. Exploraremos alguns problemas específicos e discutiremos possíveis interpretações práticas para eles.

Todos os problemas que iremos discutir são NP-difíceis, de modo que algoritmos de aproximação constituem uma boa alternativa para atacá-los.

Descreveremos nesse trabalho resultados presentes na literatura. Além de mostrar algoritmos existentes para os problemas discutidos, também descreveremos resultados de inaproximabilidade, que são teoremas que nos dizem que é impossível obter algoritmos de aproximação com certas garantias de desempenho, sob a hipótese, por exemplo, de que $\mathrm{P} \neq \mathrm{NP}$.

1.1 Histórico do problema

O problema dos k-centros é um problema clássico de clustering. Deseja-se, dado um grafo completo com custos nas arestas respeitando a desigualdade triangular, e um inteiro positivo k, encontrar um conjunto de k vértices, chamados centros, que minimize a máxima distância de um vértice ao conjunto de centros. Essa distância é por vezes chamada de *raio* da solução. Podemos considerar sua versão de decisão, em que o problema é determinar se, dada uma instância do problema dos k-centros e um número r, existe solução de raio não superior a r. Esse problema de decisão é elencado por Garey e Johnson em sua lista de problemas NP-completos [9].

Há bastante tempo se conhecem algoritmos de aproximação para esse problema. Gonzalez [10] e, independentemente, Hochbaum e Shmoys [11] mostram algoritmos que atingem um fator de aproximação igual a 2. Além disso, tais trabalhos também mostram que esse fator é o melhor possível a menos que P = NP.

Desde então, diversas variantes do problema foram propostas e estudadas. Podemos pensar que, no problema original, cada vértice se *conecta* ao centro mais próximo, e a conexão mais distante determina o raio da solução. Nas variantes com *centros capacitados*, cada vértice u tem uma capacidade finita L_u e, caso seja escolhido como centro, não pode ter mais de L_u vértices conectados a si. Assim, a escolha de quais vértices se conectam a quais centros, chamada *atribuição*, precisa ser feita explicitamente. Khuller e Sussmann [13] mostram como obter uma 6-aproximação, para o *caso uniforme*, em que todas as capacidades são iguais. Em um desenvolvimento recente na área, An et al. [2] descreveram uma 9-aproximação para o caso em que as capacidades não são uniformes. Esse resultado melhora um anterior, devido a Cygan et. al. [6], para o qual não havia um cálculo explícito do fator de aproximação, apenas uma demonstração de que era constante. O algoritmo de Cygan et. al. foi o primeiro a aplicar com sucesso arredondamento de programação linear para o problema.

Outra restrição que podemos considerar é a seguinte: entendendo os centros como prestadores de algum tipo de serviço, podemos imaginar que nossa solução seja robusta a uma falha em seu funcionamento. Assim, desejamos que os vértices que antes se conectavam a um centro que falhou possam se reconectar a outro centro não muito distante. Esse tipo de exigência nos dá as versões com *tolerância a falhas*. Para o problema dos k-centros com α-tolerância, em que levamos em conta na função objetivo a distância de um vértice a α centros distintos, onde α é parte da instância, Khuller, Pless e Sussmann [12] dão uma 3-aproximação.

Chechik e Peleg [5] combinaram os dois tipos de restrição, estudando duas variantes do problema dos k-centros. Em ambas, exige-se tolerância a falhas como na forma descrita acima, e além disso os centros têm capacidades uniformes. A diferença entre tais variantes é a forma como são feitas as novas conexões após um cenário de falhas. Na variante não conservadora, são permitidas atribuições novas para todos os vértices. Na conservadora, apenas os vértices atribuídos aos centros que falharam podem ser remanejados. Chechik e Peleg [5] descrevem algoritmos de aproximação com fatores 9 e 17, respectivamente, para tais problemas. Como parte de nosso trabalho no mestrado, consideramos também esses problemas, para os quais descrevemos algoritmos que atingem fator 6, para a variante não conservadora, e 7, para a conservadora [8]. Ainda, estudamos o caso em que as capacidades não são uniformes, para os quais obtivemos os primeiros algoritmos conhecidos.

1.2 Problemas de otimização

Um problema de otimização, para nós, é um conjunto de instâncias. Dado um problema Π, para cada instância I de Π, temos um conjunto de *soluções viáveis*, $\mathrm{Sol}(I)$. A cada solução viável $S \in \mathrm{Sol}(I)$, está associado um valor numérico, $\mathrm{val}(S)$.

Dada uma instância I, desejamos encontrar $S^* \in \mathrm{Sol}(I)$ tal que $\mathrm{val}(S^*)$ seja *mínimo*, para problemas de minimização, ou *máximo*, para problemas de maximização. Um tal S^* é chamado de *solução ótima*, e seu valor $\mathrm{val}(S^*)$ é o *valor ótimo*, denotado por $\mathrm{OPT}(I)$, ou simplesmente OPT, quando a instância está subentendida.

Se $\mathrm{Sol}(I) = \emptyset$, dizemos que a instância I é *inviável*.

Um exemplo de problema de otimização é o problema da clique de tamanho máximo. Isto é, o problema de, dado um grafo G, encontrar o maior conjunto de vértices de G que são dois a dois adjacentes. Trata-se de um problema de maximização. Uma instância desse problema é um grafo $G = (V, E)$. O conjunto de soluções viáveis é $\mathrm{Sol}(I) = \{S \subseteq V : S \text{ é clique em } G\}$. O valor de uma solução viável S é $\mathrm{val}(S) = |S|$.

Em geral, com respeito a um dado problema de otimização combinatória Π, desejamos saber se há como resolvê-los de forma eficiente. Usualmente, isso significa investigar se existe um algoritmo que, para qualquer instância I de Π, devolve em tempo polinomial na descrição de I uma solução ótima para tal instância. Se Π for um problema NP-difícil, isso não é possível a menos que P = NP. Podemos, nesse caso, fazer outras perguntas a respeito de como resolver o problema. Uma dessas perguntas é: existem bons algoritmos de aproximação para esse problema? Tal questionamento pode ser formalizado de diversas formas, a algumas das quais este trabalho se presta com respeito ao problema dos k-centros e algumas variantes. A saber, buscamos conhecer algoritmos de aproximação com fator multiplicativo e resultados de inaproximabilidade.

1.3 Algoritmos de aproximação

Há mais de uma maneira de definir algoritmos de aproximação. A única que iremos considerar de agora em diante, e que é considerada a mais comum no atual corpo de trabalho na área [16], é a seguinte.

Definição 1.1. *Considere um problema Π de otimização combinatória. Uma* α-aproximação *para Π é um algoritmo polinomial que, para toda instância I de Π, ou determina corretamente que I é inviável ou encontra uma solução $S \in \mathrm{Sol}(I)$ tal que:*

- $\mathrm{val}(S) \leq \alpha \cdot \mathrm{OPT}(I)$*, se Π é de minimização, e nesse caso $\alpha \geq 1$; ou*
- $\mathrm{val}(S) \geq \alpha \cdot \mathrm{OPT}(I)$*, se Π é de maximização, e nesse caso $\alpha \leq 1$.*

Se um algoritmo é uma α-aproximação, o número α é chamado de sua garantia de desempenho *ou* fator de aproximação*.*

Para o problema dos k-centros e as variantes que veremos, que são problemas de minimização, teremos sempre $\alpha > 1$.

Capítulo 2

Problema dos k-centros

Nessa seção, começaremos a tratar do problema de particionar elementos em conjuntos (*clusters*) de acordo com suas semelhanças. Trata-se de um problema bastante genérico que aparece em diversas aplicações, e por isso existem muitas variantes dele na literatura. A principal diferença entre elas é a função objetivo. Uma das formas mais simples desse problema é o problema dos k-centros (k-*center*), definido da seguinte maneira.

> **Problema dos k-centros** (k-CENTROS): *Dada uma coleção M de pontos com distâncias definidas para cada par de pontos pela função $d\colon M \times M \to \mathbb{Q}_{\geq 0}$, encontrar um conjunto $S \subseteq M$, com $|S| \leq k$, cujos elementos chamamos de* centros, *de forma que a maior distância de um ponto $p \in M$ ao centro mais próximo de p seja mínima.*

Note que nessa formulação, fixada uma instância (M, d, k), uma escolha de centros $S = \{s_1, s_2, \ldots, s_\ell\} \subseteq M$ dá implicitamente uma partição de M em $\ell \leq k$ conjuntos $B_1, \ldots, B_\ell$ dada por $B_i = \{p \in M\colon \text{o centro mais próximo de } p \text{ é } s_i\}$, para $i = 1, \ldots, \ell$. Se, para um dado p, houver mais de um centro que atinge a distância mínima, escolhe-se um deles arbitrariamente para determinar o cluster de p.

Uma possível interpretação desse problema seria, por exemplo, a automatização de um processo de divisão em categorias. Podemos imaginar, como exemplo, o uso de dados de compras de consumidores. Conforme os tipos e quantidades de produtos comprados, pode-se elaborar uma maneira de medir a "distância" entre dois consumidores. Nesse contexto, o problema dos k-centros pode ser visto como o problema de estabelecer k "perfis de consumidor" de maneira que os consumidores incluídos em cada perfil sejam um grupo o mais homogêneo possível. Poderíamos usar a mesma ideia para tentar encontrar formas de categorizar alunos de um curso, deputados de uma casa legislativa, etc., conforme os critérios de comparação e formulações matemáticas que modelem mais convenientemente a situação.

Outra interpretação para esse problema é como um problema de *localização de instalações* (*facility location*). Os pontos representam um conjunto de locais que precisam ser supridos por certo tipo de instalação, e suas distâncias são definidas pela rota de menor custo de um ponto a outro — onde o custo pode ser a distância no espaço, o tempo, ou uma combinação desses e outros fatores. Deseja-se abrir tais instalações em até k dos pontos de maneira que nenhum ponto esteja muito longe da instalação mais próxima. Por exemplo, imagine que temos um conjunto de hospitais públicos, e dispomos de recursos para abrir centros de tratamento de câncer em no máximo k deles. Desejamos saber em quais hospitais instalar tais centros de modo a minimizar o maior tempo de transporte entre um hospital comum e seu centro de tratamento mais próximo, a fim de que a transferência mais demorada de um paciente seja tão rápida quanto possível.

A interpretação do clustering como um problema de localização de instalações natu-

ralmente leva a variantes do problema em que a possibilidade de falha de algumas das instalações é considerada. Veremos algumas dessas mais adiante.

Antes de mostrar um algoritmo para o k-CENTROS, que apresentamos acima, veremos um problema relacionado, o k-clustering.

2.1 Minimização do máximo diâmetro de um cluster

Veremos aqui alguns resultados para problemas de clustering estudados por Gonzalez [10], adaptando e por vezes complementando seu conteúdo. Trataremos do problema de minimizar a maior distância entre qualquer par de pontos que esteja dentro de um mesmo cluster. Veremos a complexidade de algumas de suas variações e um algoritmo de aproximação.

Definição 2.1. *Seja $G = (V, E)$ um grafo com custos não-negativos nas arestas dados pela função $c\colon E \to \mathbb{Q}_{\geq 0}$. Uma* k-partição *de G é uma partição de V em k conjuntos $B_1, B_2, \ldots, B_k$. Cada B_i é um* cluster*.*

Definição 2.2. *Considere as condições da definição anterior. A k-partição $\{B_1, \ldots, B_k\}$ tem valor*

$$\operatorname{val}(B_1, \ldots, B_k) = \max\{M_1, \ldots, M_k\},$$

onde

$$M_i = \max\{c(e)\colon e \in E \text{ tem as duas pontas em } B_i\}$$

é o diâmetro *de B_i.*

A função objetivo que define o valor de uma k-partição é chamada por Gonzalez de "MM" (max-max). Em geral, pensaremos no problema de otimização $\min_{(B_1,\ldots,B_k)} \operatorname{val}(B_1, \ldots, B_k)$.

Diremos que estamos tratando do *caso métrico* quando o grafo é completo e os custos de suas arestas satisfazem a desigualdade triangular. Isto é,

$$c(ij) \leq c(ik) + c(kj), \qquad \text{para quaisquer } i, j, k \in V \text{ distintos.}$$

Note que, na verdade, não precisamos exigir que o grafo seja completo. Caso não seja, para cada $ij \notin E$, podemos incluir ij em E, e podemos estabelecer $c(ij) = d(i, j)$ para cada ij, onde $d(i, j)$ é o custo de um caminho de custo mínimo de i a j no grafo original.

A notação de Gonzalez para as variantes desse problema é a seguinte. Definem-se os problemas de clustering α-βMM, onde α é o número de clusters que desejamos formar e β representa uma descrição do espaço em que estão os pontos. Se o número de clusters for um parâmetro de entrada para o problema, então escrevemos $\alpha = k$. Se os pontos a serem divididos em clusters estão em um espaço euclidiano de dimensão m, com a distância entre pontos definida pela distância euclidiana, então $\beta = m$. Para o caso métrico, escrevemos $\beta = t$. Se não há nenhuma hipótese sobre o grafo em questão, ou seja, no caso mais geral possível, escrevemos $\beta = g$. O "MM" refere-se à nossa função objetivo, conforme já mencionamos.

Descreveremos agora uma 2-aproximação para o k-tMM, problema também conhecido simplesmente como k-clustering. Pela descrição geral que demos, esse problema se define da seguinte maneira.

> **Problema do k-clustering**: *Dado um grafo completo $G = (V, E)$ com custos $c\colon E \to \mathbb{Q}_{\geq 0}$ nas arestas respeitando a desigualdade triangular, e um inteiro k, encontrar uma partição de V em conjuntos $B_1, B_2, \ldots, B_k$ de modo que o máximo diâmetro dentre os B_i seja mínimo.*

2.1.1 Um algoritmo de aproximação para o k-clustering

Sejam M um conjunto de pontos, $G = (M, E)$ um grafo completo e d uma função distância sobre M. Isto é, considere uma instância, denotada por (M, d, k), do caso métrico do nosso problema. Suponha que $|M| > k$. Se $M \leq k$, o problema é trivial: basta tomar $\{i\}$ como conjunto da partição para cada $i \in M$. Seja $\mathrm{OPT}(M, d, k)$ o valor de uma k-partição ótima para a instância (M, d, k).

Definição 2.3. *Num grafo $G = (M, E)$, um conjunto $T \subseteq S$ é uma* clique *se, para todo par $\{i, j\}$ de pontos distintos de T, existe uma aresta com extremos i e j.*

Note que, no caso métrico, todo subconjunto de M é uma clique.

Definição 2.4. *Uma clique $T \subseteq M$ é uma* a-clique de peso h *se $|T| = a$ e $h = \min_{i \neq j \in T} d(i, j)$.*

Lema 2.5. *Se M tem uma $(k+1)$-clique de peso h, então $\mathrm{OPT}(S, d, k) \geq h$.*

Demonstração. Seja T uma $(k+1)$-clique de M de peso h. Considere uma k-partição $\{B_1, B_2, \ldots, B_k\}$ de M, e seja i tal que $1 \leq i \leq k$ e $|B_i \cap T| \geq 2$. Tal i existe pois $|T| = k + 1 > k$.

Sejam p e q elementos distintos em $B_i \cap T$. Segue da nossa hipótese que $d(p, q) \geq h$. Logo,

$$\mathrm{val}(B_1, \ldots, B_k) \geq h,$$

isto é, qualquer k-partição tem valor no mínimo h. □

Gonzalez [10] propôs o seguinte algoritmo para resolver o problema do k-clustering para uma dada instância (M, d, k).

GONZALEZ-K-CLUSTERING(M, d, k)

```
1  B_1 ← M
2  s_1 ← elemento arbitrário em B_1
3  para ℓ ← 1 até k − 1
4       v, j ← arg max{d(c_j, v) : 1 ≤ j ≤ ℓ, v ∈ B_j}
5       B_j ← B_j \ {v}
6       B_{ℓ+1} ← {v}
7       s_{ℓ+1} ← v
8       para cada v ∈ B_1 ∪ ··· ∪ B_ℓ
9            seja j tal que v ∈ B_j
10           se d(v, s_j) > d(v, s_{ℓ+1})
11                B_j ← B_j \ {v}
12                B_{ℓ+1} ← B_{ℓ+1} ∪ {v}
13 devolva (B_1, B_2, ..., B_k)
```

Observe que o algoritmo começa colocando todos os pontos em B_1 e selecionando um elemento arbitrário como o centro s_1 desse conjunto. A cada passo ℓ, o algoritmo já tem os clusters $B_1, \ldots, B_\ell$, e cria um novo cluster de acordo com o seguinte critério: tome um ponto, digamos v, que esteja à distância máxima do centro de seu cluster atual. Mova v para $B_{\ell+1}$, e tome-o como o centro $s_{\ell+1}$ desse novo cluster. Então, mova para $B_{\ell+1}$ todos os pontos que estão mais próximos de $v = s_{\ell+1}$ do que de seus respectivos centros atuais.

Teorema 2.6. *O algoritmo* GONZALEZ-K-CLUSTERING *é uma 2-aproximação para o k-clustering.*

Demonstração. Em primeiro lugar, o algoritmo é polinomial. A cada cluster criado, a lista dos pontos é percorrida para obter aquele que está à maior distância de seu centro atual. Esse passo pode ser feito em tempo $O(n)$, onde $n = |M|$. Após escolher novo centro, é necessário novamente percorrer os clusters para verificar os pontos que deverão ser movidos para o novo conjunto. Esse passo também consome tempo $O(n)$. Logo, a criação de um novo cluster custa tempo $O(n)$. O consumo de tempo do algoritmo, portanto, é $O(kn)$, que é $O(n^2)$, visto que $k \leq n$.

Além disso, é claro que o algoritmo gera uma k-partição. Resta verificar que o valor da solução gerada não é superior a $2 \cdot \mathrm{OPT}(S, d, k)$.

Seja $\{B_1, \ldots, B_k\}$ a solução gerada pelo algoritmo. Seja h a maior distância de um vértice ao centro de seu cluster, isto é, $h = \max\{d(v, s_j) \colon 1 \leq j \leq k \text{ e } v \in B_j\}$. Se p e q são vértices no mesmo cluster, digamos B_j, e são tais que $\mathrm{val}\,(B_1, \ldots, B_k) = d(p, q)$, note que, como estamos no caso métrico, vale que

$$\mathrm{val}\,(B_1, \ldots, B_k) = d(p, q) \leq d(p, s_j) + d(s_j, q) \leq h + h = 2h. \tag{2.1}$$

Sejam agora j^* tal que $1 \leq j^* \leq k$ e v^* um vértice tal que $d(v^*, s_{j^*}) = h$, onde $v^* \in B_{j^*}$, isto é, v^* é um vértice que está mais distante de seu centro s_{j^*}. Observe que, como v^* não é o centro de seu conjunto, nenhum passo do algoritmo escolheu v^* como novo centro. Ou seja, no momento em que um vértice v foi escolhido como centro de um cluster, v estava à distância pelo menos h do centro do cluster ao qual pertencia. Portanto, cada novo centro escolhido dista de no mínimo h dos centros anteriormente escolhidos. Além disso, v^* está à distância pelo menos h de cada centro, visto que está a essa distância do centro mais próximo.

Portanto, $\{s_1, \ldots, s_k, v^*\}$ é uma $(k+1)$-clique de peso pelo menos h. Logo, pelo Lema 2.5, segue que $\mathrm{OPT}(S, d, k) \geq h$. Disso e de (2.1), temos que

$$\mathrm{val}\,(B_1, \ldots, B_k) \leq 2h \leq 2 \cdot \mathrm{OPT}(S, d, k).$$

□

Antes de prosseguir, vejamos que a análise acima é justa. De fato, mostramos a seguir que é possível gerar instâncias para as quais a razão entre o valor ótimo e o valor da solução dada pela rotina Gonzalez-k-clustering é tão próxima de 2 quanto se queira.

Considere a seguinte classe de instâncias do k-1MM (k-clustering de pontos na reta real), que é um caso particular do k-tMM (k-clustering métrico):

- $S = \{0, 1, 2 - \varepsilon, 3\}$, onde $0 < \varepsilon < 1$,
- $d(i, j)$ é a distância euclidiana entre i e j (no caso 1-dimensional, $d(i, j) = |i - j|$), e
- $k = 2$.

Suponha que no primeiro passo o algoritmo tome o ponto 1 como c_1. No passo seguinte, o ponto mais distante de c_1 é o ponto 3, que é então tomado como o centro c_2 do conjunto B_2. O algoritmo termina com:

- $B_1 = \{0, 1, 2 - \varepsilon\}, B_2 = \{3\}$,
- $\mathrm{val}\,(B_1, B_2) = d(2 - \varepsilon, 0) = 2 - \varepsilon$.

Note que o valor ótimo para essa instância é $\mathrm{val}\,(\{0, 1\}, \{2 - \varepsilon, 3\}) = 1 + \varepsilon$. Portanto, a razão entre o valor da solução do algoritmo e o valor ótimo é, nesse caso, $\frac{2-\varepsilon}{1+\varepsilon}$.

2.1.2 Complexidade dos problemas de clustering

Vimos um algoritmo de aproximação para o k-tMM (k-clustering métrico), mas até agora não nos perguntamos se esse problema, ou pelo menos o k-gMM, que é mais geral, são NP-difíceis. Em geral buscamos aproximações para problemas dessa classe, por ser desconhecida uma maneira de resolvê-los eficientemente. Mas, pelo que discutimos até agora, ainda não sabemos se é difícil de obter uma solução eficiente para os problemas vistos.

Mas, como veremos agora, é verdade que esses problemas são NP-difíceis. Mostraremos uma prova, dada por Gonzalez [10], de que a versão de decisão do k-tMM é um problema NP-completo. Disso segue, como veremos, que a versão de otimização do k-tMM é NP-difícil. O mesmo vale para o k-gMM, por ser uma generalização do primeiro.

Veremos, ainda, que não existe uma $(2-\varepsilon)$-aproximação para o k-tMM, para qualquer $\varepsilon > 0$, a menos que P = NP. Ou seja, o algoritmo Gonzalez-k-clustering é o melhor possível para esse problema, a menos que P = NP.

Considere o problema da cobertura exata por conjuntos de tamanho 3, que chamaremos de XC3 ou emparelhamento 3D:

> **Problema** XC3: *Dados um conjunto finito de elementos $X = \{x_1, x_2, \ldots, x_{3q}\}$ e uma coleção C de subconjuntos de X de tamanho 3, tais que $|C| = m$ e nenhum elemento de X aparece em mais do que três conjuntos de C, decidir se C contém uma cobertura exata de X, isto é, se existe uma subcoleção $C' \subseteq C$ tal que cada elemento de X ocorre em exatamente um conjunto de C'.*

Sabe-se que esse problema é NP-completo [9]. Considere agora a seguinte restrição de XC3, denotada por RXC3.

> **Problema** RXC3: *Dada uma instância (X, C) de XC3 tal que cada elemento de X ocorre em* exatamente 3 *dos conjuntos na coleção C, decidir se C contém uma cobertura exata de X.*

O problema RXC3 também é NP-completo: Gonzalez [10] mostra uma redução polinomial de XC3 a RXC3. Mostraremos uma redução polinomial de RXC3 a uma versão de decisão do k-clustering.

> **Problema de decisão do k-clustering**: *Dados um conjunto de vértices M com distância $d(p, q)$ definida para cada par de vértices obedecendo a desigualdade triangular, um inteiro positivo k e um racional não negativo w, decidir se S admite uma k-partição de peso não superior a w.*

Teorema 2.7. *A versão de decisão do k-clustering é um problema* NP*-completo.*

Demonstração. A versão de decisão do k-clustering está em NP. Dada uma instância (M, d, k, w) do problema, um certificado para "sim" é uma k-partição $(B_1, \ldots, B_k)$ tal que val $(B_1, \ldots, B_k) \leq w$. Tal certificado tem tamanho polinomial no tamanho da entrada, e pode ser verificado em tempo polinomial.

Veremos agora uma redução polinomial do RXC3 para a versão de decisão do k-clustering.

Seja (X, C) uma instância do RXC3, com $|X| = 3q$, com q inteiro, e $|C| = m$. Considere a seguinte instância do k-tMM:

- O conjunto M de vértices contém um vértice v_i para cada $x_i \in X$, e cada $Y \in C$ introduz 9 novos vértices. Se $Y = \{x_i, x_j, x_k\}$, a estrutura mostrada na figura 2.1 representa Y.

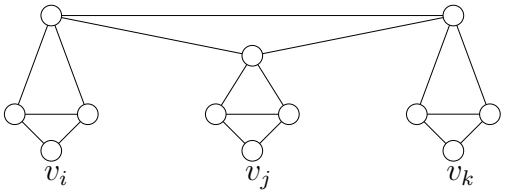
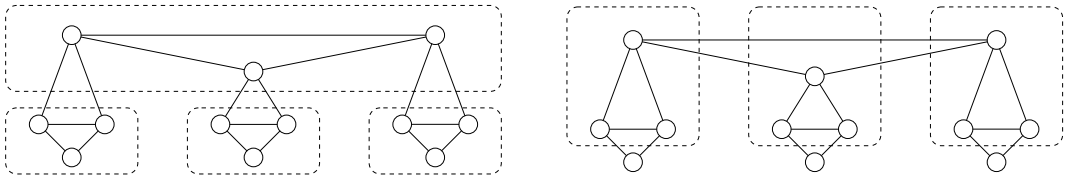


Figura 2.1: *Componente representando um* $Y = \{x_i, x_j, x_k\} \in C$.

- As distâncias d são dadas da seguinte maneira: as arestas da representação de cada um dos $Y \in C$, conforme acima, têm peso 1. As demais arestas têm peso 2;
- $k = 3m + q$; e
- $w = 1$.

Tal construção pode ser feita em tempo polinomial no tamanho da instância (X, C).

Para completar a prova, basta mostrar que M admite uma k-partição de valor no máximo 1 se, e somente se, C contém uma cobertura exata de X.



(a) Clustering dos $Y \in C'$ (b) Clustering dos $Y \in C \setminus C'$

Figura 2.2: *As duas possibilidades de divisão em clusters de um conjunto de vértices representando um* $Y \in C$.

($\Leftarrow$) Seja $C' \subseteq C$ uma cobertura exata de X. Para cada $Y \in C'$, a componente que representa Y é agrupada em quatro clusters, conforme mostrado na figura 2.2(a).

Para cada $Y \in C \setminus C'$, a componente correspondente é agrupada em 3 clusters, conforme mostrado na figura 2.2(b). Note que esses clusters definem uma partição de M, isto é, cada vértice é incluído em exatamente um cluster por essa construção.

Cada elemento em C contribui com 3 clusters, exceto aqueles em C', que contribuem com 4 clusters cada.

Como C' é uma cobertura exata de X e todos os seus elementos são 3-conjuntos (conjuntos de tamanho 3), vale que $C' = |X|/3$. Logo, o número de clusters criados é

$$3 \cdot |C| + 1 \cdot |C'| = 3m + \frac{|X|}{3} = 3m + q = k.$$

Finalmente, em todo cluster vale que os vértices estão dois a dois ligados por arestas de peso 1. Portanto, obtivemos uma k-partição de S com peso $1 \leq w$.

($\Rightarrow$) Seja $(B_1, \ldots, B_k)$ uma k-partição de S com valor $\text{val}\,(B_1, \ldots, B_k) \leq w = 1$. Note que, na instância do k-clustering, não há nenhum conjunto de quatro vértices dois a dois ligados por arestas de peso 1. Além disso, $|V| = 3q + 9m = 3(q + 3m) = 3k$.

Assim, cada B_i tem cardinalidade $|B_i| = 3$. Pela regra de construção dessa instância, toda componente que representa um $Y \in C$ está agrupada, pela nossa k-partição, como uma das opções da figura 2.2.

Como todo vértice está em exatamente um cluster, para que tenhamos k clusters, certamente há exatamente q componentes representando elementos em C que estão agrupados

como na figura 2.2(a), e cada um desses deve incluir v_ℓ's todos diferentes, uma vez que cada vértice desses só está em clusters das componentes agrupadas conforme a figura 2.2(a).

Dessas observações, segue que, se tomarmos os $Y \in C$ representados por componentes que estão agrupadas como na figura 2.2(a), teremos uma cobertura exata de X. □

Esse resultado tem duas consequências imediatas.

Corolário 2.8. *O problema de otimização k-tMM (k-clustering métrico) é* NP*-difícil.*

Corolário 2.9. *O problema de otimização k-gMM é* NP*-difícil.*

O próximo resultado segue da redução usada na prova do Teorema 2.7.

Teorema 2.10. *Não existe algoritmo de aproximação para o k-tMM com fator menor do que 2, a menos que* P = NP*.*

Demonstração. Seja (X, C) uma instância do RXC3. Seja (G, d, k) a instância do k-tMM obtida pela mesma redução usada na prova do Teorema 2.7. O valor ótimo para (G, d, k) é 1, se a resposta do RXC3 para (X, C) é "sim", e 2, caso contrário.

Fixe um $\varepsilon > 0$. Uma $(2 - \varepsilon)$-aproximação daria, para a instância (G, d, k), uma resposta exata. De fato, se o valor ótimo da instância é 2, então o algoritmo daria uma resposta de valor ≥ 2, que só pode ser, de fato, 2, uma vez que não há aresta com custo maior que 2; e se o valor ótimo da instância é 1, o algoritmo obrigatoriamente forneceria uma solução de valor $\leq (2 - \varepsilon) \cdot 1 < 2$, que no caso só pode ser, de fato, 1 (pois toda aresta tem valor 1 ou 2).

Dessa maneira, tal $(2 - \varepsilon)$-aproximação resolveria, em tempo polinomial, o problema RXC3, que é NP-completo, implicando P = NP. □

Note que os resultados sobre a complexidade e inaproximabilidade do k-tMM são especialmente importantes por tratar-se do tipo mais básico de problema de clustering. Muitas variações do clustering têm o k-tMM como um caso particular, de modo que os resultados vistos para este valem também para as variantes. Isto é, se um problema tem k-tMM como caso particular, sabemos de imediato que uma 2-aproximação para ele é o melhor que podemos esperar obter, a menos que P = NP.

2.2 Algoritmo para o problema dos k-centros

Apresentamos acima o problema do k-clustering, similar ao k-CENTROS, e mostramos uma 2-aproximação para ele. Note que a instância dos dois problemas pode ser definida da mesma maneira: um grafo métrico, isto é, um grafo completo com custos nas arestas respeitando a desigualdade triangular. Isso é o mesmo que dizer que temos um conjunto de pontos, os vértices, e uma função distância sobre eles. A diferença entre os dois problemas é que no k-centros queremos minimizar não o maior diâmetro de um cluster, mas o maior **raio**, isto é, a máxima distância de um vértice ao seu centro mais próximo.

Observe, ainda, que no algorimo GONZALEZ-K-CLUSTERING, há uma escolha explícita de centros a cada cluster formado. Isso nos leva a perguntar se o algoritmo pode servir também para o k-centros, e a resposta é sim [16].

Teorema 2.11. *O algoritmo* GONZALEZ-K-CLUSTERING *é uma 2-aproximação para o k-*CENTROS*.*

Demonstração. Tome uma instância (M, d, k) do k-CENTROS. Considere $\{s_1^*, s_2^*, \ldots, s_k^*\}$ uma solução ótima do k-centros para essa instância, e sejam $B_1^*, \ldots, B_k^*$ os clusters definidos por eles. Isto é, B_i^* é o conjunto dos vértices cujo centro mais próximo é s_i^*. Seja r^* o custo dessa solução. Cada par de pontos em cada B_i^* está à distância no máximo $2r^*$ pela desigualdade triangular.

Sejam $s_1, s_2, \ldots, s_k$ os centros escolhidos pelo algoritmo. Se cada s_i está em um cluster B_j^* diferente, então cada ponto em M está à distância no máximo $2r^*$ de seu centro mais próximo na solução do algoritmo, pelo que essa solução ótima nos diz.

Suponha, agora, que algum cluster dessa solução ótima contenha pelo menos dois dos centros dados pelo algoritmo, digamos s_i e s_j com $i < j$. O centro s_j foi escolhido porque na iteração j era o vértice mais distante do conjunto de centros até então. Portanto, se s_i e s_j estão à distância no máximo $2r^*$, então todos os vértices estão à distância no máximo $2r^*$ de algum centro escolhido pelo algoritmo. □

Outro resultado para o k-CENTROS que é semelhante ao do k-clustering é o de inaproximabilidade. O problema dos k-centros também não pode ser aproximado por um fator melhor que 2, a menos que P = NP [16].

2.3 Técnica de Hochbaum e Shmoys para problemas de gargalo

Note que, devido à forma de sua função objetivo, os problemas dos k-centros e k-clustering têm como valores de soluções viáveis os custos das arestas do grafo. Em particular, o valor ótimo é necessariamente o custo de alguma aresta do grafo.

Hochbaum e Shmoys [11] se referem a problemas com essa propriedade por **problemas de gargalo** (*bottleneck problems*), e propõem um método geral para resolvê-los. Um tal problema admite, para uma instância que diz respeito a um grafo G, um conjunto $\mathcal{G}$ de *subgrafos viáveis* de G, que definimos da seguinte maneira. Um subgrafo H de G está em $\mathcal{G}$ se existe em H uma solução viável para o problema. Portanto, basta encontrarmos $G^* \in \mathcal{G}$ cujo custo de uma aresta mais cara seja mínimo. Tais ideias ficarão mais claras ao estudarmos problemas concretos mais adiante.

Todos os problemas que vamos considerar daqui em diante são *métricos*, isto é, são definidos sobre grafos completos com custos respeitando a desigualdade triangular. Começaremos dando algumas definições e resultados que nos ajudarão a descrever a técnica.

Definição 2.12. *Um* algoritmo de decisão ρ-relaxado *para um problema de minimização* Π *é um algoritmo de tempo polinomial que, dada uma instância* I *de* Π *e um valor* x*, devolve* ou *uma solução viável* $S \in \mathrm{Sol}(I)$ *de valor* $\mathrm{val}(S) \leq \rho \cdot x$*,* ou *um certificado eficiente de que* $\mathrm{OPT}(I) > x$*.*

Definição 2.13. *Dado um grafo* $G = (V, E)$ *e um inteiro positivo* t*, a* t*-ésima potência de* G *é o grafo* $G^t = (V, E^t)$*, com*

$$E^t = \{\{u, v\} : \text{ existe em } G \text{ um caminho de } u \text{ a } v \text{ de comprimento } \leq t\}.$$

Dado um grafo $G = (V, E)$ com um custo c_e para cada aresta $e \in E$, defina

$$\max(G) = \max_{e \in E} c_e.$$

Lema 2.14. *Sejam G um grafo completo com custos c_e nas arestas respeitando a desigualdade triangular, e u, v vértices distintos. Se $P \subseteq G$ é um caminho de u a v de comprimento ℓ, então*

$$c_{uv} \leq \ell \cdot \max(P).$$

Demonstração. Para $\ell = 2$, a propriedade segue diretamente da desigualdade triangular.

Seja $\ell > 2$ e suponha que a propriedade valha para caminhos de comprimento $\ell - 1$.

Seja P um caminho $i_0 i_1 \cdots i_\ell$, de comprimento ℓ, com $u = i_0$, $v = i_\ell$. Seja $P' \subseteq P$ o caminho $i_0 i_1 \cdots i_{\ell-1}$. Por hipótese de indução, vale que

$$c_{i_0 i_{\ell-1}} \leq (\ell - 1) \max(P') \leq (\ell - 1) \max(P), \tag{2.2}$$

e, pela desigualdade triangular,

$$c_{i_0 i_\ell} \leq c_{i_0 i_{\ell-1}} + c_{i_{\ell-1} i_\ell}. \tag{2.3}$$

De (2.2) e (2.3), segue que

$$c_{i_0 i_\ell} \leq (\ell - 1) \max(P) + c_{i_{\ell-1} i_\ell} \leq (\ell - 1) \max(P) + \max(P) = \ell \cdot \max(P),$$

e a prova do lema segue por indução. □

Lema 2.15. *Seja G um grafo completo com custos c_e nas arestas respeitando a desigualdade triangular. Sejam H um subgrafo de G e t um inteiro positivo. Se $\{i, j\}$ é uma aresta em H^t, então*

$$c_{ij} \leq t \cdot \max(H).$$

Demonstração. Como $\{i, j\}$ é uma aresta em H^t, existe em H um caminho de comprimento t de i a j. Pelo Lema 2.14, segue que $c_{ij} \leq t \cdot \max(H)$. □

Lema 2.16. *Seja G um grafo completo com custos c_e nas arestas respeitando a desigualdade triangular. Sejam H um subgrafo de G e t um inteiro positivo. Se existe uma solução S em H^t para um problema de gargalo sobre (G, c), então S é uma solução em G e*

$$\operatorname{val}(S) \leq t \cdot \max(H).$$

Demonstração. Note que H^t é um subgrafo de G. Isso vem dos fatos de que o conjunto $V(H^t)$ de vértices de H^t é um subconjunto dos vértices de G, e de que G é o grafo completo. Portanto, toda solução em H^t está, de fato, em G.

Além disso, como se trata de um problema de gargalo, o valor de uma solução S em H^t é $\operatorname{val}(S) = c_e$ para alguma aresta e de H^t. Assim, do Lema 2.15, segue que $\operatorname{val}(S) \leq t \cdot \max(H)$. □

Hochbaum e Shmoys [11] propuseram o algoritmo MetodoGargalo como método geral para aproximar problemas de gargalo. O algoritmo recebe o grafo G e os custos c, além de possivelmente outras informações específicas ao problema em particular. Ele faz uso de duas rotinas que descrevemos a seguir.

Fixe um problema de gargalo Π e considere as seguintes rotinas. Dado um grafo $G = (V, E)$ com custo c_e para cada $e \in E$ e um valor x, Gargalo (G, x) devolve o grafo $G' = (V, E')$, onde $E' = \{e \in E : c_e \leq x\}$. Isto é, o grafo obtido removendo-se de G toda aresta de custo superior a x. A rotina Teste recebe H, que é um subgrafo gerador de G, e possivelmente outros argumentos necessários para o problema Π, e devolve *ou* uma solução de Π contida

em H^t, para algum inteiro positivo t, *ou* um certificado de que não existe em H solução de Π.

MetodoGargalo(G, c) ▷ para um problema de gargalo Π

1 Ordene as arestas de modo que $c_1 \leq c_2 \leq \cdots \leq c_m$
2 $i \leftarrow 1$
3 **repita**
4 $\quad G_i \leftarrow$ Gargalo(G, c_i)
5 $\quad S \leftarrow$ Teste(G_i)
6 $\quad i \leftarrow i + 1$
7 **até** S ser uma solução em G.
8 **devolva** S

Lema 2.17. *Na iteração i do algoritmo* MetodoGargalo*, as linhas 4 e 5 formam um algoritmo de decisão t-relaxado para o seguinte problema: "existe em G uma solução de Π com custo no máximo c_i?"*

Demonstração. Pela maneira como o grafo G_i é produzido pela rotina Gargalo, vale que $\max(G_i) = c_i$.

Se o S devolvido pela rotina Teste for uma solução de Π em G_i^t, então, pelo Lema 2.16, tal S é uma solução em G com valor $\text{val}(S) \leq t \cdot \max(G_i) = t \cdot c_i$.

Se S é um certificado de que não existe solução em G_i, então não existe em G solução que use apenas arestas de custo no máximo c_i, e portanto S é, de fato, um certificado de que $\text{OPT} > c_i$. □

Teorema 2.18. *O algoritmo* MetodoGargalo *é uma t-aproximação para o problema de gargalo para Π.*

Demonstração. Seja i^* a iteração em que S recebe uma solução em G. Pelo Lema 2.17, vale que $\text{val}(S) \leq t \cdot c_{i^*}$. Note que $\text{OPT} \geq c_{i^*}$:

- Se $i^* = 1$, então $c_{i^*} = c_1 \leq \text{OPT}$, uma vez que o valor ótimo é no mínimo o valor da aresta mais barata.
- Se $i^* > 1$, pelo Lema 2.17, vale que $\text{OPT} > c_{i^*-1}$. Isso porque, na iteração $i^* - 1$, a rotina Teste devolveu um certificado de falha. Como o valor ótimo é o valor de alguma aresta, $\text{OPT} > c_{i^*-1}$ implica que $\text{OPT} \geq c_{i^*}$.

Assim, $\text{val}(S) \leq t \cdot c_{i^*} \leq t \cdot \text{OPT}$. □

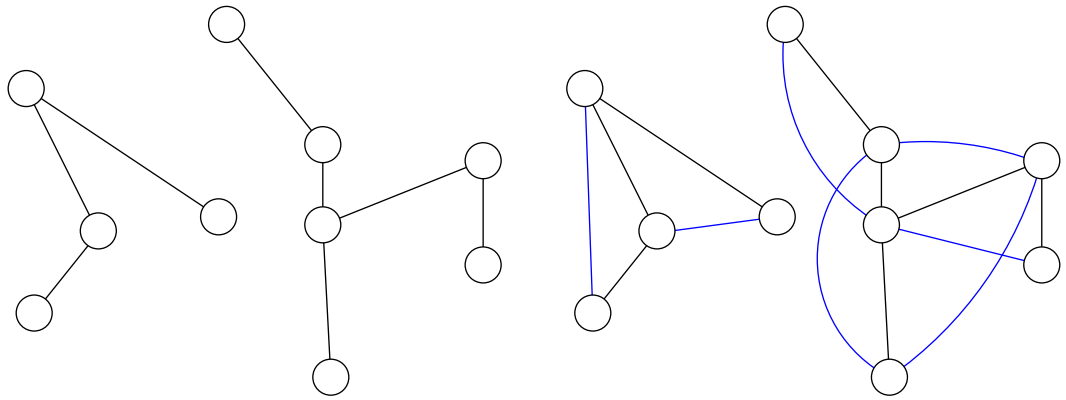
Para usar o algoritmo MetodoGargalo para um problema de gargalo, basta encontrar um t e uma rotina Teste adequados. Mostraremos a seguir como aplicar essa técnica para obter outra aproximação para o problema dos k-centros.

2.3.1 Método do gargalo para o problema dos k-centros

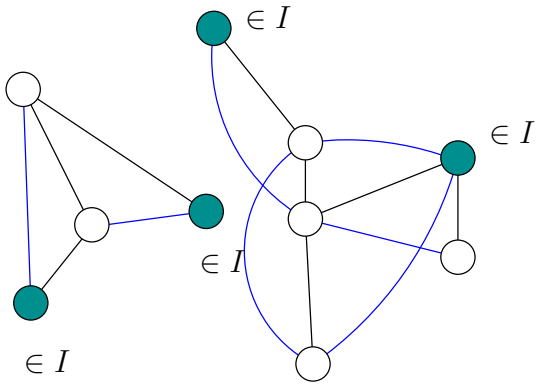
Já vimos que o algoritmo de Gonzalez para o k-clustering também é uma 2-aproximação para o k-centros. Daremos, agora, uma outra 2-aproximação para o k-centros, usando o método geral que acabamos de descrever.

Definição 2.19. *Se a instância do* k-CENTROS *consiste de um grafo completo* $G = (V, E)$ *com custos* c_e *nas arestas respeitando a desigualdade triangular e um inteiro positivo* k*, uma* solução *é um conjunto* $S \subseteq V$ *com* $|S| \leq k$*. Se* H *é um subgrafo gerador de* G*, um tal* S *é* solução em H *se todo vértice de* V *é vizinho, em* H*, de pelo menos um elemento de* S*.*

Para obter uma 2-aproximação, queremos uma rotina TESTE com a propriedade desejada para $t = 2$. No caso, é bastante simples: para o grafo G_i, a rotina constrói o grafo G_i^2, como na Figura 2.3 e devolve um conjunto independente maximal I em G_i^2, como na Figura **??**.



(a) O grafo G_i (b) O grafo G_i^2



(c) Um conjunto independente maximal I em G_i^2

Figura 2.3: *Em uma dada iteração* i *do método do gargalo, constrói-se o grafo* G_i*, e considera-se o grafo* G_i^2*, onde é escolhido um conjunto independente maximal* I*, destacado.*

Se $|I| \leq k$, então I é uma solução em G_i^2, pois todo vértice é vizinho, nesse grafo, de um centro, pela maximalidade de I. Do Lema 2.16, segue que

$$\text{val}\,(I) \leq t \cdot \max(G_i) = 2 \cdot c_i.$$

Se $|I| > k$, então I é um certificado de que OPT $> c_i$. De fato, seja $S \subseteq V$ com $|S| = k$, e suponha que todo vértice é vizinho, em G_i, de algum dos centros de S. Como $|I| > |S|$, existe um vértice $v \in S$ que é vizinho, em G_i, de pelo menos dois vértices de I, digamos u e w. Mas isso é absurdo, pois se $u, w \in I$, então u e w não têm vizinhos em comum em G_i, caso contrário seriam adjacentes em G_i^2, contrariando a hipótese de que I é um conjunto independente em G_i^2.

Assim, a rotina TESTE acima descrita tem a propriedade que desejamos: devolve ou um certificado de falha, ou uma solução de valor no máximo $t \cdot \max(G_i) = 2 \cdot c_i$. Segue

do Teorema 2.18 que o algoritmo MetodoGargalo, com a rotina teste acima, é uma 2-aproximação para o problema dos k-centros.

Podemos descrever o algoritmo de modo mais explícito, a fim de ilustrar melhor a própria técnica geral.

Gargalo-k-Center(G, c, k)

```
1  Ordene as arestas de modo que c_1 ≤ ··· ≤ c_m
2  i ← 1
3  repita
4      G_i ← Gargalo(G, c_i)
5      S ← conjunto independente maximal em G_i^2
6      i ← i + 1
7  até |S| ≤ k
8  devolva S
```

Teorema 2.20. *O algoritmo* Gargalo-k-center *é uma 2-aproximação para o problema dos k-centros.*

Demonstração. Segue da discussão anterior. □

Capítulo 3

Centros capacitados

Veremos aqui uma generalização do problema dos k-centros. A entrada para essa variante é similar à do problema original, adicionando-se um parâmetro: um inteiro L que representa a capacidade de um centro. Em vez de simplesmente escolher os centros, agora teremos que decidir também a que conjunto de vértices cada um deles irá servir, respeitando a restrição de que nenhum centro tenha atribuídos a si mais do que L vértices. Note que, nesta variante, todos os centros têm a mesma capacidade.

A motivação dessa restrição é evitar a sobrecarga de um centro. É uma preocupação, por exemplo, para quando o problema dos k-centros é interpretado como problema de localização de instalações (facilidades), e não desejamos que nenhuma instalação fique sobrecarregada. Para o problema de categorização, podemos querer proibir que uma categoria possa ser muito mais populosa do que as outras. Por isso, ao proporem esta variante pela primeira vez, Bar-Ilan, Kortzars e Peleg [4] se referiram a ela como uma versão "balanceada" dos k-centros. Os autores apresentaram uma 10-aproximação para este problema.

Posteriormente, Khuller e Sussman [13] obtiveram algoritmos de aproximação que atingem razão 6, para este mesmo problema, e 5, para uma variante um pouco mais simples. Tais garantias de desempenho ainda são as melhores que conhecemos para os respectivos problemas [5]. A seguir veremos a descrição desses problemas e os respectivos algoritmos.

3.1 Definição dos problemas

3.1.1 Problema dos k-centros capacitados

Problema dos k-centros capacitados (CKC): *Dados um grafo G completo com custos c_e nas arestas respeitando a desigualdade triangular, e inteiros positivos k e L, escolher um par (S, ϕ), onde $S \subseteq V$, $|S| \leq k$, $\phi\colon V \to S$ e*

$$|\{v \in V : \phi(v) = u\}| \leq L, \text{ para todo } u \in S, \tag{CAP}$$

que minimize o valor

$$\max_{v \in V} d_G(v, \phi(v)).$$

Definição 3.1. *Para o problema acima definido, dizemos que (S, ϕ), com $S \subseteq V$, $\phi\colon V \to S$, é* solução *se $|S| \leq k$ e ϕ respeita (CAP). Se H é um subgrafo gerador de G, um tal S é* solução em H *se todo vértice de V é vizinho, em H, de $\phi(S)$.*

O conjunto S é nosso conjunto de centros, e ϕ é a atribuição dos vértices aos centros. Note que um vértice é servido mesmo que tenha sido escolhido como centro. Além disso, não

necessariamente um centro será atribuído a si mesmo. Note o exemplo simples representado pela figura 3.1, com $k = 2$ e $L = 3$. Considere que os custos das arestas desenhadas são todos iguais a 1, e o custo de uma aresta omitida é o custo do caminho de custo mínimo no grafo da figura. A solução ótima, que tem valor 1, escolhe como centros os dois vértices destacados com uma circunferência ao redor: o "centro azul" (à esquerda) e o "centro vermelho" (à direita). Os vértices preenchidos com padrão vermelho são atribuídos ao centro vermelho, e os vértices azuis, ao centro azul. O centro vermelho é atribuído ao centro azul.

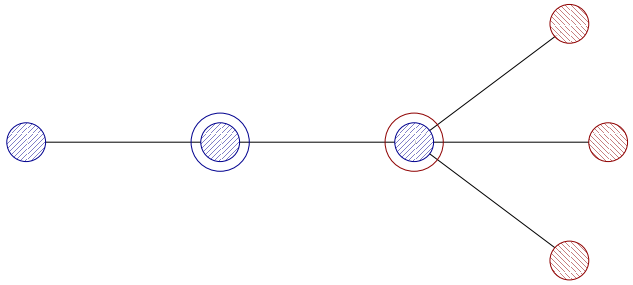


Figura 3.1: *Exemplo de instância do problema acima em que um centro não é atribuído a si mesmo numa solução ótima. O centro da direita é atribuído ao da esquerda.*

Khuller e Sussman [13] se referem a esse problema como CKC (sigla em inglês para *capacited k-center*), e apresentam uma 6-aproximação para ele. Para facilitar o entendimento do algoritmo, no entanto, antes eles consideram um outro problema, similar, para o qual mostram uma 5-aproximação que introduz várias das ideias a serem retomadas posteriormente pelo algoritmo para o CKC.

3.1.2 Problema dos k-multicentros com capacidades

Esse problema "auxiliar" é denominado *problema dos k multicentros com capacidades*. A entrada e o objetivo desse problema são iguais aos do CKC, exceto que temos permissão para "abrir" mais de um centro no mesmo vértice. Ou seja, podemos escolher um vértice como centro múltiplas vezes: S é um multiconjunto. Se escolhermos abrir r centros sobre o vértice v, então v poderá ter atribuídos a si até $r \cdot L$ vértices, pois cada centro tem capacidade L. Cada um desses r centros conta individualmente para o limite de k a serem abertos.

Note que qualquer solução viável do CKC é viável nessa variante, com o mesmo valor objetivo. Isto é, essa variante é uma relaxação do CKC. Se quisermos transformar uma solução viável do k-multicentros com capacidades em uma solução do CKC, teremos que escolher centros adicionais e distribuir os vértices atribuídos a cada multicentro escolhido mais de uma vez, preferencialmente para centros próximos no grafo, se não quisermos alterar demasiadamente o valor da solução. Além disso, é importante observar que não limitamos a quantidade de centros a serem abertos em multiplicidade – poderíamos ter os k centros todos no mesmo lugar. Isso porque, nesse problema, não balanceamos a carga entre os vértices escolhidos como centros, mas entre os centros em si. Portanto, se quisermos adaptar uma solução da relaxação para uma do problema original, deveremos ter algum cuidado tanto na escolha original dos vértices a receberem centros quanto na subsequente redistribuição, a fim de buscar garantir uma boa razão de aproximação.

Por vezes chamamos esse problema de versão flexível, ou *soft*, do CKC, pois trata-se de uma variante mais permissiva.

3.2 Algoritmos para o k-centros capacitados e sua relaxação

Essa seção apresenta os algoritmos de aproximação para o CKC e sua relaxação, dados por Khuller e Sussman [13]. Apresentamos primeiro o algoritmo para a relaxação, e a seguir descrevemos os passos adicionais necessários para obter a 6-aproximação para o CKC.

3.2.1 Um algoritmo para a versão flexível

Para esse problema, voltaremos a usar a técnica geral para problemas de gargalo de Hochbaum e Shmoys [11], descrita no Capítulo 2.3. Com isso, atingiremos a razão de aproximação 5, via uma rotina de decisão 5-relaxada que chamaremos de AtribuiCentros.

Seja (G, c, k, L) uma instância do CKC flexível. Podemos descrever brevemente tal algoritmo da seguinte maneira. O primeiro passo é construir G_i^2 e nele encontrar um conjunto independente maximal I. Esse conjunto não é escolhido arbitrariamente, mas sim através de um processo bem definido de busca em largura que descreveremos adiante. Cada vértice em I é chamado de *monarca*. A cada monarca escolhido associamos um *império*, que são seus vizinhos em G_i^2 ainda não incluídos em nenhum império anterior. Também durante a escolha dos monarcas é realizada a construção de uma *árvore de monarcas*. Não se trata de um subgrafo de G_i, mas de um grafo construído para representar uma relação de precedência entre monarcas dada pela busca em largura.

Após essa escolha do conjunto independente e construção das estruturas associadas, etapa chamada de "atribuição de monarcas", o algoritmo realiza uma primeira atribuição de vértices a centros. Cada monarca é escolhido como um centro e tenta coletar L vértices, dando prioridade àqueles em seu império. Os vértices atribuídos a um monarca nessa etapa são chamados de seu *domínio*. Se ao fim da etapa de "atribuição de domínios" ainda há vértices *descobertos*, isto é, não atribuídos a nenhum centro, o algoritmo realiza o passo final, chamado de "reatribuição". São escolhidos os monarcas que recebem centros adicionais a fim de poder-se estabelecer uma atribuição completa de vértices a centros. O nome desse passo se deve ao fato de que possivelmente alguns vértices acabam trocando de domínio, como veremos.

AtribuiCentros(G_i)

1 Para cada componente conexa G_i^c de G_i, seja n^c seu número de vértices.
2 **se** $\sum_{G_i^c} \left\lceil \frac{n_i^c}{L} \right\rceil > k$
3 **devolva** FALHA
4 $S \leftarrow \emptyset$, $\phi \leftarrow \emptyset$
5 **para cada** componente conexa G_i^c de G_i
6 $(S_i^c, I, p) \leftarrow$ EscolheMonarcas(G_i^c)
7 $\phi_i^c \leftarrow$ AtribuiDomínios(G_i^c, S_i^c, I)
8 $(S_i^c, \phi_i^c) \leftarrow$ CobreEReatribui($G_i^c, S_i^c, \phi_i^c, p$)
9 $S \leftarrow S \cup S_i^c$
10 $\phi \leftarrow \phi \cup \phi_i^c$
11 **devolva** (S, ϕ)

A rotina tenta realizar a escolha de centros e a atribuição tratando cada componente de G_i isoladamente. Isso pode ser feito porque uma solução com valor não superior a c_i não pode ter um centro ao qual sejam atribuídos vértices em componentes distintas de G_i. Ainda

devido a esse argumento, um limitante inferior para o número de centros usados é dado pelo número de centros necessário para atender a cada componente. Se a componente G_i^c tem n_i^c vértices, então uma solução de valor até c_i precisa de pelo menos $\left\lceil \frac{n_i^c}{L} \right\rceil$ centros para essa componente. Daí o critério de falha na linha 2 da rotina AtribuiCentros.

Portanto, feita a verificação desse limitante inferior para o número de centros necessários, o algoritmo consiste na realização, para cada componente conexa de G_i, dos processos que descrevemos brevemente a seguir.

EscolheMonarcas executa uma busca em largura, na dada componente de G_i, para a escolha dos monarcas e seus impérios. A busca ainda produz uma "árvore de monarcas". No algoritmo apresentado a seguir, utilizamos uma fila Q para realizar a busca em largura, e um vetor p, indexado pelos vértices do grafo, para representar a árvore de monarcas. O império de cada monarca, como veremos, será dado por dois vetores de conjuntos, Imp_1 e Imp_2, indexados pelos monarcas e armazenados, no algoritmo acima, no par $I = (\text{Imp}_1, \text{Imp}_2)$. O conjunto S no algoritmo é o conjunto de monarcas.

```
EscolheMonarcas(G_i^c)
 1  Q ← ∅
 2  S ← ∅
 3  r ← vértice arbitrário de G_i^c
 4  para cada vértice v de G_i^c
 5       visitado[v] ← FALSO
 6  p[r] ← r
 7  Q.INSERE(r)
 8  enquanto Q ≠ ∅
 9       m ← Q.REMOVE()
10       visitado[m] ← VERDADEIRO
11       S ← S ∪ {m}  ▷ m se torna um monarca
12       Imp_1[m] ← ∅, Imp_2[m] ← ∅
13       para cada vizinho v de m
14            se visitado[v] = FALSO
15                 visitado[v] ← VERDADEIRO
16                 Imp_1[m] ← Imp_1[m] ∪ {v}
17            para cada vizinho u de v
18                 se visitado[u] = FALSO
19                      visitado[u] ← VERDADEIRO
20                      Imp_2[m] ← Imp_2[m] ∪ {u}
21       para cada u ∈ Imp_2[m]
22            para cada vizinho w de u em G_i^c
23                 se visitado[w] = FALSO
24                      Q.INSERE(w)
25                      p[w] ← m
26  I ← (Imp_1, Imp_2)
27  devolva (S, I, p)
```

Para cada monarca m escolhido, definimos o império $\text{Imp}(m) = \{m\} \cup \text{Imp}_1[m] \cup \text{Imp}_2[m]$ de m. Os *súditos* que formam esse império são todos os vértices que estão à distância no máximo 2 de m e ainda não foram incluídos em qualquer outro império. Os súditos são divididos em dois níveis: $\text{Imp}_1(m)$ contém aqueles que são vizinhos de m em G_i^c, e $\text{Imp}_2(m)$, aqueles que estão à distância 2 de m em G_i^c.

Definimos o primeiro monarca, o vértice r, como a raiz da árvore. À exceção dele, um vértice w que se torna monarca é escolhido por ser vizinho ainda não marcado de um membro de $\mathrm{Imp}_2[m]$ para algum monarca m. Então w torna-se *filho* de m na árvore.

O vetor p do algoritmo, restrito aos monarcas, ou seja, ao conjunto S, nos dá a árvore de monarcas, que representa a hierarquia dada pela ordem de escolha na busca em largura.

A distância em G_i entre um monarca m e seu pai $p[m]$ é sempre igual a 3. Esta é a menor distância possível entre dois monarcas. Além disso, todo vértice está à distância no máximo 2 de um monarca; em particular, do monarca de seu império. Observe ainda que cada vértice é vizinho de no máximo um monarca. Finalmente, note que cada monarca, à exceção da raiz, é vizinho de pelo menos um membro "nível 2" do império de seu pai.

AtribuiDomínios realiza uma primeira atribuição de vértices a monarcas. Nessa fase do algoritmo, cada monarca recebe no máximo L vértices, e só recebe vértices de outros impérios caso todos os seus próprios súditos já pertençam a algum domínio. Além disso, para cada vértice v, a rotina só considera candidatos a receberem v os monarcas que estão à distância no máximo 2 de v em G_i.

Para isso, a rotina resolve um problema de fluxo máximo com custo mínimo. Isto é, encontra um fluxo de custo mínimo dentre os fluxos máximos, para a instância de fluxo que descrevemos a seguir.

Considerando o grafo $H = G_i^c$, entrada do algoritmo EscolheMonarcas, e o conjunto $S \subseteq V$ de monarcas produzido pelo algoritmo, o grafo da instância do problema de fluxo é o grafo orientado $H' = (V \cup M \cup \{s,t\}, E')$, onde M é uma cópia do conjunto S de monarcas, com $M \cap V = \emptyset$, s e t são uma fonte e um sorvedouro artificiais tais que $\{s,t\} \cap (M \cup V) = \emptyset$, e os arcos são

$$\begin{aligned} E' = &\{(m', v) \in M \times V : m' \text{ é cópia do monarca } m \text{ tal que } d_H(m, v) \leq 2\} \\ &\cup \{(s, m) : m \in M\} \\ &\cup \{(v, t) : v \in V\}. \end{aligned}$$

A ideia é que nossa primeira tentativa de atribuição seja dada por um fluxo nessa rede. Queremos definir o *domínio* $\mathrm{Dom}(m)$ do monarca m como o conjunto de vértices que recebem fluxo da cópia de m no problema de fluxo. Digamos que a cópia do monarca m envia uma unidade de fluxo para cada vértice em seu domínio. Note que apenas vértices à distância no máximo 2 do monarca m são candidatos a receberem fluxo da cópia de m.

Para respeitar as capacidades dos monarcas, basta fazer com que cada $m' \in M$, por sua vez, receba no máximo L unidades de fluxo da fonte s da rede. Assim, as capacidades dos arcos, dadas pelo vetor u, são

- Para cada $m' \in M$, $u(s, m') = L$.
- Para cada $v \in V$, $u(v, t) = 1$.
- Para cada $(m', v) \in (M \times V) \cap E'$, $u(m', v) = 1$.

Dado um fluxo máximo, isto é, que realiza o máximo número de atribuições de vértices a domínios, queremos selecionar aquele de custo mínimo. Como descrevemos anteriormente, desejamos que um monarca priorize seu próprio império na hora de coletar vértices para seu domínio. Esse intuito será codificado pelos custos dos arcos na rede.

Para cada monarca $m \in S$, e cada vértice $v \in V$ tal que $d_H(m, v) \leq 2$ mas $v \notin E(m)$, isto é, v não é súdito de m, o custo do arco (m', v), onde $m' \in M$ é a cópia de m, é 1. Os demais arcos têm custo 0.

Assim, dentre os fluxos que maximizam a quantidade de vértices satisfeitos (atribuídos a um domínio), os melhores são aqueles de custo mínimo, ou seja, aqueles em que cada monarca prefere satisfazer seus súditos antes de eventualmente satisfazer membros de outro império.

Dados o grafo H e os custos e capacidades descritos acima, encontramos um fluxo ϕ^*, que é um fluxo de custo total mínimo dentre todos os fluxos máximos sobre essa rede. Sabemos que isso pode ser feito em tempo polinomial [1]. De fato, dada a instância do fluxo, podemos encontrar em tempo polinomial um fluxo máximo, com valor, digamos, F. Então, dada a instância e fixado o valor F, temos também um algoritmo eficiente que encontra um fluxo de custo mínimo dentre os fluxos com valor F.

Fixada uma solução ótima φ para o problema de fluxo, definimos para cada monarca $m \in S$ seu domínio

$$\mathrm{Dom}(m) = \{v \in V : v \text{ recebe fluxo da cópia de } m \text{ em } \varphi\}.$$

Para cada monarca m e cada $v \in \mathrm{Dom}(v)$, definimos a primeira tentativa de atribuição $\phi(v) = m$.

A seguir, vamos dar um limitante inferior para o número de centros usados por uma solução ótima em termos do nosso fluxo φ.

Definição 3.2. *Um monarca é*

- leve*, se seu domínio tem cardinalidade* $< L$*.*
- pesado*, se seu império tem vértices não satisfeitos.*
- completo*, caso contrário.*

Lema 3.3. *Se um monarca é* pesado*, então todo vértice em seu domínio é proveniente de seu próprio império.*

Demonstração. Segue da otimalidade do fluxo fixado. De fato, seja m um monarca pesado e suponha que exista $v \in \mathrm{Dom}(m) \setminus \mathrm{Imp}(m)$.

Como m é pesado, existe $u \in \mathrm{Imp}(m) \setminus \mathrm{Dom}(m)$. Então a cópia $m' \in M$ de m poderia deixar de enviar a unidade de fluxo para v e passar a enviá-la para u. O valor do fluxo continuaria igual, e seu custo decresceria em 1 unidade, pois um arco de custo 1 deixaria de transmitir fluxo, e o arco que passa a transmitir fluxo tem custo 0. Mas o fluxo fixado tem custo mínimo, uma contradição. □

Sejam, agora, K_L o número de monarcas *leves*, n_L o número de vértices nos domínios dos monarcas leves, e n o total de vértices do grafo G_i^c considerado.

Definição 3.4. *Definimos recursivamente os conjuntos* ε_j *para* $j \geq 0$*:*

- $\varepsilon_0 = \{m \in S : m \text{ é monarca leve}\}$.
- *Para* $j > 0$, $\varepsilon_j = \varepsilon_{j-1} \cup \{m \in S : \exists v \in V \exists m' \in \varepsilon_{j-1} \ tq \ \phi(v) = m \ e \ d_H(v, m') \leq 2\}$.

Isto é, o conjunto ε_{j+1} é obtido adicionando-se a ε_j os monarcas que têm em seus domínios vértices que poderiam ter sido atribuídos a monarcas em ε_j.

Definição 3.5.

$$\varepsilon = \bigcup_{j \geq 0} \varepsilon_j.$$

Isto é, ε é o maior ε_j que conseguimos obter pelo processo descrito na definição anterior.

Lema 3.6. *O conjunto ε não contém monarcas pesados.*

Demonstração. Seja $m \in \varepsilon$ e suponha que m é pesado. Seja $j > 0$ tal que $m \in \varepsilon_j$ e $m \notin \varepsilon_{j-1}$. Tal j deve existir porque não podemos ter $m \in \varepsilon_0$, já que m não é leve.

Então existem um vértice $v \in V$ e um monarca $m' \in \varepsilon_{j-1}$ tais que $\phi(v) = m$ e $d_G(v, m') \leq 2$.

Portanto, no fluxo ótimo fixado, v recebe fluxo da cópia de m mas também há um arco da cópia de m' para v. Então, a cópia de m pode deixar de enviar fluxo para v e passar a enviar fluxo para um vértice de seu império.

Por sua vez, a cópia de m' pode passar a enviar fluxo para v, e assim v continua satisfeito. Pode ser que m' não seja leve, ou seja, que $j \neq 1$. Nesse caso, precisamos realocar um vértice no domínio de m' para o domínio de um monarca em ε_{j-2}, e assim por diante, até atingir um monarca em ε_0 (leve).

Esse procedimento aumenta em 1 o valor do fluxo. Mas nosso fluxo fixado é máximo por hipótese, uma contradição. □

Lema 3.7. *Suponha que exista uma solução ótima do CKC flexível de custo não superior a c_i para a instância (G, c, k, L) fixada.*

Considere um centro escolhido numa tal solução que cobre um monarca em ε. Tal centro não pode cobrir vértices que não estejam nos domínios dos monarcas em ε.

Demonstração. Fixe uma solução ótima (S^*, ϕ^*) do CKC flexível para a instância considerada. Seja $v \in S^*$, isto é, um vértice sobre o qual há um centro em tal solução. Suponha que v cobre os vértices $m \in \varepsilon$ e $u \in V$.

Suponha que u não pertence a nenhum domínio de monarcas em ε e seja $j = \arg\min\{i \geq 0 \colon m \in \varepsilon_i\}$.

Pela otimalidade da solução fixada, devemos ter $d_G(v, m) = d_G(v, u) = 1$. Isto é, como a solução é ótima e seu valor é no máximo c_i, sabemos que, nessa solução, todo vértice está atribuído a um centro do qual é vizinho em G_i. Isto é, (S^*, ϕ^*) é solução em G_i.

Então, v é vizinho comum a m e u em G_i, de modo que $d_G(m, u) \leq 2$.

Podemos atribuir $\phi(u) = m$. Se $j \neq 0$, passamos um vértice do domínio de m para um monarca em ε_{j-1}, e assim por diante, até chegar a um monarca em ε_0, ou seja, leve.

Esse processo aumenta o fluxo em 1 unidade, o que é uma contradição.

Então existe $m' \in S$ tal que $\phi(u) = m'$. Portanto $d(m', u) \leq 2$, e como $d(m, u) \leq 2$ e $m \in \varepsilon$, segue da definição que $m' \in \varepsilon$. □

Definição 3.8. *Denotaremos por $\mathcal{F} = S \setminus \varepsilon$ o conjunto dos monarcas que não estão em ε.*

Teorema 3.9. *Suponha que exista uma solução ótima de valor no máximo $w(e_i)$.*

Então o número necessário de centros para uma solução ótima é pelo menos $K_L + \left\lceil \frac{n - n_L}{L} \right\rceil$.

Demonstração. Não há vizinho comum a dois monarcas, pela forma como fazemos a busca que escolhe monarcas.

Então cada $m \in \varepsilon$ requer um centro próprio, e nenhum desses centros, pelo Lema 3.7, pode cobrir os monarcas em $\mathcal{F}$ ou os vértices nos domínios destes. Portanto, se n_ε é o número de vértices cobertos pelos monarcas em ε, então precisamos de pelo menos $|\varepsilon| + \left\lceil \frac{n - n_\varepsilon}{L} \right\rceil$. Então

um limitante inferior para o número necessário de centros é

$$\begin{aligned} |\varepsilon| + \left\lceil \frac{n - n_\varepsilon}{L} \right\rceil &= K_L + |\varepsilon \setminus \varepsilon_0| + \left\lceil \frac{n - n_\varepsilon}{L} \right\rceil \\ &= K_L + \frac{n_\varepsilon - n_L}{L} + \left\lceil \frac{n - n_\varepsilon}{L} \right\rceil \qquad (3.1) \\ &= K_L + \left\lceil \frac{n - n_\varepsilon + n_\varepsilon - n_L}{L} \right\rceil \\ &= K_L + \left\lceil \frac{n - n_L}{L} \right\rceil, \end{aligned}$$

onde (3.1) se justifica pelo fato de que cada monarca em $\varepsilon \setminus \varepsilon_0$ é *cheio*, isto é, tem exatamente L vértices em seu domínio. □

Até agora nossa rotina pode deixar vértices descobertos (sem atribuição). Veremos como cuidar disso usando um número de centros que não ultrapassa o limitante inferior dado pelo Teorema 3.9. Assim, se usarmos mais do que k centros, o teorema nos diz que o valor ótimo do problema do CKC flexível para a instância (G, c, k, L) considerada é superior a c_i, de onde segue que, nesse caso, a cardinalidade do nosso conjunto de centros é um certificado de falha para a iteração atual do MetodoGargalo.

A rotina CobreEReatribui irá criar novos centros para satisfazer os vértices que não foram incluídos em nenhum domínio. Essa rotina percorre a árvore de monarcas T a partir das folhas. São criados, em cada monarca m, k' novos nós, para cobrir $k' \cdot L$ vértices, eventualmente deixando $\ell < L$ vértices descobertos. Então esses ℓ vértices são atribuídos a m, e ℓ vértices "antigos", isto é, provenientes do domínio de m, passam a ser de responsabilidade do pai de m na árvore de monarcas, o vértice $p[m]$. Cada monarca passa esse restante para seu pai, até que eventualmente podemos ser obrigados a abrir um centro na raiz r que não ficará cheio.

CobreEReatribui(G_i^c, S^c, ϕ^c, p)

para cada monarca $m \in S^c$
    descobertos$(m) \leftarrow E(m) \setminus \left(\bigcup_{u \in M} \mathrm{Dom}(u)\right)$
Seja T a árvore de monarcas dada pelo vetor p.
para cada $m \in T$
    recebidos$(m) \leftarrow \emptyset$
repita
    $m \leftarrow$ folha de T
    Seja $|\text{descobertos}(m)| + |\text{recebidos}(m)| = k' \cdot L + \ell$, onde $0 \leq \ell < L$
    para $i \leftarrow 1$ **até** k'
        S^c recebe mais uma cópia de m
        Atribua a essa nova cópia L dos vértices em descobertos$(m) \cup$ recebidos(m)
    Atribua os ℓ restantes a m
    Mova ℓ vértices de $\mathrm{Dom}(m)$ para recebidos$(p[m])$.
    Remova m de T
até T ficar vazia
devolva (S^c, ϕ^c)

Teorema 3.10. *Cada vértice é atribuído a um centro que dista dele no máximo em 5 arestas em G_i. Além disso, são usados no máximo $K_L + \left\lceil \frac{n - n_L}{L} \right\rceil$ centros.*

Demonstração. Todo centro que não é um monarca leve cobre L vértices por construção. E como o domínio de um monarca leve não diminui em tamanho em relação à solução do problema de fluxo, usamos não mais do que

$$K_L + \left\lceil \frac{n - n_L}{L} \right\rceil$$

centros.

Além disso, cada vértice, ao final, fica atribuído ou ao monarca do qual recebe fluxo, ou ao pai deste. A distância em G_i é no máximo 2 no primeiro caso, e aumenta de no máximo 3. □

Corolário 3.11. *O algoritmo* AtribuiCentros *é uma rotina de decisão 5-relaxada para o problema dos k-multicentros com capacidades (CKC flexível).*

Demonstração. Pelos teoremas 3.9 e 3.10, sabemos que nosso algoritmo ou encontra um conjunto de centros e uma atribuição em que cada vértice está à distância no máximo $5 \cdot c_i$ do centro ao qual está atribuído, ou um certificado de que o valor ótimo do CKC flexível é superior a c_i. □

Ou seja, o Método do Gargalo com AtribuiCentros como rotina de teste dá uma 5-aproximação para o k-multicentros com capacidades.

3.2.2 Aproximando o CKC

Note que, antes da etapa que cuidava dos vértices deixados sem atribuição (a rotina CobreEReatribui), o algoritmo que acabamos de descrever abria apenas um centro em cada vértice. Portanto, simplesmente mudar um pouco a forma como a cobertura final é realizada pode nos dar uma solução viável para o CKC, em que não podemos abrir multicentros.

Lembre-se de que os vértices pertencentes ao império de um monarca m são chamados seus *súditos*, e são divididos em níveis: o próprio monarca m, os súditos nível 1 (aqueles pertencentes a $\text{Imp}_1(m)$), e os súditos nível 2 (em $\text{Imp}_2(m)$).

Definição 3.12. *Após a execução de* EscolheMonarcas*,*

- *Dado um monarca m, cada $w \in \text{Imp}_2(m)$ foi incluído no império, durante a busca em largura, por um vértice u que é vizinho de m em G_i. Dizemos que* $\text{contato}(w) = u$.
- *À exceção da raiz, cada monarca m foi incluído na fila por um súdito nível 2 de seu pai. Este é chamado seu* cônjuge*, e denotado por $s(m)$.*
- *Se m e n são monarcas tais que $m \neq r \neq n$ e $p[m] = p[n]$, então dizemos que m e n são* irmãos.

Note que cada monarca, exceto a raiz, tem exatamente um cônjuge. Além disso, cada súdito nível 2 de qualquer império pode ser cônjuge de no máximo um monarca. Isso porque, caso contrário, teríamos dois monarcas com um vizinho em comum em G_i, mas já vimos que dois monarcas estão à distância no mínimo 3 em G_i.

Usando essa linguagem, podemos descrever uma 7-aproximação para o CKC. Execute o algoritmo anterior, criando multicentros. Então, mova os centros adicionais de cada monarca m para seus súditos. Isso é possível, porque cada centro que atende os descobertos de m atende L vértices; e para cada filho de m, que pode lhe enviar até $L - 1$ vértices descobertos, há um súdito nível 2 diferente, o cônjuge desse filho, no império de m.

Esse procedimento aumenta em no máximo 2 a distância de um vértice ao seu centro, em relação à solução com multicentros. Assim, atingimos a razão de aproximação de 7. Para a 6-aproximação, precisamos escolher os novos centros com mais cuidado, conforme descreveremos a seguir.

Para cada monarca m, montaremos uma estrutura $T(m)$, que é uma árvore contendo o próprio monarca e alguns vértices de sua vizinhança em G_i^2. Os vértices em $T(m)$ serão os candidatos a receber centros que seriam criados sobre m. Queremos definir $T(m)$ de maneira a garantir que:

(1) Haja vértices suficientes em $T(m)$ para receber centros que atendam os vértices delegados a m por algum de seus filhos; e

(2) Nenhum vértice possa virar centro mais de uma vez.

Para tanto, exigimos que um monarca só possa estabelecer como centros seus próprios súditos ou sua vizinhança em G_i, à exceção de seu cônjuge, como explicaremos logo adiante.

Para cada monarca m, a árvore $T(m)$ terá altura 0 ou 2, e a definimos da seguinte maneira:

- A raiz de $T(m)$ é m.
- As folhas de $T(m)$ são vértices em $\text{Imp}_2(m)$ que são cônjuges de algum monarca.
- Para cada folha de $T(m)$, seu *contato* é incluído em $T(m)$ como seu pai, e filho de m. Note que esse tipo de vértice está na vizinhança de m em G_i, mas não necessariamente em $\text{Imp}_1(m)$.

Observe que, se um vértice v é cônjuge de um monarca m, então ele pode estar em duas árvores: como folha em um $T(m')$ e possivelmente como nó interno de $T(m)$, pois pode ser o contato de uma folha de $T(m)$.

Portanto, estabelecemos que um monarca m pode criar centros em qualquer vértice de $T(m)$, exceto seu cônjuge. Assim, não criamos dois centros no mesmo vértice.

A árvore $T(m)$ será usada para criar centros para atender vértices descobertos passados para m por seus filhos. Vértices passados do monarca m' para m são atribuídos a um dentre os seguintes:

- O cônjuge $s(m')$ de m'. Nesse caso, a distância máxima em G_i dos vértices transferidos para esse centro ao qual são atribuídos é no máximo 3.
- O contato do cônjuge, contato($s(m')$). Nesse caso, a distância em G_i é no máximo 4.
- O cônjuge de n, $s(n)$, onde n é um monarca irmão de m'. Nesse caso, a distância em G_i é no máximo 5.
- O contato do cônjuge de um monarca n, contato($s(n)$), onde n é irmão de m'. Nesse caso, a distância é no máximo 6.
- O monarca m, como no caso do algoritmo para o k-multicentros com capacidades, em que a distância em G_i é no máximo 5.

Um monarca, portanto, não aloca centros nos vértices que são passados a ele. Assim, podemos ter que abrir um centro num vértice já coberto, isto é, atribuído a outro centro. Nesse caso, consideramos que esse novo centro *não* atende a si mesmo, mas a L outros vértices.

3.3 Capacidades não uniformes

A seguir, veremos dois resultados, devidos a An, Bhaskara, Chekuri, Gupta, Madan e Svensson [2] para o problema dos k-centros com capacidades. Os algoritmos desta seção diferem daqueles vistos até agora na medida em que obtêm suas soluções através do arredondamento de relaxações lineares do problema. Isto é, tomaremos, aqui, para cada problema, uma formulação como programa linear inteiro (PLI), cujas restrições de integralidade relaxaremos obtendo um programa linear (PL). A seguir, resolvemos o programa linear, obtendo uma solução ótima y^* possivelmente fracionária. Então, descrevemos uma forma de obter uma solução inteira y a partir de y^*, processo ao qual damos o nome de arredondamento. Tal processo só nos dá uma aproximação caso consigamos arredondar y^* de maneira a controlar a piora no valor objetivo da solução. Como o PL é uma relaxação do problema, o valor de uma solução ótima sua é uma cota inferior para o valor ótimo da instância do problema original. Portanto, se mostrarmos que o valor da solução inteira y assim obtida está a um determinado fator do valor da solução fracionária y^*, teremos mostrado que está também a no máximo esse mesmo fator do valor ótimo do problema.

Aproximações baseadas em arredondamento de PL, conforme discutem Williamson e Shmoys na seção de problemas em aberto de seu livro [16], são interessantes por dois principais motivos. O primeiro é que, para cada instância, temos um limitante para o valor ótimo, que é o valor da solução do PL, com o qual podemos comparar o valor da solução inteira obtida. Williamson e Shmoys chamam isso de garantia *a posteriori*, em contraste com a razão de aproximação, que é uma garantia *a priori*. O segundo é que frequentemente esse tipo de algoritmo é mais fácil de se adaptar ou reaproveitar do que técnicas diretas. Isso se deve, em parte, ao fato de que, partindo da formulação como PLI de um determinado problema, muitas vezes são necessárias poucas modificações para se obterem boas formulações de problemas similares e generalizações.

An et al. [2] usam essa técnica para obter aproximações para o problema dos k-centros com capacidades não uniformes, que generaliza o problema definido na Seção 3.1.1. A diferença é que estamos interessados, aqui, no caso em que as capacidades dos centros não necessariamente são todas iguais.

Na variante chamada $\{0, L\}$-capacitada, os centros podem ter capacidade ou 0 ou L para alguma constante L dada como parte da instância. Para esse problema, veremos que é possível obter uma 6-aproximação, razão idêntica à melhor conhecida para o caso uniforme. Note que esse problema é bastante similar àquele que já apresentamos. No entanto, a possibilidade de determinar que alguns vértices tenham capacidade zero nos dá a flexibilidade adicional de eleger apenas parte dos vértices como candidatos a centros.

A versão mais geral é aquela em que as capacidades são arbitrárias. É dada como parte da entrada uma função que atribui a cada vértice uma capacidade qualquer. Para esse problema, Cygan, Hajiaghayi e Khuller [6] apresentaram a primeira aproximação conhecida com fator constante. Seu algoritmo também é baseado em arredondamento de PL, e tem uma garantia de desempenho que não é calculada explicitamente, mas é estimada na ordem das centenas. Posteriormente, An, Bhaskara e Svensson, e de forma independente Chekuri, Gupta e Madan, conseguiram melhorar tal resultado, obtendo uma 9-aproximação [2]. É o que descrevemos a seguir.

3.3.1 Capacidades arbitrárias

Veremos como usar arredondamento de PL para obter uma 9-aproximação para o problema que definimos a seguir.

Problema dos k-centros com capacidades arbitrárias: *Dados um grafo $G = (V, E)$ completo com um custo c_e para cada aresta e respeitando a desigualdade triangular, uma função de capacidade $L\colon V \to \mathbb{Z}_+$ e um inteiro positivo k, escolher um par (S, ϕ), onde $S \subseteq V$, $|S| \leq k$, $\phi\colon V \to S$ e*

$$|\{v \in V\colon \phi(v) = u\}| \leq L(u), \text{ para todo } u \in S, \tag{3.2}$$

que minimize o valor

$$\max_{v \in V} d_G(v, \phi(v)).$$

Definição 3.13. *Para o problema acima, dizemos que (S, ϕ), com $S \subseteq V$, $\phi\colon V \to S$, é* solução *se $|S| \leq k$ e ϕ respeita (3.2). Se H é um subgrafo gerador de G, um tal S é* solução em H *se todo vértice de V é vizinho, em H, de $\phi(S)$.*

Conforme a definição acima,iremos considerar que nosso problema está definido sobre um grafo H conexo e sem pesos nas arestas. O grafo pode ser suposto conexo pois caso contrário simplesmente aplicamos o algoritmo separadamente a cada componente e verificamos que o total de centros abertos ao total de todas as soluções parciais não supera k. Isso pode ser feito pois os centros de uma componente não podem servir clientes de outras, e portanto as soluções são de fato isoladas.

Mostraremos uma rotina que ou decide que não há solução em H ou obtém uma solução em H^9, isto é, uma solução de distância no máximo 9 em H. Assim, dado um grafo $H = (V, E)$ com capacidades definidas pela função L, considere o programa linear inteiro que descrevemos a seguir. Nele, para cada $u \in V$, há uma variável binária y_u que vale 1, se u é escolhido como centro, e 0, caso contrário. Ou seja, $S = \{u \in V\colon y_u = 1\}$, e $|S| = \sum_{u \in V} y_u$. Além disso, para cada par de vértices u, v, temos uma variável binária x_{uv} que vale 1 se v é atribuído a u, e 0, caso contrário. Portanto, $\phi(v) = u$ se $x_{uv} = 1$. Usualmente, daremos o nome de u a um vértice que é centro ou candidato a centro, e v a um vértice considerado como cliente.

$$\begin{aligned}
\sum_{u \in V} y_u &= k & \\
x_{uv} &\leq y_u, & \forall u, v \in V \\
\sum_{v\colon d_H(u,v) \leq 1} x_{uv} &\leq L(u) \cdot y_u, & \forall u \in V \\
\sum_{u\colon d_H(u,v) \leq 1} x_{uv} &= 1, & \forall v \in V \\
x_{uv} &\in \{0, 1\} & \forall u, v \in V \\
y_u &\in \{0, 1\} & \forall u \in V.
\end{aligned}$$

Definição 3.14. *No programa inteiro acima, chamamos y_u de* variável de abertura *de u e x_{uv} de* variável de atribuição *de v a u. Numa dada solução (x, y), y_u é o* valor de abertura *de u e x_{uv} é o* valor de atribuição *de v a u.*

Afirmamos que o PLI acima modela corretamente o problema de encontrar uma solução em H, que é um problema de viabilidade. O número de centros abertos é igual a k devido à primeira restrição. As restrições seguintes impedem um vértice de se conectar a um centro que não foi aberto, além de forçarem cada vértice a se conectar a exatamente um centro. Além disso, as capacidades dos centros são respeitadas, e um vértice não aberto não tem nada

atribuído a si pela variável x. Assim, cada solução viável do PLI corresponde a uma solução viável do problema original conforme a correspondência que estabelecemos anteriormente entre y e S e entre x e ϕ. De semelhante modo, cada solução do problema pode ser mapeada em uma solução correspondente do PLI. Consideremos agora a respectiva relaxação linear, à qual iremos nos referir por $\mathrm{PL}_k(H, L)$:

$$\begin{aligned}
\sum_{u \in V} y_u &= k \\
x_{uv} &\leq y_u, && \forall u, v \in V \\
\sum_{v:\, d_H(u,v) \leq 1} x_{uv} &\leq L(u) \cdot y_u, && \forall u \in V \\
\sum_{u:\, d_H(u,v) \leq 1} x_{uv} &= 1, && \forall v \in V \\
0 \leq x_{uv} &\leq 1 && \forall u, v \in V \\
0 \leq y_u &\leq 1 && \forall u \in V.
\end{aligned}$$

Tome uma solução (x, y) de $\mathrm{PL}_k(H, L)$. Ainda podemos garantir que cada vértice se conecta a exatamente uma unidade de abertura de centro. No entanto, como não há restrições de integralidade, essa abertura pode estar espalhada entre vários vértices. De semelhante forma, um vértice u pode ter um valor fracionário de abertura, isto é, pode ser que $0 < y_u < 1$. Assim, pelas restrições do PL, u pode ter atribuído a si um valor possivelmente fracionário de até $y_u \cdot L(u)$ clientes.

Definição 3.15. *Dada uma solução (x, y) de $\mathrm{PL}_k(H, L)$ e um vértice u, chamamos de* capacidade aberta em *u o produto $y_u \cdot L(u)$. Dizemos que a* capacidade disponível *de um vértice u é a diferença $y_u \cdot L(u) - \sum_{v \in V} x_{uv}$.*

Isto é, a capacidade disponível de um vértice u é a diferença entre a capacidade aberta em u e o valor total de atribuição de vértices a u. Note que numa solução viável a capacidade disponível em todo vértice é não negativa.

Estaremos interessados em transferir valores de abertura entre vértices de modo a obter uma quantia inteira, isto é, 0 ou 1, de abertura em todos os vértices. Quando transferimos abertura de um vértice a outro, devemos levar em conta as capacidades dos vértices envolvidos na transferência. Intuitivamente, não permitimos que a transferência acumule abertura em vértices com baixa capacidade, pois podemos assim perder viabilidade. Uma maneira de evitar esse risco é realizar o processo através das chamadas r-transferências, que definimos da seguinte forma.

Definição 3.16. *Se y e y' são vetores em $\mathbb{Q}_+^V$, chamamos y' de* r-transferência de (H, L, y) *se*

1. $\sum_{v \in V} y'_v = \sum_{v \in V} y_v$, *e*
2. $\sum_{v \in V:\, d(v,U) \leq r} L(v) y'_v \geq \sum_{u \in U} L(u) y_u$.

O valor r é chamado de distância *da transferência y'. Se y' é o vetor característico de um conjunto $S \subseteq V$, dizemos que S é uma* r-transferência (inteira) *de (H, L, y).*

Ou seja, y' é uma r-transferência se é um rearranjo dos valores de abertura de y em que, para cada conjunto U de vértices, a capacidade aberta, em y', de uma vizinhança de

tamanho r de U não é menor que a capacidade aberta de U em y. O parâmetro r define o tamanho da vizinhança para a qual a capacidade aberta de um conjunto de vértices pode se distanciar como resultado da transferência.

Uma propriedade que será útil é que a operação de transferência é passível de composição, conforme o lema a seguir.

Lema 3.17. *Se y' é uma r-transferência de (H, L, y) e y'' é uma r'-transferência de (H, L, y'), então y'' é uma $(r + r')$-transferência de (H, L, y).*

Demonstração. Basta verificar as condições da Definição 3.16. Em primeiro lugar, temos $y''(V) = y'(V) = y(V)$, como desejado. Agora fixe um conjunto U de vértices, e defina $U' = \{v \in V : d(v, U) \leq r\}$. Então

$$\begin{aligned}
\sum_{v \in V\colon d(v,U) \leq r+r'} L(v)y''_v &= \sum_{v \in V\colon d(v,U') \leq r'} L(v)y''_v \\
&\geq \sum_{u \in U'} L(v)y'_v \qquad (3.3) \\
&= \sum_{v \in V\colon d(v,U) \leq r} L(v)y'_v \\
&\geq \sum_{u \in U} L(v)y_v, \qquad (3.4)
\end{aligned}$$

o que conclui a prova. A desigualdade (3.3) vem do fato de que y'' é r'-transferência de (H, L, y'), e a desigualdade (3.4) vem do fato de que y' é uma r-transferência de (H, L, y). □

O algoritmo que veremos arredonda uma solução do programa linear $\mathrm{PL}_k(H, L)$ através de uma transferência como a definida acima. Como uma r-transferência controla o espalhamento da capacidade aberta em cada conjunto de vértices, um conjunto S que é uma r-transferência de (H, L, y), para alguma solução (x, y) do PL, é capaz de atender os vértices com uma perda controlada em relação à distância atingida pela solução fracionária do PL. Isso é formalizado no lema a seguir.

Lema 3.18. *Seja $H = (V, E)$ um grafo conexo com capacidades $L\colon V \to \mathbb{Z}_+$. Se (x, y) é uma solução viável de $\mathrm{PL}_k(H, L)$ e S é uma r-transferência de (H, L, y), então existe uma atribuição $\phi\colon V \to S$, que pode ser obtida em tempo polinomial, que respeita as capacidades dos vértices em S e tal que $d_H(v, \phi(v)) \leq r + 1$ para todo vértice v. Além disso, $|S| = k$.*

Demonstração. O vetor característico de S, que denotamos por y^S, é uma r-transferência de (H, L, y). Portanto, pela definição de r-transferência e pela viabilidade de (x, y) para $\mathrm{PL}_k(H, L)$, sabemos que

$$|S| = \sum_{v \in V} y^S_v = \sum_{v \in V} y_v = k,$$

provando a última afirmação do enunciado.

Considere, agora, o grafo (V, W)-bipartido H', em que W é construído adicionando-se $L(u)$ cópias de u para cada $u \in S$, digamos $u_1, \ldots, u_{L(u)}$, onde cada u_i é adjacente aos vértices de V que são vizinhos de u em H^{r+1}. Suponha que M seja um emparelhamento que cobre V nesse grafo. Então a cada $v \in V$ incide exatamente uma aresta de M. Essa aresta é $u_i v$ para algum $u \in S$ e algum $i \in [L(u)]$, e então definimos $\phi(v) = u$. Assim, todo vértice é atribuído a algum centro a distância no máximo $r + 1$, e para cada $u \in S$ existem no

máximo $L(u)$ arestas de M incidentes a cópias de u, de modo que ϕ respeita as capacidades dos centros:

$$\left|\phi^{-1}(u)\right| = |\{v \in V : u_i v \in M \text{ para algum } i \in [L(u)]\}| \leq |[L(u)]| = L(u).$$

Sabemos que é possível encontrar um emparelhamento máximo em tempo polinomial no tamanho do grafo [1]. Além disso, o grafo H' pode ser construído em tempo polinomial no tamanho de H. Isso porque podemos considerar, sem perda de generalidade, que para todo $u \in V$, vale que $L(u) \leq |V|$, uma vez que cada vértice pode ter atribuído a si não mais do que todos os vértices. Então, $|W| \leq |V| \cdot |S| \leq |V|^2$, que é polinomial no tamanho de H.

Resta provar que um emparelhamento máximo no grafo bipartido H' definido acima necessariamente cobre o conjunto V, garantindo assim que ϕ sempre pode ser obtida, como desejado. Para tanto, mostraremos que H' satisfaz a *condição de Hall* para o conjunto V. Isto é, que para todo conjunto $U \subseteq V$, existem em H' pelo menos $|U|$ vizinhos de U. Um resultado clássico sobre emparelhamentos, o Teorema de Hall, diz que essa condição é necessária e suficiente para a existência de um emparelhamento que cobre todos os vértices de V [1].

Tome um conjunto $U \subseteq V$ qualquer. Primeiro, note que

$$\begin{aligned}
|U| &= \sum_{v \in U} 1 \\
&= \sum_{v \in U} \sum_{u\colon uv \in E} x_{uv} && (3.5) \\
&= \sum_{u \in U \cup N_H(U)} \sum_{v \in U\colon uv \in E} x_{uv} \\
&\leq \sum_{u \in U \cup N_H(U)} \sum_{v \in V\colon uv \in E} x_{uv} \\
&\leq \sum_{u \in U \cup N_H(U)} L(u) \cdot y_u, && (3.6)
\end{aligned}$$

onde (3.5) e (3.6) vêm do fato de que (x, y) é solução de $\mathrm{PL}_k(H, L)$.

Como S é uma r-transferência de (H, L, y), tomando o conjunto $U' = U \cup N_H(U)$, temos

$$\sum_{u \in U'} L(u) \cdot y_u \leq \sum_{u \in V\colon d_H(u,U') \leq r} L(u) \cdot y_u^S,$$

ou seja,

$$\begin{aligned}
|U| &\leq \sum_{u \in V\colon d_H(u,U') \leq r} L(u) \cdot y_u^S \\
&= \sum_{u \in V\colon d_H(u,U) \leq r+1} L(u) \cdot y_u^S \\
&= \sum_{u \in S\colon d_H(u,U) \leq r+1} L(u) \\
&= |\{w \in W : d_{H'}(w, U) = 1\}|,
\end{aligned}$$

o que nos dá a condição de Hall para o conjunto V no grafo H'. □

O Lema 3.18 nos diz, em outras palavras, que, dada uma solução (x, y) do $\mathrm{PL}_k(H, L)$, uma

r-transferência inteira de (H, L, y) nos dá uma solução em H^{r+1}. Isso significa que se, para qualquer instância (H, L, k), formos capazes de obter de forma eficiente uma r-transferência inteira de (H, L, y) para qualquer solução (x, y) do $\mathrm{PL}_k(H, L)$, então temos uma $(r + 1)$-aproximação para o problema dos k-centros com capacidades arbitrárias. Adiante veremos que de fato conseguimos sempre obter uma tal r-transferência inteira com $r = 8$, o que nos dá uma 9-aproximação.

Em geral, os algoritmos que desenvolvemos para encontrar r-transferências operam realizando transferências vértice a vértice. Isto é, tomando um par de vértices s e t e, para um valor δ, subtraindo-se δ do valor de abertura de s e somando-se δ ao valor de abertura de t. Vejamos a relação entre esse tipo de passo e a distância da transferência resultante.

Definição 3.19. *Sejam s e t em V tais que $L(t) \geq L(s)$ e y um vetor em $\mathbb{Q}_+^V$. Uma* transferência pontual de s a t de (H, L, y) *é um vetor y' em $\mathbb{Q}_+^V$ tal que $y'_v = y_v$ para todo vértice $v \notin \{s, t\}$, $y'_s = y_s - \delta$ e $y'_t = y_t + \delta$, para algum $\delta \leq \min\{y_s, 1 - y_t\}$. Dizemos que $d_H(s, t)$ é a* distância *da transferência pontual y'. Dizemos que s* cede abertura *e t* recebe abertura *em y'. Se $\delta = \min\{y_s, 1 - y_t\}$, dizemos que y' é uma* transferência máxima *de s a t em (H, L, y).*

Lema 3.20. *Se y' é uma transferência pontual de s a t de (H, L, y), então y' é uma $d_H(s, t)$-transferência de (H, L, y). Se y'' é uma transferência pontual de s' a t' de (H, L, y'), com $s' \neq t$, então y'' é uma $\max\{d_H(s, t), d_H(s', t')\}$-transferência de (H, L, y).*

Demonstração. Como a transferência pontual é uma operação que preserva valor total de abertura, temos $y''(V) = y'(V) = y(V)$, como desejado. Seja U um conjunto de vértices qualquer. Se U não contém s, então

$$\sum_{v \in V\colon d_H(v,U) \leq d_H(s,t)} L(v)y'_v \geq \sum_{v \in U} L(v)y'_v \geq \sum_{v \in U} L(v)y_v.$$

Senão, $s \in U$ e portanto $d_H(t, U) \leq d_H(s, t)$, de modo que

$$\begin{aligned}
\sum_{v \in V\colon d_H(v,U) \leq d_H(s,t)} L(v)y'_v &\geq \sum_{v \in U \cup \{t\}} L(v)y'_v \\
&= \sum_{v \in U \setminus \{s,t\}} L(v)y'_v + L(s)y'_s + L(t)y'_t \\
&= \sum_{v \in U \setminus \{s,t\}} L(v)y_v + L(s)y_s + L(t)y_t + \delta(L(t) - L(s)) \\
&\geq \sum_{v \in U \setminus \{s,t\}} L(v)y_v + L(s)y_s + L(t)y_t \qquad (3.7) \\
&\geq \sum_{v \in U} L(v)y_v,
\end{aligned}$$

onde a desigualdade (3.7) vem do fato de que, pela definição de transferência pontual, temos que $L(t) \geq L(s)$ e $\delta \geq 0$. Isso conclui a prova de que y' é uma $d_H(s, t)$-transferência de (H, L, y).

Se $U \cap \{s, s'\}$ é vazio ou contém s', podemos provar que y'' satisfaz a segunda propriedade da Definição 3.16, com distância $r = \max\{d_H(s, t), d_H(s', t')\}$, de maneira análoga à que fizemos acima para y'. Considere então que $U \cap \{s, s'\} = \{s\}$. Nesse caso não garantimos

que $d_H(t', U) \leq r$, mas sabemos que $s' \neq t$ e portanto t não cede abertura em y'', ou seja,

$$\begin{aligned}\sum_{v \in V\colon d_H(v,U) \leq r} L(v) y''_v &\geq \sum_{v \in U \cup \{t\}} L(v) y''_v \\ &= \sum_{v \in U \setminus \{t\}} L(v) y''_v + L(t) y''_t \\ &\geq \sum_{v \in U \setminus \{t\}} L(v) y'_v + L(t) y'_t \qquad (3.8) \\ &= \sum_{v \in U \cup \{t\}} L(v) y'_v \geq \sum_{v \in U} L(v) y_v,\end{aligned}$$

onde a desigualdade (3.8) vale pelo fato de que $s' \notin U$ e de que $y''_t \geq y'_t$.

Se $\{s, s'\} \subseteq U$, temos

$$\begin{aligned}\sum_{v \in V\colon d_H(v,U) \leq r} L(v) y''_v &\geq \sum_{v \in V \setminus \{s',t'\}\colon d_H(v,U) \leq d_H(s,t)} L(v) y''_v + (L(t') y''_{t'} + L(s') y''_{s'}) \\ &\geq \sum_{v \in V \setminus \{s',t'\}\colon d_H(v,U) \leq d_H(s,t)} L(v) y'_v + (L(t') y'_{t'} + L(s') y'_{s'}) \\ &\geq \sum_{v \in V\colon d_H(v,U) \leq d_H(s,t)} L(v) y'_v \geq \sum_{v \in U} L(v) y_v.\end{aligned}$$

Concluímos que y'' é uma r-transferência de (H, L, y) para $r = \max\{d_H(s,t), d_H(s',t')\}$. □

Ou seja, transferências pontuais são r-transferências válidas, e uma sequência de transferências pontuais em que um vértice que recebe abertura nunca passa a ceder abertura resulta numa transferência com distância total mais controlada do que aquela que o Lema 3.17 nos dá para r-transferências arbitrárias.

Redução a transferências em árvores

Nesta seção, voltaremos a usar a divisão do grafo em impérios, definida na Seção 3.2.2 e à qual An et al. [2] se referem como "particionamento de Khuller e Sussmann". Mostraremos que, se soubermos fazer uma r-transferência inteira sobre árvores cujos vértices têm valores de abertura com determinadas propriedades, então é possível obter uma $(3r+2)$-transferência inteira em um grafo qualquer.

Definição 3.21. *Uma* árvore de transferência *é uma tripla (T, L, y), onde $T = (V, E)$ é uma árvore enraizada em que cada vértice v tem capacidade $L(v) \in \mathbb{Z}_+$, e $\mathbf{0} < y \leq \mathbf{1}$ é um vetor de abertura tal que $\sum_{u \in V} y_u = 1$, e $y_u = 1$ para cada vértice u que não é folha.*

Lema 3.22. *Suponha que exista um algoritmo que, dada uma árvore de transferência (T, L, y), obtém uma r-transferência inteira de (T, L, y) em tempo polinomial. Então existe um algoritmo que, dado um grafo conexo $H = (V, E)$ com capacidades $L\colon V \to \mathbb{Z}_+$ e um inteiro positivo k para o qual $\mathrm{PL}_k(H, L)$ tem uma solução viável (x, y), encontra uma $(3r + 2)$-transferência inteira de (H, L, y).*

Como o objetivo é usar o algoritmo para árvores de transferência como subrotina, devemos pensar em como obter uma tal árvore T a partir de nossa instância original. Uma forma possível é partir da árvore de monarcas conforme definida na Seção 3.2.2 e modificá-la. Para cada monarca v, adicionamos como folhas em T adjacentes a v todos os vértices do império

de v que têm valor de abertura não nulo. Assim, todos os vértices internos (não folhas) em T são monarcas, e o conjunto de monarcas induz em T a árvore de monarcas usual. Para configurar uma árvore de transferência, precisamos que:

1. o valor total de abertura seja inteiro. Isso é verdade, pois adicionamos a T todo vértice com valor de abertura não nulo. Ou seja, $y(T) = y(H) = k$; e

2. o valor de abertura de cada vértice interno seja 1.

O segundo requisito pode ser obtido através de uma 1-transferência, a que chamamos *agregação inicial*. Poderíamos tentar obtê-la transferindo para cada monarca v valor de abertura de sua vizinhança em H até que v tenha valor de abertura 1. Sabemos que é possível fazer composição de transferências, então bastaria fazer a agregação inicial e em seguida aplicar o algoritmo para árvores de transferência sobre T, L e os novos valores de abertura.

No entanto, o processo que acabamos de descrever pode não ser uma 1-transferência, ou seja, uma agregação inicial válida. Isso porque as capacidades dos monarcas podem ser arbitrariamente pequenas, não garantindo o segundo requisito da Definição 3.16.

Uma segunda tentativa poderia ser definir, para cada monarca v, o alvo da agregação inicial como sendo um vértice de capacidade máxima dentre aqueles na vizinhança de v em H, que denotaremos por b_v. Note que exigimos que b_v esteja na vizinhança de v, mas não necessariamente em seu império. Isso resulta numa 1-transferência válida, e podemos substituir, na definição da árvore T, cada monarca v pelo correspondente b_v. A desvantagem dessa solução é que, agora, uma aresta em T ligando dois vértices internos representa uma distância até 5 em H. Isso pode dificultar a análise da distância da transferência, ou resultar em uma análise frouxa.

A solução é modificar o grafo H criando, para cada monarca v, um vértice auxiliar a_v, com capacidade $L(a_v) = L(b_v)$ e abertura inicial $y_{a_v} = 0$. O vértice a_v é adjacente, em H, ao próprio v e a todos os vizinhos de v em H. Portanto, a_v está "no lugar" do monarca v, mas "representando" o vértice b_v. Substituindo cada monarca v pelo correspondente a_v na árvore T, mantemos a propriedade, herdada da árvore de monarcas, de que uma aresta entre dois vértices internos de T corresponde a uma distância exatamente 3 em H. A Figura 3.2 mostra um exemplo de criação de vértice auxiliar para um monarca v.

Assim, podemos realizar a agregação inicial acumulando abertura 1 em cada vértice auxiliar a_v, o que nos dá uma árvore de transferência. Após obter uma r-transferência na árvore, precisamos de uma última 1-transferência em H para que a_v "devolva" sua abertura para o vértice b_v. Dado que uma aresta em T representa distância no máximo 3 em H, a composição das três etapas de transferência nos dá uma $(1 + 3r + 1)$-transferência. O algoritmo é formalizado a seguir.

Demonstração do Lema 3.22. Considere o particionamento de H em impérios conforme visto anteriormente. Defina o grafo H' adicionando a H, para cada monarca v, um novo vértice a_v, adjacente em H' a v e a cada vizinho de v em H. Escolha $b_v \in \arg\max_{u:\, d(u,v)\leq 1} L(v)$ arbitrariamente. Note que, como as vizinhanças dos monarcas são disjuntas, garantimos que $b_{m_1} \neq b_{m_2}$ para quaisquer monarcas m_1 e m_2. Defina $L(a_v) = L(b_v)$ e $y_{a_v} = 0$.

Defina a árvore T da seguinte maneira. Comece com a árvore de monarcas obtida do grafo H. Para cada monarca v e cada u no império de v tal que $y_u > 0$, adicione u a T como uma folha adjacente a v. Ao fim do processo, substitua cada monarca v por seu auxiliar a_v. Se uv é uma aresta de T, temos duas possibilidades. Ou u e v são ambos auxiliares de monarcas, e então $d_H(u, v) = 3$, ou u é auxiliar de monarca e v está em seu império, e então $d_H(u, v) \leq 2$. Ou seja, adjacência em T corresponde à distância no máximo 3 em H.

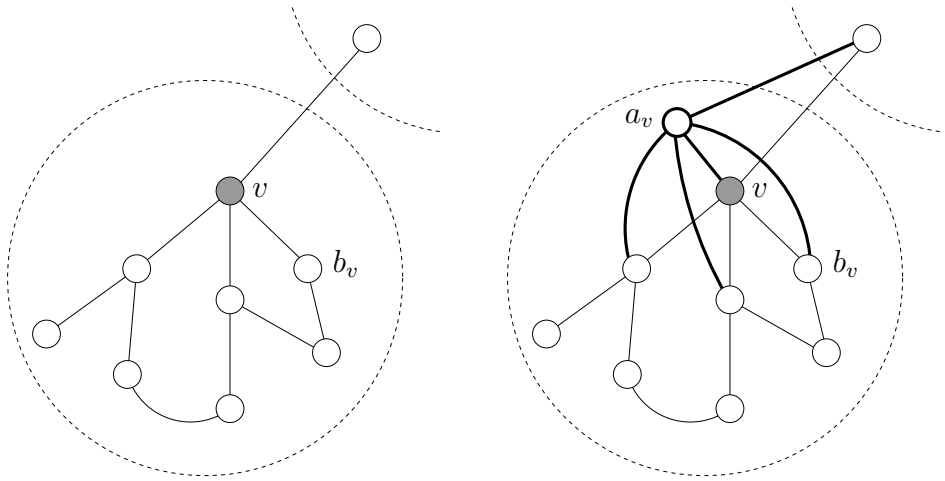


Figura 3.2: *A figura à esquerda mostra o império do monarca v em H. Escolhemos o vértice b_v como um vértice de capacidade máxima dentre aqueles na vizinhança de v. O vértice auxiliar a_v é criado em H', na figura à direita, com a mesma capacidade de b_v, e é adjacente, em H', a v e a todos os vizinhos de v. Para qualquer vértice $u \notin \{v, a_v\}$, temos $d_{H'}(u, a_v) = d_H(u, v)$.*

Tome um monarca v. Sabemos, pela viabilidade de (x, y), que o valor de abertura na vizinhança de v é pelo menos 1. O mesmo vale para a_v, cuja vizinhança em H contém propriamente a vizinhança de v em H. Obtenha uma 1-transferência $y^{(i)}$ de (H', L, y) transferindo uma unidade de abertura da vizinhança de a_v para a_v. Isso é feito da seguinte forma: para cada vizinho u de v em H, realize uma transferência pontual máxima de u a v. Quando v atingir valor de abertura 1, pare. Comece o processo pelo vértice b_v para garantir que $y^{(i)}_{b_v} = 0$. Remova de T todos os vizinhos de a_v cujo valor de abertura tornou-se 0 como resultado das transferências pontuais. Note que isso garante que o vértice b_v não faz parte de T, e assim não é envolvido na transferência feita em T: deixamos b_v como reserva ou "backup" para a transferência final.

Repetindo a agregação para cada monarca, obtemos $y^{(i)}$. Observe que o Lema 3.20 nos garante que $y^{(i)}$ é efetivamente uma 1-transferência de (H', L, y), pois é composição de transferências pontuais de distância 1, e nenhum vértice auxiliar cede abertura. Como todo vértice interno de T, que são os vértices auxiliares, tem abertura 1 em $y^{(i)}$, temos uma árvore de transferência $(T, L, y^{(i)})$. Obtenha uma r-transferência inteira $y^{(ii)}$ de $(T, L, y^{(i)})$. Pela relação entre adjacência em T e distância em H, e o fato de que T contém todo o valor de abertura de H', temos que $y^{(ii)}$ é uma $3r$-transferência de $(H', L, y^{(i)})$.

Se v é um monarca, note que necessariamente $y^{(ii)}_{b_v} = 0$, uma vez que tomamos o cuidado de zerar a abertura de b_v na agregação inicial e portanto b_v não está em T. Se a_v está na transferência inteira que realizamos na árvore, isto é, se $y^{(ii)}_{a_v} = 1$, transferimos essa abertura pontualmente para b_v. Este último passo resulta em uma 1-transferência inteira $y^{(iii)}$ que é o vetor característico de um conjunto S de vértices. Portanto, S é a composição das transferências $y^{(i)}$, $y^{(ii)}$ e $y^{(iii)}$, e então é uma $(3r + 2)$-transferência inteira de (H', L, y).

Agora observe que os vértices auxiliares têm abertura zero em y e em $y^{(iii)}$, de modo que $S \subseteq V$ é uma transferência inteira de (H, L, y). Além disso, para quaisquer vértices de H, digamos u e v, vale que $d_{H'}(u, v) = d_H(u, v)$. Isso porque qualquer caminho em H' que passa por um vértice auxiliar a_v pode em vez disso passar por v sem ter que aumentar seu comprimento, ou seja, os vértices auxiliares não alteram distâncias no grafo. Portanto, a distância da transferência S em relação a (H, L, y) é também $3r + 2$, como queríamos. □

Algoritmo para 2-transferência inteira em árvores

Nesta seção, mostraremos como obter uma r-transferência inteira em qualquer árvore de transferência. Note que, pelo Lema 3.22, quanto menor o r para o qual conseguirmos um algoritmo, menor a distância total da transferência que conseguimos em um grafo arbitrário. Ademais, pelo Lema 3.18, isso significa obter uma melhor razão de aproximação final do algoritmo. Idealmente, gostaríamos de obter uma 1-transferência para qualquer árvore, de modo a conseguir uma 5-transferência em qualquer grafo, e portanto, pelo Lema 3.18, uma 6-aproximação. Infelizmente, no entanto, isso não é possível com essa abordagem, como explicita o Lema 3.23.

Lema 3.23. *Existem árvores de transferência que não admitem 1-transferências inteiras.*

Demonstração. Seja $T = (V, E)$ uma árvore consistindo apenas de uma raiz r e seis folhas adjacentes a r. Seja L uma função de capacidade constante sobre V, isto é, que define a mesma capacidade, digamos $\ell \in \mathbb{Z}_{>0}$, para todo vértice da árvore. Defina os valores de abertura da seguinte maneira: $y_r = 1$ e $y_v = \frac{2}{3}$ para cada folha v.

Seja $S \subseteq V$ uma q-transferência inteira de (T, L, y) para algum inteiro positivo q. Então

$$|S| = y^S(V) = y(V) = 1 + 6 \cdot \frac{2}{3} = 1 + 4 = 5,$$

de modo que $U = V \setminus S$ contém exatamente dois vértices. Considere os seguintes casos.

Se U tem duas folhas, então

$$\sum_{v \in V \colon d_T(v,U) \leq 1} L(v) y_v^S = L(r) y_r^S = L(r) = \ell < \frac{4}{3} \cdot \ell = \sum_{v \in U} L(v) y_v.$$

Senão, $U = \{r, t\}$ para alguma folha t. Considere então o conjunto $\{t\}$, temos

$$\sum_{v \in V \colon d_T(v,\{t\}) \leq 1} L(v) y_v^S = L(r) y_r^S = 0 < \ell \cdot \frac{2}{3} = L(t) y_t.$$

Concluímos que $q > 1$, isto, é, S não pode ser uma 1-transferência. □

Apesar do resultado negativo acima, o próximo lema mostra que a segunda melhor possibilidade é verdadeira.

Lema 3.24. *Existe um algoritmo polinomial que, dada uma árvore de transferência (T, L, y) qualquer, obtém uma 2-transferência inteira de (T, L, y).*

Demonstração. Descreveremos um algoritmo que, dada uma árvore T com raiz r, capacidades L e valores de abertura y nos vértices, onde $y(v) = 1$ para cada vértice interno v, realiza uma 2-transferência que abre integralmente $\lfloor y(T) \rfloor$ vértices e deixa um vértice com valor de abertura $y(T) - \lfloor y(T) \rfloor$. Se (T, L, y) é árvore de transferência, ou seja, tem a propriedade adicional de que $y(T) = 1$, então o algoritmo realiza uma 2-transferência inteira.

O algoritmo é recursivo, e cada chamada devolve um conjunto S de vértices que são abertos integralmente, isto é, que têm valor de abertura 1 após a transferência, e um par de valores $(y_p, L(p))$ representando a abertura e a capacidade de um vértice aberto fracionariamente. Tal vértice é necessariamente a raiz ou um de seus filhos. Se $y_p = 0$, então S é uma 2-transferência inteira.

Existem dois possíveis casos base. O primeiro é um único vértice v com abertura $y(v) = 1$, e nesse caso o algoritmo não precisa fazer transferência, bastando devolver o conjunto $\{v\}$.

O segundo é uma árvore T_r com raiz r, à qual são adjacentes ℓ folhas, digamos $\{v_1, \ldots, v_\ell\}$, onde $L(v_1) \geq L(v_2) \geq \cdots \geq L(v_\ell)$.

Note que exigimos que vértices internos tenham abertura 1, e portanto $y_r = 1$. Se além disso a soma das aberturas das folhas é inteira, ou seja, se $y(T_r)$ é inteiro, então basta escolhermos o conjunto S como sendo os $y(T_r)$ vértices mais capacitados em T_r. Isso pode ser feito através de transferências pontuais dentro de T_r, que tem diâmetro 2, e portanto S é uma 2-transferência. Se T_r não é a árvore da instância original, seja T a árvore de onde partiu a chamada para a instância T_r. Basta resolver o problema para $T - T_r$, obtendo uma 2-transferência S', e então devolver $S' \cup S$.

Resta descrever o algoritmo no caso em que $y(T_r)$ não é inteiro. Escolhemos S como sendo os $\lfloor y(T_r) \rfloor$ vértices mais capacitados em T_r. No entanto, resta em T_r um valor fracionário de abertura $y(T_r) - \lfloor y(T_r) \rfloor$. Precisamos disponibilizar esse valor de abertura para a instância T, que gerou a chamada com T_r, para a abertura integral de um vértice. Para tanto, substituímos, em T, a árvore T_r pelo vértice artificial p, que tem abertura $y_p = y(T_r) - \lfloor y(T_r) \rfloor$ e capacidade $L(p)$ igual à do $\lceil y(T_r) \rceil$-ésimo vértice mais capacitado de T_r, isto é, o vértice mais capacitado que não entrou em S. Dizemos que p representa tal vértice de T_r. Para resolver a instância resultante, basta tratar p como uma folha qualquer de T. Se p for aberto integralmente, então o vértice que p representa é aberto integralmente.

Note que esse procedimento nos dá uma transferência inteira, mas não é imediato afirmar que se trata de uma 2-transferência. Isso porque, a princípio, o vértice p pode representar uma folha de T_r, e assim uma transferência pontual de p a uma folha de T pode ter distância até 3. No entanto, veremos diretamente que o conjunto S final é uma 2-transferência. □

Assim, podemos deduzir o algoritmo de An et. al. [2] para o problema dos k-centros com capacidades arbitrárias.

AlgoritmoAnEtAl (H, L, k)

1 $(x, y) \leftarrow$ solução de $\mathrm{PL}_k(H, L)$
2 $S \leftarrow$ 8-transferência inteira de (H, L, y)
3 $\phi \leftarrow$ atribuição de V a S em H^9
4 **devolva** (S, ϕ)

Teorema 3.25. *O algoritmo* AlgoritmoAnEtAl *acima definido é uma 9-aproximação para o problema dos k-centros com capacidades arbitrárias.*

Demonstração. Os passos do algoritmo estão bem definidos e podem ser feitos em tempo polinomial devido aos Lemas 3.18, 3.22 e 3.24. Ademais, pelo Lema 3.18, a solução devolvida é uma 9-aproximação. □

3.3.2 Capacidades não-nulas uniformes

Agora, mostraremos como é possível adaptar as ideias usadas no algoritmo da Seção 3.3.1 para o caso especial em que todas as capacidades diferentes de zero devem ser iguais. Trata-se do problema dos k-centros $\{0, L\}$-capacitados, que definimos a seguir.

> **Problema dos k-centros $\{0, L\}$-capacitados**: *Dados um grafo G completo com custos c_e nas arestas respeitando a desigualdade triangular, um conjunto de vértices V^L e inteiros positivos k e L, escolher um par (S, ϕ), onde $S \subseteq V^L$, $|S| \leq k$, $\phi\colon V \to S$ e*
>
> $$|\{v \in V : \phi(v) = u\}| \leq L, \text{ para todo } u \in S, \tag{3.9}$$

que minimize o valor

$$\max_{v \in V} d_G(v, \phi(v)).$$

Definição 3.26. *Para o problema acima definido, dizemos que (S, ϕ), com $S \subseteq V^L$, $\phi\colon V \to S$, é* solução *se $|S| \leq k$ e ϕ respeita 3.9. Se H é um subgrafo gerador de G, um tal S é* solução em H *se todo vértice de V é vizinho, em H, de $\phi(S)$.*

Definição 3.27. *Para uma instância do problema acima definido, cada vértice v em V^L é chamado de* L-vértice. *Dado um conjunto X qualquer de vértices, denotamos $X^L = X \cap V^L$.*

Note que a única alteração em comparação com o CKC já visto na Seção 3.1.1 é que é dado um conjunto $V^L \subseteq V$ de centros elegíveis. Nos termos do problema dos k-centros com capacidades arbitrárias, isso representa uma função de capacidade em que todo vértice tem capacidade ou 0 ou L, o que dá o nome da variante.

Capítulo 4

Tolerância a falhas

Nesse capítulo, mostraremos algumas variantes dos problemas dos k-centros que surgem ao considerarmos uma restrição adicional, chamada de *tolerância a falhas*. Trata-se de exigir que cada cliente esteja próximo não apenas de um centro, mas de vários. Essa restrição é uma forma de buscar robustez ante ao risco de que um centro possa falhar, isto é, deixar de ser centro em algum momento. Numa aplicação de redes de computadores, por exemplo, isso pode modelar a possibilidade de que um certo número de servidores deixe de poder prestar serviço por problemas técnicos ao longo do tempo. Nesse caso, queremos que nossa escolha de servidores seja tal que aqueles que continuam funcionando sejam capazes de acomodar os clientes daqueles que falharam. Krumke [14] foi o primeiro a propor uma restrição desse tipo à versão básica do problema dos k-centros.

Na maioria dos problemas que aqui apresentamos, faz parte da instância um inteiro α arbitrário que representa o número máximo de falhas simultâneas com as quais se exige que solução seja capaz de lidar. No entanto, veremos também dois problemas para os quais só conhecemos aproximações caso esse parâmetro α seja constante.

4.1 Definição dos problemas

Nos dois primeiros problemas deste capítulo, cada centro pode atender um número arbitrário de clientes. Isto é, esses dois problemas não trazem a restrição de capacidades como vista no Capítulo 3, ou ainda, podemos considerar que são variantes em que as capacidades são ilimitadas. Assim, para lidar com uma falha de tamanho até α, queremos que cada cliente v esteja próximo de $(\alpha+1)$ dos centros escolhidos. Dessa forma, caso os α centros mais próximos de v falhem simultaneamente, v ainda poderá ser atendido por um centro a uma distância razoável: o $(\alpha+1)$-ésimo centro mais próximo de v (com empates resolvidos arbitrariamente) sempre tem capacidade de servir v. Definimos a seguir uma notação auxiliar para a definição dos problemas.

Definição 4.1. *Seja $G=(V,E)$ um grafo conexo com custo c_e qualquer para cada aresta $e \in E$. Seja $X \subseteq V$ um conjunto de vértices com rótulos ordenados dados por uma função injetora $f\colon X \to \mathbb{N}$. Fixado um $v \in V$, seja $\left(x_1, x_2, \ldots, x_{|X|}\right)$ a ordenação dos vértices em X com respeito à distância até v. Ou seja, a ordem tal que $d_G(x_i, v) \leq d_G(x_{i+1}, v)$ para $i = 1, \ldots, |X|-1$ e, caso $d_G(x_i, v) = d_G(x_{i+1}, v)$, então x_i tem rótulo menor do que o de x_{i+1} de acordo com f. Dado um inteiro positivo ℓ, denotamos*

$$\delta_G^{\ell}(v, X) = x_{\ell}.$$

Na definição acima, a função de rótulo f serve apenas para que X tenha uma ordena-

ção explícita e assim possamos definir δ de forma determinística. Na prática, os rótulos dos elementos do conjunto X podem ser dados pela ordem em que estes são escolhidos por um algoritmo, ou simplesmente herdados do conjunto V. Portanto, podemos considerar que sempre há uma tal f dada implicitamente. Definimos a seguir a variante principal do problema com tolerância a falhas e capacidades ilimitadas, que é equivalente ao problema definido por Krumke [14], e uma versão desse problema proposta por Khuller, Pless e Sussmann [12].

Problema dos k-centros com α tolerância: *Dados um grafo $G = (V, E)$ completo com um custo c_e para cada aresta e respeitando a desigualdade triangular, e dois inteiros positivos, k e α, escolher $S \subseteq V$, $|S| \leq k$, que minimize o valor*

$$\max_{v \in V} \delta_G^{\alpha+1}(v, S).$$

Problema dos k-centros com α tolerância não estrita (ou relaxada): *Dados um grafo $G = (V, E)$ completo com um custo c_e para cada aresta e respeitando a desigualdade triangular, e dois inteiros positivos, k e α, escolher $S \subseteq V$, $|S| \leq k$, que minimize o valor*

$$\max_{v \in V \setminus S} \delta_G^{\alpha+1}(v, S).$$

Note que, na versão não estrita, os centros não são levados em conta na definição de distância da função objetivo. Interpreta-se isso como uma versão em que os centros não são clientes. Assim, uma vez que um centro falhe, ele não precisa ser atendido por um outro centro. Dentre os problemas estudados nesse capítulo, este é o único que tem essa característica.

Nos demais problemas que veremos neste capítulo, os centros têm capacidades limitadas. Essa junção de restrições foi primeiro proposta por Chechik e Peleg [5]. Para resolver tais problemas, temos que encontrar, além de um conjunto de centros S, uma atribuição de clientes ao conjunto S, tal como no Capítulo 3, que chamaremos de *atribuição inicial* ϕ_0. Essa solução precisa ter a propriedade de que, para qualquer conjunto de falhas $F \subseteq S$ existe uma nova atribuição $\phi_F \colon V \to S \setminus F$ que respeite as capacidades, e o pior raio de uma tal *reatribuição* ϕ_F é o valor da solução. Em geral, a garantia de existência de reatribuição para qualquer cenário de falhas se dá ao mostrarmos um algoritmo eficiente que encontra tal atribuição dada a instância original, a solução inicial e o cenário de falhas F.

Nesse contexto, podemos considerar, como propõem Chechik e Peleg [5], duas variações sobre as reatribuições permitidas: nas versões *não conservativas* dos problemas, a única restrição é que se respeitem as capacidades dos centros. Nas versões *conservativas*, exigimos ainda que a reatribuição coincida com a atribuição inicial ϕ_0 para todos os clientes cujo centro inicialmente designado não falhou.

Definição 4.2. *Tome um grafo $G = (V, E)$ completo com custo c_e para cada aresta $e \in E$ respeitando a desigualdade triangular, uma função de capacidade $L \colon V \to \mathbb{Z}_+$ e inteiros positivos k e α. Dizemos que (G, c, L, k, α) é uma* instância de problemas de k-centros com tolerância a falhas e capacidades.

Fixe uma tal instância e considere uma solução (S, ϕ_0), *onde $S \subseteq V$ com $|S| \leq k$, e $\phi_0 \colon V \to S$ satisfaz 3.2, isto é, respeita as capacidades dos centros.*

Seja $F \subseteq S$, com $|F| \leq \alpha$. Dizemos que F é um cenário de falhas. *Uma* reatribuição viável *para F é uma atribuição $\phi_F \colon V \to S \setminus F$ que respeita as capacidades dos centros e tal que $\phi_F^{-1}(u) = \emptyset$ para todo $u \in F$.*

Uma reatribuição ϕ_F é conservativa *com respeito a ϕ_0 se $\phi_F(v) = \phi_0(v)$ sempre que $\phi_0(v) \notin F$.*

Usando a linguagem acima, definimos a forma geral do problema dos k-centros com tolerância a falhas e capacidades.

Problema dos k-centros com tolerância a falhas e capacidades: *Dada uma instância (G, c, L, k, α) como acima definida, escolher uma solução (S, ϕ_0), como acima definida, que minimize o valor*

$$\max_{F \text{ cenário de falhas}} \min_{\phi_F} \max_{v \in V} d_G(v, \phi_F(v)).$$

Sobre a definição acima, diferenciamos quatro casos:

- Difereniamos entre aqueles em que os centros têm capacidades arbitrárias e aqueles em que as capacidades não nulas são uniformes. Isto é, temos casos com centros capacitados ou $\{0, L\}$-capacitados, respectivamente;
- Diferenciamos entre aqueles que exigem reatribuição conservativa e aqueles em que a reatribuição não precisa conservar nada da solução inicial. São os casos conservativos e não conservativos, respectivamente.

Para cada uma das quatro combinações emergentes, apresentamos um algoritmo de aproximação. Os algoritmos que conhecemos para os casos com capacidades arbitrárias, no entanto, exigem que o parâmetro α da instância seja dominado por uma constante, pois executam subrotinas exponenciais em α.

4.2 Casos com capacidades ilimitadas

Apresentamos primeiro os resultados descritos por Khuller, Pless e Sussmann [12] para os dois primeiros problemas citados acima, isto é, os casos com capacidades ilimitadas.

4.2.1 Problema dos k-centros com α-tolerância não estrita

Definição 4.3. *Para o problema dos k-centros com α-tolerância não estrita, dizemos que $S \subseteq V$ é* solução *se $|S| \leq k$. Se H é um subgrafo gerador de G, um tal S é* solução em H *se todo vértice de $V \setminus S$ é vizinho, em H, de pelo menos α elementos de S.*

Observe também que o problema dos k-centros é um caso particular desse, para $\alpha = 1$. Portanto, a menos que P = NP, a melhor razão de aproximação que podemos esperar para esse problema em geral é 2. De fato, mostraremos um algoritmo que é uma 2-aproximação.

Esse algoritmo também usa o método geral dado pelo MetodoGargalo. Usamos $t = 2$ e a rotina de teste descrita pelo algoritmo a seguir. A ideia do algoritmo é atribuir a cada vértice v um número de "cobertura", $C(v)$, que começa valendo zero. O número de cobertura corresponde ao número de centros já escolhidos dos quais o vértice é vizinho (se v é centro, define-se $C(v) = \alpha$ pois v já está "satisfeito"). O algoritmo executa α iterações, e, ao fim da iteração j, garante que cada vértice está coberto pelo menos j vezes (ou por ser um centro, ou por ser, de fato, vizinho de j dos centros já escolhidos).

TESTE-α-TOLERÂNCIA(H, k, α)

```
1  S ← ∅    ▷ o conjunto de centros escolhidos
2  para cada v ∈ V
3      C(v) ← 0    ▷ número de vezes que v é coberto
4  para j ← 1 até α
5      enquanto existe v com C(v) < j
6          Seja v tal que C(v) < j
7          S ← S ∪ {v}
8          C(v) ← α    ▷ versão não estrita: centros não precisam ser cobertos
9          para cada vizinho u de v em H²
10             C(u) ← C(u) + 1
11 devolva S
```

Note que o que o algoritmo faz, implicitamente, é encontrar, a cada iteração j, um conjunto independente em G_i^2. Isso porque, se dois vértices distintos u e v são escolhidos na mesma iteração para se tornarem centros, é porque o primeiro deles a ser escolhido como centro não aumentou o número de cobertura do outro, o que só acontece se eles não são vizinhos em G_i^2.

Lema 4.4. *Fixe uma instância (G, k, α) do problema dos k-centros com α-tolerância. O algoritmo* TESTE-α-TOLERÂNCIA *com $H = G_i$ devolve ou uma solução em G_i^2, ou um certificado de que $\mathrm{OPT}(G, k, \alpha) > c_i$.*

Teorema 4.5. *O método geral para problemas de gargalo (*METODOGARGALO*), com a rotina de teste dada por* TESTE-α-TOLERÂNCIA*, é uma 2-aproximação para o problema dos k-centros com α-tolerância.*

Demonstração. Segue do Lema 4.4 e do Teorema 2.18. □

4.2.2 Problema dos k-centros com α-tolerância

Estudaremos, nesta seção, o problema dos k-centros com α-tolerância (α-*all-neighbor k-center*). A única diferença em relação ao problema anterior é o fato de que, aqui, exigimos que *todos* os vértices sejam cobertos por α centros "próximos".

Definição 4.6. *Para o problema dos k-centros com α-tolerância, dizemos que $S \subseteq V$ é* solução *se $|S| \leq k$. Se H é um subgrafo gerador de G, um tal S é* solução em H *se todo vértice de $V \setminus S$ é vizinho, em H, de pelo menos α elementos de S, e todo vértice de S (centro) é vizinho, em H, de pelo menos $\alpha - 1$ elementos de S.*

Para esse problema, apresentaremos uma 3-aproximação. Como esse problema é também uma variação dos k-centros, espera-se conseguir encontrar uma 2-aproximação. Khuller, Pless e Sussmann [12] apresentaram 2-aproximações para os casos em que $\alpha = 2$ e $\alpha = 3$, mas ainda não se sabe se a melhor razão possível (a menos que P = NP) para esse problema é menor do que 3 no caso geral [5, 15].

O algoritmo de teste para esse problema toma a seguinte forma: dado G_i, obtém-se um conjunto independente maximal I em G_i^2. Se $\alpha \cdot |I| > k$, então não existe solução em G_i, pois cada elemento de I exige a criação de α centros diferentes. Nesse caso, I é um certificado de que $\mathrm{OPT} > c_i$.

Além disso, se algum vértice tem menos de $\alpha - 1$ vizinhos em G_i, certamente G_i também não é viável, pois não há como todos os vértices estarem conectados a α centros. Um tal vértice é um certificado para essa afirmação.

Caso contrário, isto é, se $\alpha \cdot |I| \leq k$ e todo vértice tem pelo menos $\alpha - 1$ vizinhos em G_i, então damos uma solução em G_i^3 da seguinte maneira: para cada $v \in I$, escolha como centros o vértice v e $\alpha - 1$ de seus vizinhos em G_i. Como I é independente em G_i^2, então todas essas vizinhanças são disjuntas, e portanto não corremos o risco de tentar repetir um centro nesse processo.

Vamos mostrar que este conjunto de centros é uma solução em G_i^3. Tome $v \in V$. Se $v \in I$, então v está à distância no máximo 1 em G_i dos α centros mais próximos. Se $v \in V \setminus I$, então existe $w \in I$ tal que v e w têm em G_i um vizinho u em comum (pela maximalidade de I). Portanto, todos os vizinhos de w em G_i são vizinhos de v em G_i^3. Assim, os centros escolhidos formam uma solução em G_i^3, e esse algoritmo de teste nos dá uma 3-aproximação para o problema dos k-centros com α-tolerância estrita.

4.3 Casos com capacidades limitadas

Nas seções a seguir exploramos as diversas variantes do problema dos k-centros levando em conta capacidades e tolerância a falhas. Assim como na Seção 3.3.1, aqui iremos considerar que a instância é um grafo conexo sem pesos nas arestas. Os resultados para esses problemas, bem como as figuras usadas, provêm de nosso trabalho com Fernandes e Pedrosa [8].

4.3.1 Caso $\{0, L\}$-capacitado e conservativo

Depois da ocorrência de uma falha, uma solução viável conservativa tem que reatribuir cada cliente descoberto a um centro aberto na sua vizinha e com capacidade disponível. Isso requer que algum tipo de "capacidade de centro disponível local" seja usada como backup. A próxima definição descreve um conjunto de vértices que são bons candidatos a serem abertos como centros de backup. Esse conjunto pode ser particionado em clusters de no máximo α vértices, com os clusters suficientemente separados uns dos outros. A ideia é que as falhas na vizinhança de um desses clusters não afetem centros nos outros clusters. Mais precisamente, as vizinhanças de clusters diferentes não se intersectam e, portanto, em uma solução viável conservativa, qualquer cliente que é atribuído a um centro em determinado cluster não pode ser reatribuído a um centro na vizinhança de nenhum dos outros clusters.

Definição 4.7. *Considere um grafo $G = (V, E)$ e inteiros não negativos α e ℓ. Um conjunto W de vértices de G é* (α, ℓ)-independente *se ele pode ser particionado em conjuntos $C_1, \ldots, C_t$, tais que $|C_i| \leq \alpha$ para $1 \leq i \leq t$ e $d(C_i, C_j) > \ell$ para $1 \leq i < j \leq t$.*

A seguir, nós denotamos por (G, k, L, α) uma instância do problema dos k-centros com capacidades $\{0, L\}$ e tolerância a falhas conservativo, como descrito na Seção **??**. Nós dizemos que (G, k, L, α) é viável se existe uma solução distâcia-1.

Lema 4.8. *Seja (G, k, L, α) uma instância viável para o problema dos k-centros com capacidades $\{0, L\}$ e tolerância a falhas conservativo, e seja (S^*, ϕ_0^*) a solução distâcia-1. Se $W \subseteq S^*$ é um conjunto $(\alpha, 4)$-independente em G, então $(G, k - |W|, L')$ é viável, onde $L'_u = 0$, se u pertence a W e $L'_u = L_u$, caso contrário.*

Demonstração. Como W é $(\alpha, 4)$-independente, deve existir uma partição $C_1, \ldots, C_t$ de W tal que $d(C_i, C_j) > 4$ para qualquer par i, j, com $1 \leq i < j \leq t$. Além disso, cada parte C_i tem no máximo α vértices e, assim, existe uma atribuição conservativa $\phi_{C_i}^*$ com $(\phi_{C_i}^*)^{-1}(C_i) = \emptyset$. Portanto, $\phi_{C_i}^*$ é uma solução distância-1 para a instância $(G, k - |C_i|, L^i)$ do problema dos k-centros capacitados, onde $L_u^i = 0$, se u pertence a C_i e $L_u^i = L_u$, caso contrário. Como

ϕ_0^* é conservativa, $\phi_{C_i}^*$ difere de ϕ_0^* apenas em $(\phi_0^*)^{-1}(C_i)$. Então, se um centro u em S^* é tal que$(\phi_0^*)^{-1}(u) \neq (\phi_{C_i}^*)^{-1}(u)$, temos que $u \in N^2(C_i)$. Como W é $(\alpha, 4)$-independente, $N^2(C_i) \cap N^2(C_j) = \emptyset$ para todo $j \in [t] \setminus \{i\}$. Seja ψ uma atribuição tal que, para cada cliente v,

$$\psi(v) = \begin{cases} \phi_{C_i}^*(v) & \phi_0^*(v) \in C_i \text{ para algum } i \text{ em } [t], \\ \phi_0^*(v) & \text{caso, contrário.} \end{cases}$$

Assim, o conjunto $\psi^{-1}(u)$ é vazio se $u \in W$; é $(\phi_{C_i}^*)^{-1}(u)$, se existe $i \in [t]$ tal que $u \in N^2(C_i) \setminus C_i$; e é $(\phi_0^*)^{-1}(u)$, caso contrário. Isso significa que, para L', tal como no lema, $|\psi^{-1}(u)| \leq L'_u$ para todo u, e então (S^*, ψ) é uma solução para a instância $(G, k - |W|, L')$ do problema dos k-centros capacitados. □

Um conjunto de vértices $A \subseteq V$ é 7-*independente* em G se todo par de vértices em A está a uma distância de pelo menos 7 em G. Essa definição também foi usada por Chechik e Peleg [5] e, como mostraremos, esse conjunto é útil para obter um conjunto $(\alpha, 4)$-independente em G.

Lema 4.9. *Seja A um conjunto 7-independente em G, para cada a em A, seja $B(a)$ um conjunto qualquer de α vértices em $N(a)$, e seja $B = \cup_{a \in A} B(a)$. Se (G, k, L, α) é viável para o problema dos k-centros capacitados com tolerância a falhas conservativo, então $(G, k - |B|, L')$ é viável para o problema dos k-centros capacitados, onde $L'_u = 0$, se u pertence a B, e $L'_u = L_u$, caso contrário.*

Demonstração. Seja (S^*, ϕ_0^*) uma solução para (G, k, L, α). Para cada $a \in A$, devem existir pelo menos α centros em $S^* \cap N(a)$. Seja $W(a)$ a união de $S^* \cap B(a)$ e outros $\alpha - |S^* \cap B(a)|$ centros em $S^* \cap N(a)$. Seja $W = \cup_{a \in A} W(a)$. Como A é 7-independente, $N^3(a)$ e $N^3(b)$ são disjuntos para quaisquer dois a e b em A, e então $N^2(W(a)) \cap N^2(W(b)) = \emptyset$. Assim, W é $(\alpha, 4)$-independente.

Seja L'' tal que $L''_u = 0$, se $u \notin S^*$ e $L''_u = L_u$, caso contrário. Observe que a instância (G, k, L'', α) é viável (como nós zeramos apenas as capacidades dos vértices que não são centros). Pelo Lema 4.8, a instância $(G, k - |W|, L''')$ é viável, onde $L'''_u = 0$ se $u \in W$, e $L'''_u = L''_u$, caso contrário. Note que $L'_u \geq L'''_u$ para todo u, e $|B| = |W|$. Portanto, como $(G, k - |W|, L''')$ é viável, $(G, k - |B|, L')$ também é. □

Agora nós apresentamos uma 7-aproximação para o problema dos k-centros com capacidades $\{0, L\}$ e tolerância a falhas conservativo. Nesse caso, ao invés de usar uma função de capacidade, é conveniente considerar o subconjunto de vértices com capacidade L, que é denotado por V^L. Nós denotamos por (G, k, V^L, α) e por (G, k, V^L) instâncias das versões com tolerância e sem tolerância a falhas, respectivamente. Os passos são detalhados no Algoritmo **??**, onde ALG denota um algoritmo de aproximação para o problema dos k-centros com capacidades $\{0, L\}$.

Teorema 4.10. *Se* ALG *é uma β-aproximação para o problema dos k-centros $\{0, L\}$-capacitados, então o Algoritmoé uma $\max\{7, \beta\}$-aproximação para o problema dos k-centros com capacidades $\{0, L\}$ e tolerância a falhas conservativo.*

Demonstração. Considere uma instância (G, k, V^L, α) do problema dos k-centros com capacidades $\{0, L\}$ e tolerância a falhas conservativo, onde $G = (V, E)$. Sejam A, $B(a)$ para $a \in A$, e B como definimos no Algoritmo **??**, com (G, k, V^L, α) como entrada. Suponha que (G, k, V^L, α) é viável. Como A é 7-independente, pelo Lema 4.9, a instância $(G, k - |B|, V^L \setminus B)$, onde nós zeramos as capacidades de todos os vértices em B, é também

viável para o problema dos k-centros com capacidades $\{0, L\}$. Isso significa que, se o Algoritmo **??** executa a Linha **??**, então a instância dada é de fato viável. Por outro lado, se ALG devolve uma solução (S, ϕ), então, como $|S| \leq k - |B|$, o tamanho de $S \cup B$ é no máximo k e ϕ é uma atribuição de centro inicial válida. Além disso, ϕ é tal que: (1) cada vértice u está a uma distância de no máximo β de $\phi(u)$; e (2) nenhum vértice é atribuído a B.

Seja $F \subseteq S \cup B$ com $|F| = \alpha$ um cenário de falha. A seguir descrevemos uma reatribuição de centro conservativa para $(S \cup B, \phi)$. Nós precisamos reatribuir somente os vértices que foram inicialmente atribuídos a centros em $F \setminus B$ (como nenhum vértice foi atribuído a um vértice em B). Assim, no máximo $L|F \setminus B|$ vértices precisam ser reatribuídos. Para cada vértice u, nós podemos escolher $a \in A$ a uma distância de no máximo 6 de u (como A é maximal) e fazemos $\tilde{\phi}(u) = a$. Depois, para cada $a \in A$, e para cada u com $\tilde{\phi}(u) = a$, reatribuímos u a algum centro não cheio de $B(a) \setminus F$. Note que $B(a) \setminus F$ pode absorver todos os vértices reatribuídos. De fato, a capacidade disponível de $B(a) \setminus F$ antes do evento de falha é $L|B(a) \setminus F| = L|F \setminus B(a)| \geq L|F \setminus B|$, onde nós usamos $|B(a)| = |F| = \alpha$. Como, para um vértice reatribuído u, $d(u, \tilde{\phi}(u)) \leq 6$ e u é reatribuído a algum centro $v \in N(\tilde{\phi}(u))$, a distância entre u e v é no máximo 7. Também, se um vértice u não foi reatribuído, então a distância ao seu centro é no máximo β. □

Usando a 6-aproximação para o problema dos k-centros com capacidades $\{0, L\}$ de An et al. [3], nós obtemos o corolário a seguir.

Corolário 4.11. *O Algoritmo* **??** *usando o algoritmo de An et al. [3] para o problema dos k-centros com capacidades $\{0, L\}$ é uma 7-aproximação para o problema dos k-centros com capacidades $\{0, L\}$ e tolerância a falhas conservativo.*

4.3.2 Caso com capacidades não uniformes e conservativo

Nesta seção, consideramos o problema dos k-centros com capacidades não uniformes e tolerância a falhas, caso conservativo. Ou seja, em contraste com o problema visto na Seção 4.3.1, aqui os centros possuem capacidades arbitrárias. Veremos a seguir um algoritmo que é uma aproximação para esse problema sob a hipótese de que o parâmetro α, isto é, o número de centros que podem falhar, é limitado por uma constante. Trata-se da primeira aproximação conhecida para esse problema.

Lembre que, no caso com capacidades $\{0, L\}$, um vértice v atribuído a um centro do conjunto de falhas poderia ser reatribuído a um centro operante no conjunto $B(a)$, onde a é um elemento do conjunto 7-independente A próximo de v. Cada conjunto $B(a)$ tinha capacidade livre total para receber todos os vértices que precisassem de reatribuição. No caso com capacidades arbitrárias, o conjunto B de centros pré-escolhidos precisa ser obtido com mais cautela, uma vez que os vértices com capacidades não-nulas não necessariamente têm a mesma capacidade. Uma vez que o conjunto B dos centros de backup é selecionado, é preciso garantir que a instância resultante do problema dos k-centros com capacidades seja viável. Na Seção 4.3.1, um conjunto $(\alpha, 4)$-independente é obtido a partir de A, e o Lema 4.8 é então aplicado. Note que esse lema também é válido para capacidades arbitrárias, de modo que ele também será útil aqui. Para obter um conjunto $(\alpha, 4)$-independente a partir de B, precisamos nos certificar de que B pode ser particionado de forma que quaisquer duas partes distintas estejam à distância 7, ou maior, uma da outra. Isso é feito pelo Algoritmo **??**. Denotamos por ALG um algoritmo de aproximação para o problema dos k-centros com capacidades.

O Algoritmo **??** é polinomial no tamanho da representação de G, k e L. Na linha **??**, o algoritmo realiza um teste equivalente a encontrar um conjunto $U \subseteq V$, com $|U| = \alpha$, que minimize a diferença $L(B \cap N^6(U)) - L(U)$. Se tal subproblema fosse resolvido em tempo

polinomial para qualquer α, então o Algoritmo **??** seria também polinomial para qualquer α. No entanto, como veremos na Seção **??**, tal subproblema é coNP-difícil. Quando α é fixado, ou limitado superiormente por uma constante com respeito à instância, podemos enumerar os conjuntos U em tempo polinomial. A seguir, mostramos que o Algoritmo **??** é um algoritmo de aproximação para o problema dos k-centros com capacidades e tolerância a falhas conservativo se o parâmetro α for fixado.

O lema a seguir é análogo ao Lema 4.9, mas vale para o caso com capacidades arbitrárias.

Lema 4.12. *Seja B o conjunto de vértices obtidos pelo Algoritmo **??** após a execução das linhas **??** a **??**. Se a instância (G, k, L, α) é viável para o problema dos k-centros com capacidades e tolerância a falhas conservativo, então $(G, k - |B|, L')$ é viável para o problema dos k-centros com capacidades, onde $L'_u = 0$ para cada u em B, e $L'_u = L_u$ para os demais.*

Demonstração. Vamos verificar que o conjunto B é $(\alpha, 6)$-independente. Isto é, conforme a Definição 4.7, que B pode ser particionado em conjuntos $C_1, \ldots, C_t$ de modo que cada C_i tem cardinalidade no máximo α, e $d(C_i, C_j) > 6$ para cada $1 \leq i < j \leq t$.

Seja t o número de componentes do grafo $G^6[B]$ e tome cada C_i, $1 \leq i \leq t$, como o conjunto de vértices de uma das componentes de $G^6[B]$. Mostraremos que, para cada i, vale que $|C_i| \leq \alpha$. Suponha, por contradição, que, para algum índice i tenhamos $|C_i| > \alpha$. Seja U' o conjunto de vértices de C_i que foram inseridos em B na última iteração da linha **??** do Algoritmo **??** a ter afetado C_i. Segue diretamente do algoritmo que $|U'| \leq \alpha$. Então existe um vértice v em $C_i \setminus U'$ que está em $N^6(U') \cap B$ nessa iteração. No entanto, a execução da linha **??** removeria v de B, uma contradição. Concluímos que B é de fato $(\alpha, 6)$-independente.

Agora, para cada i, $1 \leq i \leq t$, escolha um elemento arbitrário a_i do conjunto C_i. Notamos que o conjunto $A = \{a_1, a_2, \ldots, a_t\}$ é 7-independente em G. Considere uma solução (S^*, ϕ_0^*) para a instância (G, k, L, α) e note que, para cada i, $1 \leq i \leq t$, é preciso que haja pelo menos $\alpha + 1$ centros em $S^* \cap N(a_i)$. Isto é, temos $|C_i| \leq \alpha < |S^* \cap N(a_i)$. Então definimos o conjunto W_i como sendo a união de $C_i \cap S^*$ com outros $|C_i \setminus S^*|$ centros em $S^* \cap N(a_i)|$. Assim, $|W_i| = |C_i|$ e $W_i \subseteq N(C_i)$.

Seja $W = \cup_{i=1}^{t} W_i$. Observe que $|W| = |B|$. Para cada par (i, j), $1 \leq i < j \leq t$, temos que $d(W_i, W_j) > 4$, uma vez que B é $(\alpha, 6)$-independente e portanto $d(C_i, C_j) > 6$. Concluímos que W é $(\alpha, 4)$-independente.

Defina um vetor de capacidades L'' como $L''_u = L_u$, se $u \in S^*$, e $L''_u = 0$, caso contrário. Note que a instância (G, k, L'', α) é viável, uma vez que apenas alteramos para 0 as capacidades de vértices que não estão em S^*. Pelo Lema 4.8, a instância $(G, k - |W|, L''')$ é viável, onde $L'''_u = 0$ se $u \in W$, e $L'''_u = L''_u$ caso contrário. Temos $L'_u \geq L'''_u$ para todo vértice u. Como $|B| = |W|$, e como $(G, k - |W|, L''')$ é viável, segue que $(G, k - |B|, L')$ é viável. □

Teorema 4.13. *Se* ALG *é uma β-aproximação para o problema dos k-centros com capacidades, então o Algoritmo **??** é uma $(\beta + 6\alpha)$-aproximação para o problema dos k-centros com capacidades e tolerância à falhas conservativa, com parâmetro α constante.*

Demonstração. Seja (G, k, L, α) uma instância do problema dos k-centros com capacidades e tolerância a falhas. Como estamos tomando α constante, cada execução da linha **??** toma tempo polinomial em $|V|$.

Além disso, cada execução da linha **??** aumenta o valor de $L(B)$ em no mínimo uma unidade. E $L(B)$ é um inteiro cujo valor começa em 0 e não ultrapassa $|V|^2$, uma vez que a capacidade de cada vértice é no máximo $|V|$. Assim, o número de iterações é quadrático em $|V|$, e cada uma toma tempo polinomial em $|V|$. E, como ALG é um algoritmo de tempo polinomial, segue que o Algoritmo **??** é polinomial.

Pelo Lema 4.12, sabemos que, se ALG devolve FALHA na linha **??**, então a instância (G, k, L, α) é inviável. Se, no entanto, ALG devolve uma solução (S, ϕ), então $(S \cup B, \phi)$ é um conjunto válido de centros e atribuição inicial para o nosso problema e é tal que cada vértice v está à distância no máximo β de $\phi(u)$. Para completar nossa prova, argumentamos que, para cada cenário de falha F, cada cliente u de um centro em F pode ser reatribuído a outro centro que dista de u de no máximo $\beta + 6\alpha$ e que todo centro tem sua capacidade respeitada pela reatribuição.

Considere um cenário de falha $F \subseteq V$ com $|F| = \alpha$. Definimos a seguir uma rede de fluxo (H, c, s, t), com fonte s e sorvedouro t, em que um s,t-fluxo máximo nos fornece uma reatribuição válida de distância $(\beta+6\alpha)$ para os clientes de centros em F. O grafo $H = (V_H, E_H)$, conforme ilustrado na Figura 4.1, tem em seu conjunto de vértices V_H:

- uma cópia de cada y em $\phi^{-1}(F)$,
- uma cópia de cada v em F,
- uma cópia de cada u em $B \setminus F$,
- outra cópia de cada w em $B \cap F$, denotada por $\bar{w}$;

e, em seu conjunto de arcos (arestas dirigidas) E_H:

- para cada y em $\phi^{-1}(F)$, um arco (s, y) com capacidade $c(s, y) = \infty$,
- para cada v em F e cada y em $\phi^{-1}(v)$, um arco (y, v) com $c(y, v) = 1$,
- para cada v em F e cada u em $B \cap N_G^6(v)$, um arco (v, u) com $c(v, u) = \infty$,
- para cada u em $B \setminus F$, um arco (u, t) com capacidade $c(u, t) = L_u$,
- para cada w em $B \cap F$, um arco "reverso" $(\bar{w}, w)$ com $c(\bar{w}, w) = \infty$.

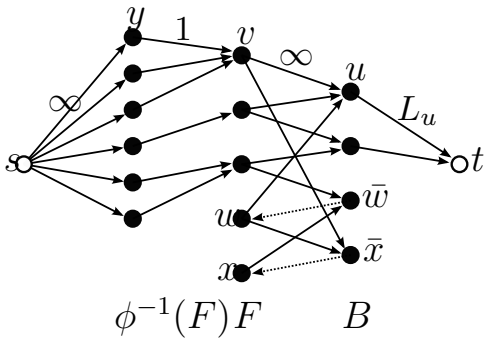


Figura 4.1: *A rede de fluxo definida a partir de L, B, ϕ e F.*

Seja C o conjunto dos arcos de um s,t-corte de capacidade mínima em H. Afirmamos que $c(C) = |\phi^{-1}(F)|$. Seja $\delta^+(X)$ o conjunto dos arcos que saem de X, isto é, arcos na forma (u, v) com $u \in X$ e $v \in \bar{X} = E_H \setminus X$.

Observe que $c(\delta^+(\phi^{-1}(F))) = 1 \cdot |\delta^+(\phi^{-1}(F))| = |\phi^{-1}(F)|$ e, portanto, $c(C) \leq |\phi^{-1}(F)|$. Como apenas arcos de $\phi^{-1}(G)$ para F e de B para t têm capacidades finitas, eles são os únicos que podem estar em C. Assim, há um conjunto $U \subseteq F$ tal que

$$C = \{(y, v) : y \in \phi^{-1}(F \setminus U) \text{ e } v = \phi(y)\} \quad \cup \quad ((N_H(U) \setminus F) \times \{t\}),$$

onde por F nos referimos à primeira cópia de cada elemento de F em V_H.

Seja $Q = N_G^6(U) \cap B \cap F$. Afirmamos que $Q \subseteq U$. Tome v em U e $\bar{w}$ em Q. Note que os arcos $(v, \bar{w})$ e $(\bar{w}, w)$ têm capacidades infinitas, de modo que nenhum deles pode estar no corte C. Segue que $w \in U$ e, assim, $Q \subseteq U$.

Como não há arco saindo de Q e entrando em t, a equação acima nos permite expressar a capacidade de C como

$$c(C) = |\phi^{-1}(F \setminus U)| + L(N_H(U) \setminus Q).$$

Observe que $N_H(U) = (N_H(U) \setminus Q) \cup Q$. Assim, pelo laço começando na linha **??** do Algoritmo **??**, temos que $L(U) \leq L(B \cap N_G^6(U)) = L(N_H(U)) = L(N_H(U) \setminus Q) + L(Q)$, e então

$$\begin{aligned} c(C) &= |\phi^{-1}(F \setminus U)| + L(N_H(U) \setminus Q) \\ &\geq |\phi^{-1}(F \setminus U)| + L(U) - L(Q) \\ &= |\phi^{-1}(F \setminus U)| + |\phi^{-1}(U)| + (L(U) - |\phi^{-1}(U)|) - L(Q) \\ &\geq |\phi^{-1}(F \setminus U)| + |\phi^{-1}(U)| + L(Q) - L(Q) \\ &= |\phi^{-1}(F \setminus U)| + |\phi^{-1}(U)| \;=\; |\phi^{-1}(F)|, \end{aligned}$$

onde a segunda desigualdade vem do fato de que ϕ não atribui nenhum vértice a Q, e de que $Q \subseteq U$.

Então o valor de um fluxo máximo em H é exatamente $|\phi^{-1}(F)|$, de modo que todo arco de $\phi^{-1}(F)$ até F tem fluxo exatamente 1. Resta obter uma atribuição $\psi : \phi^{-1}(F) \to B \setminus F$. Para cada vértice y em $\phi^{-1}(F)$, definimos $\psi(y) = u$, onde u é o centro em $B \setminus F$ que recebe a unidade de fluxo que passa por y. Na Figura 4.1, por exemplo, uma unidade de fluxo poderia passar pelo caminho $(s, y, v, \bar{x}, x, \bar{w}, w, u, t)$ e, então, definimos $\psi(y) = u$.

Como cada vértice u em $B \setminus F$ pode receber no máximo L_u unidades de fluxo, sabemos que ψ respeita as capacidades dos vértices. Além disso, como há no máximo $\alpha - 1$ elementos em $B \cap F$, cada unidade de fluxo saindo de um vértice v em F pode passar por no máximo $\alpha - 1$ arcos reversos em H sem criar um ciclo, e portanto pode passar por no máximo α arcos a partir de F antes de atingir um vértice u em $B \setminus F$. Assim, $d_G(v, u) \leq 6\alpha$.

Portanto, para cada $y \in \phi^{-1}(F)$

$$d_G(y, \psi(y)) \;\leq\; d_G(y, \phi(y)) + d_G(\phi(y), \psi(y)) \;\leq\; \beta + 6\alpha.$$

Assim, podemos definir uma reatribuição conservativa ϕ_F:

$$\phi_F(v) = \begin{cases} \phi(v) & \text{se } \phi(v) \notin F, \\ \psi(v) & \text{caso contrário.} \end{cases}$$

Verifiquemos que ϕ_F é uma reatribuição conservativa válida de distância $(\beta + 6\alpha)$. Seja u um centro aberto pelo algoritmo, isto é, $u \in S \cup B$. Se $u \in S$, então $\phi_F^{-1}(u) = \phi^{-1}(u)$. Se $u \in B$, então, $\phi_F^{-1}(u) = \psi^{-1}(u)$. Se $u \in B$, então $\phi_F^{-1}(u) = \psi^{-1}(u)$. Além disso, ϕ e ψ respeitam as capacidades dos centros aos quais atribuem clientes de centros em F. Finalmente, tome um vértice $y \in V$. Se $\phi(y) \notin F$, então $d_G(y, \phi_F(y)) = d_G(y, \phi(y)) \leq \beta$. Se, por outro lado, $\phi(y) \in F$, então $d_G(y, \phi_F(y)) = d_G(y, \psi(y)) \leq \beta + 6\alpha$. □

Usando o algoritmo de aproximação com a melhor razão conhecida para o problema dos k-centros com capacidades, obtemos o seguinte.

Corolário 4.14. *O Algoritmo* **??**, *usando o algoritmo de An et al. [3] para o problema dos k-centros com capacidades, é uma $(9+6\alpha)$-aproximação para o problema dos k-centros com capacidades e tolerância a falhas conservativo com parâmetro α fixado.*

4.3.3 Caso com capacidades não uniformes e não conservativo

Uma formulação PL inicial

Lembre que recebemos um grafo conexo sem pesos nas arestas, e nosso objetivo é decidir se existe uma solução de distância 1 (ver Seção **??**). Como no trabalho de Cygan e Kociumaka [7], usamos um PL inteiro que formula o problema. Se, após relaxar as restrições de integralidade, o PL for inviável, então sabemos que não existe solução de distância 1, caso contrário arredondamos a solução e obtemos uma aproximação.

Como na formulação para o problema dos k-centros com capacidades, temos variáveis de abertura y_u para cada vértice u, representando a escolha de u como centro, e variáveis de atribuição x_{uv}, representando a atribuição do vértice v ao centro u. No caso da variante com tolerância a falhas, para cada cenário de falha, isto é, para cada possível conjunto $F \subseteq V$ de centros que podem falhar, com $|F| \leq \alpha$, precisamos ter uma atribuição distinta dos vértices aos centros abertos por y que não estão em F, isto é, que não falharam. Uma possível maneira de formular a variante tolerante a falhas é ter variáveis de atribuição diferentes para cada F. Equivalentemente, no entanto, para obtermos uma formulação que não usa um conjunto diferente de variáveis para cada cenário de falhas, adicionamos restrições que codificam a condição de Hall, uma condição necessária e suficiente para existência de emparelhamento perfeito em grafos bipartidos, dada pelo Teorema de Hall [1] e que usamos anteriormente na Seção 3.3.1. De fato, é a condição de Hall que o PL inteiro a seguir, denotado por $PLI_{k,\alpha}(G)$, modela:

$$\begin{aligned} \textstyle\sum_{u\in V} y_u &= k \\ |U| &\leq \textstyle\sum_{u\in N_G(U)\setminus F} y_u L_u && \forall\, U \subseteq V,\ F \subseteq V : |F| = \alpha \\ y_u &\in \{0,1\} && \forall\, u \in V. \end{aligned}$$

Observe que o $PLI_{k,\alpha}(G)$ formula o problema dos k-centros com capacidades e tolerância a falhas. A primeira restrição garante que exatamente k centros sejam abertos. O segundo conjunto de restrições garante que, para cada cenário de falhas, exista uma atribuição viável dos clientes a centros abertos que não falharam. Observe que, fixado um conjunto F, a existência de uma tal atribuição é equivalente à existência de um emparelhamento M no grafo bipartido formado por clientes e unidades abertas de capacidade com a propriedade de que M cobre todo o conjunto de clientes. O segundo conjunto de restrições exige que a condição de Hall e, assim, a existência de um tal emparelhamento M, seja garantida.

Gap de integralidade. Uma primeira tentativa poderia ser relaxar $PLI_{k,\alpha}(G)$ diretamente. Quando as restrições de integralidade são relaxadas, no entanto, a fração de abertura, dada por y, de um cenário de falhas F pode ser estritamente menor do que α, isto é, $y(F) < \alpha$, mesmo se tomarmos F como um conjunto de vértices com abertura não nula. Assim, as restrições consideradas são mais fracas do que precisamos. Considere o seguinte exemplo.

Seja C_n o ciclo sobre n vértices, tomando $n = s^2$ para algum inteiro positivo par. Seja G_n o grafo obtido de C_n adicionando-se arestas entre dois vértices que distam em C_n de no máximo s. Observe que qualquer par de antípodas, ou vértices que estão à distância máxima um do outro, em C_n estão à distância $s/2$ em G_n. Se definirmos $L_u = n$ para todo u em G_n, $k = s$ e $\alpha = k - 1$, então o custo de qualquer solução para essa instância é $s/2$, uma vez

que para qualquer conjunto de k vértices em G_n, pode ser que todos, exceto um, falhem. Agora, seja y o vetor com $y_u = 1/s$ para todo u. Afirmamos que y é viável para a relaxação de $PLI_{k,\alpha}(G_n)$. Primeiro, observe que $\sum_{u \in V} y_u = s = k$. Além disso, como cada vértice possui $2s$ vizinhos, para qualquer conjunto de centros F de cardinalidade $\alpha = k-1 = s-1$, o segundo conjunto de restrições do programa é satisfeito, pois o lado direito da desigualdade é sempre pelo menos igual a n, e o lado esquerdo é no máximo n. Então y é viável e, portanto, o limitante inferior obtido pela relaxação do $PLI_{k,\alpha}(G_n)$ pode ser arbitrariamente pequena comparada à solução ótima. Em outras palavras, o problema de minimização obtido de $PLI_{k,\alpha}(G_n)$ tem gap de integralidade ilimitado.

Lidando com o gap de integralidade

Suponha que conhecêssemos um subconjunto $B \subseteq S^*$ de centros que podem falhar, onde S^* é o conjunto de centros de uma solução ótima. Então poderíamos forçar $y_u = 1$ para todo u em B, isto é, decidimos abrir u antes de resolver o PL. Caso tenhamos um cenário de falha $F \subseteq B$, evitamos o problema do gap de integralidade ilimitado, pois temos $y(F) = |F|$. Como não sabemos obter eficientemente um tal B contendo apenas centros de alguma solução ótima fixada, e como um cenário de falha F pode ter centros que não estão em B, visamos a dois objetivos mais relaxados:

(O1) escolhemos um conjunto de centros B tal que cada membro de B esteja próximo de um centro distinto em uma solução ótima; e

(O2) supomos que apenas centros em B podem falhar, e isso configura o pior caso de falha.

Para atingir tais objetivos, usaremos a separação do grafo em monarcas de Khuller e Sussmann [13], como na Seção 3. Aqui, novamente, a ideia é poder argumentar que em cada cluster, ou império, necessariamente há uma certa quantidade de centros em qualquer solução ótima. Um cenário de pior caso, como buscamos construir para a escolha de B, corresponde a uma escolha local dos centros mais capacitados, conforme explicitaremos adiante.

Lembre que a decomposição do grafo na árvore de monarcas nos dá o seguinte resultado:

Lema 4.15. *Dado um grafo conexo $G = (V, E)$, podem-se obter um conjunto de monarcas $\Gamma \subseteq V$, e uma partição de V em conjuntos $\{C_v\}_{v \in \Gamma}$, tais que*

- *existe uma árvore enraizada T sobre Γ, com $d_G(u, v) = 3$ para toda aresta (u, v) de T;*
- *$N_G(v) \subseteq C_v$ para todo v em Γ; e*
- *$d_G(u, v) \leq 2$ para todo v em Γ e todo u em C_v.*

A seleção de centros pré-abertos. Aplicamos o Lema 4.15 e obtemos a clusterização de V em monarcas. Seja $v \in \Gamma$ um monarca, e considere uma solução em G. Como até α centros nessa solução podem falhar, é preciso haver no mínimo $\alpha + 1$ centros em $N(v)$, caso contrário haveria um cenário de falha para o qual é impossível atribuir v a um vizinho. Segue que os elementos de C_v estão à distância até 3 de pelo menos $\alpha + 1$ centros pertencentes ao conjunto C_v. Além disso, uma vez que os conjuntos $N(v)$ são disjuntos para todo v em Γ, concluímos que há pelo menos $\alpha+1$ centros por cluster (império) em qualquer solução em G.

Para atingir o objetivo (O1), podemos selecionar, para cada $v \in \Gamma$, qualquer subconjunto de tamanho até $\alpha + 1$ de C_v. Para atingir (O2), precisamos pensar na capacidade total que pode ficar indisponível em um cenário de falha. Para cada cluster, a maior quantidade de capacidade que pode ser retirada em um dado cenário não é maior do que a capacidade

acumulada dos α vértices de maior capacidade no cluster. Portanto, selecionamos tais α vértices como o conjunto B de *backups*.

Formalmente, para cada $v \in \Gamma$, seja $B_v \subseteq C_v$ um conjunto com α elementos de C_v com as maiores capacidades. Empates entre vértices podem ser resolvidos arbitrariamente. Então B_v é o conjunto de centros pré-abertos para o cluster C_v. O conjunto de todos os centros pré-abertos é definido como

$$B = \cup_{v \in \Gamma} B_v.$$

Modificando a formulação PL

Nós pré-abrimos os elementos de B adicionando ao $PLI_{k,\alpha}(G)$ a restrição $y_u = 1$, para todo $u \in B$. Quando nós pré-estabelecemos uma solução, nós podemos deixar a formulação linear original inviável, uma vez que é possível que nenhuma solução de distância-1 abra os elementos de B. Contudo, como em qualquer solução de distância-1 existem pelo menos α centros em um dado cluster, cada cento nessa solução está a uma distância de 3 a um elemento distinto de B de capacidade não inferior. Assim, nós podemos converter uma solução de distância-1 em uma solução de distância-4 reatribuindo clientes a elementos de B e preservando a maior parte da estrutura no PL original.

Consertando a viabilidade. Para obter uma relaxação do PL útil, à medida que pré-abrimos o conjunto B de centros, nós modificamos o grafo G usado na formulação. Para cada cluster C_v, nós aumentamos G com arestas conectando cada cliente que poderia potencialmente ser servido por centros em C_v para cada vértice no conjunto B_v. Mais precisamente, nós definimos o grafo dirigido $G' = G'(G, \{C_v\}_{v \in \Gamma}) = (V, E')$, onde E' é o conjunto de arcos (u, w) tal que $\{u, w\} \in E$, ou existe v in Γ e t em $N(v)$ tal que $\{u, t\} \in E$ e $w \in B_v$ (veja Figura 4.2). Lembramos que usamos um grafo dirigido, porque nós queremos permitir uma reatribuição de um cliente de um centro arbitrário no cluster a um centro em B, mas não o contrário.

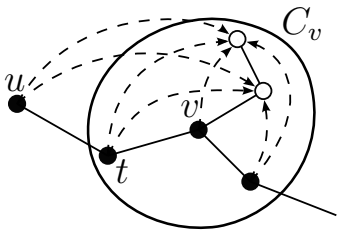


Figura 4.2: *As linhas tracejadas representam arcos adicionados ao grafo G para obter G', e as linhas contínuas representam arcos duplicados em direções opostos. Os vértices com preenchimento branco representam B_v.*

Uma nova formulação. Na nova formulação, nós consideramos apenas cenários $F \subseteq B$. Assim, em uma solução viável y, nós teremos $y(F) = |F|$ para cada cenário F. Também, para cada monarca do cluster, nós queremos abrir (fracionariamente) pelo menos um um centro que não falhou na sua vizinhança, para cada cenário de falha. No programa inteiro $PLI_{k,\alpha}(G)$, isso estava implícito pelas restrições, mas quando y não é inteiro, devem existir centros com capacidades altas que satisfazem a demanda local com menos do que uma unidade aberta. Portanto, nós temos uma restrição adicional para cada monarca v do cluster para assegurar que existe uma unidade de abertura (fracionária) em $N(v)$ excluindo qualquer abertura de B.

Nós obtemos um novo programa linear, denotado por $PL_{k,\alpha}(G, \{C_v\}_{v\in\Gamma})$.

$$\begin{array}{rll} \sum_{u\in V} y_u = k & & \\ |U| \leq \sum_{u\in N_{G'}(U)\setminus F} y_u L_u & & \forall\, U \subseteq V,\ F \subseteq B : |F| = \alpha \\ 1 \leq \sum_{u\in N_G(v)\setminus B} y_u & & \forall\, v \in \Gamma \\ y_u = 1 & & \forall\, u \in B \\ 0 \leq y_u \leq 1 & & \forall\, u \in V. \end{array}$$

Note que, ao contrário do $PLI_{k,\alpha}(G)$, o programa $PL_{k,\alpha}(G, \{C_v\}_{v\in\Gamma})$ depende do clustering obtido. O lema a seguir afirma que $PL_{k,\alpha}(G, \{C_v\}_{v\in\Gamma})$ é uma "relaxação" de $PLI_{k,\alpha}(G)$, isto é, se $PL_{k,\alpha}(G, \{C_v\}_{v\in\Gamma})$ é inviável, então nós obtemos um certificado de que não existe solução de distância-1 para G.

Lema 4.16. *Se $PLI_{k,\alpha}(G)$ é viável, então $PL_{k,\alpha}(G, \{C_v\}_{v\in\Gamma})$ é viável.*

Demonstração. Suponha que $PLI_{k,\alpha}(G)$ é viável. Seja y uma solução viável para $PLI_{k,\alpha}(G)$, e seja R o conjunto de centros correspondentes a y.

Primeiramente, nós definimos uma função injetora β de R em $R \cup B$ que cobre B. Lembre que Γ é o conjunto de midpoints. Para cada v em Γ, sejam $u_1, \ldots, u_\alpha$ os elementos de B_v em ordem não crescente de capacidade. Analogamente, sejam $w_1, \ldots, w_\alpha, \ldots$ os elementos de $R \cap N(v)$ em ordem não crescente de capacidade (lembre que cada $N(v)$ tem pelo menos $\alpha + 1$ centros em uma solução ótima). No caso de empate, os elementos em B_v deveriam vir primeiro nessa ordenação. Para cada i com $1 \leq i \leq \alpha$, nós definimos $\beta(w_i) = u_i$. Finalmente, para cada w em R cujo $\beta(w)$ ainda não foi definido, consideramos $\beta(w) = w$. Note que, por causa da regra de desempate, nesse caso, $w \notin B$. Além disso, $L_w \leq L_{\beta(w)}$ para todo w em R e a função inversa β^{-1} é bem definida na imagem de β.

Seja $R' = \beta(R)$ e seja y' o vetor característico de R'. Nós afirmamos que y' é uma solução viável para $PL_{k,\alpha}(G, \{C_v\}_{v\in\Gamma})$. Sejam $U \subseteq V$ e $F \subseteq B$ com $|F| = \alpha$. Da viabilidade de y para $PLI_{k,\alpha}(G)$, e como $|\beta^{-1}(F)| = |F| = \alpha$, nós temos

$$\begin{aligned} |U| \ &\leq\ \textstyle\sum_{u\in N_G(U)\setminus\beta^{-1}(F)} y_u L_u \ =\ \sum_{u\in(N_G(U)\setminus\beta^{-1}(F))\cap R} L_u \\ &=\ \textstyle\sum_{u\in(N_G(U)\cap R)\setminus\beta^{-1}(F)} L_u \ \leq\ \sum_{u\in\beta((N_G(U)\cap R)\setminus\beta^{-1}(F))} L_u \\ &=\ \textstyle\sum_{u\in\beta(N_G(U)\cap R)\setminus F} y'_u L_u \ \leq\ \sum_{u\in N_{G'}(U)\setminus F} y'_u L_u. \end{aligned}$$

A verificação que as outras restrições também são satisfeitas para y' é direta. □

Embora $PL_{k,\alpha}(G, \{C_v\}_{v\in\Gamma})$ tenha um número exponencial de restrições, o lema a seguir mostra que existe um algoritmo de tempo polinomial que resolve a separação para ele.

Lema 4.17. *Para α fixo, existe um algoritmo que, em tempo polinomial, decide se o vetor y é viável para $PL_{k,\alpha}(G, \{C_v\}_{v\in\Gamma})$. Se y não é viável, o algoritmo também devolve a restrição do $PL_{k,\alpha}(G, \{C_v\}_{v\in\Gamma})$ que é violada por y.*

Demonstração. Nos concentraremos no segundo conjunto de restrições, como existem polinomialmente muitas restrições de outros tipos. Note que o número de cenários distintos F é $O(|V|^\alpha)$, que é polinominal já que α é constante. Fixe um cenário de falha F e suponha que nós podemos resolver o seguinte problema:

$$\min_{U\subseteq V} \sum_{u\in N_{G'}(U)\setminus F} y_u L_u - |U|. \tag{4.1}$$

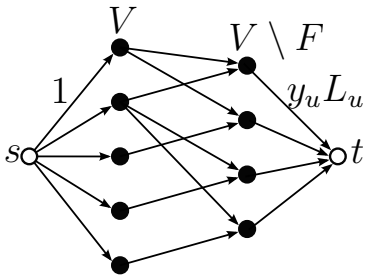


Figura 4.3: *Rede de fluxo com fonte s e sorvedouro t. Existe um arco com capacidade unitária de s para cada $v \in V$, e um arco sem capacidade de v para cada vizinho $u \in N(U) \setminus F$. Além disso, para cada $u \in V \setminus F$, existe um arco para t com capacidade $y_u L_u$.*

Se esse valor é não-negativo, então todas as restrições no segundo conjunto para esse cenário F são satisfeitas, caso contrário, existe um subconjunto U^* de V para o qual a restrição é violada, como queríamos encontrar. Nós podemos reescrever o problema de minimização anterior como o seguinte programa linear inteiro nas variáveis binárias a_u para u em V, e b_u para u em $V \setminus F$:

$$\begin{array}{lll} \min & \sum_{u \in V \setminus F} b_u(y_u L_u) + \sum_{u \in V} a_u - |V| & \\ \text{s.t.} & a_u + b_v \geq 1 & \forall\ (u,v) \in G' \\ & a_u,\ b_v \in \{0,1\} & \forall\ u \in V,\ v \in V \setminus F. \end{array}$$

A variável a_u indica que u não está em U e a variável b_v indica que existe algum u na lista de adjacência de v que está em U (isto é, $a_u = 0$). A matriz correspondente para esse problema é totalmente unimodular, então a relaxação tem solução ótima inteira, que pode ser obtida em tempo polinomial. Note que esse problema (tirando a constante $-|V|$ na função objetivo) corresponde à formulação do corte mínimo para o problema de fluxo de rede descrito na Figura 4.3. Assim é suficiente executar qualquer algoritmo de fluxo máximo e corte mínimo. □

Corolário 4.18. *Para α fixo, $PL_{k,\alpha}(G, \{C_v\}_{v \in \Gamma})$ pode ser resolvido em tempo polinomial.*

Se, para um α arbitrário, existir um algoritmo de tempo polinomial para encontrar um conjunto F com $|F| = \alpha$ que minimiza o valor de (4.1), então uma versão mais forte do Corolário 4.18 sem a restrição de α ser fixo será válida. Infelizmente, tal algoritmo existe apenas se P = coNP [8].

The algorithm

Our algorithm consists of two parts. In the first, we round a fractional solution y of $PL_{k,\alpha}(G, \{C_v\}_{v \in \Gamma})$, and obtain a set R of k centers. In the second part, for each failure scenario $F \subseteq R$ with $|F| \leq \alpha$, we have to obtain an assignment from V to $R \setminus F$.

Rounding. Since we have pre-opened centers, we round only the residual set of vertices $V \setminus B$. This phase is based on the algorithm of An et al. [3] for the capacitated (non-fault-tolerant) k-center. The main difference is that we do not allow transfers from or to vertices in the set B. The algorithm reduces the problem of rounding a general graph to the problem of rounding tree instances. There are three consecutive transfers. In the first step, we concentrate one unit of opening on one auxiliary vertex that is added at the same location as the cluster midpoint. In the second step, we create a tree instance using the auxiliary vertices as internal nodes, and obtain an integral transfer using Lemma **??**. In the last step, the opening of auxiliary vertices is transferred back to vertices of the original graph. A detailed description is presented in the following:

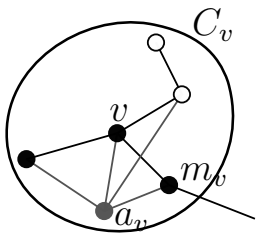


Figura 4.4: *Auxiliary vertex a_v is adjacent to each vertex in $N(v)$. The white vertices represent set B_v.*

Step 1. For each cluster C_v, choose an element m_v in the neighborhood of the midpoint v that is not pre-opened, and has the largest capacity, that is, $m_v = \arg\max_{u \in N_G(v) \setminus B} L_u$. Create an auxiliary vertex a_v at the same location as v (add an edge to a_v from each element of $N(v)$ as in Figure 4.4), with capacity $L_{a_v} = L_{m_v}$, and initial opening $y_{a_v} = 0$. Next, aggregate one unit of opening to a_v by transferring fractional openings from $N_G(v) \setminus B$ to a_v. This can be done as $\sum_{u \in N_G(v) \setminus B} y_u \geq 1$. The transfer proceeds as follows: for each u in $N_G(v) \setminus B$, decrease y_u, while increasing y_{a_v}, until y_u becomes 0. The process is interrupted once y_{a_v} reaches 1. The result is a distance-1 transfer $y^{(1)}$. The first vertex to have its fractional opening transferred is m_v, so that, at the end of this step, $y^{(1)}_{m_v} = 0$.

Step 2. Obtain a tree T from the clustering tree by replacing each midpoint v with a_v for every v in Γ. Next, for each cluster C_v, select every vertex u in C_v such that $0 < y^{(1)}_u < 1$ and add a leaf corresponding to u, connected to a_v. Finally, apply Lemma **??**, and obtain an integral T-restricted distance-2 transfer $y^{(2)}$ (starting with $y^{(1)}$). Notice that $d_G(w_1, w_2) = 3$ for each edge (w_1, w_2) of T if both w_1 and w_2 are internal nodes; and $d_G(w_1, w_2) \leq 2$ if either w_1 or w_2 is a leaf. Hence, $y^{(2)}$ can be interpreted as a distance-$(2 \cdot 3)$ transfer of $y^{(1)}$ (on the graph G).

Step 3. For each cluster C_v, transfer the opening of the auxiliary vertex a_v back to the original vertex m_v. This is possible since $y^{(2)}_{m_v} = 0$. Obtain a final integral distance-1 transfer $y^{(3)}$. Open the set of vertices $R \subseteq V$ that corresponds to the characteristic vector $y^{(3)}$.

Assignment. After opening centers R, up to α failures might occur. Our algorithm must provide a valid assignment for each failure scenario $F \subseteq R$. We will consider two cases, depending on whether $F \subseteq B$.

First, we examine the case that F is a subset of B. In this case, $PL_{k,\alpha}(G, \{C_v\}_{v \in \Gamma})$ assures the existence of an assignment from V to a set of fractionally opened centers that does not intersect F. Since the rounding algorithm obtains an integral G-restricted distance-8 transfer (by adding up the three consecutive transfers), this will lead to a distance-9 solution that does not assign to any element of F.

For the case that F is not a subset of B, we may not rely on the existence of a fractional assignment obtained from the LP. Instead, we will show that, for each F, there exists a corresponding $F' \subseteq B$, and that a distance-9 solution for failure scenario F' can be transformed into a distance-10 solution for failure scenario F. Indeed, we will show that each element u assigned to a center $v \in F \setminus F'$ in the former solution may be reassigned to a distinct element $v' \in F' \setminus F$ in the latter solution, such that v and v' are in the same cluster, and $L_{v'} \geq L_v$.

A naive analysis of the preceding strategy would yield a 13-approximation, as the distance between v and v' might be 4, and thus $d(u, v') \leq d(u, v) + d(v, v') \leq 9 + 4$. To obtain a more

refined analysis, we will bound the distance between u and the midpoint associated to v. More precisely, denote by $\delta(v)$ the midpoint of the cluster that contains v. We obtain the following lemma.

Lema 4.19. *Consider $F \subseteq B$ with $|F| = \alpha$ and let R be the integral transfer obtained from y by the rounding algorithm above. One can find, in polynomial time, an assignment $\phi : V \to R \setminus F$ such that $d_G(u, \phi(u)) \leq 9$ and $d_G(u, \delta(\phi(u))) \leq 8$ for each u in V.*

Demonstração. Let $\bar{V} = V \cup \{a_v : v \in \Gamma\}$ be the union of the vertices of G and the auxiliary vertices, and $\bar{G}$ be the graph obtained after we add the auxiliary vertices to G. Fix a subset $U \subseteq V$.

Recall that the rounding algorithm considers an initial feasible solution $y = y^{(0)}$, and obtains consecutive transfers $y^{(1)}, y^{(2)}, y^{(3)}$. In the following, for each transfer $y^{(i)}$, for $0 \leq i \leq 2$, and each $U \subseteq V$, we will consider a set $X = X^{(i)}(U)$ of vertices, excluding faulty elements, whose total installed capacity exceeds $|U|$. That is, we want to obtain X such that the value $ic^{(i)}(X) := \sum_{u \in X \setminus F} y_u^{(i)} L_u \geq |U|$. Initially, before any transfer is performed, we have that $y^{(0)} = y$ and, by the constraints of $PL_{k,\alpha}(G, \{C_v\}_{v \in \Gamma})$, we have that $|U| \leq \sum_{u \in N_{G'}(U) \setminus F} y_u L_u$, so we set $X^{(0)}(U) = N_{G'}(U)$.

In the first step, we have a distance-1 transfer. Notice that $N_{G'}(U) \setminus B = N_G(U) \setminus B$. Also, recall that the transfer is restricted to vertices in $\bar{V} \setminus B$. We obtain

$$\begin{aligned} |U| \leq ic^{(0)}(X^{(0)}(U)) &= ic^{(0)}((N_{G'}(U) \cap B) \cup (N_G(U) \setminus B)) \\ &\leq ic^{(1)}((N_{G'}(U) \cap B) \cup (N_G^2(U) \setminus B)). \end{aligned}$$

Hence we set $X^{(1)}(U) = (N_{G'}(U) \cap B) \cup (N_G^2(U) \setminus B)$.

In the second step, we have an integral T-restricted distance-2 transfer of $y^{(1)}$. Once again, since T does not include vertices of B, we obtain

$$\begin{aligned} ic^{(1)}(X^{(1)}(U)) &= ic^{(1)}((N_{G'}(U) \cap B) \cup (N_G^2(U) \setminus B)) \\ &\leq ic^{(2)}((N_{G'}(U) \cap B) \cup N_T^2(N_G^2(U) \setminus B)). \end{aligned}$$

We set $X^{(2)}(U) = (N_{G'}(U) \cap B) \cup N_T^2(N_G^2(U) \setminus B)$.

Let $\bar{R} \subseteq \bar{V}$ be the set corresponding to vector $y^{(2)}$. First consider a bipartite graph $H = (V \cup \bar{R}, D)$ with $\{u, v\}$ in D if $v \in X^{(2)}(\{u\}) \setminus F$. Then modify H by including $L_v - 1$ additional copies of each vertex v in $\bar{R}$. Notice that now, for each $U \subseteq V$, we have $|N_H(U)| = |\bigcup_{u \in U} N_H(\{u\})| = ic^{(2)}(\bigcup_{u \in U} X^{(2)}(\{u\})) = ic^{(2)}(X^{(2)}(U)) \geq |U|$. This is exactly Hall's condition for the existence of a matching in H covering V. We obtain such a matching in polynomial time, and obtain a corresponding assignment $\phi' : V \to \bar{R}$.

Now, for every vertex u in V, we show that the distance from u to $\delta(\phi'(u))$ is bounded by 8. Recall that $\phi'(u) \in N_H(\{u\}) = X^{(2)}(\{u\})$. We have two cases. First, suppose that $\phi'(u) \in N_{G'}(\{u\}) \cap B$. Since we have that $d_G(u, \phi'(u)) \leq 4$, by the construction of G', we obtain that $d_G(u, \delta(\phi'(u))) \leq 6$, and we are done. Now, assume that $\phi'(u) \in N_T^2(N_G^2(\{u\}) \setminus B)$. In this case, there must be some v in $N_G^2(\{u\}) \setminus B$ and a shortest path ρ connecting v to $\phi'(u)$ in T. We consider two possibilities. If the length of ρ is 1, then $\rho = (v, \phi'(u))$, and we deduce that $d_G(u, \delta(\phi'(u))) \leq d_G(u, v) + d_G(v, \phi'(u)) + d_G(\phi'(u), \delta(\phi'(u))) \leq 2 + 3 + 2 = 7$. If the length of ρ is 2, then there exists w such that $\rho = (v, w, \phi'(u))$, and we get that $d_G(u, \phi'(u)) \leq d_G(u, v) + d_G(v, w) + d_G(w, \phi'(u)) \leq 2 + 3 + 3 = 8$. If $\phi'(u)$ is an internal node of T, then $\delta(\phi'(u)) = \phi'(u)$, and thus $d_G(u, \delta(\phi'(u))) \leq 8$. Otherwise, w must be an internal node and $\phi'(u)$ a leaf of w. Hence $\delta(\phi'(u)) = w$, and therefore $d_G(u, \delta(\phi'(u))) \leq d_G(u, \phi'(u)) \leq 8$.

A similar analysis also allows us to deduce that $d_G(u, \phi'(u)) \leq 8$. To obtain a final assignment $\phi : V \to R$, we reassign each vertex u assigned to an auxiliary vertex a_v, to the vertex m_v, that is, for each u in V, if $\phi'(u) = a_v$ for some v in Γ, then set $\phi(u) = m_v$, otherwise set $\phi(u) = \phi'(u)$. □

Now we may obtain the approximation factor.

Teorema 4.20. *There exists a* 10*-approximation for the capacitated* α*-fault-tolerant* k*-center with fixed* α*.*

Demonstração. Consider a failure scenario $F \subseteq R$ with $|F| = \alpha$. For each cluster C_v, let F_v be the set of centers that failed in cluster C_v. Also, let F'_v be the set of the $|F_v|$ most capacitated centers in B_v, and $F' = \bigcup_{v\in\Gamma} F'_v$. We use Lemma 4.19, and obtain an assignment $\phi' : V \to R \setminus F'$. Now, for each v in Γ, obtain an ordering $\{u_1, \ldots, u_t\}$ of the vertices in $F'_v \setminus F_v$, and an ordering $\{v_1, \ldots, v_t\}$ of the vertices in $F_v \setminus F'_v$. For each w that is assigned to v_i, for some $1 \leq i \leq t$, reassign it to u_i, that is, for every w such that $\phi'(w) = v_i$, set $\phi(w) = u_i$. Notice that this leads to a valid assignment ϕ, since $L_{u_i} \geq L_{v_i}$ for every $1 \leq i \leq t$. Also, we notice that since u_i and v_i are in the same cluster, $d_G(\delta(v_i), u_i) \leq 2$, and thus $d_G(w, \phi(w)) \leq d_G(w, \delta(\phi'(w))) + d_G(\delta(\phi'(w)), u_i) \leq 8 + 2 = 10$. □

4.3.4 Caso $\{0, L\}$-capacitado e não conservativo

Para um dado L, o problema dos k-centros com capacidades $\{0, L\}$ e tolerância a falhas é a versão particular do problema dos k-centros com capacidades e tolerância a falhas no qual cada vértice tem capacidade 0 ou L. Vértices com capacidade 0 são chamados de 0-*vértices* e vértices com capacidade L são chamados de L-*vértices*. Para um dado conjunto A de vértices, nós denotamos por A^L o conjunto contendo todos os L-vértices de A.

Formulação PL

A seguir mostramos um algoritmo de arredondamento para o caso com capacidades $\{0, L\}$. Como na Seção 4.3.3, nós formulamos o problema usando $PLI_{k,\alpha}(G)$. Nesse caso, contudo, nós podemos reescrever o programa de forma que somente L-vértices apareçam na soma e todos os coeficientes sejam iguais, isto é, $PLI_{k,\alpha}(G)$ pode ser escrito como:

$$\begin{aligned} \textstyle\sum_{u\in V} y_u &= k \\ |U| &\leq \textstyle\sum_{u\in (N_G(U))^L \setminus F} y_u L \qquad && \forall\, U \subseteq V,\ F \subseteq V : |F| \leq \alpha \\ y_u &\in \{0, 1\} && \forall\, u \in V. \end{aligned}$$

Note que a segunda linha no problema acima pode ser simplificada. A observação chave é que, no pior caso, a capacidade total dos centros que falharam é sempre a constante αL. De fato, considere a solução inteira viável y e um subconjunto fixo $U \subseteq V$ tal que $U \neq \emptyset$. Seja $H = \{u \in (N_G(U))^L : y_u = 1\}$. Segue que $|H| > \alpha$, caso contrário nós teríamos $|U| \leq \sum_{u\in (N_G(U))^L \setminus H} y_u L = \sum_{u\in (N_G(U))^L \setminus H} 0 \cdot L = 0$, que é uma contradição, pois U não é vazio. Seja F' um subconjunto qualquer de H com $|F'| = \alpha$. Pela restrição de desigualdade em $PLI_{k,\alpha}(G)$, para $F = F'$, nós obtemos

$$\begin{aligned} |U| \leq \textstyle\sum_{u\in (N_G(U))^L \setminus F'} y_u L &= \textstyle\sum_{u\in (N_G(U))^L} y_u L - \sum_{u\in (N_G(U))^L \cap F'} 1 \cdot L \\ &= \textstyle\sum_{u\in (N_G(U))^L} y_u L - \alpha L. \end{aligned}$$

Portanto, o seguinte programa linear, que é denotado por $PLU_{k,\alpha}(G)$, é uma relaxação do $PLI_{k,\alpha}(G)$.

$$\begin{aligned} \textstyle\sum_{u\in V} y_u &= k \\ |U| &\leq \textstyle\sum_{u\in (N_G(U))^L} y_u L - \alpha L && \forall\, U \subseteq V,\ U \neq \emptyset \\ 1 &\leq \textstyle\sum_{u\in (N_G(v))^L} y_u && \forall\, v \in V \\ 0 \leq y_u &\leq 1 && \forall\, u \in V. \end{aligned}$$

Ao contrário do $PL_{k,\alpha}(G, \{C_v\}_{v\in\Gamma})$, podemos resolver a separação para esse programa mesmo se α é parte da instância. A diferença é que, nessa formulação, os cenários de falha não precisam ser enumerados. Dada uma solução candidata y, nós podemos calcular o valor mínimo de $\sum_{u\in N^L(U)} y_u L - |U|$ entre todos os conjuntos U e verificar se esse valor é pelo menos αL. Isso pode ser feito em tempo polinomial usando um algoritmo de fluxo máximo e corte mínimo com argumentos muito similares aos da prova do Lema 4.17. Isso significa que nós podemos resolver a separação para o $PLU_{k,\alpha}(G)$ em tempo polinomial, o que implica no lema a seguir.

Lema 4.21. *O $PLU_{k,\alpha}(G)$ pode ser resolvido em tempo polinomial mesmo se α é parte da entrada.*

Arredondamento

Para o problema dos k-centros com capacidades $\{0, L\}$ sem tolerância a falhas, An et al. [3] executam um pré-processamento adicional do grafo de entrada para obter um clustering com propriedades mais fortes. Dessa forma, eles obtêm uma 5-transferência inteira. Isto é, antes do pré-processamento descrito na Seção **??**, que produz um grafo conexo sem peso $G = (V, E)$, eles removem qualquer aresta conectando dois 0-vértices. Nós aplicamos seu algoritmo de arredondamento para a solução obtida pelo $PLU_{k,\alpha}(G)$, o que resultou no lema a seguir.

Lema 4.22. *Suponha que G é um grafo conexo tal que cada vértice é um 0-vértice ou um L-vértice, não existe aresta conectando dois 0-vértices e y é uma solução viável do $PLU_{k,\alpha}(G)$. Então, existe um algoritmo que produz uma 5-transferência inteira y' de y em tempo polinomial.*

A seguir obtemos uma 6-aproximação para o caso com $\{0, L\}$ capacidades.

Teorema 4.23. *Existe uma 6-aproximação para o problema dos k-centros com capacidades $\{0, L\}$ e tolerância a falhas (com α fazendo parte da entrada).*

Demonstração. Seja y uma solução ótima para o $PLU_{k,\alpha}(G)$, e y' uma 5-transferência inteira de y obtida pelo algoritmo do Lema 4.22. Seja R o conjunto de centros correspondentes ao vetor característico y'. Nós procedemos como na prova do Lema 4.19. Considere um subconjunto $U \subseteq V$. Seja $X(U) = \{v : y_v > 0 \text{ and } v \in (N_G(U))^L\}$ e seja $Y(U) \subseteq R$ o conjunto de centros inteiramente abertos aos quais nós transferimos uma abertura fracionária de $X(U)$.

Pelas restrições de $PLU_{k,\alpha}(G)$ e pelo fato de que y' é uma transferência inteira, nós temos

$$|U| + \alpha L \leq \sum_{u\in X(U)} y_u L \leq \sum_{u\in Y(U)} y'_u L = |Y(U)|L.$$

Agora consideremos o cenário de falha $F \subseteq V$ com $|F| = \alpha$. Nós podemos criar um grafo bipartido (como no Lema 4.19) que conecta cada vértice $u \in V$ aos vértices $Y(\{u\}) \setminus F \subseteq R$.

Usando a condição de Hall, nós obtemos uma atribuição $\phi : V \to R \setminus F$ que respeita as capacidades. Como y' é uma 5-transferência, nós sabemos que $Y(\{u\}) \subseteq N^6(\{u\})$ para todo u e, assim, $d(u, \phi(u)) \leq 6$. □

Referências Bibliográficas

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, 1993. 22, 31, 49

[2] Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. In Jon Lee and Jens Vygen, editors, *Integer Programming and Combinatorial Optimization*, volume 8494 of *Lecture Notes in Computer Science*, pages 52–63. Springer International Publishing, 2014. 2, 27, 33, 37

[3] Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. *Mathematical Programming*, 154(1):29–53, 2015. 45, 49, 53, 57

[4] J. Bar-Ilan, G. Kortsarz, and D. Peleg. How to Allocate Network Centers. *Journal of Algorithms*, 15(3):385–415, 1993. 17

[5] Shiri Chechik and David Peleg. The fault tolerant capacitated k-center problem. In Guy Even and MagnúsM. Halldórsson, editors, *Structural Information and Communication Complexity*, volume 7355 of *Lecture Notes in Computer Science*, pages 13–24. Springer Berlin Heidelberg, 2012. 2, 17, 40, 42, 44

[6] Marek Cygan, MohammadTaghi Hajiaghayi, and Samir Khuller. LP Rounding for k-Centers with Non-uniform Hard Capacities. *arXiv:1208.3054 [cs]*, August 2012. arXiv: 1208.3054. 2, 27

[7] Marek Cygan and Tomasz Kociumaka. Constant Factor Approximation for Capacitated k-Center with Outliers. In *31^{st} International Symposium on Theoretical Aspects of ComputerScience (STACS)*, volume 25, pages 251–262, 2014. 49

[8] Cristina G. Fernandes, Samuel P. de Paula, and Lehilton L. C. Pedrosa. Improved Approximation Algorithms for Capacitated Fault-Tolerant k-Center. In Evangelos Kranakis, Gonzalo Navarro, and Edgar Chávez, editors, *LATIN 2016: Theoretical Informatics*, number 9644 in Lecture Notes in Computer Science, pages 441–453. Springer Berlin Heidelberg, April 2016. DOI: 10.1007/978-3-662-49529-2_33. 2, 43, 53

[9] Michael R. Garey and David S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, 1980. 2, 9

[10] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. 2, 6, 7, 9

[11] Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33(3):533–550, 1986. 2, 12, 13, 19

[12] Samir Khuller, Robert Pless, and Yoram J. Sussmann. Fault tolerant k-center problems. *Theoretical Computer Science*, 242:237–245, 2000. 2, 40, 41, 42

[13] Samir Khuller and Yoram J. Sussmann. The capacitated k-center problem. In *In Proceedings of the 4th Annual European Symposium on Algorithms, Lecture Notes in Computer Science 1136*, pages 152–166. Springer, 1996. 2, 17, 18, 19, 50

[14] S.O. Krumke. On a generalization of the p-Center Problem. *Information Processing Letters*, 56(2):67–71, 1995. 39, 40

[15] Andrew Lim, Brian Rodrigues, Fan Wang, and Zhou Xu. k-Center problems with minimum coverage. *Theoretical Computer Science*, 332(1–3):1 – 17, 2005. 42

[16] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. 3, 11, 12, 27