Aproximação de métricas finitas por métricas arbóreas e aplicações

Murilo Santos de Lima

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

Programa: Ciência da Computação

Orientadora: Profa. Dra. Cristina Gomes Fernandes

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro do CNPq

São Paulo, novembro de 2011

Aproximação de métricas finitas por métricas arbóreas e aplicações

Esta versão definitiva da dissertação
contém as correções e alterações sugeridas pela
Comissão Julgadora durante a defesa realizada
por Murilo Santos de Lima em 15/12/2011.
O original encontra-se disponível no Instituto de
Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Profa. Dra. Cristina Gomes Fernandes (orientadora) - IME-USP
- Prof. Dr. Paulo Feofiloff - IME-USP
- Prof. Dr. Orlando Lee - IC-Unicamp

"You won't realize the distance you've walked until you take a look around and realize how far you've been."
(autor desconhecido)

Agradecimentos

Ao Regente, que conduz de forma sublime esta sinfonia dura mas instigante.

À minha mãe Maria, que sempre enfatizou a importância de princípios e o valor do estudo. Ao meu pai Wilson, à minha irmã Malu e às minhas sobrinhas Júlia e Lara, pelo carinho. Aos meus avós, em especial ao meu avô Euzébio por aquele conselho e por me ensinar a tocar "Buckets of Rain" do Dylan num sonho.

À minha orientadora Cristina Gomes Fernandes, sempre prestativa e compreensiva, e com quem aprendi um monte. (Você é conceito A, pró!) Aos demais professores do DCC-IME-USP com quem tive contato mais próximo, em especial à professora Yoshiko Wakabayashi, por quem guardo enorme admiração.

Aos amigos que me receberam em São Paulo: Robélia Meira, Richard Lopes e João Paulo Dantas.

Aos colegas do IME-USP, em especial a Joel Uchoa, Raphael Ribas e Guilherme Puglia, que considero grandes amigos.

Aos demais amigos que fiz nesse período: Leandro Borges, Eli William Lima, Caio Ravagnani, Alexander Hincapie, Adriana Murcia, Vinícius Ferreira, Henrique Quintanilha, Bruno Caique, Diana Valadares, Laszlo Lueska, Alexandre Rodrigues, Leila Magalhães, Pedrina Arruda, seu Sérgio e Daniel Meirelles. A Rodrigo Damasceno e Clarisse Lyra. Amo todos vocês.

Às pessoas que de alguma forma me apoiaram nesse período: tia Elisa, Kelly, Marlon. A L.F.C. Pinheiro pela ajuda com minhas neuroses.

A Jeff Tweedy, Robin Pecknold, Damon Albarn, Graham Coxon, Gerard Love, Norman Blake, Raymond McGinley, Rivers Cuomo e Thom Yorke, pelas canções maravilhosas. A Johann Sebastian Bach por suas *Variações Goldberg* que auxiliaram na demonstração do Lema 7.4.

Ao CNPq pelo auxílio financeiro. Ao bandejão da USP pela boa comida.

"Our love is all of God's money."
– Jeff Tweedy

Resumo

Aproximação de métricas finitas por métricas arbóreas e aplicações

Muitos problemas de otimização em grafos, em especial problemas métricos, são mais fáceis de resolver em árvores. Portanto, uma estratégia para obter um bom algoritmo para certos problemas é obter uma árvore que aproxime o grafo, e utilizar uma solução do problema nessa árvore como uma solução aproximada para o problema no grafo original. Neste trabalho é estudada a técnica de Fakcharoenphol, Rao e Talwar, que mostraram como aproximar uma métrica finita arbitrária com n pontos por uma métrica numa árvore com distorção esperada $O(\lg n)$ – o ótimo assintótico. Essa estratégia resulta em algoritmos de aproximação com boas razões de aproximação, e em algoritmos com bom fator de competitividade para diversos problemas de otimização online e distribuídos. É apresentada especificamente a aplicação da técnica ao problema do emparelhamento mínimo bipartido online, que ilustra como a aproximação de métricas auxilia na resolução de um problema e os cuidados que devem ser tomados nessa aplicação.

Palavras-chave: aproximação de métricas, métricas arbóreas, emparelhamento mínimo bipartido online.

Abstract

Approximation of finite metrics by tree metrics and applications

Many optimization problems on graphs, especially metric problems, are easier to solve on trees. Therefore, a strategy for obtaining a good algorithm for certain problems is to obtain a tree that approximates the graph, and use a solution of the problem on the tree as an approximate solution for the problem on the original graph. We study the work of Fakcharoenphol, Rao e Talwar, who showed how to approximate an arbitrary finite metric on n points by a tree metric with expected distortion $O(\lg n)$, which is asymptotically optimum. This strategy leads to algorithms with good approximation factors, and to competitive algorithms for various optimization problems, some of them online and distributed. Here, we present the application of that technique to the problem of finding a minimum online matching on a bipartite metric graph. This problem illustrates how metric approximation aids in solving a problem, and the care that must be taken when doing such an application.

Keywords: metric approximation, tree metrics, online minimum bipartite matching.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Muitos problemas de otimização em grafos, em especial problemas métricos, são mais fáceis de resolver em árvores. Portanto, uma estratégia para obter um bom algoritmo para certos problemas é obter uma árvore que aproxime o grafo, e utilizar a solução do problema nessa árvore como uma solução aproximada para o problema no grafo original.

Neste trabalho é descrito um algoritmo que aproxima uma métrica finita, ou as distâncias num grafo, pelas distâncias numa árvore. O objetivo é que as distâncias na árvore tenham uma relação com a métrica original, de forma que garantias sobre uma solução na árvore possam ser estendidas para garantias em relação à métrica inicial. Especificamente, a distância entre um par de vértices na árvore deve ser maior ou igual à distância na métrica original, mas se deseja limitar superiormente esse aumento na distância, denominado *distorção*. Esse algoritmo é utilizado como sub-rotina em algoritmos de aproximação para diversos problemas de otimização, bem como em algoritmos com bom fator de competitividade para diversos problemas online e distribuídos.

Mais adiante, é estudada neste trabalho a aplicação desse algoritmo em um problema de computação online, o problema do emparelhamento mínimo bipartido online. Esse problema foi escolhido por demonstrar claramente como a aproximação de métricas auxilia na resolução de um problema, e os cuidados que devem ser tomados nessa aplicação. Aplicações desse problema incluem problemas de balanceamento de carga e leilões de anúncios em sítios de busca na Web.

1.1 Histórico

As primeiras ideias no sentido de aproximar uma métrica por outra mais simples surgiram quando Karp tentou resolver o problema dos k servidores (*k-server problem*) no n-circuito (um circuito com n vértices), aproximando-o por caminhos [Kar89]. No entanto, qualquer aproximação determinística do n-circuito tem distorção $\Omega(n)$ [RR98], o que leva a um algoritmo cujo fator de competitividade é pelo menos kn. Karp então propôs que se aproximasse o n-circuito por uma distribuição de probabilidade sobre caminhos. Essa estratégia leva a uma distorção esperada de 2, de forma que Karp obteve um algoritmo probabilístico $2k$-competitivo para os k servidores no n-circuito. Alon *et al.* [AKPW95] adaptaram a ideia para aproximar uma métrica num grafo arbitrário por uma distribuição de probabilidade sobre árvores geradoras do grafo, mas obtiveram um limite superior de apenas $2^{O(\sqrt{\lg n \lg\lg n})}$ na distorção.

Bartal [Bar96] propôs aproximar uma métrica arbitrária através de uma distribuição de probabilidade sobre métricas arbóreas. Uma **métrica arbórea** é uma métrica induzida pelas distâncias

numa árvore, não necessariamente contida no grafo original, mas cujo conjunto de vértices contém o conjunto de vértices do grafo original. Especificamente, Bartal propôs um algoritmo que produz uma métrica arbórea, escolhida dentre uma família de métricas arbóreas segundo uma distribuição de probabilidade, de forma que a distorção esperada seja baixa. Bartal mostrou como obter uma métrica com distorção esperada $O(\lg^2 n)$, utilizando o conceito de árvores hierarquicamente bem-separadas (HSTs, do inglês *hierarchically well-separated trees*). O resultado foi melhorado pelo próprio Bartal para $O(\lg n \lg\lg n)$ [Bar98], e para $O(\lg n)$ por Fakcharoenphol, Rao e Talwar [FRT04b]. Tal resultado corresponde ao ótimo assintótico: Bartal [Bar96] havia mostrado que existem grafos para os quais qualquer aproximação probabilística por uma árvore tem distorção $\Omega(\lg n)$. Para uma descrição mais completa do histórico do problema, consulte [FRT04b, FRT04a].

Este trabalho se concentra no resultado de Fakcharoenphol, Rao e Talwar [FRT04b, FRT04a], que é apresentado no Capítulo 4.

1.2 Aplicações

A aproximação de métricas por métricas em árvores tem diversas aplicações, em especial em algoritmos de aproximação e nas versões online e distribuída de diversos problemas de otimização. A seguir, são citadas as referências encontradas de aplicações dessa técnica, para ajudar o leitor interessado.

Alguns problemas são enumerados em [Bar96, Bar98, FRT04b], destacando-se dentre eles o problema da árvore de Steiner de grupos [GKR00, AAA+06] e generalizada [AAB96, BC97]; da k-mediana [KH79], para o qual foi obtida a primeira aproximação sublinear, embora hoje seja conhecida uma 3,45-aproximação [Shm99]; da rotulação de métricas [KT02, KKMR09]; do projeto de redes de compra em atacado (*buy-at-bulk newtork design*) [AA97, HST05, CHKS06]; dos k servidores nas versões online [Kou09] e distribuída [BR97]; e da paginação distribuída [BFR95, Bar96].

Aplicações mais recentes incluem o problema da árvore de Steiner pré-projetada [IKMM04]; *caching* cooperativo hierárquico online [LPTV06]; emparelhamento mínimo bipartido online [MNP06, CP08]; ordenação de *buffers* em métricas na reta [KP06], embora uma aproximação determinística melhor seja conhecida [GS09]; corte em grafos com requisitos, que unifica diversos problemas de cortes em grafos [NR10]; variantes da k-árvore de Steiner [HKS09]; além de diversas variantes de problemas de projeto de redes [GKR09, GOS09, BH09, vZ09].

Neste trabalho é descrita em detalhes a aplicação dessa técnica ao problema do emparelhamento mínimo bipartido online, na Parte II (Capítulos 5 a 7).

1.3 Organização da Dissertação

No Capítulo 2 são apresentados alguns conceitos necessários para o restante do texto: teoria dos grafos, probabilidade, problemas de otimização, algoritmos de aproximação, algoritmos online e análise competitiva. Ali também é descrito o modelo de computação utilizado neste trabalho.

A Parte I (Capítulos 3 e 4) trata de aproximação de métricas. No Capítulo 3 são definidos os conceitos de métrica e aproximação de métricas, e são apresentados alguns resultados relacionados à aproximação de métricas por métricas arbóreas. É descrito, no Capítulo 4, o algoritmo de Fakcharoenphol, Rao e Talwar [FRT04b, FRT04a], que aproxima uma métrica finita arbitrária por uma

métrica arbórea com distorção logarítmica.

A Parte II (Capítulos 5 a 7) é dedicada ao problema do emparelhamento mínimo bipartido online. O Capítulo 5 apresenta o problema, juntamente com limitantes inferiores no fator de competitividade de qualquer algoritmo. No Capítulo 6 são apresentados algoritmos determinísticos; no Capítulo 7 é apresentado um algoritmo probabilístico com melhor fator de competitividade, que se baseia fortemente na aproximação de métricas por métricas arbóreas.

Por fim, no Capítulo 8 são feitas algumas considerações finais.

Note que são apresentadas duas versões da bibliografia: uma denominada "Bibliografia Selecionada", contendo apenas as publicações mais importantes para o desenvolvimento do texto, e outra contendo tudo que é referenciado.

Capítulo 2

Preliminares

Neste capítulo são apresentados alguns conceitos básicos, necessários para o entendimento das técnicas utilizadas nos demais capítulos. Este capítulo é baseado nos Capítulos 1 e 6, na Seção 2.1 e nos Apêndices D e E de de Carvalho *et al.* [dCCD+01], e na Seção 1.1 e no Capítulo 7 de Borodin e El-Yaniv [BEY98].

Sejam um conjunto A e uma função $f : A \to \mathbb{R}$. Dado um subconjunto finito B de A, denota-se por $f(B)$ a somatória $\sum_{b \in B} f(b)$. Diz-se que um número x é **positivo** se $x > 0$, e que x é **não-negativo** se $x \geq 0$.

2.1 Teoria dos Grafos

Um **grafo** é um par (V, E), onde V é um conjunto finito e E é um conjunto de pares de elementos de V. O conjunto V é denominado o conjunto dos **vértices** do grafo, e o conjunto E é denominado o conjunto das **arestas** do grafo. Para não carregar a notação, denota-se uma aresta $\{u, v\}$ por uv. A fim de evitar ambiguidades quando se referindo a mais de um grafo $G = (V, E)$, é usual denotar V por V_G e E por E_G.

Um grafo G pode ser representado graficamente utilizando-se um ponto para cada vértice de V_G e uma linha entre dois vértices u e v se a aresta uv está em E_G. Na Figura 2.1, por exemplo, é mostrada uma representação gráfica do grafo $(\{a, b, c, d\}, \{ab, ac, bc, cd\})$.

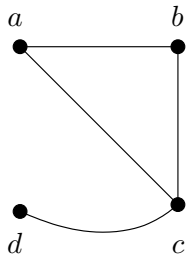


Figura 2.1: *Uma possível representação gráfica do grafo* $(\{a, b, c, d\}, \{ab, ac, bc, cd\})$.

Grafos são úteis para representar relacionamentos entre objetos. Por exemplo, uma rede de computadores pode ser representada como um grafo, em que cada computador é representado por um vértice, e existe uma aresta entre dois vértices se existir um canal de comunicação direto entre os computadores correspondentes. É comum também associar um valor numérico aos vértices e/ou às arestas de um grafo. No exemplo da rede, pode-se associar a uma aresta entre dois vértices o

tempo de transmissão de uma mensagem entre os computadores correspondentes.

Sejam u e v vértices distintos de G. Se $uv \in E_G$, então u e v são denominados **adjacentes**, e se diz que existe uma aresta **entre** u e v. Se $e = uv \in G$, então se diz que e é **incidente** a u e a v, que u e v são **incidentes** a e e que u e v são as **pontas** de e. O **grau** de um vértice $v \in V_G$ consiste no número de arestas incidentes a v em G. O **grau mínimo** de um grafo é o mínimo dentre os graus de seus vértices. Um grafo G é **completo** se há uma aresta unindo cada par de vértices de G. Denota-se por K_n o grafo completo com n vértices. O grafo K_1 é denominado grafo **trivial**, e o grafo K_0 é denominado grafo **vazio**.

Um **subgrafo** de G é qualquer grafo H tal que $V_H \subseteq V_G$ e $E_H \subseteq E_G$. Nesse caso, diz-se que G **contém** H. Seja $U \subseteq V_G$. O subgrafo de G **induzido** por U, denotado por $G[U]$, é o grafo $(U, \{uv \in E_G : u, v \in U\})$.

Um grafo G é denominado **bipartido** (*bipartite*) se V_G pode ser dividido em duas partes disjuntas U e V, tais que toda aresta de G é do tipo uv com $u \in U$ e $v \in V$. Pode-se dizer, nesse caso, que G é (U, V)-bipartido. Por exemplo, o grafo $G = (\{a, b, c, d, e, f\}, \{ad, ae, bd, bf, ce\})$, representado na Figura 2.2, é bipartido: tomando os conjuntos $U = \{a, b, c\}$ e $V = \{d, e, f\}$, representados na figura pela linha pontilhada, toda aresta de G tem uma ponta em U e outra em V. Um grafo G (U, V)-bipartido é **bipartido completo** se existe uma aresta entre cada vértice de U e cada vértice de V. Denota-se por $K_{m,n}$ o grafo bipartido completo em que uma das partes da bipartição tem m vértices e a outra tem n vértices.

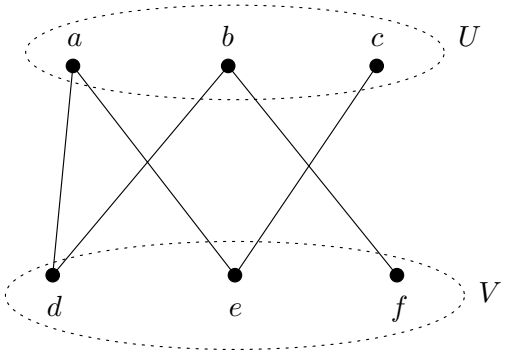


Figura 2.2: *Exemplo de grafo bipartido. Toda aresta tem uma ponta no conjunto U e outra no conjunto V.*

Um **emparelhamento** (*matching*) de um grafo G é um subconjunto M de E_G tal que, para todo vértice v de G, no máximo uma aresta incidente a v está em M. Um emparelhamento M de um grafo G é denominado **perfeito** se, para cada vértice v de G, exatamente uma aresta incidente a v está em M. Como exemplo, um emparelhamento do grafo da Figura 2.2 é o conjunto $\{ad, bf, ce\}$ (ver Figura 2.3); note que se trata de um emparelhamento perfeito.

O **problema do emparelhamento perfeito de custo mínimo** consiste em:

> **Problema** MinEP (G, c): *Dados um grafo G que possui um emparelhamento perfeito e um custo c_e não-negativo associado a cada aresta e de G, encontrar um emparelhamento perfeito M de G tal que a soma dos custos das arestas em M seja mínima.*

Esse problema pode ser resolvido em tempo $O(n^3)$, onde $n = |V_G|$, utilizando o algoritmo húngaro para grafos bipartidos [Kuh55, Law00] ou o algoritmo de Edmonds para grafos arbitrários [Edm65, CR99]. (Embora ambos algoritmos tenham a mesma complexidade assintótica, o algoritmo húngaro é mais simples e induz implementações mais eficientes, sendo preferido quando o grafo de interesse

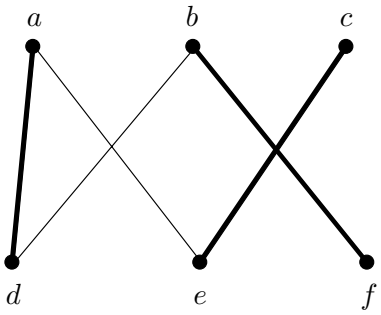


Figura 2.3: *As arestas mais escuras formam um emparelhamento perfeito do grafo da Figura 2.2.*

é bipartido.) Uma variante desse problema é estudada na Parte II.

Um **caminho** é um grafo G para o qual existe uma permutação $(v_1, v_2, \ldots, v_n)$ de V_G tal que as arestas de G são exatamente $v_{i-1}v_i$, para $i = 2, \ldots, n$. Se G é um subgrafo de um grafo H, então G é um caminho (**de** v_1 **para** v_n ou **entre** v_1 e v_n) de H. O **comprimento** de um caminho é seu número de arestas. Um exemplo é mostrado na Figura 2.4(a).

Um grafo G é **conexo** se, para todo par u, v de vértices de G, existe um caminho de u para v em G. Um grafo que não é conexo é **desconexo**. Um **componente** C de um grafo G é um subgrafo conexo maximal de G; isto é, não existe subgrafo conexo de G que contenha C propriamente.

Um **circuito** é um grafo G com ao menos 3 vértices para o qual existe uma permutação $(v_1, v_2, \ldots, v_n)$ de V_G tal que as arestas de G são exatamente v_1v_n e $v_{i-1}v_i$, para $i = 2, \ldots, n$. Se G é um subgrafo de um grafo H, então G é um circuito de H. O **comprimento** de um circuito é seu número de arestas. Um exemplo é mostrado na Figura 2.4(b). A **cintura** (*girth*) de um grafo G que contém ao menos um circuito, denotada por $g(G)$, é o comprimento de um menor circuito de G.

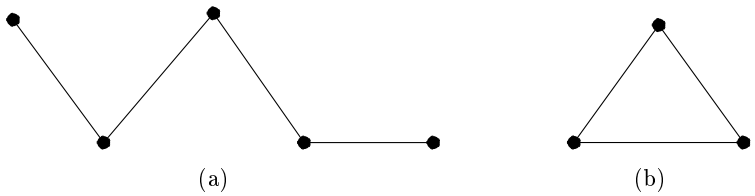


Figura 2.4: *(a) Um caminho de comprimento 4. (b) Um circuito de comprimento 3.*

O seguinte fato será usado adiante.

Fato 2.1: *Todo grafo com n vértices e ao menos n arestas tem um circuito.*

O teorema a seguir, com uma pequena variação no enunciado, é apresentado em [Bol78, p. 104].

Teorema 2.2: *Sejam $g \geq 3$, $\delta \geq 3$ e $n := (11\delta)^g$. Existe um grafo G com n vértices, cintura pelo menos g e pelo menos $(\delta - 1)n$ arestas.*

Demonstração: Considere a classe $\mathcal{G}$ de todos os grafos com conjunto de vértices $\{1, \ldots, n\}$ e δn arestas. Note que todo grafo de $\mathcal{G}$ tem um circuito, logo sua cintura está definida. Observe também que

$$|\mathcal{G}| = \binom{\binom{n}{2}}{\delta n} :$$

das $\binom{n}{2}$ arestas que o grafo pode ter, tome δn. Para $\ell = 3, \ldots, n$, o número de circuitos de compri-

mento ℓ que podem ser formados com n vértices é

$$\frac{1}{2}(\ell-1)!\binom{n}{\ell}:$$

escolhidos ℓ dos n vértices ($\binom{n}{\ell}$ possibilidades), existem $\ell!$ permutações desses vértices. Se for aplicada uma rotação e/ou uma inversão (2ℓ possibilidades) a uma dessas permutações, e somente nesses casos, o circuito obtido é o mesmo, logo $\frac{1}{2\ell}$ dessas permutações definem um circuito distinto. Note que

$$\frac{1}{2}(\ell-1)!\binom{n}{\ell} = \frac{1}{2}(\ell-1)!\frac{n!}{\ell!(n-\ell)!} = \frac{1}{2\ell}\cdot\frac{n!}{(n-\ell)!} = \frac{1}{2\ell}(n(n-1)\cdots(n-\ell+1)) < \frac{1}{2\ell}n^\ell.$$

Por argumento semelhante, fixado um circuito de comprimento ℓ, o número de grafos em $\mathcal{G}$ com esse circuito é no máximo

$$\binom{\binom{n}{2}-\ell}{\delta n-\ell}.$$

Então, o número total de circuitos de comprimento ℓ em $\mathcal{G}$, contando repetições, é menor que

$$\frac{1}{2\ell}n^\ell\binom{\binom{n}{2}-\ell}{\delta n-\ell},$$

logo a média aritmética do número de circuitos de comprimento ℓ nos grafos em $\mathcal{G}$ é menor que

$$\frac{1}{2\ell}n^\ell\binom{\binom{n}{2}-\ell}{\delta n-\ell}\binom{\binom{n}{2}}{\delta n}^{-1}.$$

Assim, a média aritmética do número de circuitos de comprimento menor que g nos grafos em $\mathcal{G}$ é menor que

$$\sum_{\ell=3}^{g-1}\frac{1}{2\ell}n^\ell\binom{\binom{n}{2}-\ell}{\delta n-\ell}\binom{\binom{n}{2}}{\delta n}^{-1},$$

que é menor que n:

$$\begin{aligned}
\sum_{\ell=3}^{g-1}\frac{1}{2\ell}n^\ell\binom{\binom{n}{2}-\ell}{\delta n-\ell}\binom{\binom{n}{2}}{\delta n}^{-1} &= \sum_{\ell=3}^{g-1}\frac{1}{2\ell}n^\ell\binom{\delta n}{\ell}\binom{\binom{n}{2}}{\ell}^{-1} && (2.1)\\
&\le \sum_{\ell=3}^{g-1}\frac{1}{2\ell}n^\ell\left(\frac{\delta n}{\ell}\right)^\ell e^\ell\left(\frac{\binom{n}{2}}{\ell}\right)^{-\ell} && (2.2)\\
&\le \sum_{\ell=3}^{g-1}\frac{(e\delta n^2)^\ell}{2\ell}\left(\frac{n^2}{4}\right)^{-\ell} && (2.3)\\
&= \sum_{\ell=3}^{g-1}\frac{(4e\delta)^\ell}{2\ell}\\
&< \sum_{\ell=3}^{g-1}(11\delta)^\ell\\
&= \frac{(11\delta)^g-(11\delta)^3}{11\delta-1}\\
&< n.
\end{aligned}$$

A igualdade (2.1) segue do fato de que, para inteiros $i \leq b \leq a$,

$$\frac{\binom{a-i}{b-i}}{\binom{a}{b}} = \frac{\binom{b}{i}}{\binom{a}{i}}.$$

Já a desigualdade (2.2) segue de

$$\binom{a}{b} \geq \left(\frac{a}{b}\right)^b$$

e de [CLRS01, Equação C.5]

$$\binom{a}{b} \leq \left(\frac{a}{b}\right)^b e^b.$$

A desigualdade (2.3) segue pois $n \geq 2$ e

$$\binom{n}{2} \geq \frac{n^2}{4}.$$

Então, pelo fato de toda população possuir um elemento cujo valor é menor ou igual ao valor da média aritmética, existe um grafo G' em $\mathcal{G}$ com menos que n circuitos de comprimento menor que g. Seja G o grafo obtido de G' pela remoção de uma aresta de cada tal circuito. Então G tem mais que $\delta n - n = (\delta - 1)n$ arestas e cintura pelo menos g, como queríamos demonstrar. □

Esse teorema tem o seguinte corolário, que será utilizado mais adiante.

Corolário 2.3: *Para todo $k \geq 33^3$, existe um grafo conexo com $n \geq k$ vértices, cintura pelo menos $\log_{33} n$ e mais que $2n$ arestas.*

Demonstração: Sejam $n := 33^{\lceil \log_{33} k \rceil}$, $\delta := 3$ e $g := \log_{33} n$; tem-se que $g \geq 3$. Seja G um grafo dado pelo Teorema 2.2, que possui $(11\delta)^g = 33^g = n$ vértices, cintura pelo menos g e mais que $(\delta - 1)n = 2n$ arestas. Caso G seja desconexo, sejam $\{C_1, \ldots, C_c\}$ seus componentes e v_i um vértice arbitrário de C_i, para $i = 1, \ldots, c$. Adicione a aresta $v_1 v_j$, para $j = 2, \ldots, c$; note que isso não cria novos circuitos. O grafo resultante tem as propriedades do enunciado. □

Um grafo sem circuitos é denominado **acíclico** ou uma **floresta**. Um grafo conexo e acíclico é denominado uma **árvore**. Numa árvore, todo vértice de grau 1 é denominado **folha**.

Fato 2.4: *Numa árvore T com n vértices, valem as seguintes propriedades:*

(1) existe exatamente um caminho entre cada par de vértices;

(2) T tem exatamente $n - 1$ arestas;

(3) se $n \geq 3$ e for adicionada uma aresta a T, o grafo resultante possui um circuito;

(4) se $n \geq 2$ e for removida uma aresta de T, o grafo resultante é desconexo;

(5) se $n \geq 2$, T tem ao menos duas folhas.

Note que todo componente de uma floresta é uma árvore, e portanto, uma floresta tem no máximo $n - 1$ arestas. Observe também que o Fato 2.1 segue dos itens (2) e (3).

Uma árvore T é denominada **enraizada** se é feita distinção de um vértice r de T. Nesse caso, r é denominado a **raiz** de T. Sejam u e v dois vértices adjacentes em T. Se u precede v no caminho de r até alguma folha, então u é denominado o **pai** de v, e v um **filho** de u. O maior comprimento

de um caminho entre r e uma folha de T é denominado a **altura** de T. Na Figura 2.5 é mostrada uma árvore enraizada de altura 2.

Seja T uma árvore enraizada com raiz r e altura h. Defina-se então o **nível** de um vértice v de T como o valor h menos o comprimento do caminho de v até r. Isto é, a raiz tem nível h e os filhos de um vértice no nível i têm nível $i-1$. Uma aresta uv tem nível igual ao máximo dentre o nível de u e o nível de v. Na Figura 2.5, o vértice v tem nível 1 e a aresta rv tem nível 2.

Definição 2.5 (HST): *Uma* **árvore k-hierarquicamente bem-separada** *(k-HST, do inglês k-*hierarchically well-separated tree*) é uma árvore enraizada em que a cada aresta é associado um custo real positivo com a seguinte restrição: arestas de mesmo nível têm o mesmo custo, e dado um vértice v que não seja raiz nem folha, a razão entre o custo da aresta de v ao pai de v e o custo de uma aresta de v a um filho de v é pelo menos k.*

Eventualmente, um vértice interno pode ter um único filho. A árvore da Figura 2.5, por exemplo, é uma 3-HST. O conceito de HST facilita o projeto de algoritmos simples, em especial algoritmos de divisão e conquista, para diversos problemas [Bar98, FRT04b, KT02, BBBT97].

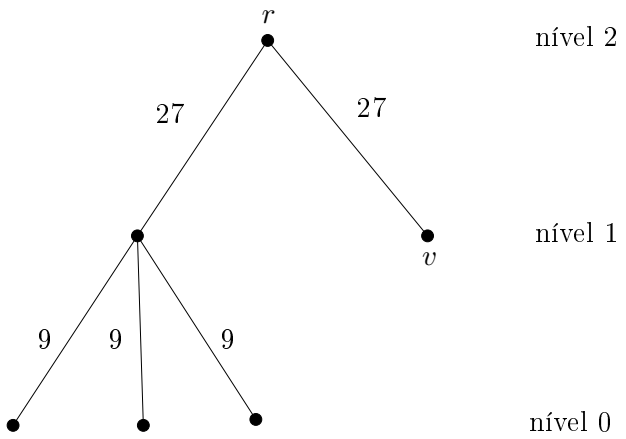


Figura 2.5: *Uma árvore enraizada de altura 2 e raiz r. O vértice v tem nível 1 e a aresta rv tem nível 2. Note que esta é uma 3-HST.*

Definição 2.6 (HST perfeita): *Uma k-HST é* **perfeita** *se a razão entre o custo de arestas consecutivas num caminho da raiz até uma folha é exatamente k.*

Definição 2.7 (HST de base t): *Uma HST tem* **base** *t se todas as folhas têm o mesmo nível, e as arestas de nível mais baixo têm custo t.*

A árvore da Figura 2.5, por exemplo, é uma 3-HST perfeita, mas não tem base definida.

2.2 Probabilidade

Uma **distribuição de probabilidade discreta** sobre um conjunto finito não-vazio Ω é uma função $\mathcal{D} : \Omega \to [0,1]$ tal que $\sum_{\omega\in\Omega} \mathcal{D}(\omega) = 1$. Um **espaço probabilístico discreto** é um par $(\Omega, \mathcal{D})$, onde Ω é um conjunto finito e $\mathcal{D}$ é uma distribuição de probabilidade discreta sobre Ω.

Exemplo 2.8 (Distribuição uniforme discreta): *A distribuição uniforme discreta sobre um conjunto finito Ω qualquer consiste na função $\mathcal{D}(\omega) := 1/|\Omega|$, para todo $\omega \in \Omega$.*

Uma **variável aleatória** sobre Ω é uma função $X : \Omega \to \{\lambda_1, \ldots, \lambda_n\}$, onde λ_i é um número real para $i = 1, \ldots, n$. Um **evento** é o conjunto $X^{-1}(S)$, para algum $S \subseteq \{\lambda_1, \ldots, \lambda_n\}$. O evento $X^{-1}(S)$ é denotado por $[X \in S]$ ou, se $S = \{x\}$, por $[X = x]$. As notações $[X \neq x]$, $[X > x]$ e $[X < x]$ são definidas de maneira análoga.

A probabilidade do evento $[X \in S]$ é o número $\mathcal{D}(X^{-1}(S))$, denotado por $\mathbb{P}[X \in S]$. A **esperança** ou **valor esperado** da variável aleatória X é o número

$$\mathbb{E}[X] := \sum_{i=1}^{n} \lambda_i \cdot \mathbb{P}[X = \lambda_i].$$

Fato 2.9 (Linearidade da Esperança): *Sejam X e Y variáveis aleatórias e α um número real. Então,*

$$\mathbb{E}[X + \alpha Y] = \mathbb{E}[X] + \alpha \cdot \mathbb{E}[Y].$$

Fato 2.10 (Desigualdade de Markov): *Seja $(\Omega, \mathcal{D})$ um espaço probabilístico discreto e X uma variável aleatória sobre Ω que assume apenas valores não-negativos. Então, para todo $\lambda > 0$, vale que*

$$\mathbb{P}[X \geq \lambda] \leq \frac{1}{\lambda} \cdot \mathbb{E}[X].$$

Demonstração: Seja $X : \Omega \to \{\lambda_1, \ldots, \lambda_n\}$. Então,

$$\mathbb{E}[X] = \sum_{i=1}^{n} \lambda_i \cdot \mathbb{P}[X = \lambda_i] \geq \sum_{\lambda_i \geq \lambda} \lambda_i \cdot \mathbb{P}[X = \lambda_i] \geq \sum_{\lambda_i \geq \lambda} \lambda \cdot \mathbb{P}[X = \lambda_i] = \lambda \cdot \mathbb{P}[X \geq \lambda].$$

□

Uma das distribuições de probabilidade utilizadas neste trabalho não é discreta, mas contínua. Para definir distribuições de probabilidade contínuas, são necessários os seguintes conceitos. Uma **σ-álgebra** de um conjunto Ω é uma coleção $\mathcal{F}$ de subconjuntos de Ω tal que:

- $\emptyset \in \mathcal{F}$;
- para todo $A \in \mathcal{F}$, vale que $\Omega \setminus A \in \mathcal{F}$;
- para toda família $A_1, A_2, \ldots$ de conjuntos em $\mathcal{F}$, vale que $\bigcup_i A_i \in \mathcal{F}$.

Uma função $\mathcal{D} : \mathcal{F} \to [0, 1]$ é **$\mathcal{F}$-aditiva** se, para toda família $A_1, A_2, \ldots$ de conjuntos dois a dois disjuntos em $\mathcal{F}$, vale que

$$\mathcal{D}(\bigcup_i A_i) = \sum_i \mathcal{D}(A_i).$$

Uma tal função $\mathcal{D}$ é uma **distribuição de probabilidade contínua** sobre $(\Omega, \mathcal{F})$ se $\mathcal{D}$ é $\mathcal{F}$-aditiva e $\mathcal{D}(\Omega) = 1$. Um **espaço probabilístico contínuo** é uma tripla $(\Omega, \mathcal{F}, \mathcal{D})$ tal que $\mathcal{F}$ é uma σ-álgebra de Ω e $\mathcal{D}$ é uma distribuição de probabilidade contínua sobre $(\Omega, \mathcal{F})$.

Seja $A \subseteq \Omega$. A **função característica** de A, denotada por f_A, é a função $f_A : \Omega \to \{0, 1\}$ tal que $f_A(\omega) = 1$ se $\omega \in A$ e $f_A(\omega) = 0$ caso contrário, para todo $\omega \in \Omega$.

Exemplo 2.11 (Distribuição uniforme contínua): *Para $\Omega \subseteq \mathbb{R}$ e uma σ-álgebra $\mathcal{F}$ de Ω, a distribuição uniforme contínua é a distribuição de probabilidade contínua $\mathcal{D}$ sobre $(\Omega, \mathcal{F})$ tal que*

$$\mathcal{D}(A) := \frac{\int_\Omega f_A(x)dx}{\int_\Omega f_\Omega(x)dx}, \quad \text{para todo } A \in \mathcal{F}.$$

Exemplo 2.12: *Seja $A = [a, b) \subseteq \mathbb{R}$ e $\mathcal{F}$ a σ-álgebra composta pelas uniões enumeráveis de subintervalos de A. Então a distribuição uniforme contínua sobre $(A, \mathcal{F})$ é a função*

$$\mathcal{D}(\bigcup_{i=1}^{n}[a_i, b_i)) := \frac{\sum_{i=1}^{n}(b_i - a_i)}{b - a},$$

onde $[a_i, b_i) \subseteq A$ para $i = 1, \ldots, n$ e $a_{i+1} \geq b_i$ para $i < n$.

Neste texto, com exceção da distribuição contínua uniforme (usada no Capítulo 4), só serão consideradas distribuições de probabilidades e espaços de probabilidade discretos. Assim sendo, a denominação "discreto(a)" será omitida no restante do texto.

2.3 Modelo de Computação

Neste trabalho é utilizado o modelo de computação RAM real (*real-RAM*) [PS85]. Nesse modelo, é possível manipular números reais arbitrários, e cada operação envolvendo números reais consome apenas uma unidade de tempo.

Adicionalmente, supõe-se que os algoritmos têm acesso a um gerador de números reais aleatórios, isto é, um algoritmo RandUni que em tempo constante devolve um real aleatório com distribuição uniforme no intervalo $[0, 1)$. Implementações aproximadas desse algoritmo podem ser vistas em Knuth [Knu98]. O algoritmo RandUni pode ser facilmente modificado em tempo constante para sortear, com distribuição uniforme, um real aleatório num intervalo limitado qualquer ou um elemento aleatório de um conjunto finito qualquer [Rob95], ou ainda, em tempo linear no tamanho do conjunto, uma permutação uniformemente aleatória de um conjunto finito [CLRS01, Seção 5.3]. Um algoritmo que utiliza RandUni é denominado um **algoritmo probabilístico** ou **aleatorizado** (*randomized algorithm*).

2.4 Problemas de Otimização e Algoritmos de Aproximação

Um **problema de otimização** consiste num conjunto $\mathcal{I}$ de **instâncias** e, para cada instância I em $\mathcal{I}$, num conjunto $\mathrm{Sol}(I)$ de **soluções viáveis** para I e numa função que atribui um valor $\mathrm{val}(S, I)$ para cada solução S em $\mathrm{Sol}(I)$. Uma instância é denominada **viável** se possui alguma solução viável. Num **problema de minimização**, deseja-se obter soluções viáveis de valor mínimo; num **problema de maximização**, soluções viáveis de valor máximo. Em ambos os casos, utiliza-se os termos "valor ótimo", "solução ótima" e "problema de otimização" no lugar de valor mínimo (máximo), solução de valor mínimo (máximo) e problema de minimização (maximização). O valor de qualquer solução ótima para uma instância viável I é denotado por $\mathrm{opt}(I)$, ou simplesmente por opt se a instância I estiver implícita. Neste trabalho, o foco é em problemas de minimização; nesse caso, é comum denominar o valor $\mathrm{val}(S, I)$ de uma solução S de I como o **custo** de S.

Seja A um algoritmo que, para um problema de minimização $\mathcal{P}$ com soluções de custo não-negativo, devolve uma solução viável $A(I)$ para cada instância viável I de $\mathcal{P}$. Dados $\alpha : \mathcal{I} \to \mathbb{R}$ e um real $d \geq 0$, diz-se que A é uma **α-aproximação assintótica** para $\mathcal{P}$ se, para toda instância viável I, vale que

$$\text{val}(A(I), I) \leq \alpha(I) \cdot \text{opt}(I) + d.$$

Se $d = 0$, então A é uma **α-aproximação** para $\mathcal{P}$. Diz-se, então, que A é um **algoritmo de aproximação** para $\mathcal{P}$. O ínfimo (sobre todas as instâncias) dos números α que satisfazem essa desigualdade é denominado **fator de aproximação** de A e, claramente, $\alpha \geq 1$. Conceitos análogos podem ser definidos para problemas de maximização.

Seja A um algoritmo probabilístico que, para um problema de minimização $\mathcal{P}$ com soluções de custo não-negativo, devolve uma solução viável $A(I)$ para cada instância viável I de $\mathcal{P}$. Note que, neste caso, $\text{val}(A(I), I)$ é uma variável aleatória, com espaço de probabilidade definido pelos sorteios realizados por A. Dados $\alpha : \mathcal{I} \to \mathbb{R}$ e um real $d \geq 0$, diz-se que A é uma **α-aproximação probabilística assintótica** para $\mathcal{P}$ se, para toda instância viável I, vale que

$$\mathbb{E}[\text{val}(A(I), I)] \leq \alpha(I) \cdot \text{opt}(I) + d.$$

Se $d = 0$, então A é uma **α-aproximação probabilística** para $\mathcal{P}$. Diz-se, então, que A é um **algoritmo de aproximação probabilístico** para $\mathcal{P}$. O ínfimo (sobre todas as instâncias) dos números α que satisfazem essa desigualdade é denominado **fator de aproximação esperado** de A. Conceitos análogos podem ser definidos para problemas de maximização.

Algoritmos de aproximação são de especial interesse em problemas de otimização NP-difíceis, para os quais não existe algoritmo polinomial que garanta obter uma solução ótima, a menos que P = NP. Para tais problemas, um algoritmo de aproximação polinomial garante devolver uma solução com custo limitado por um fator do valor ótimo, o que pode ser suficiente numa aplicação prática.

2.5 Algoritmos Online e Análise Competitiva

Informalmente, um problema de otimização é denominado **online** se a seguinte restrição adicional for satisfeita. Cada instância é vista como uma sequência de requisições, e cada requisição deve ser "atendida" sem conhecimento das requisições futuras. Um algoritmo para um problema de otimização online é um **algoritmo online**.

Como exemplo, considere o problema do escalonamento (*scheduling*) em máquinas idênticas. São dadas m máquinas e n tarefas; cada tarefa $1 \leq k \leq n$ tem um tempo de duração $t(k)$. O objetivo é encontrar uma distribuição das tarefas entre as máquinas de forma a minimizar o tempo máximo que uma máquina trabalha. Formalmente, o problema é descrito da seguinte forma.

Problema Escalonamento (m, t): *Dados um inteiro $m > 0$ e uma sequência $(t(1), \ldots, t(n))$ de números positivos, encontrar uma sequência $(s(1), \ldots, s(n))$, onde $s(k) \in \{1, \ldots, m\}$ para $1 \leq k \leq n$, de forma a minimizar* $\max_{1 \leq j \leq m} \sum_{k:s(k)=j} t(k)$.

Uma versão online desse problema consiste no seguinte. Uma instância é uma sequência $(I_0, I_1, I_2, \ldots)$ de instâncias de Escalonamento com um número m de máquinas fixo. Para $i = 0, 1, 2, \ldots$, tem-se $I_i = (m, (t(1), \ldots, t(i)))$. Um algoritmo online começa com uma solução $S_0 = ()$ para I_0 e, dada

uma solução $S_i = (s(1), \ldots, s(i))$ para I_i, escolhe $s(i+1)$ tal que $S_{i+1} = (s(1), \ldots, s(i), s(i+1))$ é uma solução para I_{i+1}. Isto é, a cada instante, chega uma nova tarefa e o algoritmo deve decidir em qual máquina escaloná-la, sem modificar o escalonamento feito para as tarefas anteriores.

As duas principais maneiras de analisar um algoritmo online são análise de caso médio e análise competitiva. Numa análise de caso médio, considera-se uma distribuição de probabilidade nas sequências de entrada e calcula-se o custo esperado da solução produzida pelo algoritmo, dada uma entrada escolhida de acordo com tal distribuição de probabilidade. Numa análise competitiva, o objetivo é estabelecer uma garantia de qualidade de um algoritmo online através da comparação com o **custo ótimo offline**, que é o custo de uma solução devolvida por um algoritmo hipotético que tem conhecimento da sequência de requisições como um todo. O foco deste texto é na análise competitiva, que é formalizada pelas seguintes definições.

Dada uma instância I de um problema de otimização online $\mathcal{P}$, denote por $\mathrm{opt}(I)$ o custo ótimo offline de I.

Sejam $\mathcal{P}$ um problema de minimização online com soluções de custo não-negativo e $\mathcal{I}$ o conjunto de instâncias de $\mathcal{P}$. Seja A um algoritmo online para $\mathcal{P}$. Dados $\alpha : \mathcal{I} \to \mathbb{R}$ e um real $d \geq 0$, se, para toda instância viável I de $\mathcal{P}$, o algoritmo A produz uma solução viável S de I que satisfaz

$$\mathrm{val}(S, I) \leq \alpha(I) \cdot \mathrm{opt}(I) + d,$$

então se diz que A é **α-competitivo**. O ínfimo (sobre todas as instâncias) dos números α que satisfazem essa desigualdade é denominado **fator de competitividade** de A. Conceitos análogos podem ser definidos para problemas de maximização online.

Embora as definições de algoritmo α-competitivo e α-aproximação assintótica sejam parecidas, a análise competitiva compara a solução de um algoritmo com o custo ótimo offline, enquanto que na análise de um algoritmo de aproximação a comparação é com uma solução ótima global. Aqui, o uso da constante d é importante para que o fator de competitividade não tenha dependência das requisições iniciais, e sim da sequência I como um todo.

Seja A um algoritmo online probabilístico para um problema de minimização online $\mathcal{P}$ com soluções de custo não-negativo e conjunto de instâncias $\mathcal{I}$. Dados $\alpha : \mathcal{I} \to \mathbb{R}$ e um real $d \geq 0$, se, para toda instância viável I de $\mathcal{P}$, o algoritmo A produz uma solução viável S de I que satisfaz

$$\mathbb{E}[\mathrm{val}(S, I)] \leq \alpha(I) \cdot \mathrm{opt}(I) + d,$$

então se diz que A é α-competitivo[1] (no sentido probabilístico). O ínfimo (sobre todas as instâncias) dos números α que satisfazem essa desigualdade é denominado **fator de competitividade esperado** de A. Conceitos análogos podem ser definidos para problemas de maximização.

Aqui vale a pena ressaltar que, diferentemente do estudo de algoritmos de aproximação polinomiais, nem sempre são feitas restrições ao consumo de tempo ou de espaço do algoritmo online, embora em geral seja desejável obter algoritmos eficientes. O objetivo é, principalmente, entender se a restrição de computação online impõe alguma perda de garantia de qualidade das soluções devolvidas e, em caso afirmativo, delimitar essa perda.

[1] O modelo de algoritmos online probabilísticos aqui utilizado corresponde ao modelo do adversário inconsciente (*oblivious adversary*) descrito em [BEY98, Seção 7.1], no qual o adversário conhece o código do algoritmo, mas não conhece os resultados dos sorteios realizados pelo mesmo em tempo de execução.

2.6 O Número Harmônico

A seguinte definição será utilizada no restante da dissertação.

Definição 2.13: *O n-ésimo* **número harmônico**, *para $n = 1, 2, \ldots$, é a somatória*

$$1 + \frac{1}{2} + \ldots + \frac{1}{n} = \sum_{i=1}^{n} \frac{1}{i},$$

a qual se denota por H_n.

Fato 2.14: *Para $n \geq 1$, vale que $\ln n < H_n < 1 + \ln n$, logo $H_n = \Theta(\lg n)$.*

Uma prova desse fato está disponível em [CLRS01, Seção A.2].

Parte I

Aproximação de Métricas

Capítulo 3

Introdução

Neste capítulo são apresentadas algumas definições sobre métricas e aproximação de métricas, bem como alguns limites inferiores sobre aproximação de métricas por métricas arbóreas.

3.1 Métricas

O conceito de **métrica** é uma formalização da noção de distância entre os elementos de um conjunto.

Definição 3.1: *Uma métrica é um par (V, d), onde V é um conjunto e d é uma função de $V \times V$ em $\mathbb{R}$ tal que, para todo u, v, w em V:*

- $d(u, v) \geq 0$;
- $d(u, v) = 0$ *se e só se* $u = v$ (identidade dos indiscerníveis);
- $d(u, v) = d(v, u)$ (simetria);
- $d(u, v) \leq d(u, w) + d(w, v)$ (desigualdade triangular).

A função d é denominada **distância**. Se V é finito, então se diz que a métrica é **finita**; nesse caso, o tamanho da métrica é o tamanho de V. Para não carregar a notação, denota-se (u, v) por uv. A fim de evitar ambiguidades quando se referindo a mais de uma métrica, é usual denotar V por V_M e d por d_M para uma métrica $M = (V, d)$.

Exemplo 3.2: *Para $t > 0$, a métrica t-uniforme com n pontos consiste em (V, d), onde V é um conjunto de tamanho n e, para $u, v \in V$, $d(u, v) := t$ se $u \neq v$, e $d(u, v) := 0$, caso contrário.*

Demonstração: As propriedades de não-negatividade, identidade dos indiscerníveis e simetria valem trivialmente. Resta mostrar que vale a desigualdade triangular.

Sejam $u, v \in V$. Se $d(u, v) = 0$, tem-se que, para todo $w \in V$, $d(u, w) \geq 0$ e $d(w, v) \geq 0$, logo $d(u, v) \leq d(u, w) + d(w, v)$. Se $d(u, v) = t$, então $u \neq v$ e, para todo $w \in V$, tem-se que $w \neq u$ ou $w \neq v$. Logo $d(u, w) + d(w, v) \geq t$ e, portanto, $d(u, v) \leq d(u, w) + d(w, v)$. □

Chama-se uma métrica de **uniforme** quando ela é t-uniforme para algum valor real $t > 0$.

A seguir é definido o conceito de **raio** de uma submétrica.

Definição 3.3: *Seja $S \subseteq V_M$ e w um elemento de V_M, não necessariamente em S. Diz-se que S tem raio r em relação a w se r é a maior distância entre w e algum elemento de S.*

Uma métrica comum em diversas aplicações é a seguinte.

Exemplo 3.4: *Seja G um grafo conexo com um custo positivo associado a cada aresta. Defina o custo de um caminho P em G como a soma dos custos das arestas em P. Seja d a função que associa a cada par ordenado de vértices (u, v) de G o custo de um caminho de custo mínimo de u a v em G. Então (V_G, d) é uma métrica.*

Demonstração: Sejam u, v e w vértices quaisquer de G. Observe que, como G é conexo, $d(u, v)$ está bem definida: existe pelo menos um caminho de u a v em G, e o número de caminhos em G é finito.

Note que o caminho trivial tem custo zero. Além disso, toda aresta tem custo positivo, logo $d(u, v) \geq 0$.

Claramente, tomando o caminho trivial, $d(u, u) = 0$. Além disso, como toda aresta tem custo maior que zero, se $u \neq v$, então $d(u, v) > 0$.

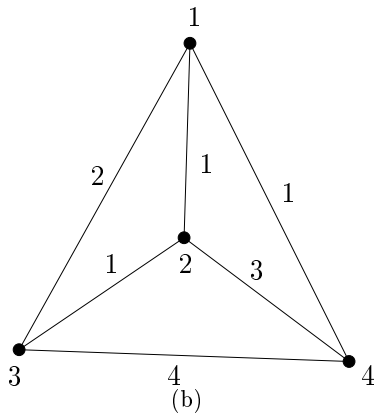
Como o reverso de um caminho de u a v é um caminho de v a u e vice-versa, todo caminho de custo mínimo de u a v corresponde a um caminho de custo mínimo de v a u. Logo $d(u, v) = d(v, u)$.

Sejam P, Q e R caminhos de custo mínimo entre u e v, u e w e w e v, respectivamente. Note que a concatenação de Q e R contém um caminho de u a v de custo no máximo $d(u, w) + d(w, v)$. Tal caminho tem custo maior ou igual ao custo de P, já que P é mínimo. Logo $d(u, v) \leq d(u, w) + d(w, v)$. □

Diz-se que essa é a métrica **induzida pelas distâncias** no grafo G, e que a função d é a **distância em** G. Dado um grafo G com custos positivos nas arestas, denota-se por d_G a distância em G. Note que toda métrica finita corresponde à métrica induzida pelas distâncias num grafo, por exemplo, o grafo completo com custos nas arestas iguais às distâncias na métrica (ver Figura 3.1).

d	1	2	3	4
1	0	1	2	1
2	1	0	1	3
3	2	1	0	4
4	1	3	4	0

(a)



(b)

Figura 3.1: *(a) Uma métrica definida sobre o conjunto $\{1, 2, 3, 4\}$; (b) um grafo correspondente a essa métrica.*

Definição 3.5: *A métrica estrela é a métrica induzida pelas distâncias no $K_{1,n}$, com custo 1 em cada aresta.*

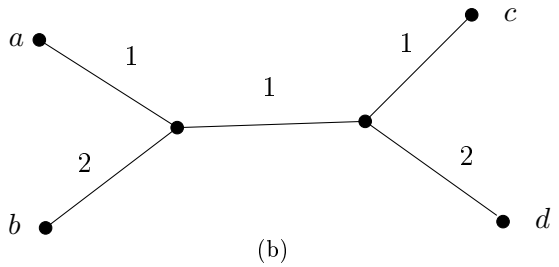
Dada uma métrica (V, d), para manter compatibilidade com a terminologia de grafos, é comum denominar os elementos de V por **vértices** e um par de vértices de V por **aresta**. O valor $d(u, v)$, para $u \neq v$, pode ser pensado como o custo da aresta uv. A maior distância entre dois vértices de uma métrica M é o **diâmetro** de M, denotado por $\text{diam}(M)$. Similarmente, dado um grafo G com custos positivos nas arestas, o **diâmetro** de G, denotado por $\text{diam}(G)$, é o valor $\max_{u,v \in V_G} d_G(u, v)$.

Definição 3.6: *Uma métrica M é arbórea* (tree metric, *ou métrica aditiva*) *se existe uma árvore T com custos positivos nas arestas tal que $V_T \supseteq V_M$ e, para quaisquer $u, v \in V_M$, vale que $d_M(u, v) = d_T(u, v)$.*

A Figura 3.2(a) mostra um exemplo de métrica arbórea. A árvore da Figura 3.2(b) contém os vértices dessa métrica, e as distâncias na métrica e na árvore são idênticas. Note que, nesse caso, a árvore possui mais vértices que a métrica. Além disso, uma mesma métrica arbórea pode ser equivalente à distância em mais de uma árvore diferente.

	a	b	c	d
a	0	3	3	4
b	3	0	4	5
c	3	4	0	3
d	4	5	3	0

(a)



(b)

Figura 3.2: *(a) Uma métrica arbórea definida sobre o conjunto $\{a, b, c, d\}$; (b) uma árvore cuja métrica induzida por suas distâncias restrita a $\{a, b, c, d\}$ é idêntica a essa métrica.*

Diversos problemas de otimização em métricas têm uma solução mais simples, menos custosa ou com melhor fator de aproximação, quando restritos a métricas arbóreas. Portanto, uma estratégia para obter bons algoritmos é aproximar a métrica de interesse por uma métrica arbórea. No Capítulo 4, é apresentado um algoritmo para obter uma tal aproximação com baixa "distorção", que resultou em algoritmos de aproximação para diversos problemas (ver Seção 1.2). A próxima seção define melhor o que seria uma métrica "aproximar" outra, e mostra como um algoritmo pode utilizar essa aproximação como sub-rotina.

3.2 Aproximação de Métricas

Nesta seção será formalizado o conceito de aproximação de métricas, e como esse conceito pode ser utilizado para obter bons algoritmos de aproximação.

Definição 3.7: *Uma métrica M'* **domina** *outra métrica M se $V_{M'} = V_M$ e*

$$d_{M'}(u, v) \geq d_M(u, v) \quad \text{para todo } u, v \in V_M.$$

Definição 3.8: *Seja $\alpha \geq 1$. Uma métrica M' α-**aproxima** outra métrica M se M' domina M e*

$$d_{M'}(u, v) \leq \alpha \cdot d_M(u, v) \quad \text{para todo } u, v \in V_M.$$

Eventualmente, α pode depender de M. O ínfimo dos números $\alpha \geq 1$ que satisfazem essa

desigualdade é denominado **distorção** *de M' em relação a M.*

Uma definição análoga para o caso probabilístico é a seguinte.

Definição 3.9: *Seja $\mathcal{S}$ uma família de métricas e $\mathcal{D}$ uma distribuição de probabilidade sobre $\mathcal{S}$. Dado um número positivo α, diz-se que $(\mathcal{S}, \mathcal{D})$* **$\alpha$-aproxima probabilisticamente** *uma métrica M se cada métrica em $\mathcal{S}$ domina M e*

$$\mathbb{E}_{M' \in_{\mathcal{D}} \mathcal{S}}[d_{M'}(u,v)] \leq \alpha \cdot d_M(u,v), \quad \textit{para todo } u, v \in V_M.$$

Claramente, a distribuição de probabilidade $\mathcal{D}$ deve ser discreta se a família $\mathcal{S}$ for finita, e contínua se $\mathcal{S}$ for infinita. Em alguns pontos do texto, será feito abuso da nomenclatura e, dada uma métrica M' escolhida de $\mathcal{S}$ com probabilidade $\mathcal{D}(M')$, dir-se-á que M' α-aproxima M.

O fato a seguir simplifica algumas demonstrações sobre aproximação de métricas.

Fato 3.10 (Transitividade da Aproximação de Métricas): *Seja M uma métrica e suponha que $(\mathcal{S}, \mathcal{D})$ α-aproxime probabilisticamente M. Para cada métrica $N \in \mathcal{S}$, seja $(\mathcal{S}_N, \mathcal{D}_N)$ que β-aproxime probabilisticamente N. Sejam então $\mathcal{S}' := \bigcup_{N \in \mathcal{S}} \mathcal{S}_N$ e $\mathcal{D}'$ a distribuição de probabilidade sobre $\mathcal{S}'$ tal que, para cada $L \in \mathcal{S}'$, $\mathcal{D}'(L) := \sum_{N : \mathcal{S}_N \ni L} \mathcal{D}(N) \cdot \mathcal{D}_N(L)$. Então $(\mathcal{S}', \mathcal{D}')$ $\alpha\beta$-aproxima probabilisticamente M.*

Demonstração: Claramente, $\mathcal{D}'$ é uma distribuição de probabilidade, pois

$$\sum_{L \in \mathcal{S}'} \sum_{N : \mathcal{S}_N \ni L} \mathcal{D}(N) \cdot \mathcal{D}_N(L) = \sum_{N \in \mathcal{S}} \sum_{L \in \mathcal{S}_N} \mathcal{D}(N) \cdot \mathcal{D}_N(L) = \sum_{N \in \mathcal{S}} \mathcal{D}(N) \sum_{L \in \mathcal{S}_N} \mathcal{D}_N(L) = \sum_{N \in \mathcal{S}} \mathcal{D}(N) = 1.$$

Para métricas $N \in \mathcal{S}$ e $L \in \mathcal{S}_N$ e vértices $u, v \in V_M$, vale que $d_L(u,v) \geq d_N(u,v) \geq d_M(u,v)$, logo toda métrica em $\mathcal{S}'$ domina M. Além disso, para todo $u, v \in V_M$,

$$\begin{aligned}
\mathbb{E}_{L \in_{\mathcal{D}'} \mathcal{S}'}[d_L(u,v)] &= \sum_{L \in \mathcal{S}'} \mathcal{D}'(L) \cdot d_L(u,v) \\
&= \sum_{L \in \mathcal{S}'} \left(\sum_{N : \mathcal{S}_N \ni L} \mathcal{D}(N) \cdot \mathcal{D}_N(L) \right) \cdot d_L(u,v) \\
&= \sum_{N \in \mathcal{S}} \sum_{L \in \mathcal{S}_N} \mathcal{D}(N) \cdot \mathcal{D}_N(L) \cdot d_L(u,v) \\
&= \sum_{N \in \mathcal{S}} \mathcal{D}(N) \left(\sum_{L \in \mathcal{S}_N} \mathcal{D}_N(L) \cdot d_L(u,v) \right) \\
&= \sum_{N \in \mathcal{S}} \mathcal{D}(N) \cdot \mathbb{E}_{L \in_{\mathcal{D}_N} \mathcal{S}_N}[d_L(u,v)] \\
&\leq \sum_{N \in \mathcal{S}} \mathcal{D}(N) \cdot \beta \cdot d_N(u,v) \\
&= \beta \cdot \mathbb{E}_{N \in_{\mathcal{D}} \mathcal{S}}[d_N(u,v)] \\
&\leq \alpha\beta \cdot d_M(u,v).
\end{aligned}$$

□

Esse teorema mostra que é possível compor algoritmos de aproximação de métricas da seguinte forma. Seja A um algoritmo que recebe uma métrica $M \in \mathcal{C}$ e devolve uma métrica $N \in \mathcal{S}$ com probabilidade $\mathcal{D}(N)$, onde $(\mathcal{S}, \mathcal{D})$ α-aproxima probabilisticamente M. Seja B um algoritmo que, para toda métrica $N \in \mathcal{S}$, devolve uma métrica $L \in \mathcal{S}_N$ com probabilidade $\mathcal{D}_N(L)$, onde $(\mathcal{S}_N, \mathcal{D}_N)$ β-aproxima probabilisticamente N. Então, o algoritmo que resulta da composição de B com A recebe uma métrica $M \in \mathcal{C}$ e devolve a métrica $L = B(A(M)) \in \bigcup_{N \in \mathcal{S}} \mathcal{S}_N$ com probabilidade $\sum_{N:\mathcal{S}_N \ni L} \mathcal{D}(N) \cdot \mathcal{D}_N(L)$. Logo L $\alpha\beta$-aproxima probabilisticamente M. Essa composição é útil, em especial, quando as métricas em $\bigcup_{N \in \mathcal{S}} \mathcal{S}_N$ têm uma estrutura específica que é utilizada por um determinado algoritmo. Assim, em vez de projetar um novo algoritmo para aproximar a métrica M, pode ser mais simples projetar um algoritmo para aproximar uma métrica em $\mathcal{S}$ por uma métrica em $\bigcup_{N \in \mathcal{S}} \mathcal{S}_N$.

Seja $\mathbf{x}$ um vetor de números reais indexado por $V_M \times V_M$, onde M é uma métrica. Denota-se por $\mathbf{x} \circ \mathbf{d}_M$ o somatório $\sum_{u,v \in V_M} \mathbf{x}(u,v) \cdot d_M(u,v)$. O teorema a seguir formaliza a ideia de compor um algoritmo de aproximação para uma classe restrita de métricas com um algoritmo de aproximação de métricas. Foi adicionada uma restrição, o item (2), em relação ao enunciado original, além de uma modificação no item (1); essas adições mostraram-se necessárias durante a tentativa de reescrever a demonstração original.

Teorema 3.11 (Bartal, 1996 [Bar96, Teorema 4]): *Seja $\mathcal{P}$ um problema de minimização em que cada instância I contém uma métrica M. Seja $I\langle M, M'\rangle$ a instância obtida pela substituição de M por M' em I. Suponha que*

(1) para cada solução viável S de I, o valor $\mathrm{val}(S, I)$ seja uma combinação linear das distâncias em M, com coeficientes não-negativos;

(2) para toda métrica M' tal que $V_{M'} = V_M$, valha que $\mathrm{Sol}(I\langle M, M'\rangle) = \mathrm{Sol}(I)$;

(3) exista uma β-aproximação B, probabilística ou determinística, para $\mathcal{P}$ quando este é restrito a uma classe de métricas $\mathcal{C}$;

(4) exista um algoritmo A que, dado M, devolve uma métrica $M' = A(M)$ de $\mathcal{S} \subseteq \mathcal{C}$ com probabilidade $\mathcal{D}(M')$, onde $(\mathcal{S}, \mathcal{D})$ α-aproxima probabilisticamente M.

Então,

$$\mathbb{E}[\mathrm{val}(B(I\langle M, A(M)\rangle), I)] \leq \alpha\beta \cdot \mathrm{opt}(I),$$

onde a esperança diz respeito às escolhas aleatórias realizadas por A e eventualmente por B, se B for probabilístico. Ou seja, A e B compostos formam uma $\alpha\beta$-aproximação probabilística para $\mathcal{P}$.

Demonstração: Sejam $M' := A(M)$, $I' := I\langle M, M'\rangle$ e $\mathrm{OPT}(I)$ e $\mathrm{OPT}(I')$ soluções ótimas quaisquer de I e I', respectivamente. Note, pelo item (2), que $B(I') \in \mathrm{Sol}(I)$ e $\mathrm{OPT}(I) \in \mathrm{Sol}(I')$.

Do item (1), para cada $S \in \mathrm{Sol}(I)$, existe um vetor $\mathbf{x}_S$ indexado por $V_M \times V_M$ com entradas não-negativas, tal que $\mathrm{val}(S, I) = \mathbf{x}_S \circ \mathbf{d}_M$ e $\mathrm{val}(S, I') = \mathbf{x}_S \circ \mathbf{d}_{M'}$. Então, utilizando a linearidade da esperança,

$$\begin{aligned} \mathbb{E}[\mathrm{val}(B(I'), I)] &= \mathbb{E}[\mathbf{x}_{B(I')} \circ \mathbf{d}_M] \\ &\leq \mathbb{E}[\mathbf{x}_{B(I')} \circ \mathbf{d}_{M'}] && \text{(do item (4) e da Definição 3.7)} \end{aligned}$$

$$
\begin{aligned}
&= \mathbb{E}[\mathrm{val}(B(I'), I')] \\
&\leq \beta \cdot \mathbb{E}[\mathrm{val}(\mathrm{OPT}(I'), I')] && \text{(do item (3), já que } M' \in \mathcal{C}) \\
&\leq \beta \cdot \mathbb{E}[\mathrm{val}(\mathrm{OPT}(I), I')] && \text{(da definição de ótimo)} \\
&= \beta \cdot \sum_{u,v \in V_M} \mathbf{x}_{\mathrm{OPT}(I)}(u,v) \cdot \mathbb{E}[d_{M'}(u,v)] \\
&\leq \beta \cdot \sum_{u,v \in V_M} \mathbf{x}_{\mathrm{OPT}(I)}(u,v) \cdot \alpha \cdot d_M(u,v) && \text{(do item (4))} \\
&= \alpha\beta \cdot \mathrm{opt}(I).
\end{aligned}
$$

□

Esse resultado pode ser adaptado para problemas de otimização online: no lugar da β-aproximação exigida no item (3), pede-se um algoritmo online β-competitivo, e o que se obtém é um algoritmo online probabilístico $\alpha\beta$-competitivo.

O foco deste trabalho é o caso em que $\mathcal{C}$ consiste na classe das métricas arbóreas. Seja então A um algoritmo que, dada uma métrica arbitrária M, obtém uma métrica arbórea $T \in \mathcal{S}$ com probabilidade $\mathcal{D}(T)$, onde $\mathcal{S} \subseteq \mathcal{C}$ e $(\mathcal{S}, \mathcal{D})$ α-aproxima probabilisticamente M. O teorema implica que A pode ser composto com uma β-aproximação para $\mathcal{P}$ restrito a métricas arbóreas, de forma a obter uma $\alpha\beta$-aproximação probabilística para o caso geral de $\mathcal{P}$. Na seção a seguir são apresentados alguns resultados específicos para aproximação de métricas por métricas arbóreas.

3.3 Aproximação de Métricas por Métricas Arbóreas

Nesta seção são apresentados alguns limites inferiores sobre aproximação de métricas por métricas arbóreas.

A demonstração do seguinte teorema não será apresentada, uma vez que envolve conceitos de topologia algébrica, que fogem do escopo deste trabalho.

Teorema 3.12 (Rabinovich e Raz, 1998 [RR98, Corolário 5.3]): *Para todo n suficientemente grande, existe um grafo conexo G com n vértices tal que, se a métrica induzida pelas distâncias em G é α-aproximada por uma métrica arbórea, então $\alpha = \Omega(n)$.*

3.3.1 Limite Probabilístico

Uma pergunta que se segue à constatação do Teorema 3.12 é se o mesmo limite se aplica a aproximações probabilísticas por métricas arbóreas. Diversos exemplos mostram que não (um deles será discutido no Capítulo 4). Portanto, é interessante obter um limite inferior também para o caso probabilístico. Nesta seção mostra-se que toda aproximação probabilística de uma métrica arbitrária por métricas arbóreas tem distorção esperada pelo menos logarítmica no número de vértices da métrica. A seção se baseia no trabalho de Bartal [Bar96].

Considere um grafo conexo G com um custo positivo $c_G(u,v)$ para cada aresta uv de G. Uma **ℓ-partição fraca** de G é uma partição P de V_G tal que, para toda parte U de P e todo par u, v de U, vale que $d_G(u,v) \leq \ell$. Note que, se $\ell \geq \mathrm{diam}(G)$, qualquer partição de V_G satisfaz essa condição. O valor ℓ é denominado o **diâmetro** das partes de P.

Dada uma distribuição de probabilidade $\mathcal{D}$ sobre as ℓ-partições fracas de G e uma aresta $uv \in E_G$, seja $x_{\mathcal{D}}(u,v)$ a probabilidade de u e v estarem em partes distintas de uma ℓ-partição fraca

de G escolhida segundo $\mathcal{D}$. Dados parâmetros $r \geq 0$, $\rho > 0$ e $\lambda > 0$, uma (r, ρ, λ)**-partição probabilística fraca** de G é uma distribuição de probabilidade $\mathcal{D}$ sobre o conjunto de $(r\rho)$-partições fracas de G tal que

$$\frac{x_{\mathcal{D}}(u,v)}{c_G(u,v)} \cdot r \leq \lambda \quad \text{para toda aresta } uv \in E_G.$$

Note que a função do parâmetro λ é delimitar o valor de $x_{\mathcal{D}}(u,v)$ em relação ao valor de $c_G(u,v)$, enquanto que o parâmetro ρ delimita o diâmetro das partes das partições fracas de G. O parâmetro r, por sua vez, serve para correlacionar esses dois limites. Informalmente, o teorema a seguir mostra que não é possível diminuir muito a probabilidade de uma aresta ter seus vértices separados sem aumentar o diâmetro das partes das partições fracas ou, reciprocamente, que não é possível diminuir muito o diâmetro das partes das partições fracas sem aumentar a probabilidade de uma aresta ter seus vértices separados. O enunciado difere um pouco do original, visto que se notaram necessárias algumas modificações.

Teorema 3.13 (Bartal, 1996 [Bar96, Teorema 15]): *Para todo n suficientemente grande, existe um grafo conexo G com n vértices e custo unitário nas arestas tal que se, para algum $\rho \geq 1$ e $\lambda \geq \frac{1}{2}$, existe uma (r, ρ, λ)-partição probabilística fraca de G para todo $r \geq 1$, então $\lambda\rho = \Omega(\lg n)$.*

Demonstração: Seja $k \geq 33^3$. Pelo Corolário 2.3, existe um grafo G com $n \geq k$ vértices que possui cintura $g(G) \geq \log_{33} n$ e $m > 2n$ arestas. Observe que, se G é um grafo com cintura g, então existe um par de vértices cuja distância em G é pelo menos $\lfloor g/2 \rfloor$.

Atribua custo 1 a cada aresta de G. Sejam $\rho \geq 1$ e $\lambda \geq \frac{1}{2}$ tais que $\lambda\rho < \lfloor \frac{1}{4} \log_{33} n \rfloor$. Tome $r := 2\lambda$; como $\lambda \geq \frac{1}{2}$, então $r \geq 1$. A prova consiste em mostrar que não existe uma (r, ρ, λ)-partição probabilística fraca de G.

Seja $\mathcal{P}$ o conjunto de todas as $(r\rho)$-partições fracas de G. Pela definição de $(r\rho)$-partição fraca de G, para toda parte U de P e todo par u, v de U, vale que $d_G(u,v) \leq r\rho = 2\lambda\rho < \lfloor \frac{1}{2} \log_{33} n \rfloor$. Seja $G[U]$ o subgrafo de G induzido por U. Note que $G[U]$ é uma floresta pois, se $G[U]$ possuísse um circuito, tal circuito teria comprimento pelo menos $\log_{33} n$, logo existiria um par $u, v \in U$ tal que $d_G(u,v) \geq \lfloor \frac{1}{2} \log_{33} n \rfloor$, uma contradição. Assim, em qualquer $r\rho$-partição fraca de G, o número de arestas entre partes distintas é pelo menos $m - (n-1) > m/2$.

Para cada $P \in \mathcal{P}$, seja $X_P(u,v) := 1$ se u e v estão em partes distintas de P, e $X_P(u,v) := 0$ caso contrário. Do parágrafo anterior, vale que $\sum_{uv \in E_G} X_P(u,v) > m/2$.

Agora suponha, por contradição, que exista uma (r, ρ, λ)-partição probabilística fraca $\mathcal{D}$ de G. Por definição, vale que $x_{\mathcal{D}}(u,v) \leq \frac{\lambda}{r} = \frac{1}{2}$ para toda aresta $uv \in E_G$, e portanto $\sum_{uv \in E_G} x_{\mathcal{D}}(u,v) \leq m/2$. Mas

$$x_{\mathcal{D}}(u,v) = \sum_{P \in \mathcal{P}} X_P(u,v) \cdot \mathcal{D}(P) = \mathbb{E}_{P \in_{\mathcal{D}} \mathcal{P}}[X_P(u,v)].$$

Logo

$$\sum_{uv \in E_G} x_{\mathcal{D}}(u,v) = \sum_{uv \in E_G} \mathbb{E}_{P \in_{\mathcal{D}} \mathcal{P}}[X_P(u,v)] = \mathbb{E}_{P \in_{\mathcal{D}} \mathcal{P}} \left[\sum_{uv \in E_G} X_P(u,v) \right] > \frac{m}{2},$$

uma contradição.

Portanto, se existem $\lambda \geq \frac{1}{2}$ e $\rho \geq 1$ tais que existe uma (r, ρ, λ)-partição probabilística fraca

de G para todo $r \geq 1$, então $\lambda\rho \geq \frac{1}{4}\lfloor \log_{33} n \rfloor$, ou seja, $\lambda\rho = \Omega(\lg n)$. □

O seguinte lema também será necessário. O mesmo é enunciado por Bartal [Bar96] sem demonstração.

Lema 3.14 (Bartal, 1996 [Bar96, segunda parte do Teorema 14]): *Toda árvore T com custos positivos nas arestas possui uma $(r, 1, 1)$-partição probabilística fraca, para todo $r \geq 1$.*

No trabalho original, esse lema é enunciado para $1 \leq r \leq \text{diam}(T)$. Caso $r > \text{diam}(T)$, basta tomar $\mathcal{D}$ que vale 1 se e só se $P = \{V_G\}$. Para tal $\mathcal{D}$, vale que $x_{\mathcal{D}}(u, v) = 0$, para toda aresta $uv \in E_G$.

O Teorema 3.13 e o Lema 3.14 são utilizados, então, como base para o teorema a seguir, que prova o limite desejado.

Teorema 3.15 (Bartal, 1996 [Bar96, Teorema 9]): *Para n arbitrariamente grande, existe um grafo conexo G com n vértices e custos positivos nas arestas tal que, se a métrica induzida pelas distâncias em G é α-aproximada probabilisticamente por métricas arbóreas, então $\alpha = \Omega(\lg n)$.*

Demonstração: Seja G um grafo conexo com um custo positivo $c_G(u, v)$ associado a cada aresta uv de G, e suponha que a métrica induzida pelas distâncias em G seja α-aproximada probabilisticamente por $(\mathcal{S}, \mathcal{D})$, onde $\mathcal{S}$ é um conjunto de métricas arbóreas. O passo principal da prova consiste em mostrar que, para todo $r \geq 1$, existe uma $(r, \alpha, 2)$-partição probabilística fraca de G.

Fixe um $r \geq 1$ e seja M uma métrica em $\mathcal{S}$, escolhida com probabilidade $\mathcal{D}(M)$. Como M é uma métrica arbórea, existe uma árvore T com custos positivos nas arestas, possivelmente com mais vértices que M (e, portanto, que G), tal que M corresponde à métrica induzida pelas distâncias em T restrita aos vértices de G. Seja $\mathcal{D}_T$ uma $(r\alpha, 1, 1)$-partição probabilística fraca de T, dada pelo Lema 3.14; logo, para toda aresta uv de T, vale que

$$\frac{x_{\mathcal{D}_T}(u, v)}{c_T(u, v)} \leq \frac{1}{r\alpha}. \tag{3.1}$$

Dada uma partição P de V_T, diz-se que P **induz** a partição P' de V_G se dois vértices u e v de G estão na mesma parte de P' se e só se estão na mesma parte de P. Note que uma partição de V_T induz uma única partição de V_G, mas mais de uma partição de V_T pode induzir a mesma partição de V_G, já que T pode ter mais vértices que G. Note também que, se P é uma ℓ-partição fraca de T, então P' é uma ℓ-partição fraca de G: para dois vértices quaisquer u e v de G numa mesma parte de P, vale que $d_T(u, v) \leq \ell$ e, como a métrica induzida pelas distâncias em T domina a métrica induzida pelas distâncias em G, vale que $d_G(u, v) \leq d_T(u, v)$ e assim $d_G(u, v) \leq \ell$.

Sejam $\mathcal{P}_G$ e $\mathcal{P}_T$ os conjuntos das $(r\alpha)$-partições fracas de G e T, respectivamente, e dada $P' \in \mathcal{P}_G$, seja $\mathcal{P}_T(P') := \{P \in \mathcal{P}_T : P \text{ induz } P'\}$. Seja $\mathcal{D}_G$ a distribuição de probabilidade sobre $\mathcal{P}_G$ tal que

$$\mathcal{D}_G(P') = \sum_{T \in \mathcal{S}} \sum_{P \in \mathcal{P}_T(P')} \mathcal{D}(T) \cdot \mathcal{D}_T(P).$$

Segue a prova de que $\mathcal{D}_G$ é uma distribuição de probabilidade:

$$\begin{aligned}
\sum_{P' \in \mathcal{P}_G} \mathcal{D}_G(P') &= \sum_{P' \in \mathcal{P}_G} \sum_{T \in \mathcal{S}} \sum_{P \in \mathcal{P}_T(P')} \mathcal{D}(T) \cdot \mathcal{D}_T(P) \\
&= \sum_{T \in \mathcal{S}} \mathcal{D}(T) \sum_{P' \in \mathcal{P}_G} \sum_{P \in \mathcal{P}_T(P')} \mathcal{D}_T(P) \\
&= \sum_{T \in \mathcal{S}} \mathcal{D}(T) \sum_{P \in \mathcal{P}_T} \mathcal{D}_T(P) \qquad (3.2) \\
&= \sum_{T \in \mathcal{S}} \mathcal{D}(T) \\
&= 1.
\end{aligned}$$

A igualdade (3.2) segue do fato de que uma partição de V_T induz uma única partição de V_G, já que $V_T \supseteq V_G$.

Mais que isso, é possível mostrar que $\mathcal{D}_G$ é uma $(r, \alpha, 2)$-partição probabilística fraca de G. Para isso, basta mostrar que

$$\frac{x_{\mathcal{D}_G}(u,v)}{c_G(u,v)} \leq \frac{2}{r}$$

para toda aresta uv de G, lembrando que $x_{\mathcal{D}_G}(u,v)$ é a probabilidade de u e v estarem em partes distintas de uma $(r\alpha)$-partição fraca de G escolhida segundo $\mathcal{D}_G$.

Para cada $T \in \mathcal{S}$ e cada $P \in \mathcal{P}_T$, seja T_P a árvore obtida pela alteração nos custos de T da seguinte forma: se uv é uma aresta de T e u e v estão em partes distintas de P, então $c_{T_P}(u,v) := r\alpha$; caso contrário, $c_{T_P}(u,v) := c_T(u,v)$. Então, para toda aresta uv de T,

$$\begin{aligned}
\mathbb{E}_{P \in_{\mathcal{D}_T} \mathcal{P}_T}[c_{T_P}(u,v)] &= (1 - x_{\mathcal{D}}(u,v)) \cdot c_T(u,v) + x_{\mathcal{D}}(u,v) \cdot r\alpha \\
&\leq c_T(u,v) + c_T(u,v) \qquad (3.3) \\
&= 2 \cdot c_T(u,v),
\end{aligned}$$

onde a desigualdade (3.3) vem da Equação (3.1). Logo, para todo $u, v \in V_G$, tem-se que

$$\mathbb{E}_{T \in_{\mathcal{D}} S \wedge P \in_{\mathcal{D}_T} \mathcal{P}_T}[d_{T_P}(u,v)] \leq 2 \cdot \mathbb{E}_{T \in_{\mathcal{D}} S}[d_T(u,v)] \leq 2\alpha \cdot d_G(u,v).$$

Se u e v estão em partes distintas de $P' \in \mathcal{P}_G$ induzida por P, então $d_{T_P}(u,v) \geq r\alpha$, já que toda aresta de T_P entre vértices de partes distintas de P' (e portanto de P) tem custo $r\alpha$. Logo $x_{\mathcal{D}_G}(u,v) \leq \mathbb{P}_{T \in_{\mathcal{D}} S \wedge P \in_{\mathcal{D}_T} \mathcal{P}_T}[d_{T_P}(u,v) \geq r\alpha]$. Mas, como $\mathbb{E}_{T \in_{\mathcal{D}} S \wedge P \in_{\mathcal{D}_T} \mathcal{P}_T}[d_{T_P}(u,v)] \leq 2\alpha \cdot d_G(u,v)$, utilizando a desigualdade de Markov (Fato 2.10),

$$\begin{aligned}
x_{\mathcal{D}_G}(u,v) \leq \mathbb{P}_{T \in_{\mathcal{D}} S \wedge P \in_{\mathcal{D}_T} \mathcal{P}_T}[d_{T_P}(u,v) \geq r\alpha] &\leq \frac{1}{r\alpha} \cdot \mathbb{E}_{T \in_{\mathcal{D}} S \wedge P \in_{\mathcal{D}_T} \mathcal{P}_T}[d_{T_P}(u,v)] \\
&\leq \frac{1}{r\alpha} \cdot 2\alpha \cdot d_G(u,v) \\
&= \frac{2}{r} \cdot d_G(u,v).
\end{aligned}$$

Logo $x_{\mathcal{D}_G}(u,v) \leq \frac{2}{r} \cdot d_G(u,v) = \frac{2}{r} \cdot c_G(u,v)$ para toda aresta $uv \in E_G$ e $\mathcal{D}_G$ é uma $(r, \alpha, 2)$-partição probabilística fraca de G.

Seja então G o grafo dado pelo Teorema 3.13. Se a métrica induzida pelas distâncias em G é α-aproximada probabilisticamente por uma distribuição de probabilidade sobre métricas arbóreas então, pelo que foi mostrado acima, para todo $r \geq 1$, existe uma $(r, \alpha, 2)$-partição probabilística fraca de G. Como $\alpha \geq 1$, pelo Teorema 3.13, vale que $2\alpha = \Omega(\lg n)$, ou seja, $\alpha = \Omega(\lg n)$. □

No Capítulo 4 é mostrado um algoritmo, proposto por Fakcharoenphol, Rao e Talwar [FRT04b, FRT04a], que mostra que o limite inferior do Teorema 3.15 é justo.

Capítulo 4

O Algoritmo FRT

Neste capítulo é descrito o funcionamento do algoritmo de Fakcharoenphol, Rao e Talwar (FRT) para aproximar uma métrica arbitrária M com n vértices por uma distribuição de probabilidade sobre métricas arbóreas. O algoritmo recebe M como entrada e devolve uma métrica arbórea $T \in \mathcal{S}$ com probabilidade $\mathcal{D}(T)$, onde $(\mathcal{S}, \mathcal{D})$ $O(\lg n)$-aproxima probabilisticamente M.

Supõe-se, sem perda de generalidade, que $d_M(u, v) \geq 1$ para todo par u, v de V_M. De fato, seja $\Delta := \min\{d_M(u, v) : u, v \in V_M, u \neq v\}$. Se $\Delta < 1$, divida cada distância em M por Δ. Execute o algoritmo sobre essa instância modificada e, ao final, multiplique por Δ cada distância na métrica arbórea devolvida.

Este capítulo é baseado nos trabalhos de Fakcharoenphol, Rao e Talwar [FRT04b, FRT04a] e na Seção 8.5 de Williamson e Shmoys [WS11]; a versão que se apresenta do algoritmo é a exposta em [FRT04a], com algumas modificações propostas em [WS11].

4.1 Decomposições em Cortes

O algoritmo se baseia nos dois seguintes conceitos. Dado um real positivo r, uma **decomposição em r-cortes** (*r-cut decomposition*) de M é uma partição de V_M tal que a métrica restrita a cada parte tem raio no máximo r em relação a um vértice de M (e, portanto, diâmetro no máximo $2r$). Seja $\delta := \lceil \lg(2 \cdot \operatorname{diam}(M)) \rceil$. Uma **decomposição hierárquica em cortes** (*hierarchical cut decomposition*) de M é uma sequência $(D_0, D_1, \ldots, D_\delta)$ de partições de V_M tal que

1. $D_\delta = \{V_M\}$;
2. para $i = 0, \ldots, \delta$, a partição D_i é uma decomposição em β_i-cortes, para $\beta_i < 2^i$;
3. para $i = 0, \ldots, \delta - 1$, a partição D_i é um refinamento de D_{i+1}, isto é, toda parte de D_i está contida em alguma parte de D_{i+1}.

Note que $\{V_M\}$ é uma decomposição em 2^δ-cortes, já que V_M tem raio no máximo $\operatorname{diam}(M) < 2 \cdot \operatorname{diam}(M) \leq 2^\delta$ em relação a qualquer vértice de M. Note também que cada parte de D_0 tem raio menor que 1, logo tem um único vértice. Além disso, as partições $D_0, D_1, \ldots, D_\delta$ da decomposição definem uma família laminar, que pode ser representada através da árvore T com conjunto de vértices $V_T := \bigcup_{i=0}^{\delta} D_i$, eventualmente com multiplicidade (ou seja, com cada conjunto aparecendo em V_T tantas vezes quanto o número de partições em que o mesmo esteja), e uma aresta entre cada

parte de D_i e cada um de seus subconjuntos em D_{i-1}, para $1 \leq i \leq \delta$. Denomina-se T a **árvore induzida por** D (ver Figura 4.1).

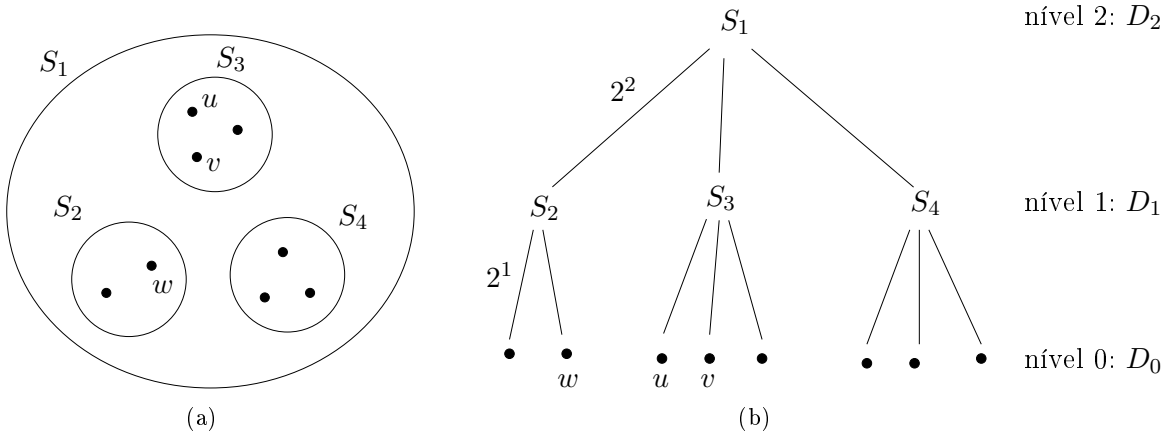


Figura 4.1: *(a) Decomposição hierárquica em cortes. (b) A árvore induzida por essa decomposição.*

Note que as partes de uma mesma decomposição em cortes D_i estão todas no mesmo nível i da árvore. Como cada folha de T consiste em um conjunto com um único vértice de M, o que se faz é identificar a folha com o vértice de M nela contido. Adicionalmente, diz-se que uma aresta uv de M é **separada em T no nível** i, e que u e v são **separados em T no nível** i, se $i+1$ é o menor índice j tal que u e v estão numa mesma parte de D_j. Por exemplo, na decomposição da Figura 4.1, uv é separada no nível 0 e u e w são separados no nível 1. Observe que toda aresta é separada em algum nível, já que $D_\delta = \{V_M\}$ e $D_0 = V_M$.

Além disso, é associado um custo 2^i a cada aresta de T entre vértices nos níveis i e $i-1$, para $1 \leq i \leq \delta$ (ver Figura 4.1). Note que T é uma 2-HST perfeita de base 2 (ver Definições 2.5, 2.6 e 2.7). Considere então a métrica induzida pelas distâncias em T, mas restrita às folhas de T, a qual será denominada a **métrica induzida por** D. Essa é a métrica utilizada pelo algoritmo FRT para aproximar a métrica M. O lema a seguir mostra que a métrica induzida por D domina M.

Lema 4.1: *Seja D uma decomposição hierárquica em cortes de uma métrica finita M cuja menor distância não-nula é pelo menos 1. Então, a métrica induzida por D domina M.*

Demonstração: Por definição, a métrica induzida por D tem o mesmo conjunto de vértices que M. Seja T a árvore induzida por D; resta mostrar que $d_T(u,v) \geq d_M(u,v)$, para todo $u, v \in V_M$.

Sejam u, v dois vértices quaisquer de M. Seja i o nível em que a aresta uv é separada em T, e S a parte de D_{i+1} que contém u e v. (Note que S é um vértice de T.) Então a distância de S a u em T é pelo menos 2^{i+1}, já que a aresta de S ao subconjunto de S que contém u no nível i tem custo 2^{i+1} (ver Figura 4.2). Por argumento similar, a distância de S a v em T é pelo menos 2^{i+1}. Como numa árvore só há um caminho entre cada par de vértices e S é o vértice de nível mais baixo que contém u e v, o caminho de u a v passa por S, logo $d_T(u,v) \geq 2 \cdot 2^{i+1} = 2^{i+2}$. No entanto, $d_M(u,v) \leq 2^{i+2}$, pois $u, v \in S$ e $S \in D_{i+1}$, cujas partes têm diâmetro no máximo 2^{i+2}. Logo $d_T(u,v) \geq d_M(u,v)$. □

O fato a seguir será útil mais adiante.

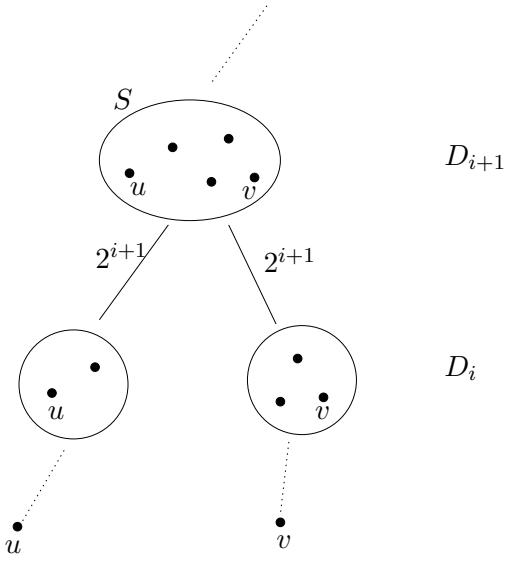


Figura 4.2: *Na árvore induzida por uma decomposição hierárquica em cortes, a distância entre vértices u e v que são separados no nível i é pelo menos 2^{i+2}.*

Fato 4.2: *Se uv é separada em T no nível i, tem-se que*

$$d_T(u,v) = 2 \cdot \sum_{j=1}^{i+1} 2^j = 4 \cdot \sum_{j=0}^{i} 2^j = 4 \cdot (2^{i+1} - 1) < 2^{i+3}.$$

Na seção a seguir, será mostrado como o algoritmo FRT obtém, para uma métrica M com menor distância não-nula pelo menos 1, uma decomposição hierárquica em cortes cuja métrica induzida possui baixa distorção esperada em relação a M.

4.2 O Algoritmo FRT

O algoritmo FRT, descrito a seguir, produz probabilisticamente uma decomposição hierárquica em cortes D de uma métrica M com menor distância não-nula pelo menos 1. Sendo T a árvore induzida por D, vale que

$$\mathbb{E}[d_T(u,v)] = \mathrm{O}(\lg n) \cdot d_M(u,v)$$

para todo $u, v \in V_M$, onde $n := |V_M|$. Lembre-se que T é uma 2-HST perfeita de base 2. Note que o algoritmo devolve D, mas T pode ser facilmente construída a partir de D. Além disso, como será visto mais adiante, para algumas aplicações é desejável que o algoritmo devolva T e não a métrica induzida por D.

A seguinte definição será utilizada nas seções a seguir.

Definição 4.3: *Sejam M uma métrica finita, π uma permutação de V_M e r um real positivo. O* **dono** *de um vértice $v \in V_M$ em relação a (π, r) é o vértice $\pi(j)$, onde j é o menor índice tal que $d_M(\pi(j), v) \leq r$.*

Note que todo vértice v de M tem um dono em relação a (π, r), pois $d_M(\pi(j), v) = 0 \leq r$ para $j = \pi^{-1}(v)$.

Inicialmente, será descrita a rotina auxiliar Refine(M, D, r, π). A rotina recebe uma métrica finita M, uma partição D de V_M, um número $r > 0$ e uma permutação π de V_M, e devolve uma

decomposição em r-cortes D' de M tal que dois vértices x e y estão na mesma parte de D' se e só se estão na mesma parte de D e têm o mesmo dono em relação a (π, r). (Em particular, D' é um refinamento de D.) Segue adiante o pseudocódigo da rotina. As invariantes são apresentadas junto com o pseudocódigo.

```
REFINE(M, D, r, π)
1  D' ← ∅;   X ← V_M;   n ← |V_M|
2  para j ← 1 até n faça
       ▷ Invariantes:
       ▷ 1. D' é uma partição V_M \ X tal que dois vértices em V_M \ X estão na mesma parte
       ▷    de D' se e só se estão na mesma parte de D e têm o mesmo dono em relação a (π, r);
       ▷ 2. X é o conjunto dos elementos de V_M cujo dono em relação a (π, r) é π(k) com k ≥ j.
3      para cada S ∈ D faça
4          S' ← {w ∈ S ∩ X : d_M(π(j), w) ≤ r}
               ▷ vértices de S cujo dono em relação a (π, r) é π(j)
5          se S' ≠ ∅ então
6              D' ← D' ∪ {S'}
7              X ← X \ S'
8  devolva D'
```

É fácil ver que as invariantes valem no início de cada iteração. As linhas 6 e 7 garantem a invariante 2, enquanto que as linhas 4, 6 e 7 garantem a invariante 1. Das invariantes 1 e 2, a correção do algoritmo é facilmente verificada. Note que, ao final do laço da linha 2, $j > n$ e, como todo vértice de M tem um dono, X é vazio, logo D' é uma partição de V_M. Como dois vértices na mesma parte de D' têm o mesmo dono (invariante 1), cada parte de D' tem raio no máximo r. Portanto, ao final do algoritmo tem-se que D' é uma decomposição em r-cortes de M com a propriedade de que dois vértices estão na mesma parte de D' se e só se estão na mesma parte de D e têm o mesmo dono em relação a (π, r).

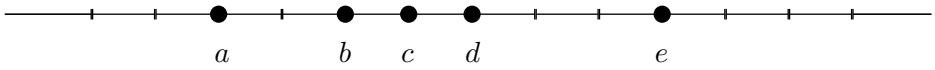
As linhas 4-7 podem ser executadas em tempo $O(|S|)$, se S for armazenado como uma lista ligada e X for representado por um vetor de bits. Como D é uma partição de V_M, tem-se que $\sum_{S \in D} |S| = |V_M|$, logo as linhas 3-7 podem ser executadas em tempo $O(n)$. Portanto, a rotina REFINE pode ser implementada de forma a consumir tempo $O(n^2)$.

Segue abaixo o pseudocódigo do algoritmo FRT.

```
FRT(M)
1  π ← permutação uniformemente aleatória de V_M
2  β_0 ← real uniformemente aleatório no intervalo [1/2, 1)
3  δ ← ⌈lg(2 · diam(M))⌉;   D_δ ← {V_M}
4  para i ← δ − 1 decrescendo até 0 faça
5      β_i ← 2^i · β_0
6      D_i ← REFINE(M, D_{i+1}, β_i, π)
7  devolva (D_0, ..., D_δ)
```

Note que, para $i = 0, \ldots, \delta - 1$, β_i é um real uniformemente aleatório no intervalo $[2^{i-1}, 2^i)$.

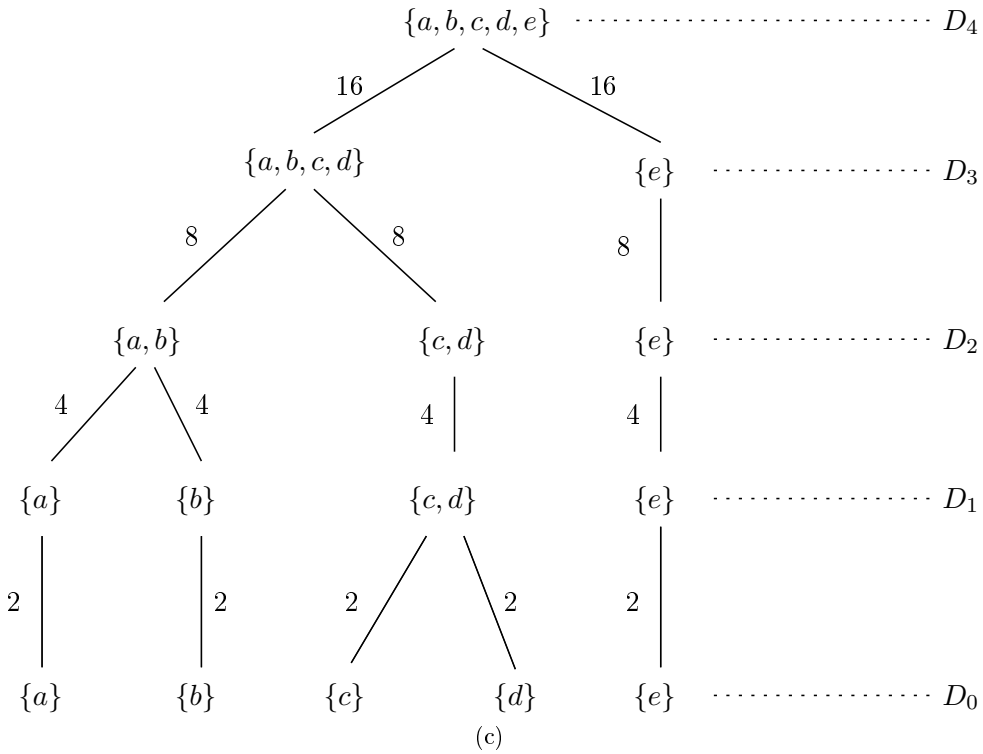
Para ilustrar o funcionamento do algoritmo, tome como exemplo (ver Figura 4.3(a)) a métrica M formada pelos pontos $\{a, b, c, d, e\}$ sobre a reta real e pela distância euclidiana. Suponha que os sorteios do algoritmo sejam $\pi = (a, e, c, b, d)$ e $\beta_0 = 5/8$. O algoritmo devolve então a decomposição $D = (D_0, D_1, D_2, D_3, D_4)$ descrita na Figura 4.3(b), onde na linha i e coluna v da tabela "Dono" está registrado o dono do vértice v em relação a (π, β_i) (ver Definição 4.3).



(a)

Nível i	β_i	Dono a	b	c	d	e	Decomposição
4							$D_4 = \{\{a, b, c, d, e\}\}$
3	5	a	a	a	a	e	$D_3 = \{\{a, b, c, d\}, \{e\}\}$
2	5/2	a	a	c	c	e	$D_2 = \{\{a, b\}, \{c, d\}, \{e\}\}$
1	5/4	a	c	c	c	e	$D_1 = \{\{a\}, \{b\}, \{c, d\}, \{e\}\}$
0	5/8	a	b	c	d	e	$D_0 = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}\}$

(b)



(c)

Figura 4.3: *(a) Pontos $\{a, b, c, d, e\}$ sobre a reta real. (b) Execução do algoritmo FRT com a métrica definida pelos pontos de (a) e pela distância euclidiana, com $\pi = (a, e, c, b, d)$ e com $\beta_0 = 5/8$. (c) Árvore induzida pela decomposição hierárquica em cortes de (b).*

No nível 4, todos os pontos estão na mesma parte. No nível 3, tem-se que $\beta_3 = 2^3 \cdot 5/8 = 5$. Como o primeiro ponto da permutação π é a, tem-se que a é o dono em relação a (π, β_3) dos pontos a, b, c e d, que estão a uma distância não mais que 5 de a. Já o ponto e está à distância 7 de a, logo tem como dono o próprio ponto e. O algoritmo cria então uma partição D_3 de V_M com duas partes, $\{a, b, c, d\}$ e $\{e\}$. O algoritmo prossegue de forma análoga até o nível 0, quando $\beta_0 < 1$ e portanto todas as partes são unitárias. A Figura 4.3(c) mostra a árvore T induzida por D. Note

que, no nível 1, apesar de os pontos b, c e d terem dono c, os pontos c e d estão numa parte diferente da do ponto b, visto que assim já estavam na decomposição do nível 2. Observe também que $d_T(3,5) = 2 \cdot (2+4+8) = 2 \cdot 14 = 28$, enquanto que $d_M(3,5) = 2$.

Note que a probabilidade de um vértice v ter dono w em relação a (π, β_i) é tanto maior quanto mais perto v está de w, e é tanto menor quanto menor for o nível i (e portanto β_i).

Da rotina REFINE tem-se que D_i é uma decomposição em β_i-cortes, com $\beta_i < 2^i$. Portanto, $(D_0, \ldots, D_\delta)$ é uma decomposição hierárquica em cortes.

Claramente, o algoritmo consome tempo $O(\delta \cdot n^2)$, o que é polinomial em $\delta = \lceil \lg(2 \cdot \mathrm{diam}(M)) \rceil$ e $n = |V_M|$. O mesmo pode ser modificado de forma a consumir tempo $O(n(n+\delta))$. Segundo Fakcharoenphol, Rao e Talwar [FRT04b], existe uma implementação do algoritmo que consome tempo $O(n^2)$. Observe que o tamanho da saída no pior caso é $\Theta(n^2)$, logo o algoritmo consome tempo linear em relação ao tamanho da saída. Resta mostrar que a distorção esperada é pequena.

4.2.1 Análise da Distorção

O teorema a seguir mostra que a métrica induzida pela decomposição hierárquica em cortes devolvida pelo algoritmo FRT tem distorção esperada logarítmica em relação à métrica de entrada. Devido ao Teorema 3.15, esse é o melhor resultado assintótico possível. O enunciado aqui exposto difere ligeiramente do apresentado em [FRT04b, FRT04a], mas o resultado é essencialmente o mesmo.

Teorema 4.4 (Fakcharoenphol, Rao e Talwar, 2004): *A métrica T induzida pela decomposição hierárquica em cortes devolvida pelo algoritmo* FRT *tendo como entrada uma métrica M com n vértices e menor distância não-nula pelo menos 1, é tal que*

$$\mathbb{E}[d_T(u,v)] \leq 16 H_n \cdot d_M(u,v) \quad \textit{para todo } u, v \in V_M.$$

Demonstração: Serão utilizadas as seguintes definições. Para $i = 0, 1, \ldots, \delta - 1$, diz-se que w é o **dono do vértice** u **no nível** i se w é o dono de u em relação a (π, β_i). Sejam u' o dono de u no nível i e v' o dono de v no nível i. O **dono da aresta** uv **no nível** i é o vértice $w := \arg\min_{\{u',v'\}} \{\pi^{-1}(u'), \pi^{-1}(v')\}$. Adicionalmente, w **corta** uv **no nível** i se o dono de u no nível i não é o mesmo que o de v. (Uma aresta pode ser cortada em vários níveis diferentes.) No exemplo da Figura 4.3, o vértice a é o dono da aresta ab no nível 2, mas não a corta; já a aresta bc tem dono a e é cortada por a no nível 2.

Como exatamente um vértice em V_M é dono de cada aresta em cada nível, no máximo um vértice em V_M corta cada aresta em cada nível. De fato, o dono da aresta no nível i é o único candidato a cortá-la, e só não a corta se as pontas da aresta tiverem o mesmo dono no nível i.

Seja uv uma aresta de M. Note que, se i é o nível em que uv é separada em T, então uv é cortada nesse nível, pois a rotina REFINE só separa vértices que têm donos diferentes. No entanto, é possível que uma aresta também seja cortada num nível diferente daquele em que ela é separada. Isso só acontece em níveis inferiores ao nível em que ela é separada (ou ela teria sido separada antes). Por exemplo, na Figura 4.3, a aresta bc é cortada nos níveis 2 e 0. Logo, do Fato 4.2,

$$d_T(u,v) \leq \max_{i=0,\ldots,\delta-1} \mathbf{1}[\exists w \in V_M : w \text{ corta } uv \text{ no nível } i] \cdot 2^{i+3},$$

onde, para uma expressão booleana Q, o valor $\mathbf{1}[Q]$ é definido como 1 se Q é verdadeira, e 0 caso contrário. Portanto,

$$d_T(u,v) \leq \sum_{i=0}^{\delta-1} \sum_{w \in V_M} \mathbf{1}[w \text{ corta } uv \text{ no nível } i] \cdot 2^{i+3}$$

e

$$\mathbb{E}[d_T(u,v)] \leq \sum_{i=0}^{\delta-1} \sum_{w \in V_M} \mathbb{P}[w \text{ corta } uv \text{ no nível } i] \cdot 2^{i+3}.$$

Seja $w_1, w_2, \ldots, w_n$ uma ordenação não-decrescente dos vértices de V_M em relação à distância em M até a aresta uv, ou seja, até o mais próximo entre u e v. Para cada $s \in [n]$ suponha, sem perda de generalidade, que $d_M(u,w_s) \leq d_M(v,w_s)$. Então

$$\begin{array}{lll} w_s \text{ corta } uv \text{ no nível } i & \text{se e só se} & d_M(u,w_s) \leq \beta_i < d_M(v,w_s) \\ & & \text{e } w_s \text{ é o dono de } uv \text{ no nível } i. \end{array}$$

Logo $\mathbb{P}[w_s \text{ corta } uv \text{ no nível } i] = \mathbb{P}\left[\begin{array}{c} d_M(u,w_s) \leq \beta_i < d_M(v,w_s) \wedge \\ w_s \text{ é o dono de } uv \text{ no nível } i \end{array}\right]$, que é igual a

$$\mathbb{P}\left[\begin{array}{c} w_s \text{ é o dono de } uv \text{ no nível } i \mid \\ d_M(u,w_s) \leq \beta_i < d_M(v,w_s) \end{array}\right] \cdot \mathbb{P}[d_M(u,w_s) \leq \beta_i < d_M(v,w_s)].$$

Segue o argumento de que a primeira probabilidade é no máximo $1/s$. Lembre-se que o dono de uv no nível i é o primeiro vértice da permutação π que está à distância no máximo β_i de u ou de v. Como $\min\{d_M(u,w_t), d_M(v,w_t)\} \leq \beta_i$ para $t \leq s$, cada w_t para $t = 1, \ldots, s$ pode ser o dono de uv no nível i. Portanto w_s é o dono de uv no nível i somente se estiver antes de w_t em π para todo $t < s$, o que ocorre com probabilidade $1/s$. Então,

$$\begin{aligned} \mathbb{E}[d_T(u,v)] &\leq \sum_{i=0}^{\delta-1} \sum_{s=1}^{n} \frac{1}{s} \cdot \mathbb{P}[d_M(u,w_s) \leq \beta_i < d_M(v,w_s)] \cdot 2^{i+3} \\ &= \sum_{s=1}^{n} \frac{1}{s} \cdot \sum_{i=0}^{\delta-1} \mathbb{P}[d_M(u,w_s) \leq \beta_i < d_M(v,w_s)] \cdot 2^{i+3}. \end{aligned}$$

Como β_i é uniformemente distribuído no intervalo $[2^{i-1}, 2^i)$, β_i está no intervalo $[d_M(u,w_s), d_M(v,w_s))$ com probabilidade

$$\frac{\ell\left([d_M(u,w_s), d_M(v,w_s)) \cap [2^{i-1}, 2^i)\right)}{\ell\left([2^{i-1}, 2^i)\right)} = \frac{\ell\left([d_M(u,w_s), d_M(v,w_s)) \cap [2^{i-1}, 2^i)\right)}{2^{i-1}},$$

onde $\ell(I)$ para um intervalo I denota o comprimento de I. Além disso, os intervalos $[2^{i-1}, 2^i)$ para $i = 0, \ldots, \delta - 1$ particionam o intervalo $[1/2, 2^\delta/2)$, o qual contém $[d_M(u,w_s), d_M(v,w_s))$, já que $\text{diam}(M) \leq 2^\delta/2$. Assim sendo,

$$\begin{aligned}
\sum_{i=0}^{\delta-1} \mathbb{P}[d_M(u,w_s) \le \beta_i < d_M(v,w_s)] \cdot 2^{i+3} &= \sum_{i=0}^{\delta-1} \frac{\ell\left([d_M(u,w_s), d_M(v,w_s)) \cap [2^{i-1}, 2^i)\right)}{2^{i-1}} \cdot 2^{i+3} \\
&= 16 \sum_{i=0}^{\delta-1} \ell\left([d_M(u,w_s), d_M(v,w_s)) \cap [2^{i-1}, 2^i)\right) \\
&= 16 \cdot \ell\left([d_M(u,w_s), d_M(v,w_s)) \cap [1/2, 2^\delta/2)\right) \\
&= 16 \cdot \ell\left([d_M(u,w_s), d_M(v,w_s))\right) \\
&= 16 \cdot (d_M(v,w_s) - d_M(u,w_s)) \\
&\le 16 \cdot d_M(u,v),
\end{aligned}$$

a última desigualdade seguindo da desigualdade triangular. Logo

$$\begin{aligned}
\mathbb{E}[d_T(u,v)] &\le \sum_{s=1}^{n} \frac{16}{s} \cdot d_M(u,v) \\
&= 16 H_n \cdot d_M(u,v),
\end{aligned}$$

para cada u, v em V_M, como queríamos demonstrar. □

4.2.2 Considerações Adicionais

Fakcharoenphol, Rao e Talwar [FRT04b] também descrevem como desaleatorizar o algoritmo FRT, para resolver uma variante mais fraca do problema que é suficiente para algumas aplicações. Especificamente, eles descrevem um algoritmo determinístico que, dados uma métrica finita M e um peso $p(u,v)$ para cada par u, v de V_M, devolve uma métrica arbórea T que domina M e é tal que

$$\sum_{u,v \in V_M} p(u,v) \cdot d_T(u,v) = \mathrm{O}(\lg n) \sum_{u,v \in V_M} p(u,v) \cdot d_M(u,v).$$

Assim, de certa forma, esse algoritmo busca minimizar a *distorção média* de T em relação a M, considerando uma média ponderada.

Como o Teorema 3.12 implica que qualquer aproximação determinística de uma métrica arbitrária tem distorção pelo menos linear, não é possível estender o resultado do Teorema 4.4 para um similar determinístico. A desaleatorização mencionada, no entanto, provê uma medida razoável da distorção da métrica arbórea devolvida, que pode ser útil na prática para algumas aplicações.

4.3 Aproximação de uma 2-HST por uma k-HST

Conforme a Definição 2.5, uma k-HST é uma árvore enraizada com custos positivos nas arestas, tal que arestas de mesmo nível têm o mesmo custo, e o custo nas arestas no caminho entre a raiz e uma folha decrescem por um fator de pelo menos k. Algoritmos de otimização em HSTs são bastante fáceis de projetar utilizando uma estratégia de divisão e conquista, dado que toda subárvore de uma k-HST também é uma k-HST.

O algoritmo FRT aproxima uma métrica finita arbitrária por uma 2-HST, mas alguns algoritmos (como o da Seção 7.2) requerem uma k-HST com valor de k maior que 2, ou mesmo valores de k

que não são constantes, mas que dependem da instância do problema. Dessa forma, é útil obter uma forma de aproximar uma 2-HST por uma k-HST para um real $k > 1$ qualquer. Nesta seção é apresentado um algoritmo que, dada uma 2-HST T e um real $k > 1$, devolve uma k-HST cuja métrica induzida pela distância entre suas folhas $\mathrm{O}(k/\lg k)$-aproxima probabilisticamente a métrica induzida pela distância entre as folhas de T. Além disso a HST devolvida é perfeita e tem base definida, isto é, os custos nas arestas no caminho da raiz a uma folha decrescem por exatamente o mesmo fator e todas as folhas têm o mesmo nível (ver Definições 2.6 e 2.7). Juntando esse algoritmo com o Teorema 4.4 e a transitividade da aproximação de métricas (Fato 3.10), o seguinte teorema segue.

Teorema 4.5: *Para todo real $k > 1$, uma métrica finita arbitrária com n vértices pode ser aproximada probabilisticamente pela métrica induzida pela distância entre as folhas de uma k-HST com distorção esperada* $\mathrm{O}(k \lg n/\lg k)$.

Esse resultado será utilizado na Seção 7.2.2 como sub-rotina de um outro algoritmo. A seção se baseia no trabalho de Bartal, Charikar e Raz [BCR01].

Supõe-se que a 2-HST T é tal que $d_T(u, v) \geq 2$ para quaisquer $u, v \in V_T$, sem perda de generalidade (vide redução do início do capítulo). Além disso, supõe-se que T tem a seguinte propriedade adicional: se h é o nível da raiz, então as arestas do caminho da raiz até uma folha qualquer têm custos $2^h, 2^{h-1}, \ldots, 2$. Note que o algoritmo FRT devolve uma 2-HST com essa propriedade, mas caso isso não seja verdade para T, faça o seguinte:

1. seja c o custo de uma aresta qualquer de T; arredonde c para $2^{\lceil \lg c \rceil}$ (lembrando que $c \geq 2$). Isso no máximo dobra o custo de cada aresta, logo a árvore resultante 2-aproxima T;

2. sejam e e f arestas consecutivas no caminho da raiz até uma folha de T, de custo 2^i e 2^j, respectivamente, com $i > j$. Se $i > j+1$, adicione um caminho entre e e f com $i-j-1$ arestas e custos $2^{i-1}, \ldots, 2^{j+1}$. Além disso, se as arestas de nível mais baixo têm custo 2^i com $i > 1$, conecte cada folha a um caminho com $i-1$ arestas e custos $2^{i-1}, \ldots, 2$. Como $\sum_{j=1}^{i-1} 2^j < 2^i$, cada caminho adicionado tem custo menor que o da aresta de nível imediatamente acima, logo a distância entre cada par de vértices de T no máximo dobra.

Assim, a árvore resultante é uma 2-HST que 4-aproxima T e satisfaz a propriedade desejada. Essa árvore resultante ainda será chamada de T no restante da seção.

O algoritmo para aproximar a 2-HST T por uma k-HST com $k > 1$ consiste no seguinte. Seja $\ell := \lceil \lg k \rceil$. Se $\ell = 1$, não há o que fazer e o algoritmo devolve a própria árvore T (que é uma 2-HST perfeita de base 2), pois uma x-HST é uma y-HST, para $x > y$. Suponha então que $\ell > 1$, e seja h o nível da raiz r de T. Primeiro, conecte r a um caminho de comprimento $\ell - 1$ e custos $2^{h+\ell-1}, \ldots, 2^{h+1}$ (o qual será denominado **caminho radical**), obtendo uma árvore $T' \supseteq T$. Em seguida, sorteie com probabilidade uniforme um número i em $\{1, \ldots, \ell\}$. Remova de T' todos os vértices internos em níveis diferentes de $i + t\ell$ para algum t inteiro, e conecte um vértice de nível j ao seu ancestral em T' no nível $j + \ell$ com uma aresta de custo $\sum_{s=j+1}^{j+\ell} 2^s$, e cada folha ao seu ancestral em T' no nível i com uma aresta de custo $\sum_{s=i-\ell+1}^{i} 2^s$. Note que, com exceção das arestas conectadas às folhas, o custo dessa aresta corresponde ao custo do caminho entre os dois vértices em T'. Como o caminho radical tem comprimento $\ell - 1$, garante-se que algum vértice desse caminho não foi removido, logo cada vértice interno de T tem um ancestral ℓ níveis acima em T'. O algoritmo devolve a árvore resultante, a qual será denotada por T''.

Para facilitar o entendimento do algoritmo, considere como exemplo a 2-HST da Figura 4.4(a), que tem as propriedades que o algoritmo supõe, e tome $k := 4$. Na Figura 4.4(b) é mostrada essa HST após a adição do caminho radical. Com $k = 4$, o sorteio do algoritmo pode ser $i = 1$ ou $i = 2$. Por isso, nessa figura os vértices de nível par não têm preenchimento, e os vértices de nível ímpar têm preenchimento preto. A Figura 4.4(c) mostra o resultado do algoritmo caso o sorteio seja $i = 1$; nesse caso são removidos os vértices sem preenchimento, os quais são exibidos com contorno pontilhado. Já a Figura 4.4(d) mostra o resultado do algoritmo caso o sorteio seja $i = 2$; nesse caso são removidos os vértices com preenchimento preto, os quais são exibidos com preenchimento cinza.

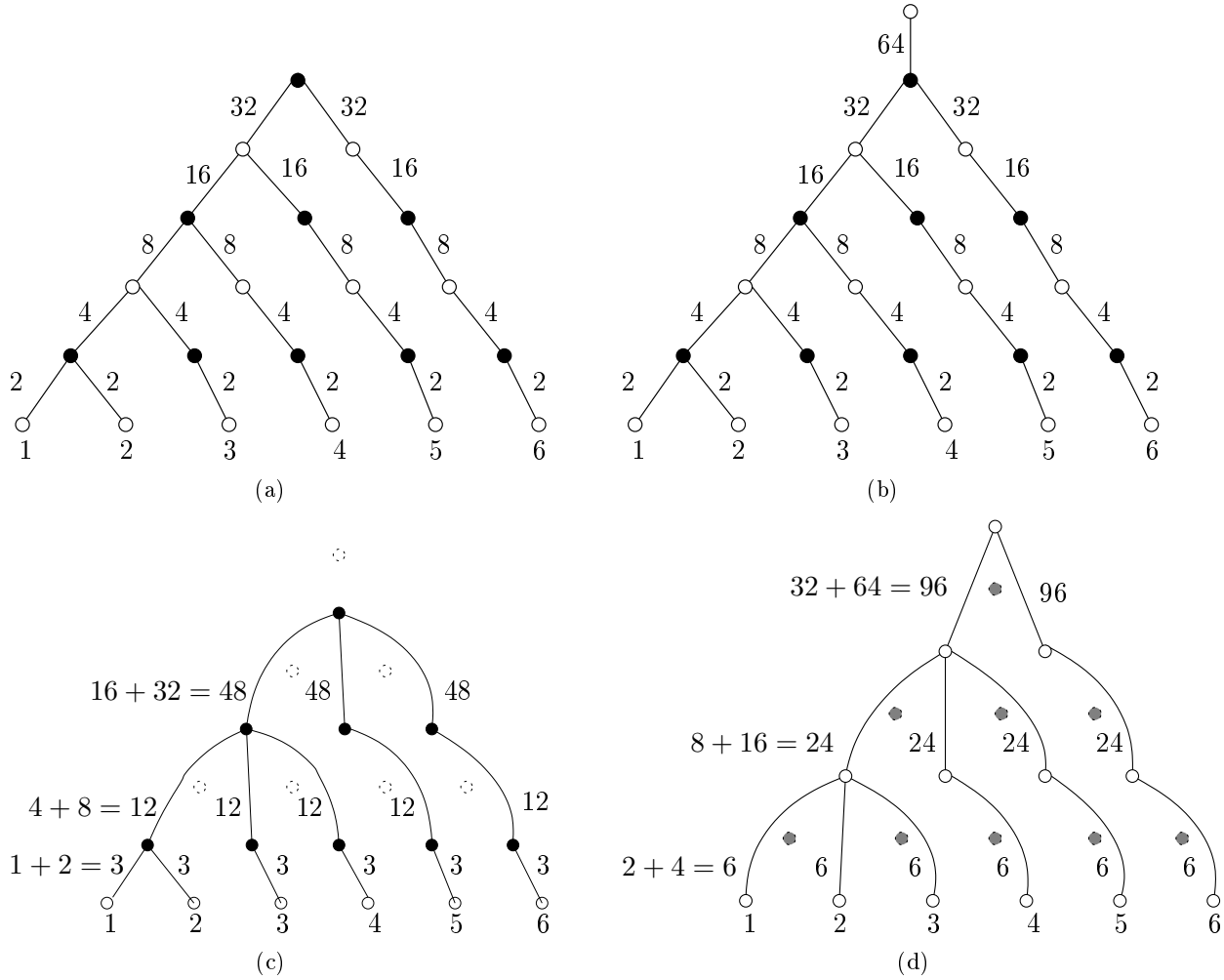


Figura 4.4: *(a) Uma 2-HST com conjunto de vértices {1, 2, 3, 4, 5, 6}. (b) Essa árvore após a adição do caminho radical. (c) A árvore resultante do algoritmo com $k = 4$ caso o sorteio seja $i = 1$. (d) A árvore resultante do algoritmo com $k = 4$ caso o sorteio seja $i = 2$.*

Será provado então que T'' é uma 2^ℓ-HST perfeita, e portanto uma k-HST, que $O(k/\lg k)$-aproxima probabilisticamente T' e portanto T, já que $d_T(u, v) = d_{T'}(u, v)$ para quaisquer folhas u e v de T. Note também que T'' tem base definida.

Para ver que T'' é uma 2^ℓ-HST perfeita, note primeiro que, claramente, toda aresta num mesmo nível de T'' tem o mesmo custo. Seja u um vértice interno de T'' e i seu nível em T'. Seja e a aresta de u a seu pai em T'' e f a aresta entre u e um de seus filhos em T''; é preciso mostrar que a razão entre o custo de e e o custo de f é 2^ℓ. Note que o custo de e é $\sum_{s=i+1}^{i+\ell} 2^s = 2^{i+1}(2^\ell - 1)$ e que o custo de f é $\sum_{s=i-\ell+1}^{i} 2^s = 2^{i-\ell+1}(2^\ell - 1)$. Logo a razão entre o custo de e e o custo de f é 2^ℓ.

Sejam então u e v duas folhas de T'. Para ver que $d_{T''}(u,v) \geq d_{T'}(u,v)$, sejam w e w' os ancestrais comuns de u e v de nível mais baixo em T' e T'', respectivamente. Note que $d_{T'}(u,v) = d_{T'}(u,w) + d_{T'}(w,v) = 2d_{T'}(u,w)$ e que $d_{T''}(u,v) = d_{T''}(u,w') + d_{T''}(w',v) = 2d_{T''}(u,w')$. Além disso, vale que $d_{T'}(u,w') \leq d_{T''}(u,w')$, lembrando que w' é um vértice tanto de T' quanto de T''. Como o nível de w' em T' é maior ou igual ao nível de w em T', vale que $d_{T'}(u,w) \leq d_{T'}(u,w') \leq d_{T''}(u,w')$, do que a afirmação segue.

Resta mostrar que $\mathbb{E}[d_{T''}(u,v)] = \mathrm{O}(k/\lg k) \cdot d_{T'}(u,v)$. Sejam j e j' os níveis em T' de w e w', respectivamente. Note que $d_{T'}(u,v) = 2\sum_{s=1}^{j} 2^s = 2^{j+2} - 4 > 2^j$ e que j' é um nível diferente dentre $\{j, \ldots, j+\ell-1\}$ para cada escolha do valor $i \in \{1, \ldots, \ell\}$ sorteado pelo algoritmo. Para um j' fixo,

$$\begin{aligned} d_{T''}(u,v) &= 2\sum_{s=i-l+1}^{0} 2^s + d_{T'}(u,v) + 2\sum_{s=j+1}^{j'} 2^s \\ &< 4 + d_{T'}(u,v) + 2\sum_{s=j+1}^{j'} 2^s, \end{aligned}$$

logo

$$\begin{aligned} \mathbb{E}[d_{T''}(u,v)] &< \frac{1}{\ell}\sum_{j'=j}^{j+\ell-1}\left(d_{T'}(u,v) + 2\sum_{s=j+1}^{j'} 2^s\right) + 4 \\ &= \frac{1}{\ell}\sum_{i=1}^{\ell}\left(d_{T'}(u,v) + 2\sum_{s=j+1}^{j+i-1} 2^s\right) + 4 \\ &= \frac{1}{\ell}\sum_{i=1}^{\ell}\left(d_{T'}(u,v) + 2\cdot 2^{j+1}\sum_{s=0}^{i-2} 2^s\right) + 4 \\ &< \frac{1}{\ell}\sum_{i=1}^{\ell}\left(d_{T'}(u,v) + 4\cdot d_{T'}(u,v)\cdot(2^{i-1}-1)\right) + 4 \\ &= \frac{1}{\ell}\sum_{i=1}^{\ell} d_{T'}(u,v)\cdot(2^{i+1}-4+1) + 4 \\ &< \frac{d_{T'}(u,v)}{\ell}\sum_{i=1}^{\ell} 2^{i+1} + 4 \\ &< \frac{d_{T'}(u,v)}{\ell}\cdot 2^{\ell+2} + 4 \\ &\leq \frac{8k}{\lg k}\cdot d_{T'}(u,v) + 4 \\ &\leq \left(\frac{8k}{\lg k} + 2\right)\cdot d_{T'}(u,v), \end{aligned}$$

já que $\ell \geq \lg k$, $2^\ell \leq 2k$ e $d_{T'}(u,v) \geq 2$.

Parte II

Emparelhamento Mínimo Bipartido Online

Capítulo 5

Introdução

Este capítulo trata de uma versão online do caso bipartido completo do problema do emparelhamento perfeito de custo mínimo (Problema MinEP, Seção 2.1). Lembrando que o problema MinEP consiste em, dado um grafo G com custos não-negativos associados às arestas, encontrar um emparelhamento perfeito de custo total mínimo de G.

Na versão online do MinEP bipartido, um grafo (R, S)-bipartido completo, com $|R| \leq |S|$, é parcialmente disponibilizado de maneira online. Inicialmente, apenas a parte S é dada; os elementos de S são denominados *servidores*. Em cada etapa, um vértice r de R, denominado uma *requisição*, é disponibilizado juntamente com o custo das arestas de r a S. Esses custos são tais que servidores e requisições correspondem a pontos de uma métrica, possivelmente com vários servidores e requisições sobre um mesmo ponto. Isto é, existe uma métrica M e uma função $\phi : R \cup S \to V_M$ tal que o custo da aresta rs é $c(r, s) = d_M(\phi(r), \phi(s))$ para todo $r \in R$ e $s \in S$. Um algoritmo online para essa versão do problema deve, ao final de cada etapa, designar um servidor não-emparelhado de S para r, estendendo o emparelhamento da etapa anterior para um emparelhamento que cubra r. Ou seja, ao final de cada etapa, o algoritmo tem nas mãos um emparelhamento que cobre a parte de R que já foi disponibilizada. O objetivo é produzir dessa maneira um emparelhamento com o menor custo possível.

Segue uma definição formal do problema.

Problema MinEPBOnline (R, S, c): *Dados uma sequência $R = (r_1, \ldots, r_n)$ e um conjunto S, com $|S| \geq n$, e uma função $c : R \times S \to \mathbb{R}_+$ tal que existem uma métrica M e uma função $\phi : R \cup S \to V_M$ para a qual $c(r, s) = d_M(\phi(r), \phi(s))$ para todo $r \in R$ e $s \in S$, encontrar $\{r_1 s_1, \ldots, r_n s_n\}$, onde $s_i \in S$ e $s_i \neq s_j$ para $i \neq j$, de forma a minimizar $\sum_{i=1}^{n} c(r_i, s_i)$, sendo que s_i deve ser calculado antes de s_{i+1} e sem a informação sobre quem é $(r_{i+1}, \ldots, r_n)$.*

Como exemplo, veja a Figura 5.1, onde é mostrada uma instância do problema com 3 servidores. Os custos correspondem à distância entre pontos na reta. Na Figura 5.1(a), chega uma primeira requisição às distâncias 1, 2 e 3 dos servidores, e a mesma é emparelhada com o servidor à distância 1. Na Figura 5.1(b), a segunda requisição está às distâncias 2, 3 e 4 dos servidores, e é escolhido o servidor à distância 3. Na Figura 5.1(c), a terceira requisição está às distâncias 3, 4 e 5 dos servidores, mas apenas o servidor à distância 5 está disponível. A solução final é mostrada na Figura 5.1(d), e consiste num emparelhamento de custo 9. Observe que toda solução para essa instância tem custo 9,

logo essa é uma solução ótima offline.

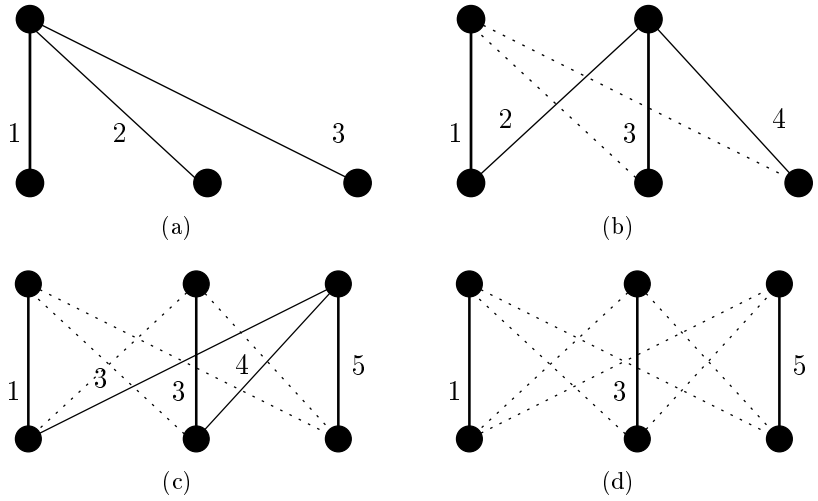


Figura 5.1: *Exemplo de instância do problema do emparelhamento mínimo bipartido online, com $n = 3$ requisições. Os custos correspondem à distância entre pontos na reta. Inicialmente, são dados apenas os servidores (vértices inferiores). (a) A primeira requisição (vértice superior) é atendida com custo 1. (b) A segunda requisição é atendida com custo 3. Embora exista um servidor à distância 2 da mesma, esse servidor já está ocupado. (c) A terceira requisição é atendida com custo 5. (d) Uma solução para essa instância, com custo total 9.*

Algumas soluções para esse problema são apresentadas nos Capítulos 6 e 7. Nas seções a seguir são apresentados o histórico do problema e alguns limites inferiores para o fator de competitividade de algoritmos que o resolvem.

5.1 Histórico

Esse problema foi estudado inicialmente por Karp, Vazirani e Vazirani [KVV90], que mostraram que não há garantia de solução para o problema para grafos bipartidos incompletos (isto é, quando algum servidor não é capaz de atender determinadas requisições): é possível garantir apenas que $n/2$ das requisições são atendidas por um algoritmo online. Além disso, se as distâncias não forem tais que os servidores e as requisições correspondam a pontos de uma métrica, não é possível obter um algoritmo cujo fator de competitividade seja independente das distâncias [KMV94].

Kalyanasundaram e Pruhs [KP93] e Khuller, Mitchell e Vazirani [KMV94] obtiveram independentemente um algoritmo determinístico $(2n - 1)$-competitivo para métricas, o *algoritmo da permutação*, e mostraram que esse é o melhor resultado determinístico possível. Esse algoritmo é descrito na Seção 6.2. O trabalho de Kalyanasundaram e Pruhs também cobre o problema do emparelhamento máximo online (emparelhamento máximo de custo máximo, para um grafo bipartido completo cujos custos nas arestas correspondem a uma métrica) [KP93].

Se a métrica do problema é dada por pontos na reta, Kalyanasundaram e Pruhs conjecturaram [KP98, p. 274] que o *Algoritmo da Função Trabalho* (WFA) de Koutsoupias e Papadimitriou [KP95] (inicialmente projetado para um problema conhecido como o problema dos k servidores) obteria um fator de competitividade constante para o emparelhamento mínimo online. Koutsoupias e Nanavati [KN04] mostraram que o fator de competitividade do WFA na verdade é $\Omega(\lg n)$ e $O(n)$. Fuchs, Hochstättler e Kern [FHK05] mostraram um algoritmo para o problema do emparelhamento mínimo bipartido online com fator de competitividade constante para a reta.

Kalyanasundaram e Pruhs [KP98, p. 274] afirmam que, para a métrica estrela (a métrica induzida pelas distâncias no $K_{1,k}$; ver Definição 3.5), o melhor fator de competitividade que pode ser obtido com um algoritmo probabilístico é pelo menos $2H_n - 1$, onde H_n é o n-ésimo número harmônico. Esse resultado é apresentado na seção a seguir. Adicionalmente, os autores conjecturam a existência de um algoritmo probabilístico com fator de competitividade $O(\lg n)$ para métricas arbitrárias.

Meyerson, Nanavati e Poplawski [MNP06] e Csaba e Pluhár [CP08] mostraram independentemente algoritmos probabilísticos gulosos simples cujo fator de competitividade é $\Theta(\lg n)$ para métricas uniformes e para uma variante do problema para $\Theta(\lg n)$-HSTs, o que pode ser generalizado para $O(\lg^3 n / \lg\lg n)$ para métricas arbitrárias, utilizando aproximação de métricas. Esse resultado é descrito em detalhes no Capítulo 7.

O melhor algoritmo probabilístico conhecido, no entanto, é o descrito por Bansal *et al.* [BBGN07], com fator de competitividade $O(\lg^2 n)$. Esse algoritmo também se baseia na aproximação de métricas por métricas arbóreas, e consiste num algoritmo $O(\lg n)$-competitivo para 2-HSTs em conjunto com o algoritmo FRT (ver Capítulo 4).

Aplicações desse problema incluem diversos problemas de balanceamento de carga online [KP98] e leilões de anúncios em sítios de busca na Web [Mar06].

5.2 Limites Inferiores

Inicialmente, serão demonstrados dois limites inferiores para algoritmos determinísticos para o problema do emparelhamento mínimo bipartido online.

Teorema 5.1 (Meyerson, Nanavati e Poplawski, 2006 [MNP06, Teorema 2.1]): *Qualquer algoritmo determinístico para o problema do emparelhamento mínimo bipartido online na métrica t-uniforme, onde t é uma constante maior que zero, tem fator de competitividade pelo menos n, onde n é o número de requisições.*

Demonstração: Considere um conjunto de n servidores em pontos distintos da métrica, e seja A um algoritmo determinístico para o problema do emparelhamento mínimo bipartido online. Inicialmente, tome uma requisição r_0 em um ponto da métrica à distância t de todos os servidores. Para $i = 1, \ldots, n-1$, tome uma requisição r_i à distância zero do servidor que a estratégia do algoritmo A determina que atenda à requisição r_{i-1}, o qual será chamado de s_{i-1}. Seja s_{n-1} o servidor que atende à última requisição. Note que o algoritmo A paga um custo nt, enquanto que o custo ótimo offline é t, pois pode-se atribuir o servidor s_{i-1} para a requisição r_i, para $i = 1, \ldots, n-1$, e o servidor s_{n-1} para a requisição r_0. Portanto o teorema segue. □

Teorema 5.2 (Khuller, Mitchell e Vazirani, 1994 [KMV94, Teorema 2.4]): *Qualquer algoritmo determinístico para o problema do emparelhamento mínimo bipartido online numa métrica arbitrária tem fator de competitividade pelo menos $2n - 1$, onde n é o número de requisições.*

Demonstração: Seja A um algoritmo determinístico para o problema do emparelhamento mínimo bipartido online. Considere a métrica que consiste de n pontos dois a dois à distância 1, e um ponto extra p_0 à distância $1/2$ de cada um desses n pontos. Os servidores estão sobre os n

pontos equidistantes, e a primeira requisição r_0 é no ponto p_0. Para $i = 1, \ldots, n-1$, tome uma requisição r_i à distância zero do servidor que a estratégia do algoritmo A determina que atenda à requisição r_{i-1}, o qual será chamado de s_{i-1}. Seja s_{n-1} o servidor que atende à última requisição. Note que o algoritmo paga um custo $1/2 + (n-1)$, enquanto que o custo ótimo offline é $1/2$, pois é possível atribuir o servidor s_{i-1} para a requisição r_i, para $i = 1, \ldots, n-1$, e o servidor s_{n-1} para a requisição r_0. O fator de competitividade é, portanto, pelo menos $\frac{1/2+n-1}{1/2} = 2n-1$. □

Note que, na prova do Teorema 5.2, não é possível obter um limite mais forte trocando $1/2$ por um valor menor, pois não seria satisfeita a propriedade de desigualdade triangular da definição de métrica.

Os seguintes limites inferiores se aplicam a algoritmos probabilísticos para o problema do emparelhamento mínimo online bipartido.

Teorema 5.3 (Meyerson, Nanavati e Poplawski, 2006 [MNP06, Teorema 2.3]): *Qualquer algoritmo probabilístico para o problema do emparelhamento mínimo bipartido online na métrica t-uniforme, onde t é uma constante maior que zero, tem fator de competitividade esperado pelo menos H_n, onde H_n é o n-ésimo número harmônico e n é o número de requisições.*

Demonstração: Seja A um algoritmo probabilístico para o problema do emparelhamento mínimo bipartido online. Tome n servidores em pontos distintos da métrica e n requisições como segue. A primeira requisição r_0 é num ponto distinto dos n servidores. Para $i = 1, \ldots, n-1$, tome uma requisição r_i num ponto que ainda não tenha nenhuma requisição e que esteja à distância zero do servidor que, segundo a estratégia do algoritmo A, tem maior probabilidade de já estar ocupado. Como o ponto é escolhido dentre $n - i + 1$ possíveis pontos e, para uma distribuição de probabilidade qualquer, a maior probabilidade é pelo menos a probabilidade na distribuição uniforme, essa probabilidade é pelo menos $\frac{1}{n-i+1}$. O custo ótimo offline é t, enquanto que o custo esperado da solução devolvida pelo algoritmo A é t mais t vezes a soma das probabilidades das requisições terem seus servidores correspondentes ocupados. Portanto, a solução devolvida pelo algoritmo A tem um custo esperado de pelo menos

$$t + t\sum_{i=1}^{n-1} \frac{1}{n-i+1} = t + t\sum_{i=2}^{n} \frac{1}{i} = tH_n,$$

do que o teorema segue. □

Teorema 5.4 (Kalyanasundaram e Pruhs, 1998): *Qualquer algoritmo probabilístico para o problema do emparelhamento mínimo bipartido online numa métrica arbitrária tem fator de competitividade esperado pelo menos $2H_n - 1$, onde n é o número de requisições e H_n é o n-ésimo número harmônico.*

Demonstração: Seja A um algoritmo probabilístico para o problema do emparelhamento mínimo bipartido online numa métrica arbitrária. Considere a mesma métrica da prova do Teorema 5.2, isto é, com n pontos dois a dois à distância 1 e um ponto extra p_0 à distância $1/2$ de cada um desses n pontos. Novamente, os servidores estão sobre os n pontos equidistantes, e a primeira requisição r_0 é no ponto p_0. Para $i = 1, \ldots, n-1$, tome uma requisição r_i num ponto que ainda não tenha nenhuma

requisição e que esteja à distância zero do servidor que, segundo a estratégia do algoritmo A, tem maior probabilidade de já estar ocupado. Por argumento semelhante ao da prova do Teorema 5.3, essa probabilidade é pelo menos $\frac{1}{n-i+1}$. O custo ótimo offline é $1/2$, enquanto que o custo esperado da solução devolvida pelo algoritmo A é $1/2$ mais a soma das probabilidades das requisições terem seus servidores correspondentes ocupados. Portanto, a solução devolvida pelo algoritmo A tem um custo esperado de pelo menos

$$\frac{1}{2} + \sum_{i=1}^{n-1} \frac{1}{n-i+1} = -\frac{1}{2} + 1 + \sum_{i=2}^{n} \frac{1}{i} = H_n - \frac{1}{2},$$

e o algoritmo A tem fator de competitividade esperado pelo menos

$$\frac{H_n - 1/2}{1/2} = 2H_n - 1.$$

□

Kalyanasundaram e Pruhs [KP98] conjecturaram que existe um algoritmo probabilístico que alcança assintoticamente esse limite inferior para métricas arbitrárias.

Capítulo 6

Algoritmos Determinísticos

Este capítulo apresenta dois algoritmos determinísticos para o problema MinEPBOnline, a saber, o algoritmo guloso (Seção 6.1) e o algoritmo da permutação (Seção 6.2). Lembrando que uma instância desse problema consiste numa tripla (R, S, c), onde $R = (r_1, \ldots, r_n)$ é uma sequência, S é um conjunto com pelo menos n elementos, e c é uma função custo $c : R \times S \to \mathbb{R}_+$ tal que existe uma métrica M e uma função $\phi : R \cup S \to V_M$ para a qual $c(r, s) = d_M(\phi(r), \phi(s))$ para todo $r \in R$ e $s \in S$. O objetivo é encontrar um emparelhamento de custo total mínimo no grafo $G := (R \cup S, R \times S)$ que cubra R, lembrando que R e c são dados de forma online.

6.1 O Algoritmo Guloso

O algoritmo guloso consiste em, para cada requisição que chega, atribuir o servidor mais próximo que ainda não está ocupado. O pseudocódigo do algoritmo é exibido a seguir.

```
Guloso(R, S, c)
1  n ← |S|
2  L ← S        ▷ L: conjunto dos servidores que ainda não foram ocupados
3  para i ← 1 até n faça
4      s_i ← arg min_{s∈L} c(r_i, s)
5      L ← L \ {s_i}
6  devolva {r_1 s_1, ..., r_n s_n}
```

Observe que o algoritmo é online, pois até a iteração i se usa apenas a informação que envolve $r_1, \ldots, r_i$. Embora esse algoritmo seja bom para métricas uniformes, ele possui um fator de competitividade exponencial no número de requisições em métricas arbitrárias.

Teorema 6.1 (Meyerson, Nanavati e Poplawski, 2006 [MNP06, Teorema 2.1]): *O algoritmo guloso para o problema do emparelhamento mínimo bipartido online é n-competitivo numa métrica uniforme, onde n é o número de requisições.*

Demonstração: Note que, se o custo ótimo offline é zero, então toda requisição está à distância zero de um servidor que pode atendê-la. O algoritmo guloso, neste caso, atribui um servidor à distância zero para cada requisição, logo também paga custo zero.

Seja $t > 0$ tal que a métrica em questão é t-uniforme. Se o custo ótimo offline não é zero, então é pelo menos t. O algoritmo guloso, por sua vez (como qualquer outro algoritmo, determinístico ou probabilístico), paga no máximo t por cada requisição. Logo o fator de competitividade é no máximo n; pelo Teorema 5.1, é exatamente n. □

Teorema 6.2 (Meyerson, Nanavati e Poplawski, 2006 [MNP06, Teorema 2.2]): *O algoritmo guloso para o problema do emparelhamento mínimo bipartido online é $\Omega(2^n)$-competitivo para métricas arbitrárias, onde n é o número de requisições.*

Demonstração: Considere a métrica definida por pontos sobre a reta e a distância euclidiana. Tome servidores $s_0 := 0$ e $s_i := 2^i - (2^i - 1)\epsilon$, com $0 < \epsilon < 1$, para $i = 1, \ldots, n-1$. Tome então requisições $r_0 := 1$ e $r_i := s_i$, para $i = 1, \ldots, n-1$. Na Figura 6.1, é mostrado um exemplo com $n = 4$. Note que o custo ótimo offline é 1, pois é possível atribuir s_i para r_i, para $i = 0, \ldots, n-1$.

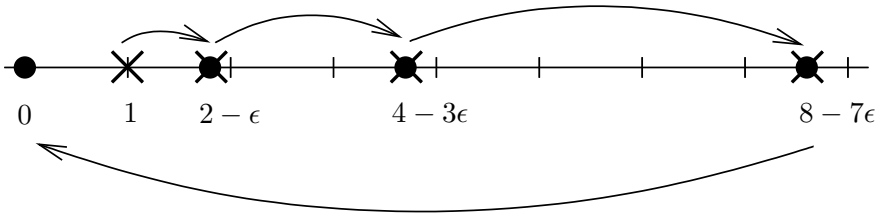


Figura 6.1: *Instância descrita no Teorema 6.2, para $n = 4$. Um ponto indica um servidor, e um '$\times$' uma requisição. As setas indicam a solução do algoritmo guloso, e são na direção da requisição para o servidor.*

O algoritmo guloso, por sua vez, procede da seguinte maneira. Ao se deparar com r_0, o algoritmo escolhe o servidor s_1, já que $c(r_0, s_1) = 2-\epsilon-1 = 1-\epsilon < c(r_0, s_0) = 1$, e paga $1-\epsilon$. Nas próximas $n-2$ requisições, a seguinte invariante é mantida: os servidores mais próximos disponíveis são s_0 de um lado e s_{i+1} do outro (ver Figura 6.1), já que os servidores $s_1, \ldots, s_i$ foram atribuídos às requisições anteriores. Como $c(r_i, s_0) = 2^i - 2^i\epsilon + \epsilon > c(r_i, s_{i+1}) = 2^{i+1} - 2^{i+1}\epsilon + \epsilon - (2^i - 2^i\epsilon + \epsilon) = 2^i - 2^i\epsilon$, o algoritmo escolhe o servidor s_{i+1}, pagando custo $2^i - 2^i\epsilon$ e deixando o servidor s_{i+1} indisponível para as requisições seguintes. No final, resta apenas o servidor s_0 para a requisição r_{n-1}, com custo $2^{n-1} - 2^{n-1}\epsilon + \epsilon$. No total, o algoritmo paga

$$\sum_{i=0}^{n-1}(2^i - 2^i\epsilon) + \epsilon = (1-\epsilon)(2^n - 1) + \epsilon = \Omega(2^n),$$

do que o teorema segue. □

Embora o algoritmo guloso não seja tão bom, o algoritmo da permutação [KP93, KMV94], que será apresentado na seção seguinte, alcança o limite do Teorema 5.2 para métricas arbitrárias.

6.2 O Algoritmo da Permutação

Nesta seção é apresentado o algoritmo da permutação [KP93, KMV94], cujo fator de competitividade é, segundo o Teorema 5.2, o melhor possível determinístico para uma métrica arbitrária.

Se A e B são conjuntos, a notação $A \oplus B$ denota a **diferença simétrica** de A e B, que consiste em $(A \setminus B) \cup (B \setminus A)$. Uma estratégia comum em diversos algoritmos para problemas de emparelhamento é considerar, para dois emparelhamentos M e N de um grafo G, o grafo $(V_G, M \oplus N)$. Observe que

todo vértice desse grafo tem grau 0, 1 ou 2. Assim, uma propriedade importante é que ele consiste apenas em vértices isolados, circuitos (de comprimento par) e caminhos, e esses circuitos e caminhos alternam entre arestas de M e de N.

Seja G um grafo (R, S)-bipartido completo com custos positivos nas arestas, tal que $n := |R| \leq |S|$, e seja $(r_1, \ldots, r_n)$ uma permutação de R. Seja $N_0 := \emptyset$ e, para $i = 1, \ldots, n$, seja N_i um emparelhamento máximo de custo mínimo de $G[S \cup \{r_1, \ldots, r_i\}]$. Se houver mais de um tal emparelhamento, tome como N_i aquele que difere no menor número de arestas de N_{i-1}. Observe que N_i cobre $\{r_1, \ldots, r_i\}$, uma vez que G, e portanto $G[S \cup \{r_1, \ldots, r_i\}]$, são grafos bipartidos completos.

O algoritmo da permutação se baseia fortemente no lema a seguir.

Lema 6.3 (Khuller, Mitchell e Vazirani, 1994 [KMV94, Lema 2.1]): *Para $i = 1, \ldots, n$, $(V_G, N_i \oplus N_{i-1})$ consiste em vértices isolados e exatamente um caminho. Ademais, esse caminho vai de r_i a um vértice de S que não é coberto em N_{i-1}.*

Demonstração: Como não há aresta incidente a r_i em N_{i-1}, tem-se que r_i incide em exatamente uma aresta de $N_i \oplus N_{i-1}$. Logo r_i é extremidade de um caminho em $(V_G, N_i \oplus N_{i-1})$.

Além disso, para $j = 1, \ldots, i-1$, o vértice r_j é coberto por N_i e N_{i-1}, logo tem grau 0 ou 2 em $(V_G, N_i \oplus N_{i-1})$. Portanto r_j não é extremidade de nenhum caminho de $(V_G, N_i \oplus N_{i-1})$. Isso significa que a outra extremidade do caminho que contém r_i está em S. Esse vértice será denotado por s_i. Claramente, o vértice s_i não é coberto por N_{i-1}, já que a aresta que incide em r_i está em N_i e as arestas do caminho alternam entre arestas de N_i e N_{i-1}.

Resta mostrar que não existe nada mais em $N_i \oplus N_{i-1}$. Se existir um circuito ou um caminho, tal caminho teria que começar e terminar em vértices de S. Em ambos os casos seria possível, trocando as arestas, melhorar o custo de N_i ou de N_{i-1} ou diminuir a diferença entre o número de arestas de N_i e de N_{i-1} (contradição). Logo o lema segue. □

A seguir é exibido o pseudocódigo do algoritmo da permutação. O algoritmo utiliza como subrotina um algoritmo para o problema do emparelhamento máximo de custo mínimo, a fim de calcular N_i. Como $(V_G, N_i \oplus N_{i-1})$ possui apenas um caminho, N_i pode ser calculado eficientemente a partir de N_{i-1} executando uma iteração do algoritmo húngaro. Uma implementação desse passo é descrita em [CCPS97, Seção 5.3, página 147], e aqui será denotado por AUMENTAR(G, c, M), sendo M um emparelhamento de G a partir do qual será buscado um emparelhamento com uma aresta a mais.

PERMUTAÇÃO(R, S, c)
1 $n \leftarrow |S|$
2 $M_0 \leftarrow N_0 \leftarrow \emptyset$
3 para $i \leftarrow 1$ até n faça
4 $\quad N_i \leftarrow$ AUMENTAR$(G[S \cup \{r_1, \ldots, r_i\}], c, N_{i-1})$
5 $\quad s_i \leftarrow$ o vértice de S que tem grau um em $(V_G, N_i \oplus N_{i-1})$
6 $\quad M_i \leftarrow M_{i-1} \cup \{r_i s_i\}$
7 devolva M_n

Note que se trata de um algoritmo online. Para ver que o algoritmo faz sentido, note, do Lema 6.3, que s_i está bem definido. Para mostrar que M_i é um emparelhamento de G, é preciso provar que s_i está livre em M_{i-1}, isto é, que nenhuma aresta em M_{i-1} incide em s_i. Isso segue do lema a seguir. Se M é um emparelhamento, denote por $S[M]$ o conjunto dos vértices de S que são pontas de arestas em M.

Lema 6.4 (Khuller, Mitchell e Vazirani, 1994 [KMV94, Lema 2.2]): *Para $0 \leq i \leq n$, vale que $S[M_i] = S[N_i]$ e que M_i é um emparelhamento.*

Demonstração: A prova é por indução em i. A afirmação vale trivialmente para $i = 0$, pois $M_0 = N_0 = \emptyset$. Suponha, por hipótese de indução, que $S[M_{i-1}] = S[N_{i-1}]$. Do Lema 6.3, $S[N_i] \setminus S[N_{i-1}] = \{s_i\}$, logo $s_i \notin S[N_{i-1}] = S[M_{i-1}]$ e M_i é um emparelhamento. O algoritmo Permutação faz $S[M_i] \leftarrow S[M_{i-1}] \cup \{s_i\}$, então $S[M_i] = S[N_i]$. □

O teorema a seguir conclui a análise do algoritmo.

Teorema 6.5 (Khuller, Mitchell e Vazirani, 1994 [KMV94, Teorema 2.3]): *O algoritmo* Permutação *é $(2n-1)$-competitivo.*

Demonstração: Do Teorema 5.2, o fator de competitividade do algoritmo é pelo menos $2n - 1$. Resta mostrar, então, que $c(M_n) \leq (2n-1) \cdot c(N_n)$.

Primeiro, note que, como os custos das arestas em G são não-negativos, vale que

$$c(N_0) \leq c(N_1) \leq \ldots \leq c(N_n);$$

se não fosse assim, seria contradita a minimalidade do custo de cada emparelhamento. Para $i = 1, \ldots, n$, seja $e_i := r_i s_i$. É claro que que $c(e_1) = c(N_1) \leq c(N_n)$.

Uma consequência de o custo ser tal que os elementos em R e S correspondem a pontos de uma métrica é que, devido à desigualdade triangular, para toda aresta rs de G, o custo de rs é menor ou igual ao custo de qualquer caminho de r a s em G. Assim, para $i \geq 2$,

$$c(e_i) \leq c(N_i \oplus N_{i-1}) \leq c(N_i) + c(N_{i-1}) \leq 2 \cdot c(N_n).$$

Logo

$$c(M_n) = \sum_{i=1}^{n} c(e_i) \leq c(N_n) + (n-1) \cdot 2 \cdot c(N_n) = (2n-1) \cdot c(N_n).$$

□

Capítulo 7

Algoritmos Probabilísticos

Neste capítulo são apresentados algoritmos probabilísticos para o problema do emparelhamento mínimo bipartido online. Na Seção 7.1, é mostrado como o algoritmo guloso probabilístico se comporta numa métrica uniforme: o fator de competitividade esperado é H_n, onde n é o número de requisições. Na Seção 7.2, é apresentada uma variante desse algoritmo para instâncias em que a métrica é dada por uma HST, com fator de competitividade $O(\lg n)$ para uma $\Omega(\lg n)$-HST. Na Seção 7.2.2, mostra-se como utilizar o algoritmo da Seção 7.2 e a técnica de aproximação de métricas do Capítulo 4 para obter um algoritmo probabilístico para uma métrica arbitrária com fator de competitividade esperado $O(\lg^3 n / \lg\lg n)$.

A fim de simplificar a descrição dos algoritmos, este capítulo utiliza uma definição um pouco diferente do problema, embora equivalente. São dados um conjunto S de pontos distintos numa métrica M (os servidores), e cada servidor $s \in S$ tem capacidade para atender a $k(s) \geq 1$ requisições distintas. O número $k(s)$ é denominado a **capacidade** de s. É dada também uma sequência R de pontos de M (as requisições), possivelmente com repetições. O objetivo é encontrar uma função $f : R \to S$ tal que $|\{r \in R : f(r) = s\}| \leq k(s)$ para todo $s \in S$ e que minimize $\sum_{r \in R} d_M(f(r), r)$. Note que é necessário que $n \leq k(S)$. Na versão online do problema, R e d_M são dados de forma online.

Segue uma definição formal do problema.

Problema MinEPBOnline2 (M, S, R, k): *Dados uma métrica M, um conjunto finito $S \subseteq V_M$, um inteiro $k(s) \geq 1$ para cada $s \in S$ e uma sequência $R = (r_1, \ldots, r_n)$ de vértices de M com $n \leq k(S)$, encontrar uma função $f : R \to S$ tal que $|\{r \in R : f(r) = s\}| \leq k(s)$ para todo $s \in S$, e que minimize $\sum_{r \in R} d_M(f(r), r)$, sendo que $f(r_i)$ deve ser calculado antes de $f(r_{i+1})$ e sem a informação sobre quem é $(r_{i+1}, \ldots, r_n)$.*

Note que uma solução para o problema nessa forma corresponde a uma solução de mesmo custo para a forma descrita no Capítulo 5: aqui, um servidor s com capacidade $k(s) > 1$ corresponde a $k(s)$ vértices servidores no mesmo ponto, e vice-versa. A representação de uma instância do problema MinEPBOnline2 pode ser assintoticamente menor que a de uma instância correspondente para o problema MinEPBOnline, por exemplo se um servidor possuir capacidade $\omega(n)$. Mas se $k(S) = O(n)$, então a representação das entradas para os dois problemas têm mesmo tamanho assintótico, já que as requisições são dadas uma a uma, e nesse caso os dois problemas são equivalentes.

7.1 O Algoritmo Guloso Probabilístico em Métricas Uniformes

O algoritmo guloso probabilístico, denotado por GP, consiste em, para cada requisição, escolher o servidor mais próximo que ainda está disponível. Se houver empate entre mais de um servidor nessa situação, o algoritmo escolhe um dentre eles com probabilidade proporcional à capacidade de cada servidor. Embora esse algoritmo não se comporte bem com métricas arbitrárias, pois o mesmo exemplo da prova do Teorema 6.2 vale para ele, ele produz um bom resultado para métricas uniformes.

Lema 7.1 (Meyerson, Nanavati e Poplawski, 2006): *O algoritmo* GP *tem fator de competitividade esperado no máximo H_n numa métrica uniforme, onde H_n é o n-ésimo número harmônico e n é o número de requisições.*

Demonstração: Suponha que cada servidor tem capacidade unitária (ou, na formulação do Capítulo 5, que não há mais de um servidor no mesmo ponto da métrica). Suponha também que existem exatamente n servidores. Seja $t > 0$ tal que a métrica em questão é t-uniforme.

Escolha uma solução ótima offline tal que, se existirem várias requisições num mesmo ponto, uma delas atendida com custo zero, a que é atendida com custo zero é a primeira delas. Defina então uma requisição como **barata** se a mesma é atendida com custo 0 nessa solução ótima offline, e como **cara** caso seja atendida com custo t. Assim, se há uma requisição cara em um ponto onde há um servidor, então houve uma outra requisição antes dessa nesse mesmo ponto. Logo, se uma requisição é cara, o algoritmo GP paga custo t para atendê-la.

Se não houver requisições caras, tanto o custo ótimo offline quanto o custo pago pelo algoritmo GP serão zero. Seja então m o número de requisições caras, e suponha que $m > 0$. Claramente o custo ótimo offline é mt. Já o algoritmo GP paga t por cada requisição cara e paga t ou 0 por cada requisição barata, dependendo das escolhas que fez anteriormente.

Sejam então r_i a i-ésima requisição barata e s_i o servidor que a atende na solução ótima offline. Note que, no algoritmo GP, para cada $j < i$, o servidor s_j foi ocupado, ou por r_j, ou por alguma requisição anterior a r_j. Seja $m(i) \leq m$ o número de requisições caras que chegaram antes de r_i. Como há $m(i)+i-1$ requisições antes de r_i, há exatamente $m(i)+i-1$ servidores ocupados quando r_i chega. Desses, $i-1$ são os servidores $s_1, \ldots, s_{i-1}$. Cada um dos demais servidores ocupados antes da chegada de r_i foi ocupado por um sorteio uniforme, e portanto cada servidor fora de $\{s_1, \ldots, s_{i-1}\}$, em especial s_i, tem probabilidade $\frac{m(i)}{n-(i-1)}$ de estar ocupado.

Portanto, o custo esperado pago pelo algoritmo GP é no máximo

$$\begin{aligned} mt + t\sum_{i=1}^{n-m} \frac{m(i)}{n-i+1} &\leq mt + t\sum_{i=1}^{n-m} \frac{m}{n-i+1} \\ &= mt + mt\sum_{i=m+1}^{n} \frac{1}{i} \\ &= mt(1 + H_n - H_m) \\ &\leq mtH_n, \end{aligned}$$

a última desigualdade seguindo do fato que $m \geq 1$ e assim $H_m \geq 1$. Portanto o lema segue.

Quando há servidores com capacidade maior que 1, pode-se fazer um argumento análogo. Interprete cada servidor com capacidade k como um servidor com k vagas. Considere, na solução ótima offline e na solução do algoritmo GP, que as vagas de cada servidor são ocupadas na mesma ordem e, em vez de pensar no servidor que atende uma requisição, pense na vaga utilizada para atender tal requisição. □

Juntando isso com o Teorema 5.3, segue o seguinte.

Teorema 7.2 (Meyerson, Nanavati e Poplawski, 2006 [MNP06, Teorema 2.3]): *O algoritmo* GP *tem fator de competitividade esperado* $H_n = \Theta(\lg n)$ *na métrica uniforme, onde* n *é o número de requisições e* H_n *é o* n*-ésimo número harmônico.*

7.2 O Algoritmo Guloso Probabilístico em HSTs

Nesta seção é tratado o problema do emparelhamento mínimo bipartido online restrito a métricas induzidas por HSTs. O objetivo é compor um algoritmo para essa versão do problema com a técnica de aproximação de métricas por métricas arbóreas, a fim de obter um algoritmo para a versão do problema com métricas arbitrárias.

Meyerson, Nanavati e Poplawski [MNP06] mostraram que o algoritmo GP é $\Theta(\lg n)$-competitivo para instâncias cuja métrica é dada pelas distâncias entre as folhas de uma λ-HST, com $\lambda \geq 1+2\ln n$, onde n é o número de requisições. Além disso, mostram que o fator de competitividade do algoritmo é independente da altura da HST se e só se $\lambda = \Omega(\lg n)$. Csaba e Pluhár [CP08] mostram que uma versão um pouco mais elaborada do algoritmo também é $O(\lg n)$-competitiva para λ-HSTs, desde que $\lambda \geq 2H_n$. A análise dessa versão do algoritmo é mais simples, e é essa versão que se apresenta aqui.

A entrada do algoritmo, denominado GP-HST, é uma quádrupla $I = (T, S, R, k)$, onde T é uma λ-HST perfeita com base t (ver Definições 2.5, 2.6 e 2.7) com $\lambda > 1$ e $t > 0$, S é um subconjunto das folhas de T, R é uma sequência de folhas de T e k é um vetor com uma entrada $k(s) \geq 1$ para cada $s \in S$. Aqui, S é o conjunto dos servidores, R é a sequência de requisições e $k(s)$ é a capacidade do servidor s; lembre-se que $|R| \leq k(S)$. O algoritmo devolve uma função $f : R \to S$ tal que $|\{r \in R : f(r) = s\}| \leq k(s)$ para todo $s \in S$.

É possível mostrar que o algoritmo GP-HST devolve o mesmo resultado do algoritmo GP, mas é escrito de forma diferente, a fim de simplificar sua análise. O algoritmo utiliza como sub-rotina o algoritmo Servir, apresentado mais adiante. O algoritmo estende o vetor k para os vértices internos, de forma que, para um vértice interno w, o valor $k(w)$, denominado também **capacidade** de w, é a soma das capacidades das folhas da subárvore com raiz em w. Se alguma folha v de T não corresponde a nenhum servidor em S, então faça $k(v) := 0$.

GP-HST(T, S, R, k)
1 estenda $k(v)$ para $v \in V_T \setminus S$
2 para $i \leftarrow 1$ até $|R|$ faça
3 $\quad f(r_i) \leftarrow$ Servir(T, k, r_i)
4 devolva f

O algoritmo Servir, descrito a seguir, tem como pré-condição a hipótese de que alguma folha

de T, e consequentemente a raiz de T, tem capacidade positiva, o que é sempre válido já que $|R| - i + 1 \leq k(S)$. Para uma execução com parâmetros (T, k, v), SERVIR devolve um vértice u de T com as seguintes propriedades:

(1) u está à mesma altura que v em T;

(2) u tem capacidade positiva;

(3) dentre os vértices que satisfazem os itens (1) e (2), u está à distância mínima de v em T;

(4) dentre os vértices que satisfazem os itens (1), (2) e (3), u é escolhido com probabilidade proporcional à sua capacidade.

Eventualmente $u = v$. Note que, se u é um vértice que satisfaz essas propriedades, então o ancestral comum de nível mais baixo de u e v é o primeiro vértice no caminho de v à raiz de T com capacidade positiva. Além disso, o algoritmo também atualiza os valores de k, a fim de manter as capacidades dos vértices consistentes. Para um vértice interno v de T, seja $F(v)$ o conjunto dos filhos de v em T.

```
SERVIR(T, k, v)
1  se k(v) > 0 então
2      k(v) ← k(v) − 1
3      devolva v
4  p ← SERVIR(T, k, pai(v))
5  u ← sorteie um dentre os filhos de p com probabilidade k(s)/k(p) para cada s ∈ F(p)
6  k(u) ← k(u) − 1
7  devolva u
```

A ideia do algoritmo consiste em, dada uma requisição num vértice v, primeiro encontrar o ancestral de v de nível mais baixo que tem capacidade positiva (linhas 1-4 de SERVIR). O algoritmo então desce na árvore, sorteando a cada passo uma subárvore com capacidade positiva, com probabilidade proporcional à capacidade da raiz da cada subárvore. Ao final, o algoritmo terá escolhido um dentre os servidores à distância mínima de v. É possível mostrar que esse algoritmo é equivalente ao algoritmo GP, mas escrito dessa forma recursiva fica mais simples argumentar sobre propriedades do algoritmo utilizando indução.

A prova de que o algoritmo SERVIR atende às propriedades (1) e (2) é por indução no comprimento c do caminho em T de v a x, o ancestral comum de nível mais baixo de u e v. Se $c = 0$, o que ocorre quando $k(v) > 0$, tem-se que $x = v$, logo as propriedades valem trivialmente. Se $c > 0$, então $k(v) = 0$ e SERVIR executa as linhas 4-7. Por hipótese de indução, o vértice p da linha 4 tem capacidade positiva e altura igual à do pai de v. Assim, qualquer filho de p tem mesma altura que v e, devido à atualização nas linhas 2 e 6, se $k(p) > 0$, então algum filho de p tem capacidade positiva, do que as propriedades seguem. A propriedade (3) segue do fato de que as chamadas recursivas de SERVIR param assim que o algoritmo encontra um ancestral de v com capacidade positiva e do fato de que, devido à forma da HST, todo vértice com mesma altura e mesmo ancestral comum em relação a v está à mesma distância de v. A propriedade (4) segue claramente do sorteio da linha 5.

Devido a essas propriedades e à forma da HST, como o algoritmo GP-HST faz uma chamada a SERVIR com um vértice folha, o mesmo devolve um vértice folha com capacidade positiva. Portanto,

o algoritmo GP-HST devolve uma solução viável. É evidente também que o algoritmo GP-HST é online.

7.2.1 Análise do Fator de Competitividade

Uma vez que o algoritmo GP-HST supõe que T é uma λ-HST perfeita de base t, todas as arestas de nível i têm custo exatamente $t\lambda^{i-1}$. Sejam r uma requisição, s um servidor e x o ancestral comum de nível mais baixo de r e s, o qual será denominado **ponto de virada** entre r e s. Seja $\mathcal{V}_T$ o conjunto dos vértices internos de T. Para uma solução f e um vértice $v \in \mathcal{V}_T$, seja $\tau_f(v)$ o número de pares $(r, s) \in R \times S$ em que $s = f(r)$ e v é o ponto de virada entre r e s. Segue que o custo de f em relação à instância I é

$$\mathrm{val}(f, I) = 2 \sum_{v \in \mathcal{V}_T} \tau_f(v) \sum_{i=1}^{h_T(v)} t\lambda^{i-1},$$

onde $h_T(v)$ é o nível de v em T. Seja f^* uma solução ótima offline; para simplificar a notação, denote $\tau_{f^*}(v)$ por $\tau(v)$.

O fator de competitividade do algoritmo é $\mathrm{O}(\lg n)$ se $\lambda = 2H_n$, e a prova é por indução na altura h de T. Se $h = 1$, então T restrita a $R \cup S$ induz a métrica $(2t)$-uniforme. Nesse caso, observe que o algoritmo é equivalente ao da Seção 7.1, e portanto tem fator de competitividade H_n, conforme o Teorema 7.2.

O passo de indução se baseia na seguinte transformação. Seja $I = (T, S, R, k)$ uma instância do problema, onde T tem altura $h > 1$, para a qual o algoritmo devolve uma solução f. Seja T' a HST obtida pela remoção das folhas de T e pela divisão do comprimento de cada aresta em T' por λ; note que T' também é uma λ-HST pefeita de base t. Defina como **instância reduzida** de I a instância $I' = (T', S', R', k')$ do problema em que as requisições e servidores numa subárvore T_v de T com raiz v e $h_T(v) = 1$ são todos aglutinados em v, e k' consiste em k restrito a S'. Seja f' a solução que f **induz** em I'; isto é, se $s = f(r)$, então $s' = f'(r')$, onde r' e s' são os pais de r e s em T, respectivamente. Note que

$$\mathrm{val}(f', I') = 2 \sum_{v \in \mathcal{V}_T : h_T(v) \geq 2} \tau_f(v) \sum_{i=1}^{h_T(v)-1} t\lambda^{i-1}.$$

Além disso, note que, se os sorteios do algoritmo forem os mesmos, então a solução devolvida pelo algoritmo para a instância I' será exatamente f'.

O seguinte fato será útil adiante.

Fato 7.3: *Sejam I uma instância do problema do emparelhamento mínimo online bipartido, f^* uma solução ótima offline de I e I' a instância reduzida de I. A solução induzida por f^* em I' é uma solução ótima offline de I'.*

Demonstração: Suponha, por contradição, que exista uma solução $\hat{f}$ de I' com custo menor que o da solução induzida por f^* em I'. Então é possível mostrar que existe uma solução viável de I que induz $\hat{f}$ em I' e tem custo menor que o de f^*, uma contradição. □

Seja então $X = \sum_{v \in \mathcal{V}_T} \tau_f(v)$; note que X é uma variável aleatória. Tem-se que

$$\begin{aligned}
\mathrm{val}(f, I) &= 2 \sum_{v \in \mathcal{V}_T} \tau_f(v) \sum_{i=1}^{h_T(v)} t\lambda^{i-1} \\
&= 2 \sum_{v \in \mathcal{V}_T} \tau_f(v) \left(t + \sum_{i=2}^{h_T(v)} t\lambda^{i-1} \right) \\
&= 2t \sum_{v \in \mathcal{V}_T} \tau_f(v) + 2 \sum_{v \in \mathcal{V}_T} \tau_f(v) \cdot \lambda \sum_{i=2}^{h_T(v)} t\lambda^{i-2} \\
&= 2t \sum_{v \in \mathcal{V}_T} \tau_f(v) + 2\lambda \sum_{v \in \mathcal{V}_T : h_T(v) \geq 2} \tau_f(v) \sum_{i=1}^{h_T(v)-1} t\lambda^{i-1} \\
&= 2tX + \lambda \cdot \mathrm{val}(f', I').
\end{aligned}$$

Da linearidade da esperança,

$$\mathbb{E}[\mathrm{val}(f, I)] = 2t \cdot \mathbb{E}[X] + \lambda \cdot \mathbb{E}[\mathrm{val}(f', I')]. \tag{7.1}$$

O lema a seguir delimita $\mathbb{E}[X]$.

Lema 7.4 (Csaba e Pluhár, 2008 [CP08, Lema 7]): *Seja f uma solução devolvida pelo algoritmo* GP-HST *para uma instância $I = (T, S, R, k)$ do problema do emparelhamento mínimo online bipartido com n requisições, onde T é uma HST perfeita de base t. Seja $X = \sum_{v \in \mathcal{V}_T} \tau_f(v)$, onde $\mathcal{V}_T$ é o conjunto de vértices internos de T e $\tau_f(v)$ é o número de requisições das quais v é o ponto de virada em f. Então*

$$\mathbb{E}[X] \leq \sum_{v \in \mathcal{V}_T} \tau(v) \sum_{i=1}^{h_T(v)} (H_n)^i,$$

onde H_n é o n-ésimo número harmônico e $\tau(v) = \tau_{f^*}(v)$ para uma solução ótima offline f^* de I.

Demonstração: Seja h a altura de T; a prova é por indução em h. Se $h = 1$, então T restrita a $R \cup S$ induz a métrica $(2t)$-uniforme. Do Teorema 7.2,

$$2t \cdot \mathbb{E}[X] = \mathbb{E}\left[2t \sum_{v \in \mathcal{V}_T} \tau_f(v) \right] = \mathbb{E}[\mathrm{val}(f, I)] \leq \mathrm{val}(f^*, I) \cdot H_n = 2t \sum_{v \in \mathcal{V}_T} \tau(v) H_n,$$

do que o lema segue para $h = 1$.

Suponha então que $h > 1$. Sejam $I' = (T', S', R', k')$ a instância reduzida de I e $X' = \sum_{v \in \mathcal{V}_T : h_T(v) \geq 2} \tau_f(v)$. Então

$$X = X' + \sum_{v \in \mathcal{V}_T : h_T(v) = 1} \tau_f(v) \tag{7.2}$$

e, por hipótese de indução,

$$\mathbb{E}[X'] \leq \sum_{v \in \mathcal{V}_T : h_T(v) \geq 2} \tau(v) \sum_{i=1}^{h_T(v)-1} (H_n)^i. \tag{7.3}$$

Seja $\mathcal{T}$ o conjunto de subárvores de T com raiz nos vértices de altura 1 em T. Para cada árvore $T_v \in \mathcal{T}$ com raiz v, considere a instância $I_v = (T'_v, S(v), R_f(v), k[S(v)])$, onde T'_v consiste em T_v acrescida de um vértice v' à distância t de v com $k(v') = 0$, $S(v)$ são os servidores em T_v, $k[S(v)]$ corresponde à restrição de k a $S(v)$, e $R_f(v)$ são as requisições que são atendidas por servidores de T_v em f. As requisições que são atendidas em T_v mas provêm de vértices em $V_T \setminus V_{T_v}$ passam a ser no vértice v' em I_v. Note que I_v depende dos sorteios realizados pelo algoritmo GP-HST para I.

Seja $f_v := \text{GP-HST}(I_v)$; note que, se os sorteios do algoritmo forem os mesmos, então $f_v(r) = f(r)$, para toda requisição r de I_v. Seja f_v^* uma solução ótima offline de I_v e $f^*[I_v]$ uma solução que utiliza as mesmas escolhas que f^* para as requisições de I_v que vêm de T_v, e os servidores restantes para as requisições de I_v em v'. Então $\text{val}(f_v^*, I_v) \leq \text{val}(f^*[I_v], I_v)$. Além disso, do Teorema 7.2, tem-se que $\mathbb{E}[\text{val}(f_v, I_v)] \leq \text{val}(f_v^*, I_v) \cdot H_n$.

Seja $\sigma_f(v)$ o número de requisições em v'; note que

$$\sum_{v \in \mathcal{V}_T : h_T(v) = 1} \sigma_f(v) = X'.$$

Além disso, $\text{val}(f_v, I_v) = 2t(\tau_f(v) + \sigma_f(v))$, já que as requisições contadas em $\sigma_f(v)$ têm ponto de virada em f num vértice de altura maior que 1, e claramente, por construção, $\text{val}(f^*[I_v], I_v) = 2t(\tau(v) + \sigma_f(v))$. Assim,

$$\begin{aligned} 2t(\mathbb{E}[\tau_f(v)] + \mathbb{E}[\sigma_f(v)]) &= \mathbb{E}[\text{val}(f_v, I_v)] \\ &\leq \mathbb{E}[\text{val}(f_v^*, I_v)] \cdot H_n \\ &\leq \mathbb{E}[\text{val}(f^*[I_v], I_v)] \cdot H_n \\ &= 2t(\tau(v) + \mathbb{E}[\sigma_f(v)]) \cdot H_n. \end{aligned}$$

Logo

$$\mathbb{E}[\tau_f(v)] \leq H_n \cdot (\tau(v) + \mathbb{E}[\sigma_f(v)]) - \mathbb{E}[\sigma_f(v)],$$

o que leva a

$$\mathbb{E}\left[\sum_{v \in \mathcal{V} : h_T(v) = 1} \tau_f(v)\right] \leq H_n \left(\sum_{v \in \mathcal{V} : h_T(v) = 1} \tau(v) + \mathbb{E}[X']\right) - \mathbb{E}[X'],$$

e portanto, juntado isso com as Equações (7.2) e (7.3),

$$\begin{aligned} \mathbb{E}[X] &\leq H_n \left(\sum_{v \in \mathcal{V} : h_T(v) = 1} \tau(v) + \mathbb{E}[X']\right) \\ &\leq H_n \left(\sum_{v \in \mathcal{V}_T : h_T(v) = 1} \tau(v) + \sum_{v \in \mathcal{V}_T : h_T(v) \geq 2} \tau(v) \sum_{i=1}^{h_T(v)-1} (H_n)^i\right) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{v\in\mathcal{V}_T:h_T(v)=1} \tau(v)H_n + \sum_{v\in\mathcal{V}_T:h_T(v)\geq 2} \tau(v) \sum_{i=1}^{h_T(v)-1} (H_n)^{i+1} \\
&= \sum_{v\in\mathcal{V}_T:h_T(v)=1} \tau(v)H_n + \sum_{v\in\mathcal{V}_T:h_T(v)\geq 2} \tau(v) \sum_{i=2}^{h_T(v)} (H_n)^{i} \\
&\leq \sum_{v\in\mathcal{V}_T} \tau(v) \sum_{i=1}^{h_T(v)} (H_n)^{i},
\end{aligned}
$$

logo o lema segue. □

O teorema a seguir conclui a análise do algoritmo. O mesmo mostra que o algoritmo GP-HST é λ-competitivo para instâncias em que a métrica é dada por uma λ-HST com $\lambda \geq 2H_n$. Meyerson, Nanavati e Poplawski [MNP06] exibiram um contraexemplo que mostra o algoritmo GP tem fator de competitividade $\omega(\lg n)$ se $\lambda = o(\lg n)$, mas tal contraexemplo não será discutido aqui. Como os algoritmos GP e GP-HST são equivalentes, esse contraexemplo mostra que o resultado a seguir é assintoticamente justo.

Teorema 7.5 (Csaba e Pluhár, 2008): *O custo esperado de uma solução f produzida pelo algoritmo* GP-HST *para uma instância $I = (T, S, R, k)$ do problema do emparelhamento mínimo online bipartido, onde n é o número de requisições de R e T é uma λ-HST perfeita de base t com $\lambda \geq 2H_n$ e $t > 0$, é tal que*

$$\mathbb{E}[\mathrm{val}(f, I)] \leq \lambda \cdot \mathrm{val}(f^*, I).$$

Demonstração: Seja $c_i = 1 - \left(\frac{1}{2}\right)^i$ para $i \geq 1$. Então

$$\mathrm{val}(f^*, I) = 2 \sum_{v\in\mathcal{V}_T} \tau(v) \sum_{i=1}^{h_T(v)} t\lambda^{i-1} \geq 2 \sum_{v\in\mathcal{V}_T} \tau(v) \sum_{i=1}^{h_T(v)} c_i t\lambda^{i-1}.$$

A prova consiste em mostrar que

$$\mathbb{E}[\mathrm{val}(f, I)] \leq 2 \sum_{v\in\mathcal{V}_T} \tau(v) \sum_{i=1}^{h_T(v)} c_i t\lambda^{i}, \tag{7.4}$$

Observe que, para $i > 1$,

$$c_i - c_{i-1} = 1 - \left(\frac{1}{2}\right)^i - 1 + \left(\frac{1}{2}\right)^{i-1} = -1 \cdot \left(\frac{1}{2}\right)^i + 2 \cdot \left(\frac{1}{2}\right)^i = \left(\frac{1}{2}\right)^i.$$

Disso deduz-se que $H_n \leq c_1\lambda$ e que, para $\ell > 1$,

$$
\begin{aligned}
\sum_{i=1}^{\ell} c_i\lambda^i - \sum_{i=1}^{\ell-1} c_i\lambda^{i+1} &= \sum_{i=1}^{\ell} c_i\lambda^i - \sum_{i=2}^{\ell} c_{i-1}\lambda^i \\
&= c_1\lambda + \sum_{i=2}^{\ell} (c_i - c_{i-1})\lambda^i
\end{aligned}
$$

$$\begin{aligned} &= c_1\lambda + \sum_{i=2}^{\ell} \left(\frac{1}{2}\right)^i \lambda^i \\ &= \sum_{i=1}^{\ell} \left(\frac{\lambda}{2}\right)^i \\ &\geq \sum_{i=1}^{\ell} (H_n)^i. \end{aligned}$$

A prova de (7.4) é por indução na altura h de T. Se $h = 1$, do Teorema 7.2 segue que

$$\mathbb{E}[\mathrm{val}(f, I)] \leq \mathrm{val}(f^*, I) \cdot H_n = 2 \sum_{v \in \mathcal{V}_T} \tau(v) \cdot tH_n \leq 2 \sum_{v \in \mathcal{V}_T} \tau(v) \cdot \frac{t\lambda}{2} = 2 \sum_{v \in \mathcal{V}_T} \tau(v) \cdot c_1 t\lambda,$$

logo (7.4) vale para $h = 1$.

Se $h > 1$, sejam I' a instância reduzida de I e f' uma solução que o algoritmo devolveria para I'. Por hipótese de indução e pelo Fato 7.3,

$$\mathbb{E}[\mathrm{val}(f', I')] \quad \leq \quad 2 \sum_{v \in \mathcal{V}_T : h_T(v) \geq 2} \tau(v) \sum_{i=1}^{h_T(v)-1} c_i t\lambda^i.$$

Isso e o Lema 7.4 aplicados a (7.1) permitem concluir que

$$\begin{aligned} \mathbb{E}[\mathrm{val}(f, I)] &= 2t \cdot \mathbb{E}[X] + \lambda \cdot \mathbb{E}[\mathrm{val}(f', I')] \\ &\leq 2t \sum_{v \in \mathcal{V}_T} \tau(v) \sum_{i=1}^{h_T(v)} (H_n)^i + 2t \sum_{v \in \mathcal{V}_T : h_T(v) \geq 2} \tau(v) \sum_{i=1}^{h_T(v)-1} c_i \lambda^{i+1} \\ &\leq 2t \sum_{v \in \mathcal{V}_T} \tau(v) \left(\sum_{i=1}^{h_T(v)} c_i \lambda^i - \sum_{i=1}^{h_T(v)-1} c_i \lambda^{i+1} \right) + 2t \sum_{v \in \mathcal{V}_T : h_T(v) \geq 2} \tau(v) \sum_{i=1}^{h_T(v)-1} c_i \lambda^{i+1} \\ &= 2t \sum_{v \in \mathcal{V}_T} \tau(v) \sum_{i=1}^{h_T(v)} c_i \lambda^i, \end{aligned}$$

que é exatamente (7.4), concluindo a prova. □

7.2.2 Aplicação de Aproximação de Métricas

Neste ponto o seguinte algoritmo para o problema MinEPBOnline2(M, S, R, k), denominado OM (de *online matching*), que compõe o algoritmo GP-HST com a aproximação de métricas por métricas arbóreas, deve ser óbvio para o leitor. O algoritmo BCR(T, k) é o algoritmo da Seção 4.3, que aproxima uma 2-HST T por uma k-HST com distorção esperada $O(k/\lg k)$ (mais precisamente, por uma $(2^{\lceil \lg k \rceil})$-HST perfeita com base definida). O algoritmo FRT(M) é o algoritmo da Seção 4.2, que aproxima uma métrica finita M por uma 2-HST com distorção esperada $O(\lg m)$, onde $m := |V_M|$. O algoritmo OM recebe como entrada uma métrica M, um conjunto S de pontos de M (os servidores), uma sequência R de pontos de M (as requisições), eventualmente com repetições, e um inteiro $k(s) \geq 1$ para cada $s \in S$ (a capacidade do servidor s). O algoritmo devolve uma função $f : R \to S$ tal que $|\{r \in R : f(r) = s\}| \leq k(s)$ para todo $s \in S$.

OM(M, S, R, k)
1 $n \leftarrow |R|$
2 $T \leftarrow \text{BCR}(\text{FRT}(M), 2H_n)$
3 devolva GP-HST(T, S, R, k)

Para simplificar a análise que segue, o algoritmo será reescrito da seguinte forma (expandindo o código de GP-HST).

OM(M, S, R, k)
1 $n \leftarrow |R|$
2 $T \leftarrow \text{BCR}(\text{FRT}(M), 2H_n)$
3 estenda $k(v)$ para $v \in V_T \setminus S$
4 para $i \leftarrow 1$ até n faça
5 $\quad f(r_i) \leftarrow \textsc{Servir}(T, k, r_i)$
6 devolva f

Claramente, se trata de um algoritmo online. Utilizando os Teoremas 4.4 e 4.5 e o Fato 3.10, tem-se que as distâncias entre as folhas de T aproximam probabilisticamente as distâncias em M com distorção esperada $\text{O}(\lg m \lg n / \lg\lg n)$, onde $n := |R|$. Assim, a composição dos algoritmos FRT e BCR corresponde ao item (4) do Teorema 3.11, e o algoritmo GP-HST corresponde ao item (3) do Teorema 3.11. Como o algoritmo GP-HST supõe que as requisições e os servidores são todos em folhas de T, vale o item (2) do Teorema 3.11, e como o valor de um emparelhamento é a soma das distâncias entre uma requisição r e o servidor $f(r)$ que a atende, vale o item (1) do Teorema 3.11. Assim, pelo Teorema 7.5, conclui-se que o algoritmo OM tem fator de competitividade esperado $\text{O}(\lg^2 n \lg m / \lg\lg n)$. Entretanto, esse algoritmo só seria prático se $m = \text{O}(n^c)$, com c constante, o que em muitos casos não é válido: em geral M é infinita.

Uma primeira alternativa que vem à mente seria utilizar $M[R \cup S]$ na linha 2. No entanto, isso faria com que o algoritmo deixasse de ser online para uma instância arbitrária, visto que seria necessário conhecer a sequência R de antemão. O algoritmo é online se for atendida a restrição de que $R \subseteq S$; tal fato será utilizando mais adiante. O pseudocódigo dessa alternativa, o algoritmo OM* para o problema MinEPBOnline2(M, S, R, k), é apresentado a seguir.

OM*(M, S, R, k)
1 $n \leftarrow |R|$
2 $T \leftarrow \text{BCR}(\text{FRT}(M[R \cup S]), 2H_n)$
3 estenda $k(v)$ para $v \in V_T \setminus S$
4 para $i \leftarrow 1$ até n faça
5 $\quad f(r_i) \leftarrow \textsc{Servir}(T, k, r_i)$
6 devolva f

Para instâncias em que $R \subseteq S$, o algoritmo OM* tem fator de competitividade esperado $\text{O}(\lg^2 n \lg |S| / \lg\lg n)$, ou $\text{O}(\lg^3 n / \lg\lg n)$ se $|S| = \text{O}(n)$.

O seguinte algoritmo é um algoritmo online para uma instância arbitrária de MinEPBOnline2. O algoritmo utiliza como base a rotina MaisPróximo(S, r_i), que calcula o servidor mais próximo de r_i em S, independente desse servidor ter ou não capacidade para atender a r_i.

OM′(M, S, R, k)
1 $n \leftarrow |R|$
2 $T \leftarrow \mathrm{BCR}(\mathrm{FRT}(M[S]), 2H_n)$
3 estenda $k(v)$ para $v \in V_T \setminus S$
4 para $i \leftarrow 1$ até n faça
5 $\quad f'(r_i) \leftarrow$ Servir$(T, k,$ MaisPróximo$(S, r_i))$
6 devolva f'

A ideia do algoritmo é, para cada requisição, executar o algoritmo OM* como se a mesma estivesse no servidor mais próximo em S, tenha esse servidor capacidade para atendê-la ou não. Dessa forma, para calcular T é necessário saber apenas a posição dos servidores.

O lema a seguir mostra que, se o fator de competitividade esperado do algoritmo OM* é κ para instâncias em que $R \subseteq S$, então o fator de competitividade esperado de OM′ é no máximo $2\kappa + 1$. Dessa forma, o algoritmo OM′ tem fator de competitividade esperado $\mathrm{O}(\lg^2 n \lg |S| / \lg\lg n)$, o que é $\mathrm{O}(\lg^3 n / \lg\lg n)$ se $|S| = \mathrm{O}(n)$.

Lema 7.6 (Csaba e Pluhár, 2008 [CP08, Lema 2]): *Se o algoritmo* OM* *tem fator de competitividade esperado κ para instâncias em que $R \subseteq S$, então o algoritmo* OM′ *tem fator de competitividade esperado no máximo $2\kappa + 1$.*

Demonstração: Dada uma instância $I = (M, S, R, k)$, seja f' a solução devolvida por OM′ tendo I como entrada. Para $r \in R$, denote por $g(r)$ o servidor devolvido por MaisPróximo(S, r). Sejam $\hat{R} := (g(r_1), \ldots, g(r_n))$, $\hat{I} := (M, S, \hat{R}, k)$ e f a solução devolvida por OM* tendo $\hat{I}$ como entrada. Note que, se os sorteios forem os mesmos, a árvore T obtida na linha 2 do algoritmo OM* para $\hat{I}$ é exatamente a árvore T obtida na linha 2 do algoritmo OM′, uma vez que $\hat{R} \subseteq S$. Assim, existe uma correspondência entre os sorteios de OM* e de OM′ de modo que, para cada $r \in R$,

$$f'(r) = f(g(r)). \tag{7.5}$$

Sejam f^* uma solução ótima offline de I e $\hat{f}$ a solução análoga a f^* em $\hat{I}$, isto é, tal que $\hat{f}(g(r)) = f^*(r)$ para cada $r \in R$. Seja $\hat{f}^*$ uma solução ótima offline de $\hat{I}$.

No melhor caso toda requisição é atendida pelo servidor mais próximo, logo um limite inferior para $\mathrm{val}(f^*, I)$ é $\sum_{i=1}^{n} d_M(r_i, g(r_i))$. Da desigualdade triangular e de (7.5), para cada $r \in R$, vale que

$$d_M(r, f'(r)) \leq d_M(g(r), f(g(r))) + d_M(r, g(r)).$$

Somando isso para todas as requisições, tem-se que

$$\mathrm{val}(f', I) \leq \mathrm{val}(f, \hat{I}) + \mathrm{val}(f^*, I). \tag{7.6}$$

Similarmente, da desigualdade triangular e do fato de que $\hat{f}(g(r)) = f^*(r)$, vale que

$$d_M(g(r), \hat{f}(g(r))) \leq d_M(r, f^*(r)) + d_M(g(r), r),$$

para cada $r \in R$. Logo

$$\text{val}(\hat{f}, \hat{I}) \leq \text{val}(f^*, I) + \sum_{i=1}^{n} d_M(g(r_i), r_i) \leq 2 \cdot \text{val}(f^*, I).$$

Pela definição de $\hat{f}^*$, tem-se que $\text{val}(\hat{f}, \hat{I}) \geq \text{val}(\hat{f}^*, \hat{I})$, logo $\text{val}(\hat{f}^*, \hat{I}) \leq 2 \cdot \text{val}(f^*, I)$.

Como OM* é κ-competitivo, vale que $\text{val}(f, \hat{I}) \leq \kappa \cdot \text{val}(\hat{f}^*, \hat{I})$. Juntando isso com a desigualdade (7.6), tem-se

$$\text{val}(f', I) \leq \kappa \cdot \text{val}(\hat{f}^*, \hat{I}) + \text{val}(f^*, I) \leq (2\kappa + 1) \cdot \text{val}(f^*, I),$$

o que conclui a prova. □

Dessa forma, chega-se ao seguinte teorema.

Teorema 7.7 (Csaba e Pluhár, 2008 [CP08, Teorema 1]): *Existe um algoritmo para o problema do emparelhamento mínimo online bipartido com fator de competitividade esperado* $O(\lg^3 n / \lg \lg n)$, *onde* n *é o número de requisições.*

Capítulo 8

Considerações Finais

Neste trabalho foi apresentada a técnica de aproximação de métricas finitas por métricas arbóreas e a aplicação dessa técnica ao problema do emparelhamento mínimo online bipartido.

Inicialmente, procurou-se definir os conceitos utilizados no restante do texto, a saber, teoria dos grafos, teoria das probabilidades, problemas de otimização, algoritmos de aproximação, algoritmos online e análise competitiva. Na versão original da dissertação, foi feito um esforço para definir algoritmos online e análise competitiva de forma coerente com as definições de problemas de otimização e algoritmos de aproximação, de forma a ter a análise competitiva de algoritmos online como um caso especial da teoria de algoritmos de aproximação. Na literatura de computação online é feito esse paralelo apenas num nível básico; num nível mais formal, as definições não mais se encaixam nas presentes na literatura de algoritmos de aproximação. Chegou-se à conclusão, no entanto, de que a definição proposta não era abrangente o suficiente e que, embora os conceitos sejam parecidos, têm diferenças fundamentais, e optou-se por manter as definições apenas num nível informal.

Em seguida, foram apresentados os conceitos de métricas e aproximação de métricas, juntamente com dois resultados que mostram como a técnica de aproximação de métricas pode ser utilizada para obter bons algoritmos de aproximação em problemas métricos. O Teorema 3.11 mereceu atenção especial, uma vez que chegou-se à conclusão de que as hipóteses no enunciado do teorema original eram insuficientes para a demonstração do mesmo. Por ser um teorema básico da técnica de aproximação de métricas, é necessário chamar a atenção para esse detalhe. Mais adiante foi apresentado o limite inferior para aproximações probabilísticas de métricas por métricas arbóreas. Aqui as principais dificuldades foram a imprecisão e a falta de formalismo em alguns pontos das demonstrações originais, em especial quanto às distribuições de probabilidade utilizadas.

O capítulo seguinte (Capítulo 4) apresenta o algoritmo FRT, que é utilizado para aproximar métricas finitas arbitrárias por métricas arbóreas. Esse foi um dos capítulos que tomou mais tempo, tendo em vista a dificuldade de encontrar uma versão do algoritmo sem brechas nas demonstrações. Houve também um esforço em esclarecer o objetivo de cada procedimento, em especial da rotina Refine. A última seção desse capítulo apresenta a aproximação de uma 2-HST por uma k-HST. O algoritmo precisou ser modificado, tanto porque as definições do trabalho original diferiam das utilizadas neste texto, como para atender às suposições feitas por algoritmos em outros capítulos que utilizam este como base.

Por fim, na parte sobre o problema do emparelhamento mínimo online bipartido (Capítulos 5 a 7), é apresentado um histórico do problema, juntamente com limites inferiores no fator de competitividade dos algoritmos para esse problema. São apresentadas soluções determinísticas e probabilísti-

cas. Uma dificuldade encontrada foi que tanto os algoritmos determinísticos como os probabilísticos foram obtidos em paralelo por dois grupos (para os algoritmos determinísticos, Kalyanasundaram e Pruhs e Khuller, Mitchell e Vazirani, e para os algoritmos probabilísticos, Csaba e Pluhár e Meyerson, Nanavati e Poplawski), de forma que houve um esforço em unificar a notação utilizada em toda a parte. Ainda assim, é utilizada uma definição diferente do problema no capítulo dos algoritmos probabilísticos. Na apresentação do algoritmo GP-HST, em especial, houve um trabalho de formalização muito intenso até se chegar ao nível de detalhe apresentado. Um grande avanço ocorreu quando se percebeu que escrever o algoritmo de forma recursiva auxiliaria no argumento da prova por indução do seu fator de competitividade. (No trabalho original o algoritmo apresentado também é recursivo, mas não existe uma correspondência tão forte entre o algoritmo e a prova por indução como no presente texto.)

O projeto inicial incluía também a aplicação da aproximação de métricas ao problema dos k servidores [Kou09]. No entanto, chegou-se à conclusão de que a simples composição do algoritmo para árvores com a aproximação de métricas não leva a um algoritmo relevante, e tal solução teria o mesmo problema abordado na Seção 7.2.2 em relação ao problema do emparelhamento mínimo online bipartido. Trabalhos mais recentes no sentido de utilizar aproximação de métricas para obter resultados melhores para esse problema ainda são incompletos e cobrem apenas parte dos casos. Assim, decidiu-se focar nos limites inferiores da aproximação de métricas e no problema do problema do emparelhamento mínimo online bipartido. Já o trabalho de Bansal *et al.* [BBGN07], que propõe um algoritmo com melhor fator de competitividade que o descrito na Seção 7.2, foi encontrado apenas às vésperas da finalização do texto, logo não houve tempo para incluí-lo.

Bibliografia Selecionada

[Bar96] Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. Em *FOCS'96: Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, páginas 184–193, 1996. 1, 2, 23, 24, 25, 26

[Bar98] Yair Bartal. On approximating arbitrary metrics by tree metrics. Em *STOC'98: Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, páginas 161–168, 1998. 2, 10

[BBGN07] Nikhil Bansal, Niv Buchbinder, Anupam Gupta, e Joseph Seffi Naor. An $O(\log^2 k)$-competitive algorithm for metric bipartite matching. Em *ESA'07: Proceedings of the 15th Annual European Conference on Algorithms*, páginas 522–533, 2007. 45, 66

[BCR01] Yair Bartal, Moses Charikar, e Ran Raz. Approximating min-sum k-clustering in metric spaces. Em *STOC'01: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, páginas 11–20, 2001. 37

[BEY98] Allan Borodin e Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998. 5, 14

[Bol78] Béla Bollobás. *Extremal Graph Theory*. Academic Press, London, England, 1978. 7

[CP08] Béla Csaba e András Pluhár. A randomized algorithm for the on-line weighted bipartite matching problem. *Journal of Scheduling*, 11(6):449–455, 2008. 2, 45, 55, 58, 63, 64

[dCCD+01] Marcelo Henriques de Carvalho, Márcia Rosana Cerioli, Ricardo Dahab, Paulo Feofiloff, Cristina Gomes Fernandes, Carlos Eduardo Ferreira, Katia Silva Guimarães, Flávio Keidi Miyazawa, José Coelho de Pina Jr., José Augusto R. Soares, e Yoshiko Wakabayashi. *Uma Introdução Sucinta a Algoritmos de Aproximação*. Publicações Matemáticas do IMPA, 2001. 5

[FRT04a] Jittat Fakcharoenphol, Satish Rao, e Kunal Talwar. Approximating metrics by tree metrics. *SIGACT News*, 35(2):60–70, 2004. 2, 28, 29, 34

[FRT04b] Jittat Fakcharoenphol, Satish Rao, e Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004. 2, 10, 28, 29, 34, 36

[Kar89] Richard M. Karp. A $2k$-competitive algorithm for the circle. Manuscript, August 1989. 1

[KMV94] Samir Khuller, Stephen G. Mitchell, e Vijay V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127(2):255–267, 1994. 44, 45, 50, 51, 52

[KP93] Bala Kalyanasundaram e Kirk Pruhs. Online weighted matching. *Journal of Algorithms*, 14(3):478–488, 1993. 44, 50

[KP98] Bala Kalyanasundaram e Kirk Pruhs. On-line network optimization problems. Em *Lecture Notes in Computer Science: Online Algorithms*, volume 1442, páginas 268–280. Springer, 1998. 44, 45, 47

[KVV90] Richard M. Karp, Umesh V. Vazirani, e Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. Em *STOC'90: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, páginas 352–358, 1990. 44

[MNP06] Adam Meyerson, Akash Nanavati, e Laura Poplawski. Randomized online algorithms for minimum metric bipartite matching. Em *SODA'06: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, páginas 954–959, 2006. 2, 45, 46, 49, 50, 55, 60

[RR98] Yuri Rabinovich e Ran Raz. Lower bounds on the distortion of embedding finite metric spaces in graphs. *Discrete and Computational Geometry*, 19(1):79–94, 1998. 1, 24

[WS11] David P. Williamson e David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. 29

Referências Bibliográficas

[AA97] Baruch Awerbuch e Yossi Azar. Buy-at-bulk network design. Em *FOCS'97: Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, páginas 542–547, 1997. 2

[AAA+06] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, e Joseph (Seffi) Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms*, 2(4):640–660, 2006. 2

[AAB96] Baruch Awerbuch, Yossi Azar, e Yair Bartal. On-line generalized Steiner problem. Em *SODA'96: Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, páginas 68–74, 1996. 2

[AKPW95] Noga Alon, Richard M. Karp, David Peleg, e Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24(1):78–100, 1995. 1

[Bar96] Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. Em *FOCS'96: Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, páginas 184–193, 1996. 1, 2, 23, 24, 25, 26

[Bar98] Yair Bartal. On approximating arbitrary metrics by tree metrics. Em *STOC'98: Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, páginas 161–168, 1998. 2, 10

[BBBT97] Yair Bartal, Avrim Blum, Carl Burch, e Andrew Tomkins. A polylog(n)-competitive algorithm for metrical task systems. Em *STOC'97: Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, páginas 711–719, 1997. 10

[BBGN07] Nikhil Bansal, Niv Buchbinder, Anupam Gupta, e Joseph Seffi Naor. An O($\log^2 k$)-competitive algorithm for metric bipartite matching. Em *ESA'07: Proceedings of the 15th Annual European Conference on Algorithms*, páginas 522–533, 2007. 45, 66

[BC97] Piotr Berman e Chris Coulston. On-line algorithms for Steiner tree problems (extended abstract). Em *STOC'97: Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, páginas 344–353, 1997. 2

[BCR01] Yair Bartal, Moses Charikar, e Ran Raz. Approximating min-sum k-clustering in metric spaces. Em *STOC'01: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, páginas 11–20, 2001. 37

[BEY98] Allan Borodin e Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998. 5, 14

[BFR95] Yair Bartal, Amos Fiat, e Yuval Rabani. Competitive algorithms for distributed data management. *Journal of Computer and System Sciences*, 51(3):341–358, 1995. 2

[BH09] Mohammad H. Bateni e Mohammad T. Hajiaghayi. A note on the subadditive network design problem. *Operations Research Letters*, 37(5):339–344, 2009. 2

[Bol78] Béla Bollobás. *Extremal Graph Theory*. Academic Press, London, England, 1978. 7

[BR97] Yair Bartal e Adi Rosén. The distributed k-server problem – a competitive distributed translator for k-server algorithms. *Journal of Algorithms*, 23(2):241–264, 1997. 2

[CCPS97] William J. Cook, William H. Cunningham, William R. Pulleyblank, e Alexander Schrijver. *Combinatorial Optimization*. Wiley-Interscience, 1997. 51

[CHKS06] Chandra Chekuri, Mohammad T. Hajiaghayi, Guy Kortsarz, e Mohammad R. Salavatipour. Approximation algorithms for nonuniform buy-at-bulk network design. *SIAM Journal on Computing*, 39(5):1772–1798, 2006. 2

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, e Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2a. edição, 2001. 9, 12, 15

[CP08] Béla Csaba e András Pluhár. A randomized algorithm for the on-line weighted bipartite matching problem. *Journal of Scheduling*, 11(6):449–455, 2008. 2, 45, 55, 58, 63, 64

[CR99] William Cook e André Rohe. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing*, 11(2):138–148, 1999. 6

[dCCD+01] Marcelo Henriques de Carvalho, Márcia Rosana Cerioli, Ricardo Dahab, Paulo Feofiloff, Cristina Gomes Fernandes, Carlos Eduardo Ferreira, Katia Silva Guimarães, Flávio Keidi Miyazawa, José Coelho de Pina Jr., José Augusto R. Soares, e Yoshiko Wakabayashi. *Uma Introdução Sucinta a Algoritmos de Aproximação*. Publicações Matemáticas do IMPA, 2001. 5

[Edm65] Jack Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17(3):449–467, 1965. 6

[FHK05] Bernhard Fuchs, Winfried Hochstättler, e Walter Kern. Online matching on a line. *Theoretical Computer Science*, 332(1-3):251–264, 2005. 44

[FRT04a] Jittat Fakcharoenphol, Satish Rao, e Kunal Talwar. Approximating metrics by tree metrics. *SIGACT News*, 35(2):60–70, 2004. 2, 28, 29, 34

[FRT04b] Jittat Fakcharoenphol, Satish Rao, e Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004. 2, 10, 28, 29, 34, 36

[GKR00] Naveen Garg, Goran Konjevod, e R. Ravi. A polylogarithmic approximation algorithm for the group Steiner tree problem. *Journal of Algorithms*, 37(1):66–84, 2000. 2

[GKR09] Anupam Gupta, Ravishankar Krishnaswamy, e R. Ravi. Online and stochastic survivable network design. Em *STOC'09: Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, páginas 685–694, 2009. 2

[GOS09] Navin Goyal, Neil Olver, e F. Bruce Shepherd. Dynamic vs. oblivious routing in network design. Em *ESA'09: Proceedings of the 17th Annual European Symposium on Algorithms*, páginas 277–288, 2009. 2

[GS09] Iftah Gamzu e Danny Segev. Improved online algorithms for the sorting buffer problem on line metrics. *ACM Transactions on Algorithms*, 6(1):1–14, 2009. 2

[HKS09] Mohammad T. Hajiaghayi, Guy Kortsarz, e Mohammad R. Salavatipour. Approximating buy-at-bulk and shallow-light k-Steiner trees. *Algorithmica*, 53(1):89–103, 2009. 2

[HST05] Ara Hayrapetyan, Chaitanya Swamy, e Éva Tardos. Network design for information networks. Em *SODA'05: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, páginas 933–942, 2005. 2

[IKMM04] Nicole Immorlica, David Karger, Maria Minkoff, e Vahab S. Mirrokni. On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. Em *SODA'04: Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, páginas 691–700, 2004. 2

[Kar89] Richard M. Karp. A $2k$-competitive algorithm for the circle. Manuscript, August 1989. 1

[KH79] Oded Kariv e S. Louis Hakimi. An algorithmic approach to network location problems. II: The p-medians. *SIAM Journal on Applied Mathematics*, 37(3):539–560, 1979. 2

[KKMR09] Howard Karloff, Subhash Khot, Aranyak Mehta, e Yuval Rabani. On earthmover distance, metric labeling, and 0-extension. *SIAM Journal on Computing*, 39(2):371–387, 2009. 2

[KMV94] Samir Khuller, Stephen G. Mitchell, e Vijay V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127(2):255–267, 1994. 44, 45, 50, 51, 52

[KN04] Elias Koutsoupias e Akash Nanavati. The online matching problem on a line. Em *Lecture Notes in Computer Science: Approximation and online algorithms*, volume 2909, páginas 179–191. Springer, 2004. 44

[Knu98] Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, 3a. edição, 1998. 12

[Kou09] Elias Koutsoupias. The k-server problem. *Computer Science Review*, 3(2):105–118, 2009. 2, 66

[KP93] Bala Kalyanasundaram e Kirk Pruhs. Online weighted matching. *Journal of Algorithms*, 14(3):478–488, 1993. 44, 50

[KP95] Elias Koutsoupias e Christos H. Papadimitriou. On the k-server conjecture. *Journal of the ACM*, 42(5):971–983, 1995. 44

[KP98] Bala Kalyanasundaram e Kirk Pruhs. On-line network optimization problems. Em *Lecture Notes in Computer Science: Online Algorithms*, volume 1442, páginas 268–280. Springer, 1998. 44, 45, 47

[KP06] Rohit Khandekar e Vinayaka Pandit. Online sorting buffers on line. Em *STACS'06: Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science*, páginas 584–595, 2006. 2

[KT02] Jon Kleinberg e Éva Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and Markov random fields. *Journal of the ACM*, 49(5):616–639, 2002. 2, 10

[Kuh55] Harold W. Kuhn. The Hungarian Method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955. 6

[KVV90] Richard M. Karp, Umesh V. Vazirani, e Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. Em *STOC'90: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, páginas 352–358, 1990. 44

[Law00] Eugene Lawler. *Combinatorial Optimization: Networks and Matroids*. Dover Science, 10th edição, 2000. 6

[LPTV06] Xiaozhou Li, C. Greg Plaxton, Mitul Tiwari, e Arun Venkataramani. Online hierarchical cooperative caching. *Theory of Computing Systems*, 39(6):851–874, 2006. 2

[Mar06] Sushma Maramreddy. Online matching problem with application: Google adwords problem. Dissertação de Mestrado, University of Cincinnati, 2006. 45

[MNP06] Adam Meyerson, Akash Nanavati, e Laura Poplawski. Randomized online algorithms for minimum metric bipartite matching. Em *SODA'06: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, páginas 954–959, 2006. 2, 45, 46, 49, 50, 55, 60

[NR10] Viswanath Nagarajan e R. Ravi. Approximation algorithms for requirement cut on graphs. *Algorithmica*, 56(2):198–213, 2010. 2

[PS85] Franco P. Preparata e Michael I. Shamos. *Computation Geometry: An Introduction*. Springer, 1985. 12

[Rob95] Eric S. Roberts. *The Art and Science of C: a Library-Based Introduction to Computer Science*. Addison-Wesley, 1995. 12

[RR98] Yuri Rabinovich e Ran Raz. Lower bounds on the distortion of embedding finite metric spaces in graphs. *Discrete and Computational Geometry*, 19(1):79–94, 1998. 1, 24

[Shm99] David B. Shmoys. Approximation algorithms for clustering problems. Em *COLT'99: Proceedings of the 12th Annual Conference on Computational Learning Theory*, páginas 100–101, 1999. 2

[vZ09] Anke van Zuylen. Deterministic sampling algorithms for network design. *Algorithmica*, 2009. 2

[WS11] David P. Williamson e David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. 29

Índice Remissivo