Clustering de trajetórias

Marcio Takashi Iura Oshiro

Tese apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
Obtenção do Título
de
Doutor em Ciências

Programa: Ciência da Computação
Orientadora: Profa. Dra. Cristina Gomes Fernandes

— São Paulo, 15 de outubro de 2015 —

– Durante a realização desse trabalho, o aluno recebeu apoio financeiro da CAPES. –

Clustering de trajetórias

Este exemplar correponde à redação final da tese devidamente corrigida e defendida por Marcio Takashi Iura Oshiro e aprovada pela comissão julgadora

São Paulo, 15 de outubro de 2015

Comissão Julgadora:

Profa. Dra. Cristina Gomes Fernandes (Presidente)	IME-USP
Prof. Dr. Carlos Eduardo Ferreira	IME-USP
Prof. Dr. Alexandre da Silva Freire	EACH-USP
Prof. Dr. Flávio Keidi Miyazawa	Unicamp
Profa. Dra. Márcia Rosana Cerioli	UFRJ

Agradecimentos

Agradeço

à minha família por todo apoio e suporte que sempre me deram;

à Cris, minha orientadora, por toda a ajuda no desenvolvimento desta tese e por nunca desistir, mesmo diante de várias dificuldades;

aos membros da banca examinadora pela revisão e sugestões;

aos amigos do IME-USP. Sem eles eu não teria aguentado esses 5 anos de doutorado. Em especial, agradeço ao Botler e ao Mota que me ajudaram a renovar minha motivação ao trabalhar paralelamente em outro problema;

à Cris (Sato) e ao Marcel por me aguentarem em momentos de desespero;

ao Carlinhos por permitir que eu fosse o técnico para a Maratona de Programação todos esses anos;

ao pessoal do MaratonIME, que certamente manterá a tradição da USP na Maratona de Programação.

Resumo

Este trabalho teve como objetivo estudar problemas cinéticos de *clustering*, ou seja, problemas de *clustering* nos quais os objetos estão se movendo. O trabalho se concentrou no caso unidimensional, em que os objetos são pontos se movendo na reta real. Diversas variantes desse caso foram abordadas.

Em termos do movimento, consideramos o caso em que cada ponto se move com uma velocidade constante num dado intervalo de tempo, o caso em que os pontos se movem arbitrariamente e temos apenas as suas posições em instantes discretos de tempo, o caso em que os pontos se movem com uma velocidade aleatória em que se conhece apenas o valor esperado da velocidade, e o caso em que, dada uma partição do intervalo de tempo, os pontos se movem com velocidades constantes em cada subintervalo.

Em termos do tipo de *clustering* buscado, nos concentramos no caso em que o número de *clusters* é um dado do problema e consideramos diferentes medidas de qualidade para o *clustering*. Duas delas são tradicionais para problemas de *clustering*: a soma dos diâmetros dos *clusters* e o diâmetro máximo de um cluster. A terceira medida considerada leva em conta a característica cinética do problema, e permite, de uma maneira controlada, que o *clustering* mude com o tempo.

Para cada uma das variantes do problema, são apresentados algoritmos, exatos ou de aproximação, alguns resultados de complexidade obtidos, e questões que ficaram em aberto.

Abstract

This work aimed to study kinetic problems of clustering, i.e., clustering problems in which the objects are moving. The study focused on the unidimensional case, where the objects are points moving on the real line. Several variants of this case have been discussed.

Regarding the movement, we consider the case where each point moves at a constant velocity in a given time interval, the case where the points move arbitrarily and we only know their positions in discrete time instants, the case where the points move at a random velocity in which only the expected value of the velocity is known, and the case where, given a partition of the time interval, the points move at constant velocities in each sub-interval.

Regarding the kind of clustering sought, we focused in the case where the number of clusters is part of the input of the problem and we consider different measures of quality for the clustering. Two of them are traditional measures for clustering problems: the sum of the cluster diameters and the maximum diameter of a cluster. The third measure considered takes into account the kinetic characteristic of the problem, and allows, in a controlled manner, that a cluster change along time.

For each of the variants of the problem, we present algorithms, exact or approximation, some obtained complexity results, and open questions.

Sumário

Agradecimentos ii
Resumo iv
Abstract vi

Sumário **viii**

1 Problemas de *clustering* **1**
1.1 Medidas de similaridade e de qualidade 2
1.2 Cinético e Dinâmico 3
1.3 Instabilidade 4
1.4 Definições e notações básicas 4
1.5 Organização da tese 5

2 Modelos de movimentação **7**
2.1 O modelo unidimensional 7
2.1.1 Cálculo da região e do diâmetro 9
2.1.2 Variantes do modelo unidimensional 9
2.2 Modelos gerais de movimentação 10

3 Minimização da soma dos diâmetros **13**
3.1 Buracos 14
3.2 Determinação dos buracos 15
3.3 Algoritmo para k fixo 18
3.4 Clustering bem-separados 22
3.5 Aproximação para $k = 3$ 25

4 Minimização do maior dos diâmetros **27**
4.1 Uma primeira aproximação 29
4.2 Aproximação gulosa 32

5 Mudança controlada de velocidade **37**
5.1 Movimento variável controlado 37

5.1.1 Cálculo de ε ótimo em um caso particular 41
5.2 Movimento aleatório controlado 46

6 Clustering com instabilidade **49**
6.1 Clustering dinâmico e instabilidade 50
6.2 Instabilidade mínima para diâmetros fixos 52
6.3 Minimização da soma dos diâmetros 55
6.3.1 Trajetórias sem cruzamentos 58
6.4 Minimização do diâmetro máximo dos clusters 63
6.4.1 Uma 2-aproximação para $\gamma = 0$ 66

7 Considerações finais **71**

A Discos no espaço métrico do diâmetro **73**

B Programas lineares inteiros para clustering cinético e dinâmico **75**

C Exemplo não bem-separado para kCC1D-S **79**

Referências Bibliográficas **81**

Capítulo 1

Problemas de *clustering*

Cluster é uma palavra da língua inglesa que significa grupo de objetos ou pessoas semelhantes. *Clustering* é o nome dado a uma partição de um conjunto de objetos em clusters[1]. O termo clustering também se refere a uma bem conhecida classe de problemas computacionais cujo objetivo é particionar um conjunto de forma a agrupar elementos "semelhantes" nos mesmos conjuntos da partição. Os problemas dessa classe diferem principalmente em dois aspectos: a medida de similaridade entre os elementos e a medida de qualidade da partição.

Como exemplo, considere que temos um conjunto de livros. Para cada livro, associamos um vetor de números reais indexados por gêneros literários. Esse vetor quantifica o quanto de cada gênero literário está presente no livro. Queremos distribuir esses livros em três estantes de forma que os livros em uma mesma estante sejam os mais parecidos possíveis em gênero. Esse é um problema de clustering, onde queremos particionar o conjunto em três clusters, cada cluster correspondendo a uma estante. Uma possível medida de similaridade entre dois livros seria a norma da diferença entre os vetores associados aos livros. Uma possível medida de qualidade seria a maior dessas normas entre pares de vetores associados a livros em uma mesma estante. Quanto menor esse valor, menor é a diferença entre pares de livros em uma mesma estante.

Problemas de clustering aparecem em diversas aplicações envolvendo classificação, reconhecimento de padrões, mineração de dados, localização, entre outros, e estão muito presentes, por exemplo, na área de aprendizado de máquina e análise de dados genéticos. Entre os problemas mais conhecidos podemos citar: k-centros (k-*center*) [Gon85, HS86, HN79, KH79a, MS84], k-medianas (k-*median*) [CW14, JRS07, KH79b, MS84], k-médias (k-*means*) [Das07, MNV09], localização de instalações (*facility location*) [CW14, JV01] e corte máximo (*max-cut*) [GJS76, KLS13], entre outros [Bru78, CP04].

[1]Decidimos por manter as palavras *cluster* e *clustering* em inglês por serem termos bem estabelecidos na Ciência da Computação. Por causa da grande frequência de utilização dessas palavras, deixaremos de escrevê-las em itálico para não sobrecarregar o texto.

Podemos considerar dois contextos para os problemas de clustering: um estático e um dinâmico. Um problema de clustering é dito estático se os elementos do conjunto a ser particionado não se movem, ou mais precisamente, se a medida de similaridade entre cada par de elementos é sempre a mesma. O exemplo dos livros corresponde a um problema de clustering estático, pois, uma vez escrito, um livro não muda de gênero literário. Em problemas dinâmicos, os dados do problema podem mudar com o tempo. Um exemplo seria considerar um conjunto de pessoas e tomando como medida de similaridade a distância física entre duas pessoas. Os problemas de clustering considerados nesta tese são dinâmicos.

1.1 Medidas de similaridade e de qualidade

A escolha de quais medidas de similaridade e de qualidade utilizar depende da aplicação de interesse e das particularidades do modelo adotado. Como no nosso caso consideramos pontos localizados na reta real, utilizamos a distância entre eles como medida de similaridade. Isto é, quanto menor o valor absoluto da diferença entre as posições de dois pontos, maior será a semelhança entre eles. Dado um conjunto de pontos, a maior distância entre um par de pontos do conjunto é chamada de diâmetro.

Com relação à medida de qualidade, Cormack [Cor71] chama a atenção para a distinção entre dois tipos de aplicações, chamadas classificação (*classification*) e dissecação (*dissection*). Para classificação, é importante que os elementos de um cluster estejam distantes, com relação à medida de similaridade, dos elementos dos outros clusters. Já para dissecação, a distância entre elementos de clusters distintos não é relevante, como por exemplo em problemas de localização de instalações.

Segundo Cormack, qualquer conjunto pode ser dissecado, no entanto, não é todo conjunto que pode ser classificado. Por exemplo, considere que temos um conjunto de casas e conhecemos as distâncias entre cada par de casas. Se esse conjunto consiste de casas nas cidades de São Paulo e Rio de Janeiro, podemos classificá-las de acordo com suas cidades, pois a distância entre São Paulo e Rio de Janeiro é significativamente grande. Mas se o conjunto consiste de casas nas cidades de São Paulo e São Caetano do Sul, não é possível classificá-las, pois são cidades densas e vizinhas.

Consideramos duas medidas de qualidade diferentes para um clustering $\mathcal{C}$: a soma dos diâmetros dos clusters em $\mathcal{C}$ e o diâmetro máximo de um cluster em $\mathcal{C}$. Em ambos os casos, quanto menor o valor da medida de qualidade, melhor será o clustering.

A Figura 1.1 mostra como clusterings com número fixo de clusters e que minimizam essas duas medidas de qualidade podem ser diferentes. Na Figura 1.1(a) podemos ver que a soma dos diâmetros dos clusters parece ser mais apropriada para classificação. Para problemas de localização, o diâmetro máximo de um cluster parece ser mais apropriado.

A Figura 1.1(b) mostra um exemplo de conjunto de pontos onde qualquer classificação será artificial, pois a distância relativa dos pontos não é grande o suficiente para separá-los em três classes de significados distintos.

(a) 2-clusterings (b) 3-clusterings

Figura 1.1: Exemplo de clustering para pontos estáticos na reta. As marcações curvas indicam um clustering que minimiza a soma dos diâmetros dos clusters e as marcações retas indicam um clustering que minimiza o diâmetro máximo de um cluster.

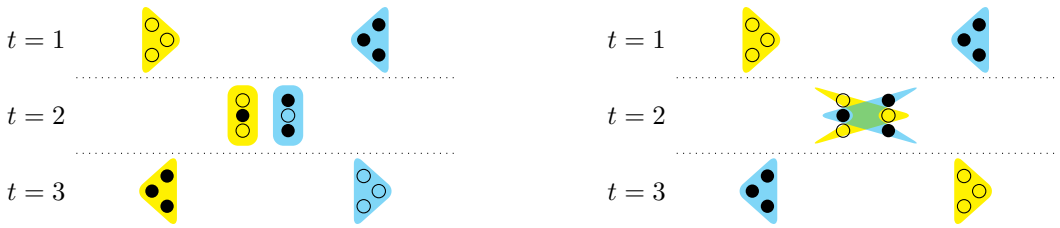
1.2 Cinético e Dinâmico

Chamamos cada elemento do conjunto a ser particionado de trajetória, pois estamos interessados no movimento dos elementos e não no elemento em si. Problemas de clustering onde a medida de similaridade entre um par de trajetórias muda continuamente ao longo do tempo são chamados de cinéticos. Em geral, as trajetórias representam o movimento de objetos ou seres vivos, ou representam dados que mudam ao longo do tempo.

Na literatura, um problema de clustering é dito dinâmico se, ao considerarmos o tempo, ocorrem mudanças de qualquer tipo no conjunto a ser particionado. Assim como no clustering cinético, a medida de similaridade pode mudar ou o próprio conjunto pode mudar, perdendo trajetórias ou ganhando novas trajetórias ao longo do tempo. Neste trabalho, clustering dinâmico tem um significado mais restrito. Utilizamos o termo dinâmico para diferenciar os problemas de clustering cinético nos quais o clustering buscado não é necessariamente estático, ou seja, o clustering pode mudar ao longo do tempo. A Figura 1.2 dá uma intuição da diferença entre clustering cinético e clustering dinâmico.

Dependendo da aplicação de interesse, uma abordagem por clustering cinético é mais adequada do que por clustering dinâmico e vice-versa. Por exemplo, dado um conjunto de animais, separá-los em bandos. Supondo que os bandos de animais têm laços sociais muito fortes e nunca se alteram, o clustering cinético parece ser o mais adequado. Agora, supondo que os animais podem se perder do seu bando original e se juntar a um outro bando, o clustering dinâmico parece ser o mais adequado.

Outra distinção importante de ser feita é que, em geral, tanto problemas de clustering cinético quanto de clustering dinâmico são abordados também em um contexto *on-line*. Isto é, não se tem toda a informação sobre as mudanças, na medida de similaridade ou no conjunto de elementos, de antemão e tais mudanças devem ser tratadas à medida que



(a) Clustering dinâmico (b) Clustering cinético

Figura 1.2: Nos três instantes, os pontos pretos se movem para a esquerda e os pontos vazios se movem para a direita. (a) Partição em cada instante mantendo os pontos mais próximos no mesmo cluster. (b) Partição única considerando a distância entre os pontos nos três instantes.

elas ocorrem. Os problemas que consideramos não são *on-line*, ou seja, consideramos que toda informação sobre as mudanças são conhecidas desde o início.

1.3 Instabilidade

No clustering dinâmico, as trajetórias não precisam pertencer ao mesmo cluster o tempo todo, podendo mudar de cluster. Em algumas aplicações, deseja-se saber qual é o melhor clustering estático a cada momento. Em outras, pode não ser interessante permitir que os clusters mudem com tanta frequência. Tal mudança visa melhorar a medida de qualidade do clustering. No exemplo dos animais, apresentado na Seção 1.2, mesmo que os bandos tenham laços sociais fortes, um animal poderia temporariamente se afastar do bando, se juntando a ele em seguida. Neste caso, pode não fazer muito sentido considerar que o animal mudou de bando e logo em seguida voltou ao seu bando original. Por outro lado, um animal que inicialmente parece pertencer a um bando e em algum momento se afasta, passa a conviver com um outro bando por um longo período, pode indicar uma possível mudança de bando autêntica. Para modelar situações como essa, em que se quer permitir mudanças do clustering, mas com um caráter mais duradouro, introduzimos o conceito de instabilidade de um clustering dinâmico e estudamos uma variante do problema de clustering dinâmico que leva em conta a instabilidade do clustering juntamente com a medida de qualidade do clustering em questão.

1.4 Definições e notações básicas

Um **conjunto** é uma coleção de elementos distintos. Uma coleção que considera repetição de elementos ou cópias de um mesmo elemento é chamada de **multiconjunto**. Dado um número inteiro positivo i, o conjunto de números inteiros $\{1, 2, \ldots, i\}$ é denotado

por $[i]$. Uma **partição** de um conjunto X é uma divisão de todos os elementos de X em subconjuntos dois-a-dois disjuntos.

A ordem dos elementos de um conjunto $X = \{x_1, x_2, \ldots, x_n\}$ é irrelevante. Se a ordem dos elementos for relevante, dizemos que X é uma **sequência** e escrevemos $X = (x_i)_{i=1}^{n}$.

Um **intervalo** é um subconjunto de $\mathbb{R}$ que contém todos os números reais entre um par de números reais a e b, com $a \leqslant b$. Se o intervalo contém os números a e b, o intervalo é **fechado** e denotado por $[a, b]$. Se o intervalo não contém a nem b, o intervalo é **aberto** e denotado por (a, b).

Um **grafo** G é um par (V, E), onde V é um conjunto qualquer e E é um conjunto de pares de elementos distintos em V. Chamamos V de conjunto de **vértices** e E de conjunto de **arestas**. Um **caminho** em um grafo é uma sequência de arestas $(\{u_i, v_i\})_{i=1}^{p}$ tal que $u_i \neq u_j$, $v_i \neq v_j$ e $v_i = u_{i+1}$, para $1 \leqslant i < j \leqslant p$. Equivalentemente, um caminho pode ser visto como uma sequência de vértices distintos $(v_i)_{i=1}^{p+1}$ tal que $\{v_i, v_{i+1}\}$ é uma aresta do grafo, para $1 \leqslant i \leqslant p$. Neste caso, dizemos que o caminho vai do vértice v_1 ao v_{p+1}. Se existir uma ordem nos vértices das arestas, isto é, se cada aresta for um par ordenado, o grafo é **dirigido**. A definição de caminhos se estende naturalmente para grafos dirigidos.

Dado um conjunto X, dizemos que $d\colon X \times X \to \mathbb{R}$ é uma **função de distância** sobre X se, para todo x, y e z em X,

- $d(x, x) = 0$;
- $d(x, y) = d(y, x)$; e
- $d(x, z) \leqslant d(x, y) + d(y, z)$.

A desigualdade do último item é conhecida como **desigualdade triangular**.

1.5 Organização da tese

No Capítulo 2, apresentamos os modelos de movimentos com os quais trabalhamos. Os modelos variam na definição de tempo, que pode ser contínuo ou discreto, e na continuidade do movimento, isto é, o movimento pode ter uma velocidade conhecida ou simplesmente temos mudanças arbitrárias de posição.

No Capítulo 3, discutimos um primeiro problema de clustering cinético onde queremos particionar o conjunto de trajetórias em um número determinado de clusters minimizando a soma de uma medida que chamamos de diâmetro. Apresentamos algoritmos exatos e de aproximação para o problema. O primeiro, Seção 3.3, tem tempo polinomial no tamanho da entrada se considerarmos que o número desejado de clusters é um número fixo que não faz parte da entrada. Na Seção 3.4, definimos o que é um clustering bem-separado e

apresentamos um algoritmo polinomial para encontrar clusterings bem-separados ótimos. Provamos que, para três clusters, esse mesmo algoritmo para clusterings bem-separados funciona como um algoritmo de aproximação para clusterings gerais.

No Capítulo 4, discutimos um segundo problema de clustering cinético onde, em vez de minimizar a soma dos diâmetros, queremos minimizar o diâmetro máximo de um cluster. Para este problema apresentamos dois algoritmos de aproximação.

Para os dois problemas apresentados, consideramos que as trajetórias possuem velocidade constante. No Capítulo 5, discutimos como lidar com velocidades variáveis, mas não totalmente arbitrárias.

No Capítulo 6, apresentamos um estudo inicial de problemas de clustering dinâmicos com instabilidade. Consideramos dois problemas que utilizam medidas de qualidade do clustering parecidas com as dos problemas cinéticos e mostramos alguns resultados para casos particulares extremos.

Capítulo 2

Modelos de movimentação

Muitos resultados heurísticos existentes na literatura sobre problemas de clustering com pontos em movimentos consideram modelos de movimentação genéricos para dimensões arbitrárias, como os sugeridos por Atallah [Ata85] e Basch, Guibas e Hershberger [BGH99]. Como nosso interesse é em resultados teóricos, nos restringimos a um modelo mais simples, para uma dimensão, que chamamos de modelo cinético unidimensional. Tal modelo é um caso particular dos modelos de Atallah e de Basch *et al.* e se mostra desafiador o suficiente para ser o ponto de partida para estudos teóricos de problemas de clustering com pontos em movimento.

Considerar apenas uma dimensão parece restringir demais o modelo, tornando os problemas triviais. Isso não é verdade, como veremos neste trabalho. É comum considerar um modelo unidimensional para abordar problemas difíceis [CW14, HT91, JRS07]. Em particular, vale a pena mencionar um trabalho recente de Karpinski, Lingas e Sledneu [KLS13], apresentando um algoritmo polinomial para o problema do corte máximo. A dificuldade do problema do corte máximo para o espaço métrico unidimensional estava em aberto até então.

Neste capítulo apresentaremos o modelo cinético unidimensional e os modelos de Atallah e de Basch *et al.* Os dois últimos são apresentados apenas para manter a completude deste trabalho, mas não serão utilizados em suas formas completas. Também discutimos variações do modelo cinético unidimensional.

2.1 O modelo unidimensional

No modelo cinético unidimensional, ou apenas modelo unidimensional, n pontos se movem com velocidade retilínea uniforme durante um intervalo de tempo contínuo. Sem perda de generalidade, podemos considerar que o intervalo de tempo é sempre o intervalo $[0, 1]$. Cada ponto i em $[n]$ tem uma posição inicial $x_i(0)$ e sua velocidade é dada por um valor v_i. Como os pontos estão em $\mathbb{R}$, uma velocidade $v_i > 0$ indica que o ponto i

está se movendo para a direita e $v_i < 0$ indica que o ponto i está se movendo para a esquerda.

Em um dado instante t, a posição $x_i(t)$ de um ponto i com posição inicial $x_i(0)$ e velocidade v_i é dada pela função

$$x_i(t) = x_i(0) + v_i t. \tag{2.1}$$

Para t no intervalo $[0, 1]$, esta função representa um segmento de reta no plano cartesiano. Chamaremos tal segmento de **trajetória** do ponto i. Nas figuras, consideramos que o eixo horizontal do plano cartesiano representa a posição x e o eixo vertical representa o tempo t. Como o intervalo de tempo é sempre $[0, 1]$, a faixa do plano entre $t = 0$ e $t = 1$ é chamada de **faixa de tempo**.

Por conveniência, trataremos os pontos e suas respectivas trajetórias como o mesmo objeto. A menos que dito o contrário, vamos supor que não existem dois pontos com a mesma trajetória, pois, em geral, tais pontos podem ser considerados como um mesmo ponto.

Dado um conjunto finito S de trajetórias, um **cluster** é um subconjunto de S e um ***k*-clustering** é uma partição de S em k clusters. O lado esquerdo de um cluster não vazio C é definido pela função linear por partes $\min_{i \in C} x_i(t)$, para t no intervalo $[0, 1]$. Analogamente, o lado direito de C é definido por $\max_{i \in C} x_i(t)$, para t no intervalo $[0, 1]$. A **região** de um cluster C, região(C), é vazia se C é vazio, caso contrário é a área da faixa de tempo limitada pelos lados esquerdo e direito de C. Definimos o **diâmetro** de C como a área da sua região, denotado por diam(C). Um exemplo é ilustrado pela Figura 2.1.

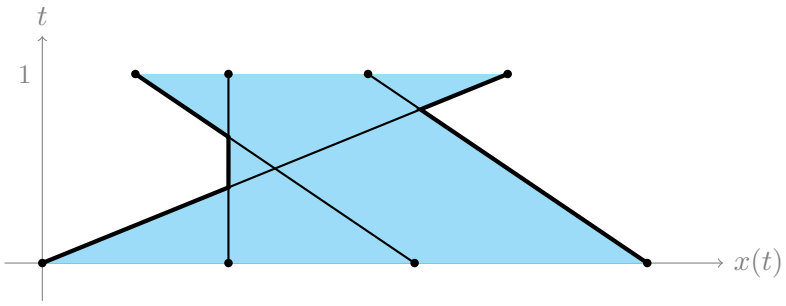


Figura 2.1: Representação de um conjunto de pontos em movimento por suas trajetórias. A área destacada é a região do conjunto.

Neste trabalho, o diâmetro será a principal medida de similaridade entre as trajetórias. Quanto menor os diâmetros dos clusters de um k-clustering, melhor será o k-clustering. Existem várias medidas de qualidade que formalizam a noção de melhor k-clustering e cada uma gera um problema de otimização diferente. Por exemplo, duas medidas de qualidade comuns são a soma dos diâmetros dos clusters e o diâmetro máximo de um cluster.

O motivo de escolhermos o diâmetro como medida de similaridade é porque consideramos o tempo como um intervalo contínuo. Em versões estáticas de problemas de clustering, o diâmetro, definido como a maior distância entre pares de pontos no cluster, é uma medida bastante comum. Então, ao considerar um intervalo de tempo contínuo, tomamos a integral dessa distância ao longo do intervalo para calcular o diâmetro de um cluster C, como mostra a Equação (2.2).

$$\int_0^1 \left(\max_{i \in C} x_i(t) - \min_{i \in C} x_i(t) \right) \mathrm{d}t = \text{área}(\text{região}(C)) = \text{diam}(C). \tag{2.2}$$

2.1.1 Cálculo da região e do diâmetro

A região de um cluster vazio é vazia e a região de um cluster que contém apenas uma trajetória é a própria trajetória. Ambas têm diâmetro zero e são casos triviais de identificar.

Dado um cluster C com $m > 1$ trajetórias, é possível determinar sua região em tempo polinomial em m. Para encontrar a função $\min_{i \in C} x_i(t)$, para t em $[0, 1]$, que define o lado esquerdo de C, utilizamos um algoritmo de divisão-e-conquista com complexidade de tempo $\mathrm{O}(m \log m)$ [Ata85]. O mesmo pode ser feito para o lado direito de C. Note que a região de um cluster será um polígono e, portanto, o diâmetro de C pode ser calculado em tempo linear no número de vértices desse polígono [O'R98].

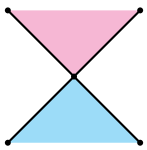


Figura 2.2: Esta região consiste de dois triângulos.

Um caso especial ocorre quando o lado esquerdo de um cluster intersecta o lado direito. Por exemplo, se um cluster tem apenas duas trajetórias que se intersectam em algum instante t, com $0 < t < 1$. Ver Figura 2.2. Como o lado esquerdo de um cluster é o mínimo de um conjunto de valores e o lado direito é o máximo do mesmo conjunto de valores, tais funções só podem se intersectar em no máximo um ponto. Nesse caso, a região do cluster pode ser dividida em dois polígonos.

2.1.2 Variantes do modelo unidimensional

Existem algumas variantes do modelo unidimensional que consideramos neste trabalho. Uma delas é relaxar a restrição de que a velocidade dos pontos deve ser constante e permitir que a posição dos pontos possa ser descrita por uma função contínua não necessariamente linear. Claro que, para tal função, deve existir uma representação de

tamanho polinomial no número de pontos considerados, caso contrário não temos como determinar as posições dos pontos eficientemente. Chamamos esta variante de modelo unidimensional com **movimento variável** e este será melhor detalhado no Capítulo 5.

O modelo unidimensional com movimento variável é mais geral que o modelo unidimensional, mas supõe conhecimento exato da movimentação dos pontos. Em muitos casos, não temos conhecimento exato da movimentação, mas temos uma certa ideia de como é essa movimentação. Uma maneira de modelar esses casos é considerando mudanças aleatórias na velocidade. Nesse modelo, chamado modelo unidimensional **aleatório**, consideramos que em intervalos uniformes de tempo, os pontos mudam de velocidade. Dentro de cada um desses intervalos a velocidade é constante e é dada por uma variável aleatória com valor esperado conhecido. O modelo unidimensional aleatório é explorado na Seção 5.2.

Em uma outra variante do modelo unidimensional, os pontos podem se movimentar arbitrariamente. Neste caso, em vez de considerarmos que o movimento é contínuo em um intervalo de tempo, consideramos um conjunto discreto de instantes e conhecemos apenas a posição dos pontos nos instantes desse conjunto. Este modelo é chamado de modelo unidimensional **discreto** e é tratado no Capítulo 6.

2.2 Modelos gerais de movimentação

Estudando problemas de geometria computacional para pontos em movimento, Atallah [Ata85] propõe um modelo de movimento para pontos no espaço euclidiano d-dimensional. Neste modelo, cada uma das d coordenadas de cada ponto é uma função polinomial do tempo. Suponha que tais polinômios têm grau no máximo m. A posição $x_i(t)$ do ponto i no instante t é dada por

$$x_i(t) = \sum_{p=0}^{m} \vec{c_{i,p}} t^p,$$

onde $\vec{c_{i,p}}$ é um vetor d-dimensional. Note que para $m = 1$ e $d = 1$ a posição do ponto i no instante t é $x_i(t) = c_{i,0} + c_{i,1}t$. Logo, o modelo unidimensional corresponde ao modelo de Atallah para $m = 1$ e $d = 1$.

O modelo de Basch *et al.* [BGH99] descreve a movimentação dos pontos de forma similar ao modelo de Atallah. Cada ponto i tem movimento contínuo ao longo do tempo, descrito por uma função $x_i(t)$, chamada de plano de voo. A diferença para o modelo de Atallah é que os planos de voo podem ser atualizados *on-line*. Ou seja, não se tem toda informação sobre a movimentação dos pontos de antemão e é necessário saber lidar com as mudanças nos planos de voo assim que elas ocorrem. Para isso, Basch *et al.* propõe um modelo de estrutura de dados que chamam de *Kinetic Data Structure* (*KDS*).

No estudo de problemas envolvendo um conjunto de pontos em movimento, em geral, estamos interessados em algum atributo desse conjunto. Como exemplo desses atributos podemos citar o par de pontos mais próximos e o casco convexo. O *KDS* considera uma descrição combinatória desses atributos. Para os dois exemplos citados, a descrição combinatória seria respectivamente um par de pontos mais próximos e um conjunto de vértices do casco convexo. A ideia principal dessas estruturas de dados é se aproveitar do fato de que, apesar dos pontos se moverem continuamente, o atributo de interesse também muda de forma contínua, mas a descrição combinatória do atributo de interesse muda apenas em alguns instantes específicos. Com isso, as estruturas de dados que seguem o modelo *KDS* precisam manter, além da descrição combinatória do atributo, um conjunto de certificados que garantem a correção da descrição combinatória atual. Tais certificados são calculados a partir dos planos de voo dos pontos e possuem um prazo de validade. Ao atingir o prazo de validade, os certificados não garantem mais a correção da descrição combinatória atual e novos certificados, ou até mesmo uma nova descrição combinatória, precisam ser calculados. Em geral, os prazos de validade correspondem aos instantes onde ocorrem mudanças na descrição combinatória do atributo de interesse.

Gao, Guibas, Hershberger, Zhang, e Zhu [GGH+03] consideram o problema do conjunto dominante de cardinalidade mínima para um grafo específico e propõem uma estrutura de dados no modelo *KDS* que mantém um conjunto dominante ao longo do tempo. Em tal grafo, cada vértice corresponde a um ponto no plano e é adjacente a todo vértice que está a uma determinada distância dele. Logo, as arestas do grafo podem mudar com o tempo. Os vértices do conjunto dominante mantido podem mudar com o tempo, mas os autores provam que são mudanças suaves e o conjunto mantido é uma aproximação de fator constante para o instante atual.

Har-Peled [HP04] considera o problema dos k-centros com o modelo de movimento de Atallah. O algoritmo probabilístico apresentado por Har-Peled encontra um clustering com no máximo k^{m+1} clusters, onde m é o grau máximo dos polinômios que descrevem o movimento. O valor de tal k^{m+1}-clustering é competitivo, por um fator constante, contra k-clusterings ótimos das versões estáticas para cada instante de tempo.

Lee, Han, e Whang [LHW07] consideram um modelo diferente, onde uma trajetória é definida como uma sequência de pontos em um espaço multidimensional. Os autores observam que agrupar essas trajetórias como um todo pode não dar um bom resultado, pois elas podem ser parecidas em certos trechos, mas diferentes nos demais trechos. Um arcabouço é proposto, onde primeiro cada trajetória é particionada em segmentos ligados por dois pontos da trajetória. Em seguida, um problema de clustering é resolvido para esses segmentos.

Degener, Gehweiler e Lammersen [DGL10] desenvolvem uma estrutura de dados *KDS* para o problema de localização de instalações com pontos se movendo continuamente em

um espaço euclidiano multidimensional. A estrutura mantém um conjunto de pontos que representa as instalações no momento atual cujo custo é no máximo um fator constante maior que o custo de um conjunto ótimo de instalações para o momento atual. Os pontos podem ficar alterando de papel, entre instalação e cliente.

Uma resenha escrita por Hartmann, Kappes e Wagner [HKW14] apresenta vários trabalhos sobre clustering dinâmico, no sentido mais geral, discutindo as diferenças em abordagens utilizadas para as várias formas de considerar a dinâmica dos pontos.

Capítulo 3

Minimização da soma dos diâmetros

Esta seção trata do problema k-Clustering Cinético 1-Dimensional de Soma Mínima (kCC1D-S), no qual são dados n pontos em $\mathbb{R}$, cada um com uma posição inicial e uma velocidade constante. Deseja-se encontrar um k-clustering cuja soma dos diâmetros de seus clusters seja mínima. Para simplificar, podemos redefinir o problema kCC1D-S através das trajetórias dos pontos. Dado um k-clustering $\mathcal{C}$, denotamos por

$$\mathrm{sd}(\mathcal{C}) = \sum_{C \in \mathcal{C}} \mathrm{diam}(C) \tag{3.1}$$

a soma dos diâmetros dos clusters em $\mathcal{C}$.

Problema 3.1 (kCC1D-S)**.** Dado um conjunto de trajetórias S e um número inteiro positivo k, encontrar um k-clustering $\mathcal{C}$ de S com $\mathrm{sd}(\mathcal{C})$ mínimo.

Seja $\mathcal{C}^*$ um k-clustering ótimo para um dado conjunto S de trajetórias e um dado número inteiro positivo k, isto é, $\mathrm{sd}(\mathcal{C}^*)$ é mínimo entre todos os k-clusterings de S. Denotamos por $\mathrm{sd}^*(S,k) = \mathrm{sd}(\mathcal{C}^*)$ o valor de um k-clustering de S ótimo.

Na versão estática do kCC1D-S, são dados n pontos em $\mathbb{R}$ e queremos encontrar um k-clustering cuja soma dos diâmetros de seus clusters seja mínima. Nesta versão, o diâmetro é definido como a maior distância entre um par de pontos no cluster. Podemos resolver a versão estática em tempo polinomial através de um algoritmo guloso [Bru78] que encontra as $k-1$ maiores distâncias entre pares de pontos adjacentes. Tais pares ficam como extremos de clusters distintos, como ilustrado na Figura 3.1. Para verificar que esse algoritmo guloso sempre devolve um k-clustering ótimo para a versão estática do kCC1D-S, note que nunca vale a pena ter um extremo de um cluster posicionado entre os dois extremos de outro cluster. Caso contrário, estaríamos somando um mesmo intervalo de $\mathbb{R}$ repetidamente.

O caso particular do kCC1D-S no qual as trajetórias em S não se intersectam pode ser reduzido à sua versão estática e, consequentemente, resolvido em tempo polinomial

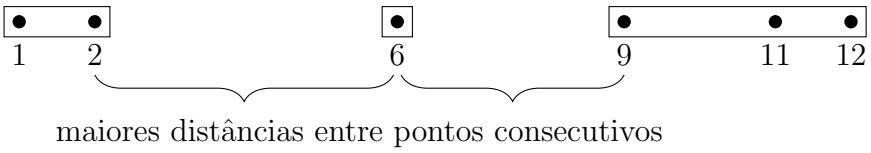


Figura 3.1: 3-clustering ótimo para a versão estática do kCC1D-S.

da seguinte forma. Como as trajetórias não se intersectam, elas podem ser ordenadas da esquerda para a direita. Considere que a sequência ordenada das trajetórias de S é $(s_i)_{i=1}^n$. Tomando $x_1 = 0$ e $x_i = x_{i-1} + \text{diam}(\{s_{i-1}, s_i\})$, para $2 \leqslant i \leqslant n$, obtemos uma instância da versão estática do kCC1D-S que corresponde ao conjunto de trajetórias S. Uma solução ótima dessa instância é uma solução ótima da instância original do kCC1D-S.

Mostraremos que o problema kCC1D-S também pode ser resolvido em tempo polinomial se considerarmos o valor de k fixo ou se queremos um k-clustering bem separado. No entanto, não podemos utilizar essa abordagem gulosa nesses casos, pois, diferentemente da versão estática, o diâmetro de um cluster não é definido por um par de trajetórias cuja região tem maior área. O diâmetro de um cluster pode ser maior que isso.

3.1 Buracos

Seja S um conjunto de trajetórias. Essas trajetórias dividem a faixa de tempo em regiões poligonais minimais. Chamamos essas regiões de **buracos** de S. Note que sempre existem dois buracos com região ilimitada, um à esquerda da região de S e outro à direita. Tais buracos são chamados de **buracos ilimitados**. Os demais buracos são **limitados**. O conjunto de buracos de S é denotado por $B(S)$, ou simplesmente B quando não houver dúvidas sobre o conjunto de trajetórias considerado.

Cada buraco b particiona S em duas partes, $S_e(b)$ e $S_d(b)$ (Figura 3.2), onde

$$S_e(b) = \{s \in S \mid x_s(t_b) \leqslant x_b \text{ para todo ponto } (x_b, t_b) \in b\}, \text{e}$$

$$S_d(b) = \{s \in S \mid x_s(t_b) \geqslant x_b \text{ para todo ponto } (x_b, t_b) \in b\}.$$

Para simplificar, dizemos que $S_e(b)$ é o conjunto das trajetórias de S à esquerda de b e $S_d(b)$ é o conjunto das trajetórias de S à direita de b.

O número de buracos de um conjunto de trajetórias é polinomial na quantidade de trajetórias. Isso segue como um corolário do seguinte resultado, que é bem conhecido.

Proposição 3.2. Um conjunto de n retas divide o plano em no máximo $\frac{n(n+1)}{2}+1$ regiões.

Demonstração. A prova é por indução em n. Se $n = 0$, o plano todo é contado como uma região. Então, suponha que $n \geqslant 1$. Seja ℓ uma reta do conjunto. Sem ℓ, por hipótese de indução, as outras $n - 1$ retas dividem o plano em no máximo $\frac{(n-1)n}{2} + 1$ regiões.

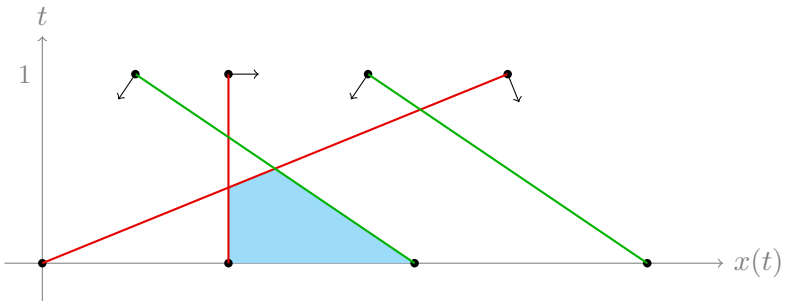


Figura 3.2: O buraco b, destacado, particiona as trajetórias nos conjuntos $S_e(b)$, em vermelho, e $S_d(e)$, em verde.

Como duas retas distintas se intersectam no máximo uma vez, o número máximo de regiões é obtida se ℓ intersecta todas as outras $n-1$ retas, evitando pontos de intersecção já existentes. Neste caso, ℓ pode aumentar o número de regiões em até n, pois cada intersecção pode apenas dividir uma região em duas. Portanto, o número total de regiões é no máximo

$$\frac{(n-1)n}{2} + 1 + n = \frac{n(n-1+2)}{2} + 1 = \frac{n(n+1)}{2} + 1. \qquad \square$$

Corolário 3.3. Um conjunto não vazio com n trajetórias tem no máximo $\frac{n(n+1)}{2} - 1$ buracos limitados.

3.2 Determinação dos buracos

Dado um conjunto S de n trajetórias, mostraremos como encontrar os buracos limitados de S. O algoritmo que apresentaremos também pode encontrar os buracos ilimitados, mas tais buracos não nos interessam.

Consideraremos as trajetórias como segmentos de retas no plano cartesiano com eixos X (horizontal) e T (vertical). Cada trajetória s de S tem extremidades nos pontos $(x_s(0), 0)$ e $(x_s(1), 1)$ do plano. Para um dado t em $[0, 1]$, definimos uma ordem total $<^t$ sobre S da seguinte forma. Sejam r e s duas trajetórias distintas de S. Então, $r <^t s$ se e somente se $x_r(t) < x_s(t)$, ou $x_r(t) = x_s(t)$ e $x_r(t-\varepsilon) < x_s(t-\varepsilon)$, para todo $\varepsilon > 0$. Dizemos que as trajetórias r e s são **adjacentes** em t se elas são consecutivas na ordem $<^t$.

Note que uma trajetória nunca tem mais do que duas trajetórias adjacentes, podendo ser uma à sua esquerda e outra à sua direita.

O algoritmo que vamos propor é baseado no algoritmo de linha de varredura de Bentley e Ottmann [BO79] para encontrar os pontos de intersecção entre pares de segmentos de reta dentro de um conjunto de segmento de retas. Para cada buraco b de S encontrado, o algoritmo também encontra o subconjunto $S_e(b)$. O subconjunto $S_d(b)$ é implicitamente encontrado, pois $S_d(b) = S \setminus S_e(b)$.

A linha de varredura será uma reta paralela ao eixo X, movendo-se ao longo do intervalo $[0, 1]$ no eixo T. A ideia geral do algoritmo é que a linha de varredura no instante t será representada pela ordenação $<^t$ das trajetórias. Note que essa ordenação só muda após a linha de varredura passar por um ponto de intersecção entre trajetórias. Portanto, podemos simular o movimento da linha de varredura processando-a para cada ponto de intersecção entre trajetórias, ordenadas pela coordenada T. Durante a varredura, trajetórias adjacentes correspondem a arestas de um mesmo buraco. A cada ponto de intersecção atingido pela linha de varredura, completamos a representação de pelo menos um buraco e imediatamente iniciamos a representação de um novo buraco.

Inicialmente a linha de varredura começa em $t = 0$ e é representada pela sequência das n trajetórias de S ordenadas por suas posições no eixo X no instante t. Denotaremos por s_i^t a i-ésima trajetória na sequência que representa a linha de varredura no instante t. Caso no instante t existam trajetórias na mesma posição, o desempate é feito pela posição dessas trajetórias em um instante $t^- < t$, como mostrado na Figura 3.3. Note que, para qualquer $t^- < t$ escolhido, o resultado do desempate sempre será o mesmo, pois duas trajetórias se intersectam no máximo uma vez.

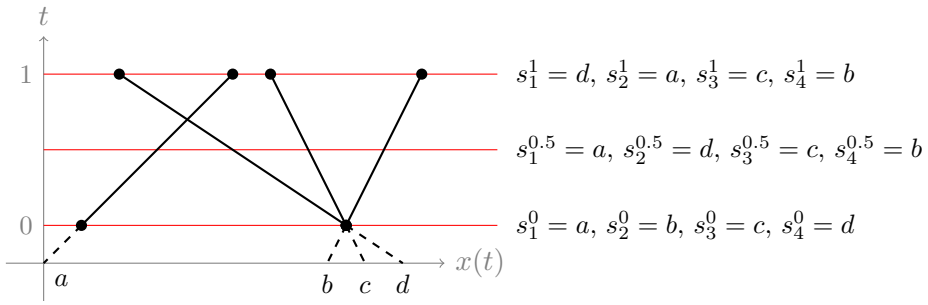


Figura 3.3: Sequências de quatro trajetórias, a, b, c e d, representando a linha de varredura nos instantes 0, 0.5 e 1.

Sejam $j \geqslant 2$ e $s_1, s_2, \ldots, s_j$ trajetórias de S. Suponha que essas trajetórias são as únicas que se intersectam em um ponto (x', t'). Sem perda de generalidade, podemos supor que

$$x_{s_1}(t' - \varepsilon) < x_{s_2}(t' - \varepsilon) < \cdots < x_{s_j}(t' - \varepsilon),$$

para $\varepsilon > 0$ tendendo a zero. Logo, temos que s_i é adjacente a s_{i+1}, para $1 \leqslant i < j$. Ademais,

$$x_{s_1}(t' + \varepsilon) > x_{s_2}(t' + \varepsilon) > \cdots > x_{s_j}(t' + \varepsilon).$$

Essa propriedade é uma invariante no algoritmo. Por causa dela, a cada ponto de intersecção que a linha de varredura passa, basta inverter a subsequência das trajetórias que se intersectam nesse ponto para atualizar a linha.

Os buracos limitados são representados pela lista de seus vértices ordenados em sentido anti-horário. Uma coisa a se notar é que tais buracos são polígonos convexos, pois

são o casco convexo de um subconjunto de pontos extremos de trajetórias e pontos de intersecção entre trajetórias. Logo, para todo $1 \leqslant i < n$ e $t \in [0,1]$, temos que existe um buraco com uma aresta contida em s_i^t e outra contida em s_{i+1}^t. Ademais, se $x_{s_i^t}(t) = x_{s_{i+1}^t}(t) = x$, então o ponto (x,t) corresponde ao vértice mais baixo de um buraco se $t < 1$, e ao vértice mais alto de outro buraco se $t > 0$.

Mostraremos uma forma de obter os buracos limitados de um conjunto S de trajetórias utilizando $n-1$ pilhas e $n-1$ filas, denotadas por $P_e(i)$ e $F_d(i)$, $2 \leqslant i \leqslant n$, respectivamente.

Inicialmente, para $2 \leqslant i \leqslant n$, enfileiramos $(x_{s_i^0}(0), 0)$ em $F_d(i)$ e, se $x_{s_i^0}(0) \neq x_{s_{i-1}^0}(0)$, também empilhamos $(x_{s_{i-1}^0}(0), 0)$ em $P_e(i)$. Ao longo da varredura, fazemos o seguinte. Seja (x,t) um ponto de intersecção entre as trajetórias $s_p^t, s_{p+1}^t, \ldots, s_q^t$, para $p < q$, tal que s_{p-1}^t e s_{q+1}^t não se intersectam nesse ponto ou não estão definidas. O ponto (x,t) é o vértice mais alto de $q-p$ buracos. Para cada b, com $p \leqslant b < q$, obtemos um buraco B da seguinte forma. Primeiro, desenfileiramos os pontos em $F_d(b)$ e os inserimos em B na ordem em que foram desenfileirados e inserimos (x,t) no final de B. Com isso, B corresponde à sequência de vértices, no sentido anti-horário, do lado direito de um buraco. Os vértices do lado esquerdo do buraco estão em $P_e(b)$. Logo, basta desempilhar os pontos em $P_e(b)$ e inseri-los, em ordem, no final de B. Note que agora, $P_e(b)$ e $F_d(b)$ estão vazias. Enfileiramos (x,t) em $F_e(b)$, pois esse ponto corresponde ao ponto mais baixo de um novo buraco. No final, quando $t = 1$, enfileiramos e empilhamos os extremos das trajetórias de forma similar ao instante inicial e obtemos os buracos restantes pelo mesmo processo descrito anteriormente.

Portanto, para encontrar os buracos, executamos a varredura de Bentley e Ottmann [BO79] para encontrar os pontos de intersecção entre as trajetórias de S com as seguintes modificações. Iniciamos executando a sub-rotina IniciaBuraco que recebe S, a representação da linha de varredura, as filas F_d e as pilhas P_e. A varredura encontra os pontos de intersecção em ordem crescente da coordenada t. Cada vez que um ponto de interseção é encontrado, para $t < 1$, executamos a sub-rotina Buraco que recebe S, o ponto de intersecção (x,t), a representação da linha de varredura, as filas F_d e as pilhas P_e e devolve uma sequência de vértices do buraco B. No final, quando $t = 1$, executamos mais uma vez IniciaBuraco seguido de Buraco.

IniciaBuraco(S, $\{s_1^t, \ldots, s_n^t\}$, F_d, P_e)
    para $i \leftarrow 2$ **até** n **faça**
        InsereFila($(x_{s_i^t}(t), t), F_d(i)$)
        se $x_{s_i^0}(0) \neq x_{s_{i-1}^0}(0)$ **então**
            InserePilha($(x_{s_{i-1}^t}(t), t), P_e(i)$)

```
BURACO(S, (x,t), {s_1^t, ..., s_n^t}, F_d, P_e)
    sejam s_p^t, s_{p+1}^t, ..., s_q^t as trajetórias que se intersectam em (x,t)
    B ← {}
    para i ← p até q faça
        enquanto F_d(i) não está vazia faça
            p ← REMOVEFILA(F_d(i))
            B ← B + p
        B ← B + (x,t)
        enquanto P_e(i) não está vazia faça
            p ← REMOVEPILHA(P_e(i))
            B ← B + p
        INSEREFILA((x,t), F_d(i))
    devolve B
```

Seja m o número de pares de trajetórias em S que se intersectam. A varredura de Bentley e Ottmann encontra essas intersecções em tempo $O((n+m)\log n)$. O algoritmo INICIABURACO consome tempo $O(n)$ e só é executado em duas ocasiões, quando $t = 0$ e quando $t = 1$. O algoritmo BURACO consome tempo $O(nv)$ no pior caso, onde v é o número de vértices do buraco sendo construído. Ele é executado para cada ponto de intersecção encontrado. Uma análise amortizada mostra que a sequência de execuções do algoritmo BURACO tem tempo $O(m)$ no total. Isso porque, para cada par de trajetórias que se intersectam, a intersecção é inserida na fila e na pilha no máximo uma vez e é processada em no máximo uma chamada ao algoritmo BURACO. Portanto, o tempo total do algoritmo continua sendo $O((n+m)\log n)$.

Para finalizar, a medida que encontramos o ponto mais baixo de um buraco b, podemos obter o conjunto de trajetórias à esquerda de b que é $S_e(b) = \{s_1^t, s_2^t, \ldots, s_i^t\}$, onde i e t são tais que o vértice mais baixo de b foi inserido em $F_d(i)$ ou $P_e(i)$ no instante t.

3.3 Algoritmo para k fixo

Uma das aplicações de problemas de clustering é análise de dados, onde temos uma quantidade enorme de dados e queremos juntar dados semelhantes em um mesmo grupo para podermos analisar os grupos em vez de cada dado individualmente. Por isso, em geral, queremos um número de clusters pequeno em relação ao número de dados (no nosso caso, trajetórias). Ademais, quanto maior o número de clusters, maior será a frequência de clusters com uma única trajetória nos clusterings ótimos.

Considerando isso, uma hipótese razoável de se adotar é fixar o valor de k no problema kCC1D-S. Isto é, ele não faz parte da entrada do problema. Mostraremos que, com essa hipótese, o kCC1D-S pode ser resolvido em tempo polinomial.

Sejam n e k números inteiros não negativos. O número de Stirling de segundo tipo conta de quantas formas diferentes é possível particionar um conjunto com n elementos em k subconjuntos não vazios. Tal número é denotado por

$$\left\{ {n \atop k} \right\} = \frac{1}{k!} \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} i^n.$$

Rennie e Dobson [RD69] provaram a seguinte desigualdade:

$$\left\{ {n \atop k} \right\} \geqslant \frac{1}{2}(k^2 + k + 2)k^{n-k-1} - 1 = \Omega(k^{n-k+1}).$$

Ou seja, dado um conjunto S de n trajetórias e um inteiro não negativo k, levaria tempo exponencial no tamanho da entrada para verificar todos os possíveis k-clusterings de S em busca de um ótimo, mesmo considerando k como um valor fixo. Mostraremos que não precisamos verificar todos os k-clusterings.

Dado um k-clustering $\mathcal{C}$ de S, um buraco que está dentro da região de um cluster de $\mathcal{C}$ está **coberto** por $\mathcal{C}$, caso contrário ele está **descoberto**. Um buraco b **separa** os clusters C_1 e C_2 de $\mathcal{C}$ se $C_1 \subseteq S_e(b)$ e $C_2 \subseteq S_d(b)$, ou $C_1 \subseteq S_d(b)$ e $C_2 \subseteq S_e(b)$. Quando não houver dúvidas, a referência para o k-clustering será omitida. Com isso, podemos provar os seguintes lemas.

Lema 3.4. Sejam S um conjunto de trajetórias e $\mathcal{C}^*$ um k-clustering ótimo para o kCC1D-S. Dois clusters distintos de $\mathcal{C}^*$ devem ser separados por um buraco de B.

Demonstração. Suponha que existam clusters C_1 e C_2 em $\mathcal{C}^*$ tais que não exista um buraco de B separando-os. Então, região$(C_1) \cap$ região$(C_2) \neq \emptyset$, caso contrário existiria um buraco de B os separando. Ademais, região$(C_1) \cup$região$(C_2) =$ região$(C_1 \cup C_2)$. Logo, $\text{diam}(C_1) + \text{diam}(C_2) > \text{diam}(C_1 \cup C_2)$. Logo, juntando C_1 e C_2 em um único cluster resultaria em um k-clustering melhor, o que é um absurdo dado que $\text{sd}(\mathcal{C}^*)$ é mínimo. Portanto, para quaisquer dois clusters distintos em $\mathcal{C}^*$, existe um buraco separando-os. □

O Lema 3.4 não garante que sempre existirá um k-clustering ótimo com $k-1$ buracos de B descobertos. Isso porque é possível que todos os buracos que separam dois determinados clusters estejam cobertos por outros clusters. Um exemplo disso é ilustrado pela Figura 3.4 e um caso concreto é apresentado no Apêndice C. No entanto, podemos garantir a existência de pelo menos um buraco descoberto por um k-clustering ótimo, para $k > 1$.

Seja S um conjunto de trajetórias. Dizemos que uma trajetória s em S é a **mais à esquerda** de S se $x_s(0)$ é mínimo em S e, em caso de empates, $x_s(1)$ é mínimo entre as trajetórias empatadas.

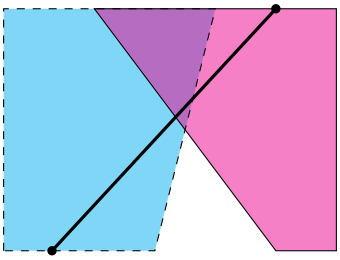


Figura 3.4: Um 3-clustering que não é bem-separado.

Lema 3.5. Sejam S um conjunto de trajetórias e $\mathcal{C}^*$ um k-clustering ótimo para o kCC1D-S. Para todo $k > 1$, existe pelo menos um buraco de B descoberto.

Demonstração. Suponha que todo buraco de B esteja coberto por $\mathcal{C}^*$. Então,

$$\mathrm{sd}(\mathcal{C}^*) = \sum_{C \in \mathcal{C}^*} \mathrm{diam}(C) \geqslant \sum_{b \in B} \text{área}(b).$$

Seja s a trajetória mais à esquerda de S. Existe um buraco b_s separando $\{s\}$ e $S \setminus \{s\}$. Considere o k-clustering $\mathcal{C}$ com os clusters $\{s\}$, $S \setminus \{s\}$ e os demais clusters vazios. Então,

$$\mathrm{sd}(\mathcal{C}) = \sum_{b \in B} \text{área}(b) - \text{área}(b_s) < \sum_{b \in B} \text{área}(b) \leqslant \mathrm{sd}(\mathcal{C}^*).$$

Isso contradiz a otimalidade de $\mathcal{C}^*$. Portanto, existe um buraco descoberto por $\mathcal{C}^*$. □

Seja $\mathcal{C}^*$ um k-clustering de S ótimo, para $k > 1$. Pelo Lema 3.5, sabemos que existe um buraco b em B que está descoberto. Logo, todo cluster de $\mathcal{C}^*$ está contido em $S_e(b)$ ou está contido em $S_d(e)$, nunca intersectando $S_e(b)$ e $S_d(b)$ simultaneamente. Suponha que k_e dos clusters de $\mathcal{C}^*$ estão contidos em $S_e(b)$ e k_d dos clusters de $\mathcal{C}^*$ estão contidos em $S_d(b)$. Claramente, $k_e \geqslant 1$ e $k_d = k - k_e \geqslant 1$. Como o diâmetro de um cluster não interfere no diâmetro de outro cluster, temos que $\mathcal{C}^*$ consiste de um k_e-clustering ótimo de $S_e(b)$ unido com um k_d-clustering ótimo de $S_d(b)$. Ou seja, para $k \geqslant 1$ e convencionando que $\mathrm{sd}^*(S, k') = \infty$ se $k' \leqslant 0$ ou $k' > |S|$, vale a seguinte recorrência

$$\mathrm{sd}^*(S, k) = \min_{\substack{b \in B, \\ k_e + k_d = k}} \{\mathrm{sd}^*(S_e(b), k_e) + \mathrm{sd}^*(S_d(b), k_d)\}. \tag{3.2}$$

Uma implementação direta da recorrência (3.2) resulta em um algoritmo com consumo de tempo $\Omega(2^n)$, pois teríamos que considerar todos os subconjuntos de S. Entretanto, a recorrência ainda pode ser aproveitada para construir um algoritmo para o kCC1D-S com complexidade de tempo polinomial, se considerarmos o valor de k fixo.

Considerando $\mathcal{C}_0 = \{S\}$, chamamos uma sequência $((b_i, C_i))_{i=1}^{k-1}$ de **k-sequência separadora** de S se, para $1 \leqslant i \leqslant k-1$,

$$\begin{aligned} &C_i \in \mathcal{C}_{i-1}, \\ &b_i \in B \text{ está contido em região}(C_i), \text{ e} \\ &\mathcal{C}_i = (\mathcal{C}_{i-1} \setminus \{C_i\}) \cup \{(C_i)_e(b_i), (C_i)_d(b_i)\}. \end{aligned} \tag{3.3}$$

Para $i \geqslant 1$, cada $\mathcal{C}_i$ é um $(i+1)$-clustering obtido ao particionar o cluster $C_i \in \mathcal{C}_{i-1}$ através do buraco b_i. Uma vez que a sequência $((b_i, C_i))_{i=1}^{k-1}$ esteja bem definida, podemos deduzir qual é a sequência $(\mathcal{C}_i)_{i=0}^{k-1}$. Logo, dizemos que a k-sequência separadora $((b_i, C_i))_{i=1}^{k-1}$ define um k-clustering de S.

Teorema 3.6. Seja S um conjunto de trajetórias. Todo k-clustering ótimo de S para o kCC1D-S é definido por alguma k-sequência separadora.

Demonstração. Prova por indução em k. Para $k = 1$, a afirmação é claramente verdade, pois $\mathcal{C}_0 = \{S\}$ é o único 1-clustering de S. Então, considere $k > 1$. Seja $\mathcal{C}^*$ um k-clustering ótimo de S.

Pelo Lema 3.5, existe um buraco b_1 descoberto. O buraco b_1, por ser descoberto, separa S em $S_1 = S_e(b_1)$ e $\bar{S}_1 = S_d(b_1)$, de modo que, para todo $C \in \mathcal{C}^*$, ou $C \subseteq S_1$ ou $C \subseteq \bar{S}_1$. Sejam $\mathcal{D} = \{C \in \mathcal{C}^* \mid C \subseteq S_1\}$, $k_1 = |\mathcal{D}|$, $\bar{\mathcal{D}} = \{C \in \mathcal{C}^* \mid C \subseteq \bar{S}_1\}$ e $\bar{k}_1 = |\bar{\mathcal{D}}|$. Note que $k_1 \geqslant 1$, $\bar{k}_1 \geqslant 1$ e $k_1 + \bar{k}_1 = k$. Ademais, $\mathcal{D}$ é um k_1-clustering ótimo de S_1 e $\bar{\mathcal{D}}$ é um $\bar{k}_1$-clustering ótimo de $\bar{S}_1$, caso contrário, $\mathcal{C}^* = \mathcal{D} \cup \bar{\mathcal{D}}$ não seria ótimo para S.

Por hipótese de indução, existe uma k-sequência separadora $((d_i, C_i))_{i=1}^{k_1-1}$ definindo $\mathcal{D}$ e existe uma k-sequência separadora $((\bar{d}_i, \bar{C}_i))_{i=1}^{\bar{k}_1-1}$ definindo $\bar{\mathcal{D}}$. Note que cada d_i é um buraco de S_1. Se d_i não é também um buraco de S, então d_i corresponde à união de dois ou mais buracos de S. Qualquer um desses buracos de S dentro de d_i separa S_1 da mesma forma que d_i. Logo, podemos trocar d_i por um dos buracos de S dentro dele. O mesmo pode ser feito com cada buraco $\bar{d}_i$ de $\bar{S}_1$.

Seja $((b_i, C_i))_{i=1}^{k-1}$ a concatenação de (b_1, S), $((d_i, C_i))_{i=1}^{k_1-1}$ e $((\bar{d}_i, \bar{C}_i))_{i=1}^{\bar{k}_1-1}$ nesta ordem, ou seja, $(b_1, S), (d_1, C_1), \ldots, (d_{k_1-1}, C_{k_1-1}), (\bar{d}_1, \bar{C}_1), \ldots, (\bar{d}_{\bar{k}_1-1}, \bar{C}_{\bar{k}_1-1})$. Note que S é um cluster de $\mathcal{C}_0$. Também temos que $C_1 = S_e(b_1)$ e $\bar{C}_1 = S_d(b_1)$ são clusters de $\mathcal{C}_1$. Em particular, $\bar{C}_1$ é um cluster de $\mathcal{C}_i$, para $1 \leqslant i \leqslant k_1$. Logo, $((b_i, C_i))_{i=1}^{k-1}$ é uma k-sequência separadora que define $\mathcal{C}^*$. □

Logo, em vez de verificarmos todos os $\left\{{n \atop k}\right\} = \Omega(k^{n-k+1})$ possíveis k-clusterings de S, podemos verificar todos os k-clusterings definidos por uma k-sequência separadora. O Lema 3.7 nos garante que a quantidade de tais sequências é polinomial no tamanho da entrada, para valores fixos de k.

Lema 3.7. Um conjunto com n trajetórias tem $O(n^{2(k-1)}(k-1)!)$ k-sequências separadoras.

Demonstração. Sabemos pelo Corolário 3.3 que $|B| = \mathrm{O}(n^2)$. Para um elemento (b_i, C_i) de uma k-sequência separadora, temos no máximo $|B| - i + 1$ opções de buracos e no máximo $|\mathcal{C}_{i-1}| = i$ opções de C_i. Logo, uma delimitação superior para o número de k-sequências separadoras é

$$\prod_{i=1}^{k-1} (|B| - i + 1)\, i = \mathrm{O}\left(\prod_{i=1}^{k-1} n^2 i\right) = \mathrm{O}\big(n^{2(k-1)}(k-1)!\big)\,. \qquad \square$$

Como mencionado na Seção 2.1.1, podemos calcular o diâmetro de um cluster com m trajetórias em tempo $\mathrm{O}(m \log m)$. Portanto, a complexidade de um algoritmo que verifica todos os k-clustering definidos por k-sequências separadoras é $\mathrm{O}((k-1)!n^{2k-1} \log n)$.

3.4 Clustering bem-separados

Como discutido na Seção 1.1, existem aplicações de problemas de clustering, como classificação, onde podemos querer que as "similaridades" entre clusters distintos sejam as menores possíveis. Por exemplo, um cluster cuja região está contida na união das regiões de dois outros clusters pode ser uma configuração indesejada, mesmo que isso ocorra em todo clustering ótimo para uma certa entrada.

Além disso, Daniely, Linial e Saks [DLS12], em seu artigo intitulado *"Clustering is difficult only when it does not matter"*, argumentam que, do ponto de vista teórico, problemas de clustering geralmente são tratados como problemas NP-difíceis por focarem na otimização de certas funções objetivo. No entanto, segundo Daniely *et al.*, o propósito prático de problemas de clustering não é otimizar tais funções objetivo, mas encontrar uma partição significativa do conjunto de dados considerado. Para isso os autores defendem que primeiro deve-se definir o que é um bom clustering antes de tentar determinar quando é possível encontrar um.

Considerando essas ideias, definimos que um k-clustering é **bem-separado** se seus clusters são dois-a-dois separados por um buraco descoberto. A Figura 3.4 mostra um exemplo de um 3-clustering que não é bem-separado, pois não existe um buraco descoberto separando a trajetória escura do cluster azul. No apêndice C, apresentamos um exemplo concreto onde o único 3-clustering ótimo não é bem-separado.

Podemos adaptar o algoritmo apresentado na Seção 3.3 para encontrar k-clusterings bem-separados ótimos para o kCC1D-S. Para isso, basta exigir que todo buraco b_i de uma k-sequência separadora seja descoberto. No entanto, apresentaremos uma forma melhor de encontrar k-clusterings bem-separados ótimos. Utilizaremos programação dinâmica e o algoritmo resultante terá complexidade de tempo polinomial, mesmo se considerarmos o valor de k como parte da entrada.

Denotamos por $\mathscr{C}_B = \bigcup_{b \in B} \{S_e(b), S_d(b)\}$ a coleção de clusters de S que são obtidos pela bipartição de S por algum buraco de B. Considere a ordem parcial $\preceq$ sobre $\mathscr{C}_B$

definida da seguinte forma: para todo C_1 e C_2 em $\mathscr{C}_B$, $C_1 \preceq C_2$ se e somente se $C_1 \subseteq C_2$. Denotamos por D_B o grafo acíclico dirigido (*dag*) que representa esta ordem parcial. Veja a Figura 3.5 para um exemplo. Note que $|V(D_B)| = |\mathscr{C}_B| \leqslant 2|B|$.

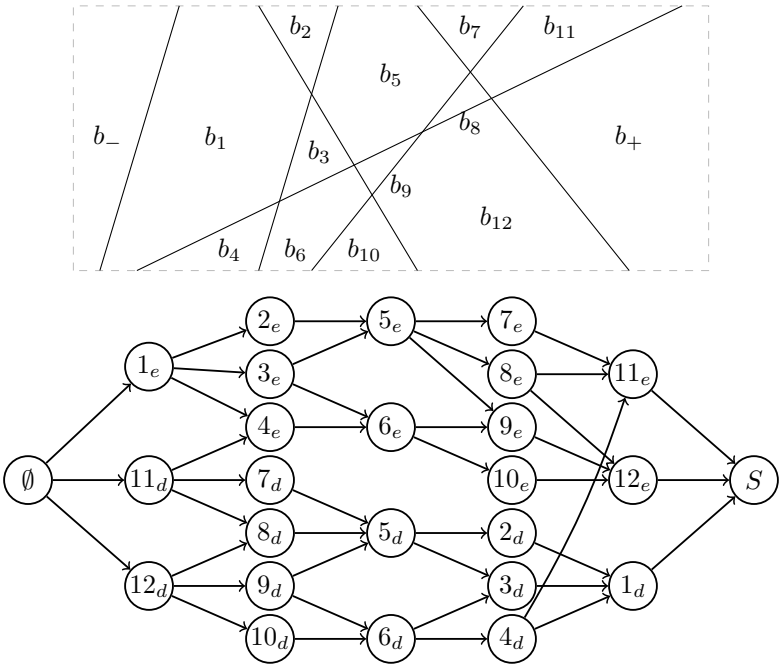


Figura 3.5: *Dag* D_B. Vértices i_e e i_d representam $S_e(b_i)$ e $S_d(b_i)$, respectivamente.

Lema 3.8. Seja $\mathcal{C}$ um k-clustering bem-separado de um conjunto S de trajetórias. Podemos ordenar os clusters de $\mathcal{C}$ em $C_1, C_2, \ldots, C_k$, de modo que, para todo $1 \leqslant i \leqslant k-1$,

$$\bigcup_{j=1}^{i} C_j \quad \text{e} \quad \bigcup_{j=i+1}^{k} C_j$$

são separados por algum buraco descoberto b_i de B.

Demonstração. Seja $\tilde{B}$ o conjunto de buracos de S descobertos por $\mathcal{C}$. Suponha que $k \geqslant 2$, caso contrário é trivial. Considere a seguinte construção.

$$A_1 = \operatorname*{arg\,min}_{A \in \mathscr{C}_{\tilde{B}}} |A|, \quad \text{e} \quad A_i = \operatorname*{arg\,min}_{\substack{A \in \mathscr{C}_{\tilde{B}}, \\ A_{i-1} \subset A}} |A|, \text{ para } 2 \leqslant i \leqslant k-1.$$

Considere também um buraco $b_i \in \tilde{B}$ tal que $A_i \in \{S_e(b_i), S_d(b_i)\}$, para $1 \leqslant i \leqslant k-1$.

Tome $C_1 = A_1$, $C_i = A_i \setminus A_{i-1}$, para $2 \leqslant i \leqslant k-1$, e $C_k = S \setminus A_{k-1}$. Vale que $\bigcup_{j=1}^{i} C_j$ e $\bigcup_{j=i+1}^{k} C_j$ são separados por b_i, para todo $1 \leqslant i \leqslant k-1$. Falta mostrar que essa construção está bem definida, isto é, nenhum dos conjuntos envolvidos é vazio, e que $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$.

Como $\mathcal{C}$ é bem-separado, temos que $|\tilde{B}| \geqslant k-1 \geqslant 1$. Logo, C_1 existe e não é vazio. Ademais C_1 é um cluster de $\mathcal{C}$. Caso contrário existiriam pelo menos dois clusters de $\mathcal{C}$ contidos em C_1. Mas esses dois clusters devem ser separados por um buraco descoberto b, implicando que $\min(|S_e(b)|, |S_d(b)|) < |C_1|$, contradizendo a escolha de C_1.

Seja $2 \leqslant i \leqslant k-1$. Suponha que $C_1, C_2, \ldots, C_{i-1}$ não são vazios e são clusters de $\mathcal{C}$. Como $\mathcal{C}$ é um k-clustering bem-separado, os outros $k-i+1 \geqslant 2$ clusters de $\mathcal{C}$ estão contidos em $S \setminus A_{i-1}$. Logo, existe um buraco b em $\tilde{B}$ separando algum par de clusters contidos em $S \setminus A_{i-1}$. Assim, podemos concluir que A_{i-1} está propriamente contido em $S_e(b)$ ou em $S_d(b)$, implicando que A_i e C_i não são vazios. Como A_i tem cardinalidade mínima e $A_{i-1} \subset A_i$, pelo mesmo argumento utilizado anteriormente para C_1, temos que $A_i \setminus A_{i-1}$, de fato, é um cluster de $\mathcal{C}$. Portanto, $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$. □

Uma k-**cadeia** é uma sequência $(C_i)_{i=1}^{k-1}$ de $k-1$ elementos distintos de $\mathscr{C}_B \setminus \{S\}$ tal que $C_1 \preceq C_2 \preceq \cdots \preceq C_{k-1}$. Note que, para qualquer k-cadeia, $C_{k-1} \preceq S$. Uma k-cadeia define o k-clustering

$$\{C_1,\, C_2 \setminus C_1,\, C_3 \setminus C_2, \ldots,\, C_{k-1} \setminus C_{k-2},\, S \setminus C_{k-1}\}.$$

Teorema 3.9. Todo k-clustering bem-separado de um conjunto de trajetórias S é definido por uma k-cadeia.

Demonstração. Seja $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ um k-clustering bem-separado de S. Considere que os clusters de $\mathcal{C}$ estão ordenados como no Lema 3.8. Para todo $1 \leqslant i \leqslant k-1$, existe um buraco descoberto b_i tal que $S_i = \cup_{j=1}^{i} C_j \in \{S_e(b_i), S_d(b_i)\}$. Considere $S_k = S$. Logo, $S_i \in \mathscr{C}_B$ e $S_i \subseteq S_{i+1}$. Portanto, $(\cup_{j=1}^{i} C_j)_{i=1}^{k-1}$ é uma k-cadeia que define $\mathcal{C}$. □

Observe que $\emptyset$ e S são a fonte e o sorvedouro de D_B, respectivamente. Para cada C em $\mathscr{C}_B$ e cada $j \geqslant 1$, seja sd-rec(C, j) o mínimo da soma dos diâmetros dos clusters de um k-clustering bem-separado de $S \setminus C$. Vale a seguinte recorrência.

$$\text{sd-rec}(C, j) = \begin{cases} 0, & \text{se } C = S \\ \operatorname{diam}(S \setminus C), & \text{se } j = 1 \\ \min\limits_{C \preceq C'} \{\, \operatorname{diam}(C' \setminus C) + \text{sd-rec}(C', j-1) \,\}, & \text{caso contrário.} \end{cases} \tag{3.4}$$

Dado um conjunto de trajetórias S e um número inteiro k,

$$\text{sd}^*(S, k) = \text{sd-rec}(\emptyset, k).$$

Note que, para calcular o valor de sd-rec$(\emptyset, k)$, a recorrência (3.4) basicamente verifica todas as k-cadeias de S.

Uma implementação direta, usando programação dinâmica, da recorrência (3.4) para calcular sd-rec$(\emptyset, k)$ consiste em preencher uma matriz $|\mathscr{C}_B| \times k$, cujas linhas representam

elementos de $\mathscr{C}_B$ e cujas colunas representam valores de j na recorrência (3.4). Cada posição na linha S da matriz pode ser preenchida em tempo $O(1)$ e cada posição na coluna $j = 1$ pode ser preenchida em tempo $O(n \log n)$. No primeiro caso, sabemos que o valor é sempre zero e no segundo caso precisamos calcular o diâmetro de um conjunto de trajetórias. As demais posições da matriz são preenchidas em uma ordem na qual, ao preencher a posição (C, j), todas as outras posições (C', j'), para $C \preceq C'$ e $j' \leqslant j$, já estão preenchidas. Dessa forma, o cálculo de sd-rec(C, j) leva tempo $O(|\mathscr{C}_B|\, n \log n)$. Portanto, é possível calcular sd-rec$(\emptyset, k)$ em tempo $O(k|\mathscr{C}_B|^2\, n \log n) = O(n^6 \log n)$.

3.5 Aproximação para $k = 3$

Para o caso particular no qual $k = 3$, conseguimos provar que um 3-clustering bem-separado ótimo para o kCC1D-S é uma 2-aproximação para o caso geral.

Lema 3.10. Sejam S um conjunto de n trajetórias e $\mathcal{C}^*$ um 3-clustering ótimo de S. Se $\mathcal{C}'$ é um 3-clustering bem-separado de S que é ótimo entre todos os 3-clusterings bem-separados de S, então

$$\mathrm{sd}(\mathcal{C}') \leqslant 2\,\mathrm{sd}(\mathcal{C}^*).$$

Demonstração. Para o caso em que $\mathrm{sd}(\mathcal{C}^*) = \mathrm{sd}(\mathcal{C}')$, não há nada a ser provado. Então, suponha que $\mathrm{sd}(\mathcal{C}^*) < \mathrm{sd}(\mathcal{C}')$.

Neste caso, temos que $\mathcal{C}^*$ não é bem-separado. Mas, pelo Lema 3.5, sabemos que $\mathcal{C}^*$ tem pelo menos um buraco descoberto. Suponha, sem perda de generalidade, que esse buraco deixa os clusters C_1 e C_2 de $\mathcal{C}^*$ à sua esquerda e deixa o cluster C_3 de $\mathcal{C}^*$ à sua direita.

Pelo Lema 3.4, existe pelo menos um buraco separando C_1 e C_2. Seja B' o conjunto de buracos de S que separam C_1 de C_2. Os buracos em B' são cobertos por C_3. Logo, $\sum_{b \in B'} \text{área}(b) \leqslant \mathrm{diam}(C_3)$. Note que existe um buraco b' em $C_1 \cup C_2$ que separa uma única trajetória que compõe o lado esquerdo de $C_1 \cup C_2$ das demais trajetórias de $C_1 \cup C_2$ e que não é coberto por C_3. Sejam D_1 um conjunto contendo apenas tal trajetória e $D_2 = (C_1 \cup C_2) \setminus D_1$. Note que

$$\mathrm{diam}(D_1) + \mathrm{diam}(D_2) \leqslant \mathrm{diam}(C_1 \cup C_2) \leqslant \mathrm{diam}(C_1) + \mathrm{diam}(C_2) + \sum_{b \in B'} \text{área}(b).$$

Tome $\mathcal{D} = \{D_1, D_2, C_3\}$. Observe que $\mathcal{D}$ é um 3-clustering bem-separado de S. Ademais,

$$\begin{aligned}
\mathrm{diam}(D_1) + \mathrm{diam}(D_2) + \mathrm{diam}(C_3) &\leqslant \mathrm{diam}(C_1) + \mathrm{diam}(C_2) + \mathrm{diam}(C_3) + \sum_{b \in B'} \text{área}(b) \\
&\leqslant \mathrm{diam}(C_1) + \mathrm{diam}(C_2) + 2\,\mathrm{diam}(C_3) \\
&\leqslant 2\,\mathrm{sd}(\mathcal{C}^*).
\end{aligned}$$

Como $\mathcal{C}'$ é um 3-clustering ótimo dentre os 3-clusterings bem-separados de S, temos que $\mathrm{sd}(\mathcal{C}') \leqslant \mathrm{sd}(\mathcal{D})$. Portanto, $\mathrm{sd}(\mathcal{C}') \leqslant 2\,\mathrm{sd}(\mathcal{C}^*)$. □

O algoritmo da Seção 3.4 encontra um 3-clustering bem-separado ótimo em tempo $\mathrm{O}(n^5 \log n)$. É de se notar que o algoritmo da Seção 3.3 encontra um 3-clustering ótimo na mesma complexidade de tempo. No entanto, isso ocorre pelo fato de, no pior caso, $|\mathscr{C}_B| = \mathrm{O}(n^2)$. Se $|\mathscr{C}_B| = \mathrm{o}(n^2)$, encontrar um 3-clustering bem-separado ótimo é bem mais rápido do que encontrar um 3-clustering ótimo.

Capítulo 4

Minimização do maior dos diâmetros

Agora trataremos do k-Clustering Cinético 1-Dimensional de Maior Diâmetro Mínimo (kCC1D-M), no qual são dados n pontos em $\mathbb{R}$, cada um com uma posição inicial e uma velocidade constante. Deseja-se encontrar um k-clustering cujo maior diâmetro de seus clusters seja mínimo. Podemos redefinir o problema kCC1D-M através das trajetórias dos pontos. Dado um k-clustering $\mathcal{C}$, denotamos por

$$\mathrm{md}(\mathcal{C}) = \max_{C \in \mathcal{C}} \mathrm{diam}(C) \tag{4.1}$$

o maior dos diâmetros dos clusters em $\mathcal{C}$.

Problema 4.1 (kCC1D-M)**.** Dado um conjunto de trajetórias S e um número inteiro positivo k, encontrar um k-clustering $\mathcal{C}$ de S com $\mathrm{md}(\mathcal{C})$ mínimo.

Seja $\mathcal{C}^*$ um k-clustering ótimo para um dado conjunto S de trajetórias e um dado número inteiro positivo k, isto é, $\mathrm{md}(\mathcal{C}^*)$ é mínimo entre todos os k-clusterings de S. Denotamos por $\mathrm{md}^*(S,k) = \mathrm{md}(\mathcal{C}^*)$ o valor de um k-clustering ótimo de S.

Na versão estática do kCC1D-M, temos n pontos em $\mathbb{R}$ e queremos encontrar um k-clustering onde o valor do maior diâmetro de seus clusters seja mínimo. Nesta versão, o diâmetro é definido como a maior distância entre um par de pontos no cluster. Podemos resolver essa versão estática em tempo polinomial em n reduzindo-a ao problema de encontrar um caminho em um grafo, com pesos nas arestas, tal que o maior peso de uma aresta nesse caminho seja mínimo, como mostraremos a seguir.

Considere que o conjunto de pontos é $[n]$ e que os pontos estão ordenados de forma crescente em relação às suas posições, isto é, $x_1 < x_2 < \cdots < x_n$. Seja $G = (V, E)$ um grafo tal que $V = \{(x_0, 0), (x_n, k)\} \cup \{(x_i, j) \mid 1 \leqslant j < k \text{ e } j \leqslant i \leqslant j + n - k\}$ e $E = \{\{(x_i, j), (x_\ell, j+1)\} \in V \times V \mid 0 \leqslant i < \ell \leqslant n \text{ e } 0 \leqslant j < k\}$. Note que cada aresta $\{(x_i, j), (x_\ell, j+1)\}$ em E pode ser interpretada como o conjunto $\{i+1, i+2, \ldots, \ell\}$. Logo, para cada aresta $\{(x_i, j), (x_\ell, j+1)\}$, associamos um peso igual ao diâmetro do conjunto correspondente, ou seja, $x_\ell - x_{i+1}$. Ver Figura 4.1.

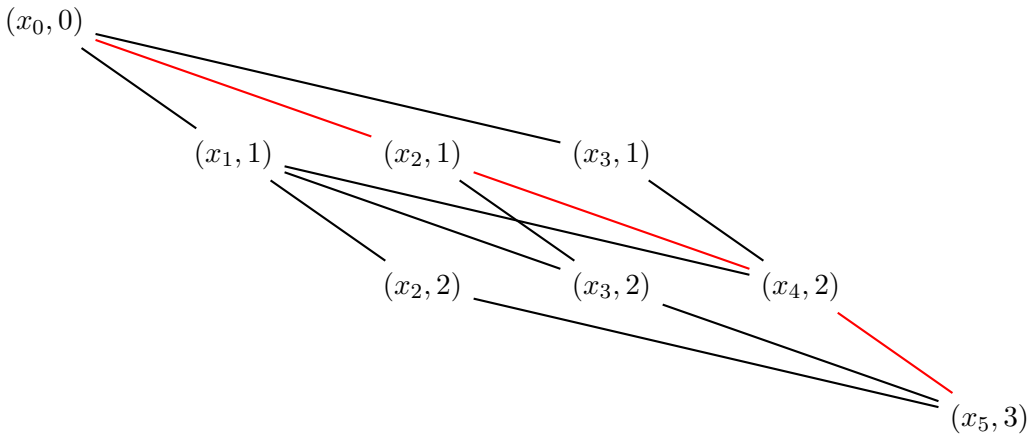


Figura 4.1: Grafo G para $n = 5$ pontos e $k = 3$. O caminho em vermelho corresponde ao 3-clustering $\{\{1, 2\}, \{3, 4\}, \{5\}\}$.

Note que todo caminho de $(x_0, 0)$ a (x_n, k) tem exatamente k arestas e juntas correspondem a um k-clustering dos n pontos. Brucker [Bru78] provou que sempre existe um k-clustering ótimo para a versão estática do kCC1D-M que corresponde a um desses caminhos. Assim, basta encontrar um caminho cujo maior peso de uma de suas arestas seja mínimo. Isso pode ser feito em tempo linear no número de arestas do grafo [Pun91], que nesse caso é $O(n^3)$.

Assim como no kCC1D-S, o caso particular do kCC1D-M no qual as trajetórias em S não se intersectam pode ser reduzido à sua versão estática e, consequentemente, resolvido em tempo polinomial. Considere a sequência $(s_i)_{i=1}^n$ das trajetórias de S ordenadas da esquerda para a direita. Tomando $x_1 = 0$ e $x_i = x_{i-1} + \text{diam}(\{s_{i-1}, s_i\})$, para $2 \leqslant i \leqslant n$, obtemos uma instância para a versão estática do kCC1D-M que corresponde ao conjunto de trajetórias S.

Podemos adaptar a programação dinâmica apresentada na Seção 3.4 para encontrar um k-clustering bem-separado cujo diâmetro máximo de um cluster seja mínimo. Essa programação dinâmica não encontrará um k-clustering ótimo para o kCC1D-M no caso geral. Já o algoritmo para k fixo, apresentado na Seção 3.3, não pode ser adaptado para o kCC1D-M, pois existem k-clusterings ótimos para o kCC1D-M que não são definidos por uma k-sequência separadora. A Figura 4.2 mostra um exemplo onde o único 2-clustering ótimo não é bem-separado e não é definido por uma k-sequência separadora. Nesse exemplo, o cluster vermelho e o preto têm ambos diâmetro 1. Qualquer outro 2-clustering terá diâmetro maior que 1.

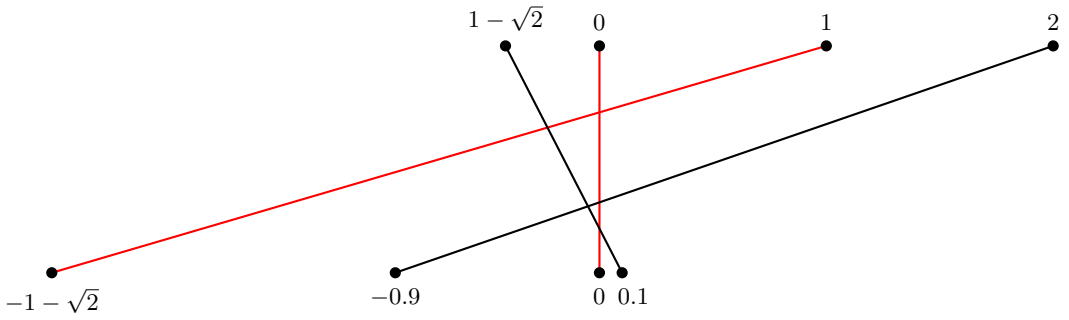


Figura 4.2: Para essas 4 trajetórias, o 2-clustering indicado é a única solução ótima para o kCC1D-M e não é definido por nenhum dos buracos.

4.1 Uma primeira aproximação

Nosso primeiro algoritmo de aproximação para o kCC1D-M é obtido por uma redução ao problema k-centros métrico.

Problema 4.2 (k-centros métrico)**.** Dado um conjunto finito S, um número inteiro positivo k e uma função de distância d sobre os elementos de S, encontrar um subconjunto X de S, com até k elementos, tal que $\max\{d(s, X) \mid s \in S\}$ seja mínimo, onde $d(s, X) = \min\{d(s, x) \mid x \in X\}$.

Os elementos do conjunto X procurado são chamados de **centros**. Basicamente, queremos encontrar até k centros tais que a maior distância de um elemento a um centro mais próximo a ele seja mínima. Note que cada centro x em X induz um cluster

$$C_x = \{s \in S \mid d(s, x) \leqslant d(s, x') \text{ para todo } x' \in X\}.$$

Caso um elemento s de S pertença a clusters de centros diferentes, o mantemos em um cluster escolhido arbitrariamente e o removemos dos demais. Dizemos que X **induz** um k-clustering $\mathcal{C} = \{C_x \mid x \in X\}$. De forma análoga, dizemos que um k-clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ é induzido por um conjunto de centros $X(\mathcal{C}) = \{x_1, x_2, \ldots, x_k\}$ tal que x_i é um elemento de C_i, para $1 \leqslant i \leqslant k$, que minimiza $\max_{s \in C_i} d(s, x_i)$.

O k-centros métrico é conhecidamente NP-difícil [Gon85, KH79a]. No entanto, existem 2-aproximações simples [Gon85, HS86], o que é o melhor possível a menos que P=NP [Gon85, HN79]. Para o nosso algoritmo de aproximação para o kCC1D-M, qualquer 2-aproximação do k-centros métrico pode ser utilizada.

A partir de uma instância (S, k) do kCC1D-M, construiremos uma instância para o k-centros métrico definindo a distância entre duas trajetórias s_1 e s_2 de S como $\text{diam}(\{s_1, s_2\})$. O Lema 4.3, abaixo, mostra que a função diam restrita a conjuntos de duas trajetórias satisfaz a desigualdade triangular. Assim, (S, k, diam) é uma instância do k-centros métrico, para a qual podemos aplicar uma 2-aproximação para obter um

conjunto X de centros e devolvemos um k-clustering qualquer induzido por X. O Teorema 4.6, mais adiante, mostra que esse algoritmo é uma $2(2+\sqrt{2})$-aproximação[1] para o kCC1D-M.

Lema 4.3. Seja S um conjunto de trajetórias. Para quaisquer três trajetórias a, b e c de S, temos que $\operatorname{diam}(\{a,b\}) + \operatorname{diam}(\{b,c\}) \geqslant \operatorname{diam}(\{a,c\})$.

Demonstração. Dadas duas trajetórias de S, digamos r e s, podemos escrever

$$\operatorname{diam}(\{r,s\}) = \int_0^1 \left(\max_{i\in\{r,s\}} x_i(t) - \min_{i\in\{r,s\}} x_i(t) \right) \mathrm{d}t = \int_0^1 |x_r(t) - x_s(t)| \, \mathrm{d}t.$$

Usando a linearidade de integrais, temos que

$$\begin{aligned} \operatorname{diam}(\{a,b\}) + \operatorname{diam}(\{b,c\}) &= \int_0^1 (|x_a(t) - x_b(t)| + |x_b(t) - x_c(t)|) \, \mathrm{d}t \\ &\geqslant \int_0^1 |x_a(t) - x_c(t)| \, \mathrm{d}t \\ &= \operatorname{diam}(\{a,c\}). \end{aligned}$$

□

Seja s uma trajetória de S. Para facilitar a leitura, abusaremos da notação e, pelo resto do capítulo, consideraremos $s^t = x_s(t)$ como uma posição e também consideraremos $s^t = (x_s(t), t)$ como um ponto no plano. Será claro pelo contexto se s^t denotará uma posição ou um ponto. Em particular, para $t = 1/2$, usaremos a notação $\bar{s} = (s^0 + s^1)/2$ em vez de $s^{1/2}$.

Lema 4.4. Se s e v são duas trajetórias tais que $|\bar{s} - \bar{v}| > r$ para algum $r > 0$, então $\operatorname{diam}(\{s,v\}) > r$.

Demonstração. Seja p uma trajetória paralela a s que passa por $\bar{v}$. Para qualquer t no intervalo $[0,1]$, temos que $\operatorname{diam}(\{s,p\}) = |s^t - p^t| > r$. Se $v = p$, então não há mais nada a provar. Assim, suponha que $v \neq p$.

Como p e v se intersectam em $t = 1/2$, então $|v^0 - p^0| = |v^1 - p^1|$. Logo, a região de $\{v,p\}$ consiste de dois triângulos congruentes com o vértice $\bar{v}$ em comum. Considere o caso no qual $\bar{s} < \bar{v}$ e $v^0 > p^0$, como mostra a Figura 4.3. Os demais casos são simétricos.

Sejam A, B, C e D as regiões indicadas na Figura 4.3. Se s não intersecta v ou a intersecta em um de seus extremos, então A é vazia. Note que

$$\begin{aligned} \operatorname{diam}(\{s,v\}) &= \text{área}(A) + \text{área}(C) + \text{área}(D), \\ \operatorname{diam}(\{p,v\}) &= \text{área}(A) + \text{área}(B) + \text{área}(D), \\ \operatorname{diam}(\{s,p\}) &= \text{área}(B) + \text{área}(C). \end{aligned}$$

[1] $2(2+\sqrt{2}) \approx 6{,}83$.

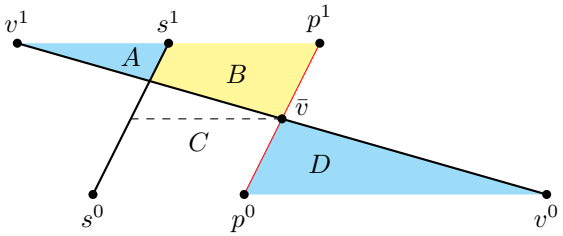


Figura 4.3: Cálculo do $\text{diam}(\{s, v\})$ em função de $\text{diam}(\{p, v\})$.

Ademais, como os dois triângulos que compõem a região de $\{v, p\}$ são congruentes, vale que $\text{área}(D) = \text{área}(A) + \text{área}(B)$ e, consequentemente, $\text{área}(D) \geqslant \text{área}(B)$. Portanto, $\text{diam}(\{s, v\}) = \text{área}(A) + (\text{diam}(\{s, p\}) - \text{área}(B)) + \text{área}(D) \geqslant \text{diam}(\{s, p\}) > r$. □

Lema 4.5. Sejam S um conjunto de trajetórias e s uma trajetória de S. Se r é tal que $\text{diam}(\{s, a\}) \leqslant r$ para toda trajetória a de S, então $\text{diam}(S) \leqslant (2 + \sqrt{2})r$.

Demonstração. Note que, para qualquer trajetória a tal que $|a^0 - s^0| > (1 + \sqrt{2})r$, temos que $\text{diam}(\{s, a\}) > r$, logo a não é uma trajetória de S. Qualquer trajetória a em S tal que $|a^0 - s^0| > 2r$ intersecta s e $|a^1 - s^1| < 2r$. Ambas observações também são verdadeiras se trocarmos a^0 com a^1 e s^0 com s^1.

Pelo Lema 4.4, para toda trajetória $a \in S$, temos que $|\bar{s} - \bar{a}| \leqslant r$. Note que o lado esquerdo de S é convexo e o lado direito é côncavo em relação ao eixo vertical. Com isso, podemos concluir que a região de S está contida no polígono P destacado na Figura 4.4.

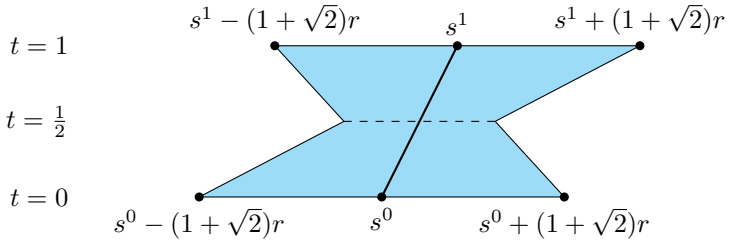


Figura 4.4: Polígono P contendo a região de S.

A área de P é dada pela soma das áreas de dois trapézios simétricos de altura $1/2$:

$$\text{área}(P) = 2\left(\frac{1}{2} \cdot \frac{2(1 + \sqrt{2})r + 2r}{2}\right) = (2 + \sqrt{2})r.$$

Portanto $\text{diam}(S) \leqslant \text{área}(P) \leqslant (2 + \sqrt{2})r$. □

Teorema 4.6. Sejam S um conjunto de trajetórias e $k \geqslant 2$ um número inteiro positivo. Se X é o conjunto de centros devolvido por uma 2-aproximação do k-centros aplicada à instância (S, k, diam), temos que o k-clustering induzido por X é uma $2(2 + \sqrt{2})$-aproximação para o kCC1D-M.

Demonstração. Sejam X^* um conjunto de centros que representa uma solução ótima para a instância (S, k, diam) do k-centros. Considere $r^* = \max\{d(s, X^*) \mid s \in S\}$ e $r = \max\{d(s, X) \mid s \in S\}$. Da 2-aproximação do k-centros, temos que $r \leqslant 2r^*$.

Seja $\mathcal{C}^* = \{C_1^*, C_2^*, \ldots, C_k^*\}$ um k-clustering ótimo para o kCC1D-M. Considere um conjunto de centros $X(\mathcal{C}^*) = \{x_i \in C_i^* \mid 1 \leqslant i \leqslant k\}$ que induz $\mathcal{C}^*$. Então,

$$r^* \leqslant \max_{s \in S} d(s, X(\mathcal{C}^*)) = \max_{C_i^* \in \mathcal{C}^*} \max_{s \in C_i^*} d(s, x_i) \leqslant \max_{C_i^* \in \mathcal{C}} \text{diam}(C_i^*) = \text{md}^*(S, k).$$

Seja $\mathcal{C}$ um k-clustering induzido por X. Note que cada cluster em $\mathcal{C}$ satisfaz as condições do Lema 4.5. Logo,

$$\text{md}(\mathcal{C}) = \max_{C \in \mathcal{C}} \text{diam}(C) \leqslant (2 + \sqrt{2})r \leqslant 2(2 + \sqrt{2})r^* \leqslant 2(2 + \sqrt{2})\,\text{md}^*(S, k).$$

Portanto, um k-clustering induzido por um conjunto de centros devolvido por uma 2-aproximação para o k-centros é uma $2(2 + \sqrt{2})$-aproximação para o kCC1D-M. □

4.2 Aproximação gulosa

Nesta seção apresentaremos um algoritmo que depende de um valor $\varepsilon > 0$ e tem um fator de aproximação $(2 + \sqrt{2}/2) + \varepsilon$, para qualquer $\varepsilon > 0$, para o kCC1D-M[2]. Esse algoritmo é inspirado no método de Hochbaum e Shmoys [HS86] para problemas de gargalo. Ele também pode ser visto como um refinamento do algoritmo anterior, pois basicamente adaptamos a ideia de uma particular 2-aproximação para o k-centros.

Primeiro descreveremos o algoritmo PartiçãoGulosa, que recebe um conjunto S de n trajetórias e um número positivo D. O algoritmo devolve um k'-clustering $\mathcal{C}_D$ para o qual $\text{md}(\mathcal{C}_D) \leqslant (2+\sqrt{2}/2)D$. Se $k' > k$, então podemos obter do algoritmo um certificado de que $\text{md}^*(S, k) > D$. Se $k' \leqslant k$ e $D = \text{md}^*(S, k)$, o algoritmo devolve uma $(2 + \sqrt{2}/2)$-aproximação para o kCC1D-M. No algoritmo principal faremos uma busca binária para encontrar um valor de D próximo a $\text{md}^*(S, k)$.

PartiçãoGulosa(S, D)
    se $S = \emptyset$ **então devolva** $\emptyset$
    seja s a trajetória mais à esquerda de S
    $C \leftarrow \{\}$
    para cada $s' \in S$ **faça**
        se $\text{diam}(\{s, s'\}) \leqslant D$ **então** $C \leftarrow C \cup \{s'\}$
    devolva $\{C\} \cup$ PartiçãoGulosa($S \setminus C$, D)

[2] $(2 + \sqrt{2}/2) \approx 2{,}71$.

Como, em cada chamada de PartiçãoGulosa, pelo menos uma trajetória é removida do conjunto de trajetórias, o algoritmo termina com no máximo $n-1$ chamadas recursivas. Em cada chamada, o algoritmo itera por todas as trajetórias do conjunto atual de trajetórias. O diâmetro de um conjunto de duas trajetórias pode ser calculado em tempo constante. Achar a trajetória mais à esquerda de S também pode ser feito em tempo constante se preprocessarmos S em tempo $O(n \log n)$, ordenando as trajetórias apropriadamente. Logo, o algoritmo PartiçãoGulosa tem complexidade de tempo $O(n^2)$.

Lema 4.7. Sejam S um conjunto de trajetórias e D um número positivo. Cada cluster construído por PartiçãoGulosa(S, D) tem diâmetro no máximo $(4+\sqrt{2})D/2$.

Demonstração. Seja s a trajetória mais à esquerda de S. Seja C o cluster construído por PartiçãoGulosa(S, D) que contém s. Qualquer trajetória u em C tem $u^0 \geqslant s^0$, já que s é a mais à esquerda. Também temos que $\text{diam}(\{s, u\}) \leqslant D$, então, pelos mesmos argumentos apresentados na prova do Lema 4.5, vale que $s^1 - (1+\sqrt{2})D \leqslant u^1 \leqslant s^1 + 2D$ e $u^0 \leqslant s^0 + (1+\sqrt{2})D$. Assim, a região de C está contida no polígono P destacado na Figura 4.5, cuja área é a soma de dois trapézios de altura $1/2$:

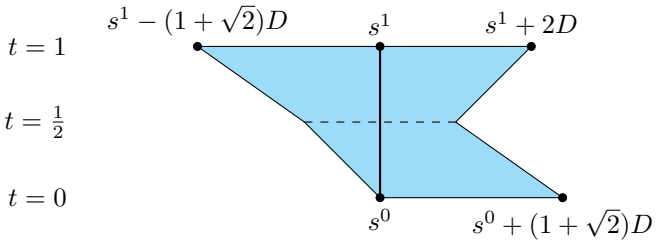


Figura 4.5: Polígono P contendo a região de C.

$$\text{área}(P) \leqslant \frac{((1+\sqrt{2})D + 2D)\frac{1}{2}}{2} + \frac{((2D + (1+\sqrt{2})D) + 2D)\frac{1}{2}}{2} = \left(2+\frac{\sqrt{2}}{2}\right) D.$$

Portanto, $\text{diam}(C) \leqslant \text{área}(P) \leqslant (2+\sqrt{2}/2)D$. □

Lema 4.8. Seja S um conjunto de trajetórias. Sejam k um número inteiro positivo e D um número real positivo. Suponha que o algoritmo PartiçãoGulosa(S, D) devolve um clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_{k'}\}$. Para $1 \leqslant i \leqslant k'$, seja s_i a trajetória mais à esquerda de C_i. Se $k' > k$, então $\{s_1, s_2, \ldots, s_{k'}\}$ é um certificado de que $\text{md}^*(S, k) > D$.

Demonstração. Sejam s_i e s_j duas trajetórias distintas de $\{s_1, s_2, \ldots, s_{k'}\}$. Como s_i e s_j pertencem a clusters diferentes, $\text{diam}(\{s_i, s_j\}) > D$, caso contrário PartiçãoGulosa as colocaria no mesmo cluster. Se $k' > k$, então, em qualquer k-clustering de S, algum cluster conterá pelo menos duas trajetórias em $\{s_1, s_2, \ldots, s_{k'}\}$. Logo, o diâmetro desse cluster será maior que D. Portanto, se $k' > k$, então $\text{md}^*(S, k) > D$. □

Como mostrado pelos Lemas 4.7 e 4.8, PartiçãoGulosa é similar ao que Hochbaum e Shmoys [HS86] chamam de procedimento de decisão relaxado para problemas de gargalo. Isto é, ou PartiçãoGulosa dá um certificado de que $\mathrm{md}^*(S,k) > D$, ou devolve uma aproximação de uma solução ótima. No entanto, diferentemente do que ocorre em problemas de gargalo, não podemos restringir os possíveis valores de $\mathrm{md}^*(S,k)$ a um conjunto de tamanho polinomial. Por isso aplicamos uma busca binária aproximada para definir o valor do parâmetro D diferindo por no máximo $\varepsilon > 0$ de $\mathrm{md}^*(S,k)$. Isso é feito pelo algoritmo k-clusteringBB1.

k-clusteringBB1(S, k, ε)
    $a \leftarrow 0$
    $b \leftarrow \mathrm{diam}(S)$
    $\delta \leftarrow \frac{2\varepsilon}{4+\sqrt{2}} \min_{u,v \in S,\, u \neq v} \mathrm{diam}(\{u,v\})$
    enquanto $(b - a > \delta)$ **faça**
        $D \leftarrow \frac{a+b}{2}$
        $\mathcal{C} \leftarrow$ PartiçãoGulosa(S, D)
        se $|\mathcal{C}| > k$ **então**
            $a \leftarrow D$
        senão
            $b \leftarrow D$
    se $|\mathcal{C}| > k$ **então**
        devolva PartiçãoGulosa(S, b)
    senão
        devolva $\mathcal{C}$

Inicialmente o intervalo de busca para o valor de D é $[0, \mathrm{diam}(S)]$. Após i iterações, o tamanho do intervalo de busca cai pela metade i vezes, ficando com comprimento $2^{-i}\,\mathrm{diam}(S)$. Assim, com $i \leqslant \log \mathrm{diam}(S) - \log \delta$ iterações, para δ definido como no algoritmo, a busca termina com um intervalo de comprimento δ.

Teorema 4.9. Sejam S um conjunto de n trajetórias e k um número inteiro positivo. Para todo $\varepsilon > 0$, k-clusteringBB1(S, k, ε) é uma $((2 + \sqrt{2}/2) + \varepsilon)$-aproximação para o kCC1D-M.

Demonstração. Primeiro, note que o algoritmo sempre devolve um k-clustering, pois o valor de b no algoritmo começa por $\mathrm{diam}(S)$. Ademais, pelo Lema 4.8, sabemos que, em toda iteração, o valor $\mathrm{md}^*(S,k)$ está no intervalo $[a,b]$ e no final $b - a \leqslant \delta$. Assim, ao final, $b \leqslant \mathrm{md}^*(S,k) + \delta$.

Suponha que $k < n$, caso contrário encontrar uma solução ótima é trivial. Então, um dos clusters em qualquer solução ótima contém pelo menos duas trajetórias. Logo, $\mathrm{md}^*(S,k) \geqslant \min\{\mathrm{diam}(\{u,v\}) \mid u, v \in S,\, u \neq v\}$.

Como $\delta = \frac{2\varepsilon}{4+\sqrt{2}} \min\{\operatorname{diam}(\{u,v\}) \mid u,v \in S,\, u \neq v\}$, a busca terminará em no máximo

$$\log \operatorname{diam}(S) + \log(4+\sqrt{2}) - \log(2\varepsilon \min\{\operatorname{diam}(\{u,v\}) \mid u,v \in S,\, u \neq v\})$$

iterações. Ademais, ao final do algoritmo, temos que

$$\begin{aligned}
\operatorname{md}(\mathcal{C}) &\leqslant \frac{4+\sqrt{2}}{2}\, b \leqslant \frac{4+\sqrt{2}}{2}(\operatorname{md}^*(S,k) + \delta) \\
&= \frac{4+\sqrt{2}}{2} \operatorname{md}^*(S,k) + \frac{4+\sqrt{2}}{2} \frac{2\varepsilon}{4+\sqrt{2}} \min_{\substack{u,v \in S \\ u \neq v}} \operatorname{diam}(\{u,v\}) \\
&= \frac{4+\sqrt{2}}{2} \operatorname{md}^*(S,k) + \min_{\substack{u,v \in S \\ u \neq v}} \operatorname{diam}(\{u,v\})\varepsilon \\
&\leqslant \left(\frac{4+\sqrt{2}}{2} + \varepsilon \right) \operatorname{md}^*(S,k).
\end{aligned}$$

Portanto, k-CLUSTERINGBB1(S,k,ε) é de fato uma $((2+\sqrt{2}/2)+\varepsilon)$-aproximação para o kCC1D-M. □

Capítulo 5

Mudança controlada de velocidade

No nosso modelo unidimensional consideramos que a velocidade dos pontos é sempre constante. Isso parece uma restrição muito forte. No entanto, podemos mostrar que, se a velocidade não for constante, mas for "controlada", então é possível aproximar tal velocidade por uma que seja constante e, usando o modelo unidimensional, obter soluções boas para o kCC1D-S e o kCC1D-M.

5.1 Movimento variável controlado

Consideramos agora o modelo unidimensional com movimento variável. Diferentemente do modelo unidimensional, a velocidade das trajetórias não precisa mais ser constante. O modelo é definido a seguir.

Primeiro, estendemos a definição de trajetória. No modelo unidimensional, uma trajetória consistia de uma função linear no intervalo $[0, 1]$. No modelo unidimensional com movimento variável, uma **trajetória** consiste de uma função contínua no intervalo $[0, 1]$. No caso particular dessa função contínua ser linear, dizemos que a trajetória é **linear**. Denotamos por $\mathcal{S}$ o conjunto de todas as trajetórias e por $\mathcal{S}' \subseteq \mathcal{S}$ o conjunto de todas as trajetórias lineares.

Note que a definição de diâmetro de um cluster de trajetórias lineares continua válida para trajetórias não lineares, pois a função que descreve o movimento é contínua no intervalo $[0, 1]$. Logo, os problemas kCC1D-S e kCC1D-M também podem ser estendidos para considerar trajetórias não necessariamente lineares. No restante do capítulo, consideramos que kCC1D-S e kCC1D-M correspondem aos respectivos problemas com trajetórias não necessariamente lineares. Ainda não temos resultados sobre a dificuldade do kCC1D-S com trajetórias não lineares. No caso do kCC1D-M, mostraremos que ele é difícil com trajetórias não lineares. Para demonstrar isso introduziremos o problema Partição, conhecidamente NP-completo [Kar72].

Problema 5.1 (Partição)**.** Dados m números inteiros positivos, decidir se é possível particioná-los em dois multiconjuntos tais que a soma dos números de cada multiconjunto seja a mesma.

Teorema 5.2. O kCC1D-M é NP-difícil com trajetórias não lineares.

Demonstração. Considere uma instância do Partição que consiste no multiconjunto de números inteiros positivos $A = \{a_1, a_2, \ldots, a_m\}$, para $m \geqslant 3$. Mostraremos como construir uma instância do kCC1D-M com trajetórias não lineares a partir dela. Tome $n = m$ e $k = 2$. Para cada $1 \leqslant i \leqslant n$, considere uma trajetória s_i que faz pelo menos duas mudanças de velocidade, definida pela função

$$x_{s_i}(t) = \begin{cases} 0, & \text{se } t \leqslant \frac{i-1}{n} \text{ ou } t \geqslant \frac{i}{n}, \\ 4a_i n(nt - i + 1), & \text{se } \frac{i-1}{n} < t \leqslant \frac{i-1}{n} + \frac{1}{2n}, \\ 4a_i n(i - nt), & \text{se } \frac{i-1}{n} + \frac{1}{2n} < t < \frac{i}{n}. \end{cases}$$

Note que $x_{s_i}(t) = 0$ exceto no intervalo aberto $((i-1)/n, i/n)$ e é contínua no intervalo $[0, 1]$. Ademais, a integral de x_{s_i} no intervalo $[(i-1)/n, i/n]$ vale a_i. A Figura 5.1 ilustra um exemplo com as trajetórias s_1 e s_2 considerando $n = 4$, $a_1 = 1$ e $a_2 = 2$.

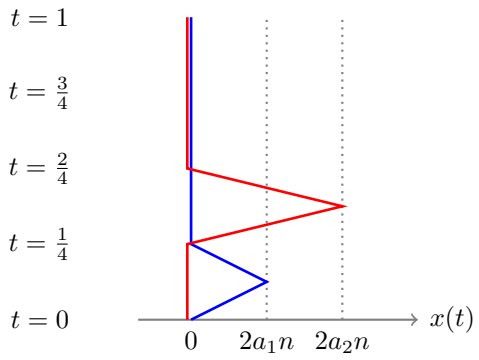


Figura 5.1: Trajetórias s_1 (azul) e s_2 (vermelha) para $n = 4$, $a_1 = 1$ e $a_2 = 2$. O pequeno espaço entre as trajetórias é apenas para facilitar a visualização.

Essa construção é polinomial pois, para cada trajetória s, a função x_s é linear por partes com no máximo quatro partes lineares. Logo, x_s pode ser representada pela sequência de pontos que são extremos de cada parte linear.

Seja $\mathcal{C}^* = \{C_1, C_2\}$ um 2-clustering ótimo para o kCC1D-M dessas n trajetórias. A partir de $\mathcal{C}^*$, particione A em

$$A_1 = \{a_i \in A \mid 1 \leqslant i \leqslant n \text{ e } s_i \in C_1\} \quad \text{e} \quad A_2 = \{a_i \in A \mid 1 \leqslant i \leqslant n \text{ e } s_i \in C_2\}.$$

Note que, de fato, $\{A_1, A_2\}$ é uma partição de A. Também não é difícil observar que, para $1 \leqslant j \leqslant 2$, se $|C_j| \geqslant 2$, $\operatorname{diam}(C_j) = \sum_{a \in A_j} a$. Se $|C_j| \leqslant 1$, $\operatorname{diam}(C_j) = 0$.

Suponha, sem perda de generalidade, que $\text{diam}(C_1) \geqslant \text{diam}(C_2)$. Sejam $M = \sum_{i=1}^{n} a_i$ e $D = \text{diam}(C_1)$. Como $n \geqslant 3$, temos que $|C_1| \geqslant 2$ e, consequentemente, $D \geqslant M/2$. Se $D = M/2$, a resposta para o PARTIÇÃO é positiva para A, pois $M/2 = \text{diam}(C_1) = \sum_{a \in A_1} a$. Logo, $\sum_{a \in A_2} a = M - M/2 = M/2$.

Agora, suponha que existe uma partição $\{B_1, B_2\}$ de A tal que $\sum_{b \in B_1} b = \sum_{b \in B_2} b = M/2$, ou seja, a resposta do PARTIÇÃO para A é positiva. Por um processo inverso ao utilizado para construir A_1 e A_2, podemos obter um 2-clustering $\mathcal{C}' = \{C'_1, C'_2\}$ tal que $\max(\text{diam}(C'_1), \text{diam}(C'_2)) = M/2$, implicando que $\mathcal{C}'$ é ótimo para o kCC1D-M com pelo menos três trajetórias.

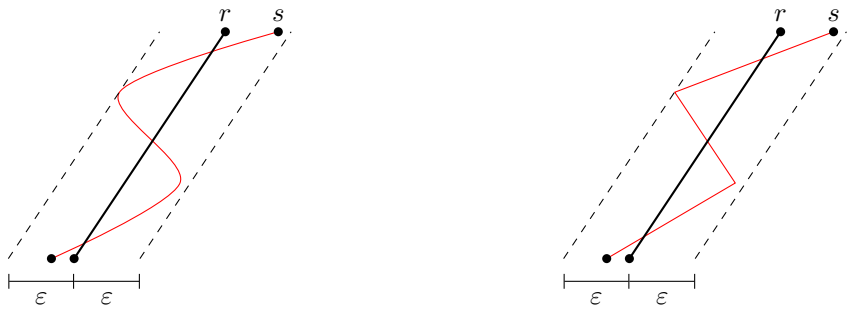
Portanto, o PARTIÇÃO tem resposta positiva para A se e somente se $D = M/2$. Logo, kCC1D-M com trajetórias não lineares é tão difícil quanto o PARTIÇÃO. □

Seja $\varepsilon > 0$. Dizemos que uma trajetória r é uma **ε-representante linear** de uma trajetória s qualquer se r é linear e, para todo t em $[0, 1]$,

$$x_r(t) - \varepsilon \leqslant x_s(t) \leqslant x_r(t) + \varepsilon.$$

A Figura 5.2 mostra um exemplo. Para toda trajetória s, seja $R_\varepsilon(s)$ o conjunto de todas as ε-representantes lineares de s. Um conjunto de trajetórias lineares $S_\varepsilon \subseteq \mathcal{S}'$ também é chamado de **ε-representante linear** de $S \subseteq \mathcal{S}$ se existir uma bijeção $r : S \to S_\varepsilon$ tal que, para todo s em S, a trajetória $r(s)$ é uma ε-representante linear de s. Ou seja, S_ε é um sistema de representantes da coleção $\{R_\varepsilon(s) \mid s \in S\}$. Nos casos em que não houver ambiguidade, denotaremos $r(s)$ apenas por r.

Se o valor de ε for muito pequeno, pode existir s em S com $R_\varepsilon(s)$ vazio. Neste caso, S não possui ε-representante linear. No entanto, não é difícil observar que sempre existe um $\varepsilon_0 > 0$, que depende apenas de S, tal que S tem um ε-representante linear para todo $\varepsilon \geqslant \varepsilon_0$. O menor valor de ε para o qual S possui um ε-representante linear é chamado de ε **ótimo** de S.



(a) (b)

Figura 5.2: Exemplos de trajetórias não lineares e respectivas ε-representantes lineares.

Lema 5.3. Seja $\varepsilon > 0$. Se S_ε é um ε-representante linear de um conjunto de trajetórias S, então

$$|\operatorname{diam}(S) - \operatorname{diam}(S_\varepsilon)| \leqslant 2\varepsilon.$$

Demonstração. Pela definição de diâmetro e linearidade de integrais, temos que

$$|\operatorname{diam}(S) - \operatorname{diam}(S_\varepsilon)| = \left| \int_0^1 \left(\max_{s\in S} x_s(t) - \min_{s\in S} x_s(t) - \max_{r\in S_\varepsilon} x_r(t) + \min_{r\in S_\varepsilon} x_r(t) \right) \mathrm{d}t \right|.$$

Para qualquer $t \in [0,1]$, temos que

$$-\varepsilon \leqslant \max_{s\in S} x_s(t) - \max_{r\in S_\varepsilon} x_r(t) \leqslant \varepsilon,$$
$$-\varepsilon \leqslant \min_{s\in S} x_s(t) - \min_{r\in S_\varepsilon} x_r(t) \leqslant \varepsilon.$$

Portanto,

$$|\operatorname{diam}(S) - \operatorname{diam}(S_\varepsilon)| \leqslant \int_0^1 2\varepsilon \,\mathrm{d}t = 2\varepsilon. \qquad \square$$

O Lema 5.3 nos diz que, ao utilizarmos um conjunto S_ε de ε-representante linear como aproximação de um conjunto S de trajetórias, obtemos um erro aditivo de no máximo 2ε no diâmetro do conjunto. Esse lema, apesar de simples, nos permite provar outros resultados mais interessantes.

Teorema 5.4. Sejam S um conjunto de trajetórias e S_ε um ε-representante linear de S, para algum $\varepsilon > 0$. Para qualquer inteiro positivo k,

$$|\operatorname{sd}^*(S,k) - \operatorname{sd}^*(S_\varepsilon,k)| \leqslant 2k\varepsilon \quad \text{e} \quad |\operatorname{md}^*(S,k) - \operatorname{md}^*(S_\varepsilon,k)| \leqslant 2\varepsilon.$$

Demonstração. Seja $\mathcal{C}^*$ um k-clustering de S com $\operatorname{sd}(\mathcal{C}^*)$ mínimo. Seja $\mathcal{C}_\varepsilon$ o k-clustering de S_ε correspondente a $\mathcal{C}$. Logo,

$$\begin{aligned}
\operatorname{sd}^*(S,k) &= \sum_{C\in\mathcal{C}^*} \operatorname{diam}(C) \\
&\geqslant \sum_{C_\varepsilon\in\mathcal{C}_\varepsilon} (\operatorname{diam}(C_\varepsilon) - 2\varepsilon) && (5.1)\\
&\geqslant \operatorname{sd}^*(S_\varepsilon,k) - 2k\varepsilon. && (5.2)
\end{aligned}$$

A desigualdade (5.1) segue do Lema 5.3.

Agora, considere $\mathcal{C}_\varepsilon^*$ um k-clustering de S_ε com $\operatorname{sd}(\mathcal{C}_\varepsilon^*)$ mínimo e $\mathcal{C}$ o k-clustering de S correspondente. Logo,

$$\begin{aligned}
\operatorname{sd}^*(S,k) &\leqslant \sum_{C\in\mathcal{C}} \operatorname{diam}(C) \\
&\leqslant \sum_{C_\varepsilon\in\mathcal{C}_\varepsilon^*} (\operatorname{diam}(C_\varepsilon) + 2\varepsilon) && (5.3)\\
&= \operatorname{sd}^*(S_\varepsilon,k) + 2k\varepsilon. && (5.4)
\end{aligned}$$

A desigualdade (5.3) segue do Lema 5.3.

Juntando (5.2) e (5.4), temos que $|\operatorname{sd}^*(S,k) - \operatorname{sd}^*(S_\varepsilon,k)| \leqslant 2k\varepsilon$. Analogamente, trocando a soma pelo máximo, temos que $|\operatorname{md}^*(S,k) - \operatorname{md}^*(S_\varepsilon,k)| \leqslant 2\varepsilon$. □

Portanto, mesmo que a velocidade dos pontos não seja constante, podemos aproximar as trajetórias por ε-representantes lineares, que possuem velocidade constante, de modo que o valor de um k-clustering das ε-representantes lineares difere do valor do k-clustering das trajetórias originais correspondente por uma diferença proporcional a ε.

Para utilizar essa abordagem para o kCC1D-S e para o kCC1D-M, de forma que os algoritmos continuem polinomiais em n, o número de trajetórias em S, devemos observar o seguinte. Primeiramente, as funções que definem as trajetórias devem ter uma representação de tamanho polinomial em n, caso contrário, não tem como os algoritmos continuarem polinomiais em n. Em geral, também é preciso que a bijeção r seja dada, pois não conhecemos meios eficientes de encontrar S_ε para o ε ótimo de S com trajetórias arbitrárias. Na Seção 5.1.1, mostraremos como encontrar o ε ótimo para trajetórias lineares por partes.

Problemas semelhantes, onde é dado um conjunto de pontos no plano e deseja-se representar esses pontos por uma ou mais retas, foram estudados por Megiddo e Tamir [MT82, MT83], utilizando definições diferentes para ε-representante linear.

5.1.1 Cálculo de ε ótimo em um caso particular

Seja s uma trajetória que é descrita por uma função x_s linear por partes no intervalo $[0,1]$. Chamamos um ponto do plano (a,t) de **vértice** de s se $x_s(t) = a$ e x_s não é diferenciável em t. Isto é, $t = 0$, ou $t = 1$, ou no instante t ocorre uma mudança no trecho linear de x_s. Vamos assumir que s é dada pela sequência $s^{t_1} = s^0, s^{t_2}, \ldots, s^{t_m} = s^1$, de seus vértices. No exemplo da Figura 5.2(b), a trajetória s tem quatro vértices.

Seja r um ε-representante linear de s, para algum ε. Para cada $1 \leqslant i \leqslant m$, o valor $\varepsilon_i = |x_r(t_i) - s^{t_i}|$ é a **x-distância** do i-ésimo vértice de s a r. Note que $\max\{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m\} = \varepsilon$.

Primeiramente vamos apresentar algumas propriedades de r quando ε é ótimo para s.

Lema 5.5. Sejam s uma trajetória linear por partes e r uma ε-representante linear de s com ε ótimo. Os vértices de s não ficam todos de um mesmo lado de r, isto é, r intersecta s.

Demonstração. Suponha que todos os vértices de s estão de um mesmo lado de r, como mostrado na Figura 5.3(a)). Considere uma trajetória r' paralela a r, mas deslocada de $\delta < \varepsilon$ em direção a s. Cada vértice i de s, para $1 \leqslant i \leqslant m$, tem x-distância $\varepsilon'_i = |\varepsilon_i - \delta|$ de r'. Logo, a trajetória r' é uma ε'-representante de s, para $\varepsilon' = \max\{\varepsilon'_1, \varepsilon'_2, \ldots, \varepsilon'_m\} < \varepsilon$. Mas isso contradiz a otimalidade de ε. Portanto, para ε ótimo, os vértices de s não podem ficar todos de um mesmo lado de r. □

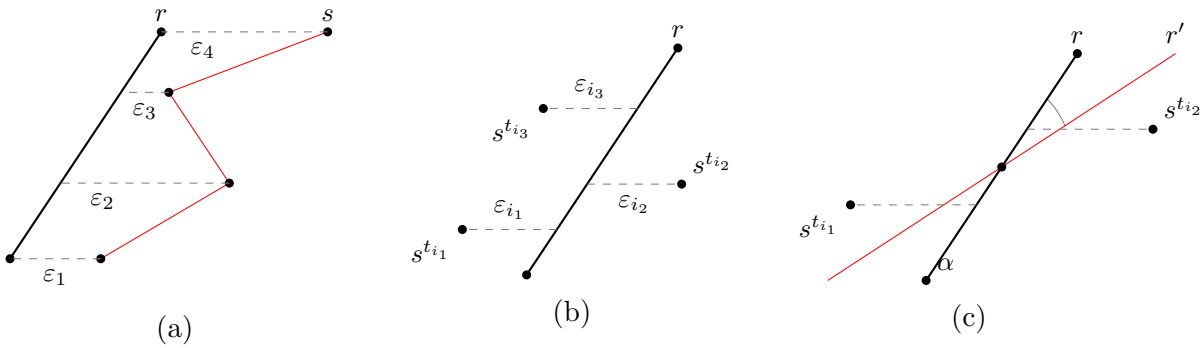


Figura 5.3: Exemplos para os Lemas 5.5 e 5.6.

Lema 5.6. Sejam s uma trajetória linear por partes e r uma ε-representante linear de s com ε ótimo. Se s tem $m \geqslant 3$ vértices, então existem índices $1 \leqslant i_1 < i_2 < i_3 \leqslant m$ tais que os vértices $s^{t_{i_1}}$ e $s^{t_{i_3}}$ ficam de um mesmo lado de r, o vértice $s^{t_{i_2}}$ fica do outro lado de r (Figura 5.3(b)) e $\varepsilon = \varepsilon_{i_1} = \varepsilon_{i_2} = \varepsilon_{i_3}$.

Demonstração. O Lema 5.5 garante a existência de pelo menos um vértice de cada lado de r. Como ε é ótimo, temos que existe um vértice à esquerda de r e um vértice à direita de r tais que ambos têm x-distância ε de r. Caso contrário, deslocar r para o lado com maior x-distância de um vértice de s diminuiria o valor de ε, o que é um absurdo, pois ε é ótimo.

Seja i_2 o menor índice de um vértice à direita de r com $\varepsilon_{i_2} = \varepsilon$. Suponha que existe um vértice à esquerda de r com índice $i < i_2$ e $\varepsilon_i = \varepsilon$. Seja i_1 um tal índice o maior possível. Ou seja, $i_1 < i_2$ e não existe vértice s^{t_i}, $t_{i_1} < t_i < t_{i_2}$, com $\varepsilon_i = \varepsilon$. O caso no qual i_2 está à esquerda de r e i_1 está à direita é análogo.

Seja $t_{\bar{i}} = (t_{i_1} + t_{i_2})/2$. Considere uma trajetória r' obtida da rotação de r, no sentido horário, centralizada no ponto $(x_r(t_{\bar{i}}), t_{\bar{i}})$ e de amplitude $\alpha > 0$ suficientemente pequena para não aumentar o valor de ε. Ver Figura 5.3(c). Note que, para r', a x-distância dos vértices de s à esquerda de r' diminui se possuírem índice menor que $\bar{i}$ e aumentam se seu índice for maior que $\bar{i}$. Para os vértices de s à direita de r', a x-distância aumenta se seu índice for menor que $\bar{i}$ e diminui se for maior.

Agora, suponha que não existe um vértice $s^{t_{i_3}}$ do mesmo lado de r que $s^{t_{i_1}}$, com $i_3 > i_2$ e $\varepsilon_{i_3} = \varepsilon$. Pela escolha de i_1 e i_2, todo vértice s^{t_i} com $\varepsilon_i = \varepsilon$ está à esquerda de r e $i \leqslant i_1$, ou à direita de r e $i \geqslant i_2$. Logo, r' seria uma ε'-representante com $\varepsilon' < \varepsilon$. Uma contradição à otimalidade de ε.

Portanto, existem i_1, i_2 e i_3 como descrito no enunciado. □

Pelo Lema 5.6, uma ε-representante linear r, com ε ótimo, de uma trajetória s é caracterizada por três vértices de s. Suponha que tais vértices são $s^{t_{i_1}}$, $s^{t_{i_2}}$ e $s^{t_{i_3}}$, com $i_1 < i_2 < i_3$, como ilustrado na Figura 5.3(b). Como $\varepsilon_{i_1} = \varepsilon_{i_3}$, então r deve ser paralela

à reta $\tilde{r}$ que passa por $s^{t_{i_1}}$ e $s^{t_{i_3}}$. Com isso, já sabemos o coeficiente angular de r. Seja d a x-distância de $s^{t_{i_2}}$ à reta $\tilde{r}$. Como $\varepsilon_{i_1} = \varepsilon_{i_2}$, temos que $\varepsilon = \varepsilon_{i_1} = \varepsilon_{i_2} = \varepsilon_{i_3} = d/2$. Assim, r é uma trajetória linear paralela a $\tilde{r}$ transladada de $d/2$ em direção a $s^{t_{i_2}}$. Logo, dados os vértices $s^{t_{i_1}}$, $s^{t_{i_2}}$ e $s^{t_{i_3}}$, podemos encontrar r em tempo O(1).

Um algoritmo para encontrar o valor ótimo de ε, para uma dada trajetória, e uma ε-representante linear r é simplesmente executar o procedimento descrito no parágrafo anterior para toda tripla de vértices que não sejam colineares. Note que para triplas de vértices colineares obtemos uma ε-representante linear com $\varepsilon = 0$, implicando que s é linear, mas trajetórias lineares possuem apenas dois vértices. Portanto, triplas de vértices colineares podem ser desconsideradas. Esse algoritmo testaria $\binom{m}{3}$ triplas de vértices de s e, para cada uma, precisaria verificar se a x-distância da trajetória encontrada para cada vértice de s é menor ou igual a $d/2$. Portanto, o consumo de tempo desse algoritmo é $O(m^4)$.

Podemos calcular o valor do ε ótimo de forma mais eficiente. Para isso introduziremos o conceito de casco convexo. O **casco convexo** de um conjunto X de pontos no plano é o *menor* polígono convexo[1] contendo os pontos em X. Mais precisamente, é o polígono resultante da intersecção de todos os polígonos convexos contendo os pontos em X. Não é difícil notar que os pontos inferior esquerdo e superior direito de X sempre estão no casco convexo e que os vértices do casco convexo são pontos de X. Denotamos por CH o casco convexo do conjunto de vértices de s. O casco convexo pode ser separado em dois lados, o direito e o esquerdo. Todo vértice em CH pertence a um dos lados e os conjuntos de vértices de cada lado são disjuntos, a menos dos pontos s^{t_1} e s^{t_m}.

Podemos refinar o Lema 5.6 na forma do seguinte corolário.

Corolário 5.7. Seja s uma trajetória linear por partes no intervalo $[0, 1]$ com $m \geqslant 3$ vértices. Um ε-representante linear r de s com ε ótimo pode ser caracterizado por uma aresta $s^{t_{i_1}} s^{t_{i_3}}$ e um vértice $s^{t_{i_2}}$ em lados opostos de CH com $t_{i_1} < t_{i_2} < t_{i_3}$.

Demonstração. Pelo Lema 5.6, existem vértices $s^{t_{i_1}}$, $s^{t_{i_2}}$ e $s^{t_{i_3}}$, com $t_{i_1} < t_{i_2} < t_{i_3}$, que caracterizam r. Primeiro mostraremos, por contradição, que $s^{t_{i_2}}$ está em CH. De forma análoga, podemos provar que $s^{t_{i_1}}$ e $s^{t_{i_3}}$ também estão em CH.

Suponha que $s^{t_{i_2}}$ não está em CH. Considere que $s^{t_{i_2}}$ está à esquerda de r. O caso em que $s^{t_{i_2}}$ está à direita de r é análogo. Como $s^{t_{i_2}}$ não está em CH, existem vértices s^p e s^q em CH tais que $a = s^p s^q$ é uma aresta de CH, $p < t_{i_2} < q$ e $s^{t_{i_2}}$ está à direita de a. Veja a Figura 5.4. Note que $x_s(t_{i_2}) - x_a(t_{i_2}) = \delta > 0$. Pelo fato de r ser um ε-representante linear de s, sabemos que $x_r(p) - x_s(p) \leqslant \varepsilon$. Como $s^{t_{i_2}}$ está à direita de a, temos que $x_r(t_{i_2}) - x_a(t_{i_2}) > \varepsilon$. Logo, a intersecção de a com r ocorre em $t < t_{i_2}$ e,

[1] Consideramos que $|X| \geqslant 3$ e os pontos não são todos colineares. Caso contrário, o casco convexo seria um segmento de reta ou apenas um ponto, caso $|X| = 1$.

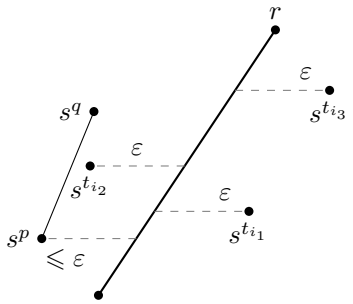


Figura 5.4: Ilustração do caso analisado.

consequentemente, temos que $x_r(q) - x_s(q) > x_r(t_{i_2}) - x_a(t_{i_2}) > \varepsilon$. Mas isso contradiz o fato de r ser ε-representante linear. Portanto, $s^{t_{i_2}}$ é um vértice em CH.

Se $s^{t_{i_1}}$, $s^{t_{i_2}}$ e $s^{t_{i_3}}$ estivessem do mesmo lado de CH, os vértices do outro lado teriam x-distância a r maior que ε, o que seria um absurdo. Logo, $s^{t_{i_1}}$ e $s^{t_{i_3}}$ estão no lado oposto, de CH, ao de $s^{t_{i_2}}$.

Agora, basta verificar que $s^{t_{i_1}}$ e $s^{t_{i_3}}$ correspondem a uma aresta de CH. Note que, se existir um vértice $s^{t'}$ no mesmo lado de CH que $s^{t_{i_1}}$ e com $t_{i_1} < t' < t_{i_3}$, então $s^{t'}$ está na reta que passa por $s^{t_{i_1}}$ e $s^{t_{i_3}}$. Caso contrário, a x-distância de $s^{t'}$ a r seria maior que ε ou o casco não seria convexo. Logo, se $t' < t_{i_2}$, poderíamos trocar $s^{t_{i_1}}$ por $s^{t'}$, caso contrário, trocamos $s^{t_{i_3}}$ por $s^{t'}$. No final, temos que $s^{t_{i_1}}$ e $s^{t_{i_3}}$ correspondem a uma aresta de CH. □

Através do Corolário 5.7, sabemos que não precisamos testar todas as $\binom{m}{3}$ triplas de vértices de s. Basta verificar as que estão no casco convexo de s. Mais ainda, para cada candidato à $s^{t_{i_2}}$ só existe uma aresta do casco convexo no lado oposto ao de $s^{t_{i_2}}$ nas condições do Corolário 5.7. Logo, o número de triplas que precisam ser testadas é linear em m.

Graham [Gra72] apresentou um algoritmo com complexidade de tempo $O(m \log m)$ para encontrar o casco convexo de um conjunto de m pontos no plano. Esse algoritmo é baseado em uma ordenação dos pontos pelo ângulo formado com um eixo horizontal. Andrew [And79] mostra que é possível adaptar a ideia do algoritmo de Graham de forma que basta ordenar os pontos pelas coordenadas t, obtendo o casco convexo em dois passos. Em cada passo, encontra-se um dos lados do casco convexo. Como a sequência dos vértices de s já está ordenada pelo tempo, o algoritmo de Andrew consome tempo $O(m)$ para encontrar CH.

A seguir apresentamos o algoritmo Calcula-ε que recebe uma trajetória s linear por partes no intervalo $[0, 1]$. Usamos como sub-rotina os algoritmos CascoEsquerdo e CascoDireito. O algoritmo CascoEsquerdo recebe s e devolve a sequência E correspondendo à sequência de vértices do lado esquerdo de CH, percorrida de s^{t_1} a s^{t_m} no sentido horário. O algoritmo CascoDireito recebe s e devolve a sequência D corres-

pondendo à sequência de vértices do lado direito de CH, percorrida de s^{t_1} a s^{t_m} no sentido anti-horário. Essas duas sub-rotinas correspondem ao algoritmo de Andrew para encontrar o casco convexo e, neste caso em particular, consomem tempo $O(m)$. O algoritmo CALCULA-ε devolve o valor do ε ótimo para s e três vértices de CH que caracterizam um ε-representante linear para o ε ótimo.

CALCULA-$\varepsilon(s)$
  $s^{e_1}, s^{e_2}, \ldots, s^{e_p} \leftarrow$ CASCOESQUERDO(s)
  $s^{d_1}, s^{d_2}, \ldots, s^{d_q} \leftarrow$ CASCODIREITO(s)
  $\varepsilon^* \leftarrow 0 \qquad i \leftarrow 2 \qquad j \leftarrow 2$
  enquanto $i \leqslant p$ ou $j \leqslant q$ **faça**
    se $e_i < d_j$ **então**
      $h_1 \leftarrow d_{j-1}, \qquad h_2 \leftarrow e_i, \qquad h_3 \leftarrow d_j$
      $i \leftarrow i + 1$
    senão
      $h_1 \leftarrow e_{i-1}, \qquad h_2 \leftarrow d_j, \qquad h_3 \leftarrow e_i$
      $j \leftarrow j + 1$
    $\varepsilon \leftarrow x$-distância entre s^{h_2} e a aresta $s^{h_1}s^{h_3}$
    se $\varepsilon > \varepsilon^*$ **então**
      $\varepsilon^* \leftarrow \varepsilon$
      ${h_1}^* \leftarrow h_1, \qquad {h_2}^* \leftarrow h_2, \qquad {h_3}^* \leftarrow h_3$
  devolva $\frac{\varepsilon^*}{2}$, ${h_1}^*$, ${h_2}^*$, ${h_3}^*$

O algoritmo CALCULA-ε termina em menos de $p+q$ iterações, pois uma vez que $i = p$, o valor de i não aumenta mais e o mesmo acontece com j quando $j = q$. Isso é devido ao fato de que não existem vértices de s com a mesma coordenada t e $e_p = d_q = 1$ é o valor máximo de uma coordenada t de um vértice. Como $p+q = m+2$ e verificar a x-distância de um vértice à uma reta pode ser feito em tempo $O(1)$, o algoritmo CALCULA-ε tem complexidade de tempo $O(m)$.

A correção do algoritmo CALCULA-ε segue dos seguintes invariantes que garantem que os vértices s^{h_1}, s^{h_2} e s^{h_3} satisfazem as hipóteses do Corolário 5.7.

(i) s^{h_2} é um vértice em CH;

(ii) $s^{h_1}s^{h_3}$ é uma aresta do casco convexo de s em lado oposto ao de s^{h_2};

(iii) $h_1 < h_2 < h_3$.

Pelas atribuições feitas nas variáveis h_1, h_2 e h_3, podemos verificar que (i) e (ii) são, de fato, invariantes. Para a primeira iteração, (iii) vale, pois $h_1 = s^0$ independentemente do resultado da comparação $e_i < d_j$. Para as iterações seguintes, note que enquanto o

valor de h_3 não muda, (iii) continua valendo. Suponha, sem perda de generalidade, que no início de uma determinada iteração temos $h_2 = e_{i-1}$ e $e_i > d_j$. Após verificar que $e_i > d_j$, atualizamos os valores das variáveis para $h_1 = e_{i-1}$, $h_2 = d_j$ e $h_3 = e_i$. Da iteração anterior, sabemos que $e_{i-1} < d_j$. Portanto, (iii) é um invariante do algoritmo.

5.2 Movimento aleatório controlado

Agora, consideraremos uma outra variante do modelo unidimensional: o modelo unidimensional aleatório. Nesta variante não sabemos exatamente a velocidade de cada trajetória, mas sabemos o valor esperado de sua velocidade. Mais precisamente, considere que o intervalo de tempo $[0, 1]$ está dividido em T partes iguais, para algum número positivo T. Seja $\mathcal{T} = \{1/T, 2/T, \ldots, 1\}$ o conjunto de instantes que dividem o intervalo de tempo. Para cada trajetória i e cada instante t em $\mathcal{T}$, a velocidade da trajetória i entre os instantes $t - (1/T)$ e t é dada por uma variável aleatória $X_{i,t}$. Note que, entre cada instante consecutivo em $\mathcal{T}$, a velocidade do ponto é constante. Ou seja, as trajetórias neste modelo são lineares por partes, como ilustrado na Figura 5.5.

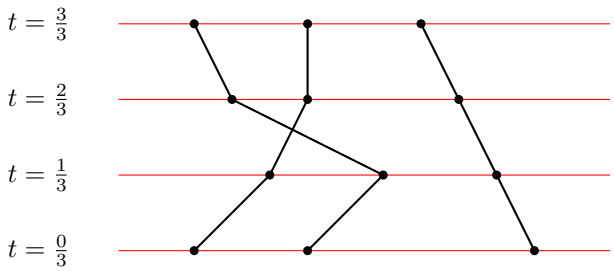


Figura 5.5: Três trajetórias com mudança de velocidade nos instantes 0, 1/3, 2/3 e 1.

Para garantir um certo controle sobre a mudança de velocidade, cada trajetória i tem uma velocidade esperada v_i constante. Para todo t em $\mathcal{T}$, as variáveis aleatórias $X_{i,t}$ podem ter qualquer distribuição de probabilidade desde que sejam independentes, seu valor seja limitado a um intervalo $[a, b]$, com $a < b$, e $\mathbb{E}[X_{i,t}] = v_i$. A sequência de variáveis aleatórias para uma mesma trajetória i é chamada de **processo** da trajetória i e é denotada por $\mathcal{X}_{i,\mathcal{T}} = (X_{i,t})_{t\in\mathcal{T}}$. Apesar de não aparecer na notação, consideramos implicitamente que todas as variáveis aleatórias de um processo tomam valores em um intervalo $[a, b]$.

Para o próximo resultado, utilizaremos uma desigualdade que fornece um limitante superior para a probabilidade da soma de variáveis aleatórias diferir, por um certo valor δ, do seu valor esperado. Sejam $X_1, X_2, \ldots, X_m$ variáveis aleatórias independentes, com X_i assumindo valores no intervalo $[a_i, b_i]$, para $1 \leqslant i \leqslant m$. Seja $SX(m) = X_1 + X_2 + \cdots + X_m$.

Então,

$$\mathbb{P}\left(|SX(m) - \mathbb{E}[SX(m)]| \geqslant \delta\right) \leqslant 2\exp\left(-\frac{2\delta^2}{\sum_{i=1}^{m}(b_i - a_i)^2}\right). \tag{5.5}$$

A desigualdade (5.5) é conhecida como **desigualdade de Hoeffding** [Hoe63].

Queremos afirmar que a probabilidade de uma trajetória desviar muito de sua posição esperada é baixa. Portanto, podemos supor que todas as trajetórias começam na posição zero no instante zero. Para cada $1 \leqslant i \leqslant n$ e t em $\mathcal{T}$, a posição e a posição esperada da trajetória i no instante t são, respectivamente,

$$S_i(t) = \sum_{t' \in \mathcal{T},\, t' \leqslant t} \frac{X_{i,t'}}{T} \qquad \text{e} \qquad \mathbb{E}[S_i(t)] = \sum_{t' \in \mathcal{T},\, t' \leqslant t} \frac{\mathbb{E}[X_{i,t'}]}{T} = tT\frac{v_i}{T} = v_i t.$$

Note que o denominador T, no somatório para $S_i(t)$, aparece pelo fato de que entre dois instantes consecutivos em $\mathcal{T}$, $t_1 < t_2$, temos que a trajetória mantém a velocidade X_{i,t_2} apenas por uma fração $1/T$ de tempo.

Lema 5.8. Sejam n um número inteiro positivo e $\mathcal{T}$ um conjunto com $T = \omega(\ln n)$ instantes. Sejam também a e b números reais com $a < b$. Para cada $1 \leqslant i \leqslant n$, seja $\mathcal{X}_{i,\mathcal{T}}$ um processo que descreve uma trajetória i com $x_i(0) = 0$. Seja S o conjunto dessas n trajetórias. Para qualquer $\varepsilon > 0$,

$$\mathbb{P}\left(\exists i \in S, t \in \mathcal{T} \text{ t.q. } |S_i(t) - v_i t| > \varepsilon\right) = \mathrm{o}(1). \tag{5.6}$$

Demonstração. Sejam i uma trajetória de S e t um instante em $\mathcal{T}$. Tome $d = b - a$. Observe que $S_i(t)$ é a soma de tT variáveis aleatórias $X_{i,t'}/T$ com valores em $[a/T, b/T]$, para t' em $\mathcal{T}$ com $t' \leqslant t$. Logo, aplicando a desigualdade de Hoeffding (5.5) e lembrando que $t \leqslant 1$, temos que

$$\mathbb{P}\left(|S_i(t) - v_i t| > \varepsilon\right) \leqslant 2\exp\left(-\frac{2\varepsilon^2}{tT\frac{d^2}{T^2}}\right) \leqslant 2\exp\left(-\frac{2\varepsilon^2 T}{d^2}\right).$$

Portanto,

$$\begin{aligned}
\mathbb{P}\left(\exists i \in S, t \in \mathcal{T} \text{ t.q. } |S_i(t) - v_i t| > \varepsilon\right) &\leqslant \sum_{i \in S,\, t \in \mathcal{T}} \mathbb{P}\left(|S_i(t) - v_i t| > \varepsilon\right) \\
&\leqslant 2nT\exp\left(-\frac{2\varepsilon^2 T}{d^2}\right) \\
&= 2\exp\left(\ln n + \ln T - \frac{2\varepsilon^2 T}{d^2}\right) \\
&= \mathrm{o}(1).
\end{aligned}$$

A última igualdade segue do fato de que $T = \omega(\ln n)$, implicando que $2\varepsilon^2 T/d^2$ é o termo dominante no exponencial. □

Nesta generalização probabilística do modelo unidimensional, temos que o intervalo de tempo é particionado em T subintervalos uniformes. No início de cada subintervalo, cada trajetória muda de velocidade de acordo com uma variável aleatória e mantém essa velocidade até o próximo subintervalo. A velocidade esperada de cada trajetória é constante para todo o intervalo.

Considerando valores grandes de n, pelo Lema 5.8, temos que, nesse processo de mudança de velocidade, com alta probabilidade, a posição de uma trajetória difere de sua posição esperada de no máximo um valor ε. Assim, com alta probabilidade, para cada trajetória i, a trajetória linear r_i com velocidade constante v_i e começando na mesma posição inicial de i é uma ε-representante linear de i. Pelo Teorema 5.4, podemos utilizar as trajetórias lineares r_i, em vez das trajetórias i, para obter uma solução aproximada com um erro aditivo de no máximo $2k\varepsilon$ para o kCC1D-S, ou com erro aditivo de no máximo 2ε para o kCC1D-M.

Capítulo 6

Clustering com instabilidade

Muitos dos trabalhos sobre clustering dinâmico encontrados na literatura buscam encontrar o melhor clustering para cada instante considerado. Considerando tempo discreto, isso seria equivalente a resolver um problema de clustering para cada instante. No entanto, com a informação sobre a movimentação dos objetos, é possível fazer isso de forma mais eficiente, por exemplo, identificando os possíveis instantes nos quais um clustering anteriormente ótimo deixa de ser ótimo, e atualizando o clustering apenas nesses instantes, como é feito no *KDS* [BGH99].

Nossa abordagem apresentada na Seção 2.1 faz o oposto, no sentido de que procuramos um único clustering que seja bom para o intervalo de tempo inteiro. Ambas as abordagens são válidas, sendo a escolha da mais adequada dependente da aplicação.

Por exemplo, se os pontos representam pessoas utilizando telefones celulares e cada cluster representa um grupo de pessoas sendo servidas por uma mesma antena naquele instante, então a primeira abordagem parece ser melhor, pois a antena que serve um telefone celular pode mudar dependendo da localização do aparelho. Agora, se os pontos representam animais e cada cluster representa um bando de animais, então a segunda abordagem parece melhor, pois, em geral, os animais não mudam de bando.

Essas duas abordagens são extremas no que diz respeito à possibilidade dos clusters mudarem. A primeira abordagem pode ser considerada caótica demais, pois os clusters podem mudar à vontade, de um instante para o outro, de acordo com a função objetivo. A segunda abordagem pode ser considerada rígida demais, pois os clusters não mudam do início ao fim.

Neste capítulo apresentaremos um modelo que tenta equilibrar esses dois extremos, permitindo algo intermediário. Apresentaremos também alguns resultados de como o modelo se comporta nesses dois extremos.

6.1 Clustering dinâmico e instabilidade

Para a movimentação, neste capítulo consideramos o modelo unidimensional discreto. Neste modelo consideramos n pontos em $\mathbb{R}$ que se movem arbitrariamente. Diferentemente do modelo unidimensional, onde tínhamos informação completa da movimentação dos pontos durante um intervalo de tempo, neste modelo são conhecidas apenas as posições dos pontos em determinados instantes de tempo. Ou seja, o tempo deixa de ser um intervalo contínuo e passa a ser discreto.

Consideramos T instantes de tempo e assumimos que o conjunto desses instantes é $[T]$. Para instantes t e t' em $[T]$, se $t < t'$, então o instante t ocorre antes do instante t'.

A **trajetória** de um ponto i é uma sequência $(x_i^t)_{t=1}^T$, onde x_i^t é a posição do ponto i no instante t, para t em $[T]$. Dizemos que a trajetória $(x_i^t)_{t=1}^T$ tem **duração** T. Denotamos por S o conjunto das n trajetórias de interesse. Assumiremos que não existem duas trajetórias idênticas em S, isto é, duas trajetórias distintas devem ter posições diferentes em pelo menos um instante.

Um **cluster dinâmico** de S com **duração** T é uma sequência $C = (C^t)_{t=1}^T$ na qual C^t é um subconjunto de S, para t em $[T]$. Cada C^t é chamado de **cluster** C no instante t. Um **k-clustering dinâmico** de S com duração T é uma sequência $\mathscr{C} = (C_j)_{j=1}^k$ de k clusters dinâmicos com duração T na qual $\mathcal{C}^t = \{C_1^t, C_2^t, \ldots, C_k^t\}$ é uma partição de S para cada t em $[T]$. Cada uma dessas partições $\mathcal{C}^t$ é um **k-clustering** de S no instante t.

Para simplificar a leitura, a menos que dito o contrário, sempre consideramos que as trajetórias, clusters dinâmicos e k-clusterings dinâmicos têm duração T e omitiremos tal informação sempre que possível, daqui em diante.

O **diâmetro** de um cluster dinâmico C é a soma dos diâmetros de seus clusters C^t para cada t em $[T]$:

$$\operatorname{diam}(C) = \sum_{t=1}^{T} \operatorname{diam}(C^t) = \sum_{t=1}^{T} \left(\max_{i \in C^t} x_i^t - \min_{i \in C^t} x_i^t \right).$$

Note que o diâmetro de um cluster em determinado instante continua sendo a maior distância entre um par de trajetórias do cluster. A soma do diâmetro para todos os instantes é uma extensão natural da integral na Equação 2.2 para tempo discreto. Apesar de utilizarmos a mesma notação para diâmetro de um cluster dinâmico e diâmetro de um cluster em determinado instante, ficará claro de qual dos dois se trata pela ausência ou presença, respectivamente, do superíndice t indicando o instante.

Assim como para o clustering cinético, para o clustering dinâmico consideramos duas medidas de qualidade: a soma dos diâmetros dos clusters dinâmicos e o diâmetro máximo de um cluster dinâmico. A soma dos diâmetros dos clusters de um cluster dinâmico C é denotada por $\operatorname{sd}(C)$. A soma dos diâmetros dos clusters dinâmicos de um k-clustering

dinâmico $\mathscr{C}$ é denotada por $\mathrm{sd}(\mathscr{C})$.

$$\mathrm{sd}(\mathscr{C}) = \sum_{j=1}^{k} \mathrm{sd}(C_j) = \sum_{j=1}^{k} \sum_{t=1}^{T} \mathrm{diam}(C_j^t).$$

O diâmetro máximo de um cluster dinâmico em um k-clustering dinâmico $\mathscr{C}$ é denotado por $\mathrm{md}(\mathscr{C})$:

$$\mathrm{md}(\mathscr{C}) = \max_{j \in [k]} \mathrm{diam}(C_j) = \max_{j \in [k]} \sum_{t=1}^{T} \mathrm{diam}(C_j^t).$$

Pela definição de cluster dinâmico, não é necessário existir alguma relação entre os clusters C^t e C^{t+1} para um instante $1 \leqslant t < T$. Se uma trajetória s está em C^t, mas não está em C^{t+1}, dizemos que s **muda** de cluster dinâmico no instante $t+1$. A **instabilidade** é uma medida de quanto os clusters dinâmicos de um k-clustering dinâmico $\mathscr{C}$ mudam ao longo do tempo. Denotamos por $\mathrm{instab}(\mathcal{C}^t, \mathcal{C}^{t'})$ a instabilidade entre os k-clusterings $\mathcal{C}^t$ e $\mathcal{C}^{t'}$ de $\mathscr{C}$, por $\mathrm{instab}(\mathscr{C}, t)$ a instabilidade de $\mathscr{C}$ no instante t para $t > 1$, e por $\mathrm{instab}(\mathscr{C})$ a instabilidade de $\mathscr{C}$ para todo $t > 1$:

$$\mathrm{instab}(\mathcal{C}^t, \mathcal{C}^{t'}) = \sum_{j=1}^{k} \left| C_j^{t'} \setminus C_j^t \right|,$$

$$\mathrm{instab}(\mathscr{C}, t) = \mathrm{instab}(\mathcal{C}^{t-1}, \mathcal{C}^t),$$

$$\mathrm{instab}(\mathscr{C}) = \sum_{t=2}^{T} \mathrm{instab}(\mathscr{C}, t).$$

Logo, a instabilidade de $\mathscr{C}$ é a quantidade de trajetórias que mudam de cluster dinâmico de um instante para o seguinte.

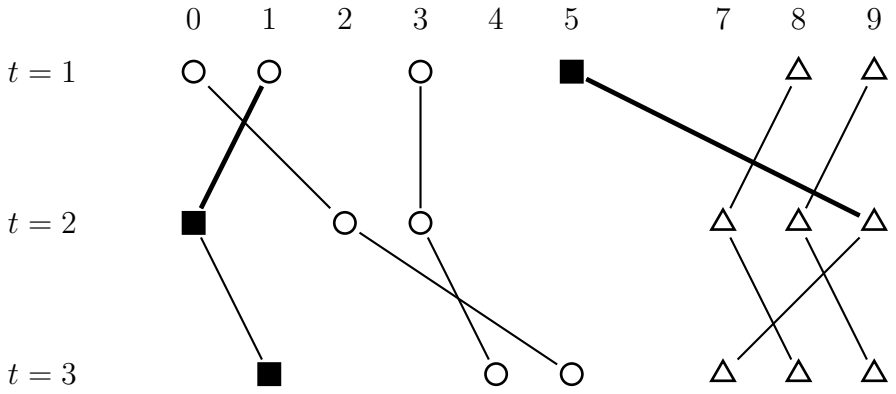


Figura 6.1: Exemplo de k-clustering dinâmico para $n = 6$, $T = 3$ e $k = 3$. Os segmentos em negrito indicam mudança de cluster.

No k-clustering dinâmico $\mathscr{C}$ do exemplo da Figura 6.1, cada tipo de figura geométrica representa um cluster dinâmico diferente, $C_\circ, C_\triangle, C_\blacksquare$, e os segmentos indicam as trajetórias. De $t = 1$ a $t = 2$, a única trajetória que estava em $C_\blacksquare$ muda para $C_\triangle$ e uma

trajetória que estava em $C_\circ$ muda para $C_\blacksquare$. De $t = 2$ a $t = 3$ não ocorrem mudanças. Os diâmetros dos clusters dinâmicos são $\mathrm{diam}(C_\circ) = 5$, $\mathrm{diam}(C_\blacksquare) = 0$ e $\mathrm{diam}(C_\triangle) = 5$. Logo, temos que $\mathrm{sd}(\mathscr{C}) = 10$ e $\mathrm{md}(\mathscr{C}) = 5$. A instabilidade de $\mathscr{C}$ é $\mathrm{instab}(\mathscr{C}) = 2$ devido às duas mudanças que ocorrem em $t = 2$.

6.2 Instabilidade mínima para diâmetros fixos

Seja S um conjunto de n trajetórias. Considere uma bijeção $\mathrm{ind}\colon S \to [n]$ arbitrária, que será utilizada apenas como critério de desempate para definir uma ordem total das trajetórias. Para cada instante t em $[T]$ e para trajetórias s e s' em S, escrevemos $s \prec^t s'$, se $x_s^t < x_{s'}^t$, ou $x_s^t = x_{s'}^t$ e $\mathrm{ind}(s) < \mathrm{ind}(s')$.

Para cada instante t, considere a função bijetora $\pi^t\colon [n] \to [S]$ que corresponde à permutação das trajetórias em S de acordo com a ordem $\prec^t$. Ou seja, $\pi^t(i) = s$ significa que a trajetória s é a i-ésima menor trajetória de S com relação a $\prec^t$.

Seja C_j^t um cluster no instante t de um k-clustering dinâmico. Denotamos por e_j^t e por d_j^t as trajetórias mínima e máxima em C_j^t, respectivamente, com relação a $\prec^t$. Chamamos e_j^t de **extremo esquerdo** de C_j^t e d_j^t de **extremo direito** de C_j^t.

Considere uma sequência de k pares ordenados de números inteiros em $[n]$, denotada por $((p_j, q_j))_{j=1}^k$, tal que $p_j \leqslant q_j$, $\bigcup_{j=1}^k [p_j, q_j] \supseteq [n]$ e $p_j \neq p_{j'}$, $q_j \neq q_{j'}$ e $p_j \neq q_{j'}$ para todo $j' \neq j$. Chamamos tal sequência de k**-cobertura por intervalos** de $[n]$. Também utilizaremos a nomenclatura k**-cobertura por intervalos** para uma sequência de T k-coberturas por intervalos $(((p_j^t, q_j^t))_{j=1}^k)_{t=1}^T$.

Um k-clustering $\mathcal{C}^t = \{C_1^t, C_2^t, \ldots, C_k^t\}$ com clusters não vazios define uma única k-cobertura por intervalos da seguinte forma. Sejam e_j^t e d_j^t os extremos esquerdo e direito, respectivamente, do cluster C_j^t, para j em $[k]$. Tome p_j^t e q_j^t em $[n]$, tais que $\pi^t(p_j^t) = e_j^t$ e $\pi^t(q_j^t) = d_j^t$. Não é difícil ver que $((p_j^t, q_j^t))_{j=1}^k$ é uma k-cobertura por intervalos. De fato, para todo j em $[k]$, temos $p_j^t \leqslant q_j^t$ já que um extremo esquerdo nunca é maior que um extremo direito na ordem $\prec^t$. Como $\mathcal{C}^t$ é uma partição de um conjunto com n trajetórias, então os extremos são todos dois-a-dois disjuntos, com exceção do caso no qual um cluster possui uma única trajetória. Ademais, como toda trajetória s pertence a algum cluster, ou s é um dos extremos, ou $e_j^t \prec^t s \prec^t d_j^t$ para algum j. Logo, $\bigcup_{j=1}^k [p_j, q_j] \supseteq [n]$. De forma análoga, um k-clustering dinâmico define uma k-cobertura por intervalos $(((p_j^t, q_j^t))_{j=1}^k)_{t=1}^T$.

Seja $X = (((p_j^t, q_j^t))_{j=1}^k)_{t=1}^T$ uma k-cobertura por intervalos de um k-clustering dinâmico $\mathscr{C}'$. É fácil calcular o diâmetro de um cluster C_j^t em $\mathscr{C}'$ a partir de X, pois $\mathrm{diam}(C_j^t) = x_d^t - x_e^t$, onde $e = \pi^t(p_j^t)$ e $d = \pi^t(q_j^t)$. Logo, X contém a informação dos diâmetros de todos os clusters e de todos os clusters dinâmicos de $\mathscr{C}'$. Porém, a partir de X não é possível reconstruir $\mathscr{C}'$. Uma k-cobertura por intervalos não possui informações sobre trajetórias que não são extremos. Entretanto, através de X, podemos derivar um k-

clustering dinâmico $\mathscr{C}$ tal que instab($\mathscr{C}$) é mínima entre os k-clusterings dinâmicos que têm X como k-cobertura por intervalos.

Para isso, a primeira coisa a se notar é que, pelo fato dos extremos dos clusters estarem fixos, os diâmetros dos clusters são sempre os mesmos, independentemente das possíveis alterações dos clusters dinâmicos. Logo, basta encontrar o menor número de mudanças de clusters dinâmicos que seja compatível com os extremos definidos. Outra observação importante é que o número de vezes que uma determinada trajetória muda de cluster dinâmico não depende dos clusters em que estão as demais trajetórias, pois os extremos dos clusters já estão definidos. Portanto, mostraremos como calcular a quantidade de mudanças de cluster para uma única trajetória e depois basta repetir o mesmo procedimento para as demais.

Uma trajetória s no cluster C_j^t, ou é um extremo do cluster, ou é tal que $e_j^t \prec^t s \prec^t d_j^t$. Dizemos que uma trajetória s é (j,t)**-compatível** se, no instante t, a trajetória s é um extremo de C_j^t, ou $e_j^t \prec^t s \prec^t d_j^t$ e s não é extremo de nenhum outro cluster no instante t. Dada uma k-cobertura por intervalos $(((p_j^t, q_j^t))_{j=1}^k)_{t=1}^T$, lembrando que $\pi^t(p_j^t) = e_j^t$ e $\pi^t(q_j^t) = d_j^t$, podemos verificar se uma trajetória s é (j,t)-compatível em tempo O(k), para todo j em $[k]$ e todo t em $[T]$.

Denotamos por $\gamma^*(s,j,t)$ o número mínimo de mudanças de cluster feitas pela trajetória s, do instante 1 ao t, quando s pertence ao cluster C_j^t. Caso a trajetória s não possa pertencer ao cluster C_j^t, por não ser (j,t)-compatível, estabelecemos que $\gamma^*(s,j,t) = \infty$. Basicamente $\gamma^*(s,j,t)$ é a contribuição mínima da trajetória s para instab($\mathscr{C}$), se s pertence ao cluster C_j^t.

Para $t = 1$, se uma trajetória s é $(j,1)$-compatível, então $\gamma^*(s,j,1) = 0$. Caso contrário, $\gamma^*(s,j,1) = \infty$. Para $t \geqslant 2$, se s é (j,t)-compatível, temos que $\gamma^*(s,j,t) = \min_{j' \in [k]}\{\gamma^*(s,j',t-1) + \bar{\mathbb{1}}(j,j')\}$, onde a função $\bar{\mathbb{1}}(j,j')$ tem valor 1 se $j \neq j'$, ou valor 0 se $j = j'$. Caso contrário, temos que $\gamma^*(s,j,t) = \infty$. Resumindo,

$$\gamma^*(s,j,t) = \begin{cases} 0, & \text{se } t = 1 \text{ e } s \text{ é } (j,1)\text{-compatível} \\ \min\limits_{j' \in [k]}\{\gamma^*(s,j',t-1) + \bar{\mathbb{1}}(j,j')\}, & \text{se } t \geqslant 2 \text{ e } s \text{ é } (j,t)\text{-compatível} \\ \infty, & \text{caso contrário.} \end{cases} \tag{6.1}$$

Logo,

$$\text{instab}(\mathscr{C}) = \sum_{s \in S} \min_{j \in [k]} \gamma^*(s,j,T). \tag{6.2}$$

O algoritmo MinInstabilidade recebe um conjunto de trajetórias S e uma sequência $(((e_j^t, d_j^t))_{j=1}^k)_{t=1}^T$ de extremos de clusters de um k-clustering dinâmico. Tal sequência nada mais é que a aplicação de π^t a cada p_j^t e q_j^t para todo t em $[T]$ e j em $[k]$. O valor devolvido pelo algoritmo é a instabilidade mínima de um k-clustering dinâmico cujos extremos são iguais aos da sequência dada. Para obter um k-clustering dinâmico correspondente com instabilidade mínima, basta alterar o algoritmo para armazenar as escolhas feitas.

```
MinInstabilidade(S, (((e_j^t, d_j^t))_{j=1}^k)_{t=1}^T)
    para cada s ∈ S faça
        para j ← 1 até k faça
            se s é (j, 1)-compatível então
                g_{s,j,1} ← 0
            senão
                g_{s,j,1} ← ∞
    sum ← 0
    para cada s ∈ S faça
        h_s ← ∞
        para j ← 1 até k faça
            para t ← 2 até T faça
                g_{s,j,t} ← ∞
                se s é (j, t)-compatível então
                    para j' ← 1 até k faça
                        aux ← g_{s,j',t-1}
                        se j' ≠ j então aux ← aux + 1
                        se aux < g_{s,j,t} então g_{s,j,t} ← aux
            se g_{s,j,T} < h_s então h_s ← g_{s,j,T}
        sum ← sum + h_s
    devolva sum
```

O algoritmo MinInstabilidade utiliza a técnica de programação dinâmica para calcular o valor da equação (6.2). Inicialmente são calculados os casos base da recorrência (6.1), isto é, o caso $t = 1$. São nk casos base e para cada um é preciso verificar se uma trajetória i é $(j, 1)$-compatível, o que pode ser feito em tempo $O(k)$. Logo, os casos base são calculados em tempo $O(nk^2)$. Para cada um dos demais $nk(T - 1)$ casos restantes, precisamos novamente verificar se uma trajetória s é (j, t)-compatível e, se for, procuramos o mínimo entre k casos já calculados. Logo, no total, esses casos são calculados em tempo $O(nk^3T)$. O valor da variável *sum* é obtido com n somas. Portanto, o algoritmo MinInstabilidade calcula o valor da equação (6.2) em tempo $O(nk^3T)$.

Podemos usar o algoritmo MinInstabilidade para resolver problemas de k-clustering dinâmico que envolvem minimização de uma combinação linear entre diâmetros de clusters dinâmicos e instabilidade. Isso pode ser feito enumerando todas as possíveis k-coberturas por intervalos e construindo um k-clustering dinâmico com instabilidade mínima a partir de cada uma delas. No final, selecionamos um k-clustering de custo mínimo. Para determinar a complexidade de tempo deste método, precisamos contar o número de possíveis k-coberturas por intervalos, ou equivalentemente, o número de possíveis sequências de extremos. Para isso, primeiro considere que todo cluster tem pelo menos

duas trajetórias. No primeiro instante, $t = 1$, não precisamos nos preocupar com a ordem dos clusters, logo temos $\binom{n}{2k}$ escolhas de extremos de clusters. Escolhida as $2k$ trajetórias que serão extremos, a menor das trajetórias, com relação a $\prec^1$, será o extremo esquerdo do primeiro cluster, sobrando $2k-1$ opções para o extremo direito. Definidos os extremos do primeiro cluster, a menor das trajetórias restantes, com relação a $\prec^1$, será o extremo esquerdo do segundo cluster, sobrando $2k-3$ opções de escolha para o extremo direito e assim sucessivamente. Logo, o número de possibilidades de escolha de extremos é[1]

$$\binom{n}{2k}(2k-1)!! = \binom{n}{2k}\frac{(2k)!}{(2k)(2k-2)\cdots 2} = \binom{n}{2k}\frac{(2k)!}{2^k\,k!} < \frac{n^{2k}}{2^k\,k!}.$$

Para os demais instantes $t \geqslant 2$, a contagem é similar. A diferença está no fato de que agora a ordem dos clusters faz diferença, pois indica as mudanças de cluster. Logo, o número de possíveis extremos para cada um desses instantes é menor que $n^{2k}/2^k$. No total, temos menos de $(1/k!)(n^{2k}/2^k)^T$ possibilidades de extremos, se cada cluster contém pelo menos duas trajetórias.

Para considerar os casos nos quais existem clusters contendo uma única trajetória, podemos usar o mesmo raciocínio anterior, mas permitindo que uma mesma trajetória possa ser escolhida duas vezes. Então, basicamente, trocamos n por $2n$ e concluímos que existem menos do que $(1/k!)(2^k\,n^{2k})^T$ possibilidades. Note que essa é uma estimativa grosseira e o número verdadeiro é bem menor.

Assim, testar todas as possibilidades de extremos e aplicar o MinInstabilidade para cada uma resulta em um algoritmo que consome tempo $\mathrm{O}(2^{kT}n^{2kT+1}k^3T/k!)$. Um algoritmo de força-bruta ingênuo teria que testar todos os $k^{nT}/k!$ possíveis k-clusterings dinâmicos, gastando tempo $\Omega(k^{nT}(n+k)T/k!)$, pois o cálculo do custo de um k-clustering dinâmico gasta tempo $\mathrm{O}((n+k)T)$. A complexidade de tempo da força-bruta ingênua é considerada pior por ser exponencial na quantidade de trajetórias. Isso porque, em geral, o número de trajetórias será muito maior do que o número de clusters dinâmicos.

6.3 Minimização da soma dos diâmetros

O primeiro problema de clustering dinâmico que apresentamos é o de minimizar a soma dos diâmetros dos clusters dinâmicos. Como também queremos, de alguma maneira, controlar a quantidade de mudanças de clusters dinâmicos, definimos o custo de um k-clustering dinâmico $\mathscr{C}$ por

$$\text{custo-s}_\gamma(\mathscr{C}) = \mathrm{sd}(\mathscr{C}) + \gamma\,\mathrm{instab}(\mathscr{C}),$$

onde γ é um número real não negativo.

[1]Seja n um número inteiro positivo ímpar. Denotamos $n!! = n(n-2)(n-4)\cdots 1$.

Problema 6.1 (kCD-S)**.** Dados um número inteiro positivo k, um número real não negativo γ e um conjunto S de n trajetórias de duração T, encontrar um k-clustering dinâmico $\mathscr{C}$ de S com custo-s$_\gamma(\mathscr{C})$ mínimo.

O parâmetro γ do kCD-S tem a função de controlar a quantidade de mudanças nos clusters dinâmicos ao longo do tempo. Se $\gamma = \infty$, isto é, se γ for grande o suficiente, então em um k-clustering dinâmico ótimo todas as trajetórias sempre se mantêm nos mesmos clusters dinâmicos. Ou seja, os clusters dinâmicos são na verdade estáticos neste caso, e o problema torna-se o kCC1D-S. Qualquer valor de γ maior ou igual a $\sum_{t=1}^{T}(\max_{i\in S} x_i^t - \min_{i\in S} x_i^t)$ é suficientemente grande, pois equivale ao valor de um k-clustering dinâmico que mantém todas as trajetórias em um único cluster o tempo todo. Se $\gamma = 0$, então as trajetórias podem mudar de cluster à vontade. Este caso equivale a encontrar k-clusterings ótimos em cada instante independentemente dos demais instantes. No entanto, note que essa equivalência só é verdade porque na função sd($\mathscr{C}$) não importa a qual cluster dinâmico um cluster pertence, importa apenas o valor de seu diâmetro.

Proposição 6.2. Seja $\mathscr{C}^*$ um k-clustering dinâmico ótimo para uma instância do kCD-S com $\gamma > 0$ e seja $\mathscr{C}_0$ um k-clustering dinâmico, para a mesma instância, mas que minimiza custo-s$_0(\mathscr{C}_0)$. Existem instâncias do kCD-S tais que custo-s$_\gamma(\mathscr{C}_0)$ pode ser $\Omega(n)$ vezes maior que custo-s$_\gamma(\mathscr{C}^*)$.

Demonstração. Seja $\varepsilon < 1/12$ e considere um conjunto de $n = 2m$ trajetórias, para algum $m \geqslant 2$. Tome $T = 3$. No instante $t = 1$, a trajetória i está na posição $x_i^1 = (-1)^i\varepsilon\gamma$, para $i \leqslant m$, e $x_i^1 = (2 + (-1)^i\varepsilon)\gamma$, para $i > m$. No instante $t = 2$, a trajetória i está na posição $x_i^2 = (1 + (-1)^i\varepsilon)\gamma$, para todo i em $[n]$. No instante $t = 3$, a trajetória i está na posição $x_i^3 = (2 + (-1)^i\varepsilon)\gamma$, para $i \leqslant m$, e $x_i^3 = (-1)^i\varepsilon\gamma$, para $i > m$. Basicamente, as trajetórias $i \leqslant m$ estão se movimentando para a direita e as trajetórias $i > m$ estão se movimentando para a esquerda, todas se encontrando no instante $t = 2$.

O 2-clustering dinâmico ótimo $\mathscr{C}^*$ para este conjunto de trajetórias consiste dos clusters dinâmicos $\{1, 2, \ldots, m\}$ e $\{m + 1, m + 2, \ldots, 2m\}$ nos três instantes, com um custo-s$_\gamma(\mathscr{C}^*) = 3 \times 2 \times 2\varepsilon\gamma = \Theta(\gamma)$. Note que qualquer outra solução contabilizaria um diâmetro de $(2-2\varepsilon)\gamma$ no instante $t = 1$ ou $t = 3$, ou pelo menos γ por mudança de cluster dinâmico.

Um 2-clustering dinâmico $\mathscr{C}_0$ para estas trajetórias que minimiza custo-s$_0(\mathscr{C}_0)$ consiste de $C_1^1 = C_1^3 = \{1, 2, \ldots, m\}$, $C_2^1 = C_2^3 = \{m+1, m+2, \ldots, 2m\}$, $C_1^2 = \{1, 3, \ldots, 2m-1\}$ e $C_2^2 = \{2, 4, \ldots, 2m\}$. A soma dos diâmetros dos clusters dinâmicos dessa sequência é $2 \times 2 \times 2\varepsilon\gamma$ e ocorrem $2m$ mudanças de clusters dinâmicos. Portanto, custo-s$_\gamma(\mathscr{C}_0) = 8\varepsilon\gamma + n\gamma = \Omega(n\gamma)$. □

Como consideramos que as posições estão na reta real, uma instância com $\gamma = 0$ pode ser resolvida pelo mesmo método guloso apresentado na Seção 3, aplicado independente-

mente a cada instante de tempo. No entanto, pela Proposição 6.2, a menos que $\gamma = 0$, encontrar k-clusterings ótimos para cada instante, em geral, não dá uma boa aproximação para o kCD-S.

O lema seguinte mostra que quase toda k-cobertura por intervalos pode corresponder a um $(k+1)$-clustering dinâmico ótimo de n trajetórias de duração $T+1$ para o $(k+1)$CD-S. Tal resultado sugere que não existe alguma estrutura comum às k-coberturas por intervalos de k-clusterings ótimos que possa ser explorada para desenvolver algoritmos polinomiais para o kCD-S.

Lema 6.3. Sejam k, n e T números inteiros positivos com $n \geqslant 2k$. Considere uma k-cobertura por intervalos de $[n]$ arbitrária, $X = (((p_j^t, q_j^t))_{j=1}^k)_{t=1}^T$. Existe um conjunto de n trajetórias de duração $T+1$ tal que o único $(k+1)$-clustering dinâmico ótimo para o $(k+1)$CD-S com $\gamma > 0$ é obtido da k-cobertura por intervalos X.

Demonstração. Seja $\varepsilon < 1/T(k+1)n$. Considere o seguinte conjunto de n trajetórias. Para $1 \leqslant i \leqslant k$ e $1 \leqslant t \leqslant T$, a trajetória $2i-1$ é tal que $x_{2i-1}^t = p_i^t \varepsilon\gamma$ e $x_{2i-1}^{T+1} = 2i\gamma - \varepsilon\gamma/2$. A trajetória $2i$ é tal que $x_{2i}^t = q_i^t \varepsilon\gamma$ e $x_{2i}^{T+1} = 2i\gamma + \varepsilon\gamma/2$. Para $i > 2k$, a trajetória i é tal que $x_i^t = \phi^t(i)\varepsilon\gamma$, para $1 \leqslant t \leqslant T$, onde ϕ^t é uma função injetora qualquer de $\{2k+1, 2k+2, \ldots, n\}$ para $[n] \setminus \{p_1^t, q_1^t, p_2^t, q_2^t, \ldots, p_k^t, q_k^t\}$, e $x_i^{T+1} = 2(k+1)\gamma$. A Figura 6.2 mostra um exemplo dessa construção.

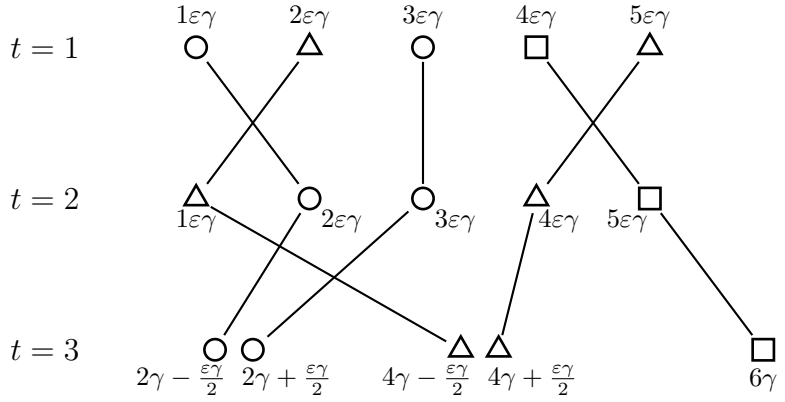


Figura 6.2: Exemplo para $n = 5$, $k = 2$ e $T = 2$ e $X = \{\{(1,3),(2,5)\},\{(2,3),(1,4)\}\}$.

Considere o $(k+1)$-clustering dinâmico $\mathscr{C}$ onde, para $1 \leqslant j \leqslant k$ e $1 \leqslant t \leqslant T+1$, temos que $C_j^t = \{2j-1, 2j\}$ e $C_{k+1}^t = \{2k+1, 2k+2, \ldots, n\}$. Note que, para $1 \leqslant j \leqslant k$ e $1 \leqslant t \leqslant T$, $\operatorname{diam}(C_j^t) \leqslant (n-1)\varepsilon\gamma$, $\operatorname{diam}(C_{k+1}^t) \leqslant (n-1)\varepsilon\gamma$, $\operatorname{diam}(C_j^{T+1}) = \varepsilon\gamma$ e $\operatorname{diam}(C_{k+1}^{T+1}) = 0$. Como não ocorrem mudanças de cluster em $\mathscr{C}$, a instabilidade de $\mathscr{C}$ é zero. Logo, custo-s$_\gamma(\mathscr{C}) \leqslant T(k+1)(n-1)\varepsilon\gamma + k\varepsilon\gamma \leqslant T(k+1)n\varepsilon\gamma < \gamma$.

Qualquer $(k+1)$-clustering dinâmico diferente de $\mathscr{C}$ teria ou um cluster com diâmetro pelo menos $(2-\varepsilon)\gamma > \gamma$ no instante $T+1$, ou pelo menos uma trajetória mudando de cluster. Em ambos os casos, o custo de tal $(k+1)$-clustering dinâmico seria maior que γ.

Portanto, para a instância construída, $\mathscr{C}$ é o único $(k+1)$-clustering dinâmico ótimo. Além disso, note que os k primeiros elementos da $(k+1)$-cobertura definida por $\mathscr{C}$ são exatamente os elementos de X. □

6.3.1 Trajetórias sem cruzamentos

Aqui, consideramos um caso particular do kCD-S no qual as trajetórias não se cruzam. Um conjunto $S = \{s_1, s_2, \ldots, s_n\}$ de trajetórias de duração T **não tem cruzamentos** se $s_1 \prec^t s_2 \prec^t \cdots \prec^t s_n$ para todo t em $[T]$. Isto é, as trajetórias mantém suas posições relativas em todos os instantes. Note que neste caso particular, $\pi^1(i) = \pi^2(i) = \cdots = \pi^T(i)$ para todo i em $[n]$. Portanto, dada uma k-cobertura por intervalos $(((p_j^t, q_j^t))_{j=1}^k)_{t=1}^T$, vamos abusar da notação e considerar que p_j^t e q_j^t são as trajetórias $e_j^t = \pi^t(p_j^t)$ e $d_j^t = \pi^t(q_j^t)$, respectivamente.

Esta restrição sobre as trajetórias poderia simplificar bastante o problema, tornando-o fácil. Infelizmente, não é este o caso. Mesmo que a posição relativa das trajetórias seja sempre a mesma, surpreendentemente a posição relativa dos clusters dinâmicos pode mudar.

Fato 6.4. Existe um conjunto de cinco trajetórias sem cruzamentos para o qual a posição dos clusters no único 2-clustering dinâmico ótimo, para o kCD-S, muda. Esse exemplo pode ser generalizado para $n \geqslant 5$ trajetórias, $k \geqslant 2$ clusters e $T \geqslant 2$.

Demonstração. Tome $n = 5$, $k = 2$, $T = 2$ e $\gamma > 0$. Considere as seguintes cinco trajetórias: $(x_1^t)_{t=1}^2 = \{0, 3\gamma\}$, $(x_i^t)_{t=1}^2 = \{3\gamma, 3\gamma\}$, para $2 \leqslant i \leqslant 4$, e $(x_5^t)_{t=1}^2 = \{3\gamma, 6\gamma\}$. Sejam $C_1^1 = \{1\}$, $C_1^2 = \{5\}$, $C_2^1 = \{2, 3, 4, 5\}$ e $C_2^2 = \{1, 2, 3, 4\}$. O 2-clustering dinâmico $\mathscr{C} = \{\{C_1^1, C_1^2\}, \{C_2^1, C_2^2\}\}$ tem custo 2γ, pois ocorrem duas mudanças de cluster dinâmico e todos os clusters têm diâmetro 0. Qualquer outro 2-clustering dinâmico ou tem algum cluster com diâmetro pelo menos 3γ, ou envolverá mais do que duas mudanças de cluster dinâmico. Logo, $\mathscr{C}$ é o único 2-clustering dinâmico ótimo para a instância construída. Note que, em $t = 2$, a posição relativa dos clusters dinâmicos muda pois C_1, que estava à esquerda de C_2, passa a ficar à direita de C_2. □

Observe que esse fenômeno da troca de posição relativa de clusters dinâmicos observado neste caso particular do problema onde as trajetórias não se cruzam corresponde a dois clusters dinâmicos que se fundem, deixando espaço para um novo cluster dinâmico surgir em um outro lugar arbitrário, em uma outra posição relativa arbitrária.

Essa mudança na posição relativa dos clusters dinâmicos dificulta o desenvolvimento de algoritmos polinomiais para o kCD-S. Para $T = 2$ ou $k = 2$, existem k-clusterings dinâmicos ótimos de trajetórias sem cruzamentos que são bem estruturados. No entanto, isso já não vale para $k \geqslant 4$ e $T \geqslant 3$. Antes de formalizar essa afirmação, apresentaremos mais algumas definições e resultados auxiliares.

Dizemos que uma k-cobertura por intervalos $((p_j, q_j))_{j=1}^k$ **não tem sobreposições** se os intervalos $[p_j, q_j]$ são dois-a-dois disjuntos. Também dizemos que dois clusters se **sobrepõem** no instante t se os intervalos correspondentes aos respectivos clusters se intersectam. Isto é, um dos extremos de um dos clusters está entre os extremos do outro na ordem $\prec^t$. No exemplo da Figura 6.2, em $t = 1$, o cluster das trajetórias circulares e o cluster das trajetórias triangulares se sobrepõem.

Lembre que, para um k-clustering dinâmico $(C_j)_{j=1}^k$, denotamos o seu k-clustering no instante t por $\mathcal{C}^t = (C_j^t)_{j=1}^k$.

Lema 6.5. Considere um k-clustering dinâmico de trajetórias sem cruzamentos. As seguintes afirmações são verdadeiras.

(i) Se dois clusters A^t e B^t se sobrepõem, então $\text{diam}(A^t \cup B^t) \leqslant \text{diam}(A^t) + \text{diam}(B^t)$;

(ii) Se dois clusters A^t e B^t se sobrepõem, e $\{\tilde{A}^t, \tilde{B}^t\}$ é uma partição de $A^t \cup B^t$ sem sobreposição, então $\text{diam}(\tilde{A}^t) + \text{diam}(\tilde{B}^t) \leqslant \text{diam}(A^t) + \text{diam}(B^t)$;

(iii) Se $\mathcal{C}^{t_1}$, $\mathcal{C}^{t_2}$ e $\mathcal{C}^{t_3}$ são k-clusterings, $\text{instab}(\mathcal{C}^{t_1}, \mathcal{C}^{t_3}) \leqslant \text{instab}(\mathcal{C}^{t_1}, \mathcal{C}^{t_2}) + \text{instab}(\mathcal{C}^{t_2}, \mathcal{C}^{t_3})$.

Demonstração. Se os clusters A^t e B^t se sobrepõem, seus respectivos intervalos $[p_a, q_a]$ e $[p_b, q_b]$ intersectam. Considere as trajetórias $p = \min\{p_a, p_b\}$ e $q = \max\{q_a, q_b\}$. Observe que $x_q^t - x_p^t \leqslant (x_{q_a}^t - x_{p_a}^t) + (x_{q_b}^t - x_{p_b}^t)$, logo $\text{diam}(A^t \cup B^t) \leqslant \text{diam}(A^t) + \text{diam}(B^t)$, provando (i).

Seja $\{\tilde{A}^t, \tilde{B}^t\}$ uma partição sem sobreposição de $A^t \cup B^t$. Suponha que $\tilde{A}^t$ e $\tilde{B}^t$ não são vazios, caso contrário (ii) equivale a (i). Sejam $[\tilde{p}_a, \tilde{q}_a]$ e $[\tilde{p}_b, \tilde{q}_b]$ seus respectivos intervalos. Sem perda de generalidade, suponha que $\tilde{p}_a \prec^t \tilde{p}_b$ e, consequentemente, $x_{\tilde{q}_a}^t - x_{\tilde{p}_b}^t \leqslant 0$. Logo, $\text{diam}(\tilde{A}^t) + \text{diam}(\tilde{B}^t) = (x_{\tilde{q}_a}^t - x_{\tilde{p}_a}^t) + (x_{\tilde{q}_b}^t - x_{\tilde{p}_b}^t) \leqslant x_{\tilde{q}_b}^t - x_{\tilde{p}_a}^t = \text{diam}(A^t \cup B^t)$ e, aplicando (i), obtemos (ii).

Agora, para mostrar (iii), note que toda trajetória que contribui para $\text{instab}(\mathcal{C}^{t_1}, \mathcal{C}^{t_3})$ mudou de cluster. Logo, essas trajetórias também devem mudar de cluster entre $\mathcal{C}^{t_1}$ e $\mathcal{C}^{t_2}$, ou entre $\mathcal{C}^{t_2}$ e $\mathcal{C}^{t_3}$, contribuindo para $\text{instab}(\mathcal{C}^{t_1}, \mathcal{C}^{t_2})$ ou para $\text{instab}(\mathcal{C}^{t_2}, \mathcal{C}^{t_3})$. □

Agora mostraremos que se $T = 2$ ou $k = 2$, sempre existe um k-clustering dinâmico ótimo para o kCD-S restrito a trajetórias sem cruzamentos, definido por uma k-cobertura por intervalos sem sobreposições. Para $T = 1$ e $k = 1$, é trivial.

Proposição 6.6. Considere um conjunto S de n trajetórias de duração $T = 2$ sem cruzamentos. Existe um k-clustering dinâmico ótimo para o kCD-S definido por uma k-cobertura por intervalos sem sobreposições.

Demonstração. Seja $\mathscr{C}^*$ um k-clustering dinâmico de S que é ótimo para o kCD-S. Seja $X = (((p_j, q_j))_{j=1}^k)_{t=1}^2$ a k-cobertura por intervalos correspondente a $\mathscr{C}^*$. Denote por w o

número de pares de intervalos em X, para um mesmo t, que se sobrepõem. Provaremos por indução em w que sempre existe um k-clustering dinâmico ótimo cuja k-cobertura por intervalos não tem sobreposições.

Se $w = 0$, não há nada a ser provado. Considere que $w \geqslant 1$. Suponha, sem perda de generalidade, que em $t = 2$ existem clusters C_1^2 e C_2^2, cujos intervalos são respectivamente $[p_1^2, q_1^2]$ e $[p_2^2, q_2^2]$, que se sobrepõem. Se os intervalos $[p_1^1, q_1^1]$ e $[p_2^1, q_2^1]$, correspondentes respectivamente aos clusters C_1^1 e C_2^1, também se sobrepõem, podemos unir C_1^2 com C_2^2 e C_1^1 com C_2^1, deixando um dos clusters vazio. Isso claramente diminui o número de sobreposições. Pelo Lema 6.5 (i), o diâmetro do k-clustering dinâmico resultante não aumenta com essa operação, que também não gera nenhuma mudança de cluster a mais. Logo, por hipótese de indução, existe uma k-cobertura por intervalos como desejada.

Se C_1^1 e C_2^1 não se sobrepõem, podemos supor, sem perda de generalidade, que $p_1^1 \prec^1 p_2^1$. Considere o cluster $\widetilde{C}_1^2$ contendo as trajetórias s em $C_1^2 \cup C_2^2$ tais que $s \preceq^2 q_1^1$ e o cluster $\widetilde{C}_2^2$ contendo as demais trajetórias de $C_1^2 \cup C_2^2$. Note que $\widetilde{C}_1^2$ e $\widetilde{C}_2^2$ não se sobrepõem. Pelo Lema 6.5 (ii), trocar C_1^2 e C_2^2 por $\widetilde{C}_1^2$ e $\widetilde{C}_2^2$ não aumenta o diâmetro. Essa troca também não aumenta a quantidade de sobreposições com outros clusters, pois os intervalos de $\widetilde{C}_1^2$ e de $\widetilde{C}_2^2$ estão contidos nos intervalos de C_1^2 e de C_2^2, respectivamente. Logo, qualquer outro cluster que se sobrepõem com $\widetilde{C}_j^2$ também se sobrepõe com C_j^2, para $1 \leqslant j \leqslant 2$. Ademais, não ocorre nenhuma nova mudança de cluster, já que as trajetórias não se cruzam. Logo, o k-clustering dinâmico resultante também é ótimo e com menos sobreposições. Por hipótese de indução, existe uma k-cobertura por intervalos como desejada.

Portanto, para $T = 2$, sempre existe um k-clustering dinâmico ótimo cuja k-cobertura por intervalos não tem sobreposições. □

Proposição 6.7. Considere um conjunto S de n trajetórias de duração T sem cruzamentos. Existe um 2-clustering dinâmico ótimo para o kCD-S, definido por uma 2-cobertura por intervalos sem sobreposições.

Demonstração. Seja $\mathscr{C}^*$ um 2-clustering dinâmico de S que é ótimo para o kCD-S. Denote por w o número de pares de intervalos na 2-cobertura por intervalos de $\mathscr{C}^*$, para um mesmo t, que se sobrepõem. Provaremos por indução em w que sempre existe um 2-clustering dinâmico ótimo cuja 2-cobertura por intervalos não tem sobreposições.

Se $w = 0$, não há nada a ser provado. Considere que $w \geqslant 1$. Note que existe um instante onde os dois clusters dinâmicos de $\mathscr{C}^*$ não se sobrepõem. Caso contrário, unindo os dois clusters dinâmicos obtemos um 1-clustering dinâmico de custo menor, contradizendo a otimalidade de $\mathscr{C}^*$.

Suponha, sem perda de generalidade, que existe um instante $t < T$ onde os dois clusters de $\mathcal{C}^{t+1}$ se sobrepõem, mas os dois clusters de $\mathcal{C}^t$ não. Trocando $\mathcal{C}^{t+1}$ por uma cópia de $\mathcal{C}^t$, pelo Lema 6.5 (ii), não aumentamos a soma dos diâmetros dos clusters no

k-clustering dinâmico resultante. A nova instabilidade será

$$I = \text{instab}(\mathscr{C}^*) - \text{instab}(\mathcal{C}^t, \mathcal{C}^{t+1}) - \text{instab}(\mathcal{C}^{t+1}, \mathcal{C}^{t+2}) + \text{instab}(\mathcal{C}^t, \mathcal{C}^{t+2}),$$

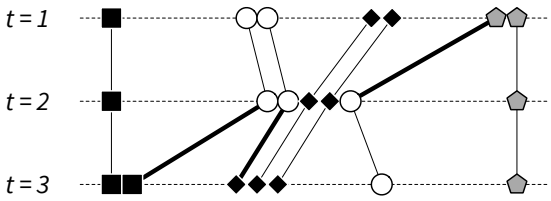
considerando $\text{instab}(\cdot, \mathcal{C}^{t+2}) = 0$ se $t + 2 > T$. Pelo Lema 6.5 (iii), $\text{instab}(\mathcal{C}^t, \mathcal{C}^{t+2}) \leqslant \text{instab}(\mathcal{C}^t, \mathcal{C}^{t+1}) + \text{instab}(\mathcal{C}^{t+1}, \mathcal{C}^{t+2})$. Logo, $I \leqslant \text{instab}(\mathscr{C}^*)$. Portanto, o k-clustering dinâmico resultante também é ótimo e tem menos sobreposições. Por hipótese de indução, existe um k-clustering dinâmico ótimo cuja 2-cobertura por intervalos não tem sobreposições. □

Fato 6.8. Existe um conjunto de $n = 7$ trajetórias de duração $T = 3$ sem cruzamentos para o qual todo 4-clustering dinâmico ótimo para o kCD-S tem dois clusters que se sobrepõem em algum instante.

Demonstração. Considere as trajetórias dadas pela Figura 6.3(a), para $\gamma > 0$ e $\Delta > 3\gamma$.

	x_1^t	x_2^t	x_3^t	x_4^t	x_5^t	x_6^t	x_7^t
$t = 1$	0	Δ	Δ	2Δ	2Δ	3Δ	3Δ
$t = 2$	0	Δ	Δ	Δ	Δ	Δ	2Δ
$t = 3$	0	0	Δ	Δ	Δ	2Δ	3Δ

(a)



(b)

Figura 6.3: Um exemplo onde os dois únicos 4-clusterings dinâmicos contêm clusters que se sobrepõem. Em (b), as linhas destacadas indicam mudança de cluster.

Existem exatamente dois 4-clusterings dinâmicos ótimos para estas trajetórias com custo 3γ. Um consiste nos 4-clusterings

$$\mathcal{C}^1 = \{\{1\}, \{2, 3\}, \{4, 5\}, \{6, 7\}\},$$
$$\mathcal{C}^2 = \{\{1\}, \{\mathbf{2}, \mathbf{3}, \mathbf{6}\}, \{\mathbf{4}, \mathbf{5}\}, \{7\}\} \text{ e}$$
$$\mathcal{C}^3 = \{\{1, 2\}, \{3, 4, 5\}, \{6\}, \{7\}\}.$$

O outro difere apenas em $t = 2$ com o 4-clustering $\mathcal{C}^2 = \{\{1\}, \{\mathbf{2}, \mathbf{6}\}, \{\mathbf{3}, \mathbf{4}, \mathbf{5}\}, \{7\}\}$. Em ambos, os clusters 2 e 3 (em negrito) se sobrepõem em $t = 2$. □

Assim, sabemos que, para trajetórias sem cruzamentos, sempre existe um 2-clustering dinâmico ótimo para o kCD-S sem sobreposições. Isso já não é verdade para 4-clusterings dinâmicos. Não sabemos o que acontece para 3-clusterings dinâmicos. Saber que sempre existe um k-clustering ótimo sem sobreposições é útil para o desenvolvimento de algoritmos mais eficientes, pois elimina a necessidade de considerar todas as k-coberturas por intervalos possíveis.

Como só temos a garantia de existência de k-clusterings dinâmicos ótimos sem sobreposição para poucos casos, vamos definir uma estrutura mais geral. Chamamos um k-clustering (dinâmico) de **laminar** se, para cada instante, a k-cobertura por intervalos correspondente é uma família laminar[2] e se toda trajetória pertence ao cluster de menor diâmetro cujo intervalo a cobre. Uma k-cobertura por intervalos é laminar se seu k-clustering (dinâmico) correspondente é laminar. Assim, uma k-cobertura por intervalos laminar define um único k-clustering (dinâmico). Note que k-coberturas por intervalos sem sobreposição são laminares. Note também que o k-clustering dinâmico do exemplo da Figura 6.3 é laminar.

Dada uma k-cobertura por intervalos de um k-clustering $\mathcal{C}$, considere a sequência ordenada de seus extremos, $s_1 \prec^1 s_2 \prec^1 \cdots \prec^1 s_{2k}$. Agora, considere uma cadeia de $2k$ caracteres, onde o i-ésimo caractere é '(' se s_i é extremo esquerdo, ou ')' caso contrário. Não é difícil observar que tal cadeia de parênteses é bem formada se e somente se $\mathcal{C}$ é laminar. A quantidade de cadeias bem formadas de k pares de parênteses é dada pelo número de Catalan $c_k = 1/(k+1)\binom{2k}{k}$ [Cat38]. Portanto, a quantidade de k-clusterings laminares é no máximo

$$L = k!\binom{n}{2k-2}c_k \leqslant \frac{n^{2k-2}}{(k+1)!}.$$

Considere uma enumeração, de 1 a L, dos k-clusterings laminares de n trajetórias. Denotaremos por custo-s$_\gamma(\ell, t)$ o custo de um k-clustering dinâmico ótimo restrito aos t primeiros instantes e tal que ℓ é o k-clustering laminar do instante t. Logo, para ℓ em $[L]$ e t em $[T]$,

$$\text{custo-s}_\gamma(\ell, t) = \begin{cases} \operatorname{sd}(\ell), & \text{se } t = 1 \\ \min\limits_{\ell' \in [L]} (\text{custo-s}_\gamma(\ell', t-1) + \operatorname{instab}(\ell', \ell) + \operatorname{sd}(\ell)), & \text{se } t \geqslant 2. \end{cases} \tag{6.3}$$

O custo mínimo de um k-clustering dinâmico laminar é $\min_{\ell \in [L]} \text{custo-s}_\gamma(\ell, T)$. Logo, é possível implementar um algoritmo de programação dinâmica para encontrar um k-clustering dinâmico laminar ótimo que consome tempo $\mathrm{O}(L^2 T) = \mathrm{O}(n^{4k-4}T/((k+1)!)^2)$.

Infelizmente, assim como ocorre com as k-coberturas por intervalos sem sobreposição, não é sempre que existe um k-clustering dinâmico ótimo que é laminar. A Figura 6.4 mostra um exemplo de tal caso, para $\gamma > 0$ e Δ suficientemente grande. Foi determinado computacionalmente que para esse exemplo os únicos 6-clusterings dinâmicos ótimos são os dois ilustrados na figura.

Como mencionado anteriormente, sabemos resolver o kCD-S para $\gamma = 0$, pois esse caso equivale a resolver T instâncias de um único instante independentemente. Outro caso que sabemos resolver em tempo polinomial é o caso de trajetórias sem cruzamentos e $\gamma = \infty$. Para resolver este caso, basta notar que, pelo fato das trajetórias não se

[2]Uma família $\mathcal{F}$ é laminar se, para todo A e B em $\mathcal{F}$, ou $A \subseteq B$, ou $B \subseteq A$, ou $A \cap B = \emptyset$.

	$x_1^t..x_3^t$	$x_4^t..x_6^t$	$x_7^t..x_9^t$	x_{10}^t	x_{11}^t	x_{12}^t	x_{13}^t	x_{14}^t
$t=1$	0	0	Δ	2Δ	3Δ	3Δ	4Δ	5Δ
$t=2$	0	Δ	Δ	Δ	Δ	2Δ	3Δ	4Δ
$t=3$	0	Δ	2Δ	2Δ	2Δ	3Δ	3Δ	4Δ
$t=4$	0	Δ	2Δ	3Δ	4Δ	5Δ	5Δ	5Δ

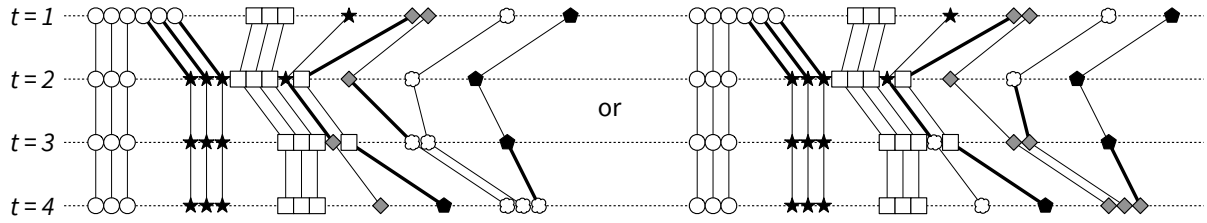


Figura 6.4: Exemplo com 14 trajetórias de duração 4 que não se cruzam, para o qual todos os 6-clusterings dinâmicos ótimos não são laminares (no instante $t = 2$). As arestas destacadas indicam mudanças de cluster.

cruzarem e não ocorrer mudanças de cluster em um k-clustering dinâmico ótimo, podemos reduzir o problema para uma instância do caso estático (Capítulo 3), ou seja, com $T = 1$. Nessa instância reduzida, a distância entre cada trajetória é a soma das distâncias entre elas para todo instante, como ilustrado na Figura 6.5. Observe que, como a posição relativa das trajetórias não muda, os pontos nessa instância reduzida estão, de fato, em $\mathbb{R}$.

Para qualquer valor de γ intermediário, não sabemos se é possível resolver o kCD-S com trajetórias sem cruzamentos em tempo polinomial. Suspeitamos que não é possível, pois apesar de parecer um caso bem simples, o fato dos clusters não necessariamente manterem a posição relativa (Fato 6.4) dificulta a aplicação de técnicas como programação dinâmica.

6.4 Minimização do diâmetro máximo dos clusters

Sejam S um conjunto de n trajetórias e $\mathscr{C}$ um k-clustering dinâmico de S. Em vez de considerar a soma dos diâmetros de clusters dinâmicos no custo de $\mathscr{C}$, podemos também considerar o diâmetro máximo de um cluster dinâmico em $\mathscr{C}$. Para isso, definimos o custo como

$$\text{custo-m}_\gamma(\mathscr{C}) = \max_{j\in[k]} \operatorname{diam}(C_j) + \gamma \operatorname{instab}(\mathscr{C}).$$

Problema 6.9 (kCD-M)**.** Dados um número inteiro positivo k, um número real não negativo γ e um conjunto S de n trajetórias de duração T, encontrar um k-clustering dinâmico $\mathscr{C}$ de S com custo-m$_\gamma(\mathscr{C})$ mínimo.

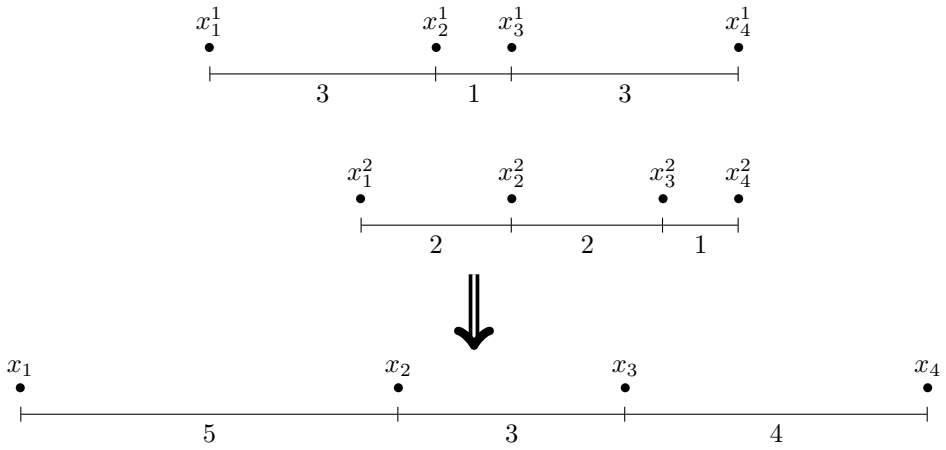


Figura 6.5: Redução de uma instância com 4 trajetórias de duração 2 para uma instância com trajetórias de duração 1.

O parâmetro γ cumpre o mesmo papel que cumpre no kCD-S. Para $\gamma = \infty$ e trajetórias sem cruzamentos, podemos fazer a mesma redução apresentada no final da Seção 6.3.1. Com isso obtemos uma instância equivalente com $T = 1$, que sabemos resolver em tempo polinomial (Capítulo 4). Já para $\gamma = 0$, diferentemente do kCD-S, não podemos simplesmente resolver independentemente cada instante, pois para o kCD-M a ordem dos clusters na sequência de um k-clustering dinâmico é relevante. Caso contrário, não temos como identificar o cluster dinâmico com maior diâmetro. Na verdade, conseguimos provar que o kCD-M é NP-difícil tanto para $\gamma = 0$ quanto para $\gamma = \infty$. Para mostrar isso, usaremos o problema Partição (Problema 5.1), que é NP-completo [Kar72].

Teorema 6.10. O kCD-M é NP-difícil para $\gamma = 0$, mesmo restrito a trajetórias sem cruzamentos.

Demonstração. Considere uma instância do Partição que consiste no multiconjunto de números inteiros positivos $A = \{a_1, a_2, \ldots, a_m\}$, para $m \geqslant 2$. Mostraremos como construir uma instância do kCD-M com $\gamma = 0$ a partir dela. Tome $n = 3$, $k = 2$ e $T = m$. Para cada t em $[T]$, as posições das três trajetórias são $x_1^t = -a_t$, $x_2^t = 0$ e $x_3^t = a_t$. Essa construção claramente é polinomial. A Figura 6.6 mostra um exemplo da construção.

Seja $\mathscr{C}^*$ um 2-clustering dinâmico ótimo para o kCD-M dessas três trajetórias. Como $\gamma = 0$, o diâmetro total máximo de um cluster de $\mathscr{C}^*$ é o menor possível. Dessa forma, não é difícil concluir que em nenhum instante vale a pena um cluster conter as três trajetórias. Para cada instante, um dos clusters conterá duas trajetórias e o outro, apenas uma. Ademais, qualquer mudança de cluster que ocorrer será sempre com a trajetória 2. Logo, podemos chamar de C_1 o cluster que sempre contém a trajetória 1 e de C_2 o cluster que sempre contém a trajetória 3.

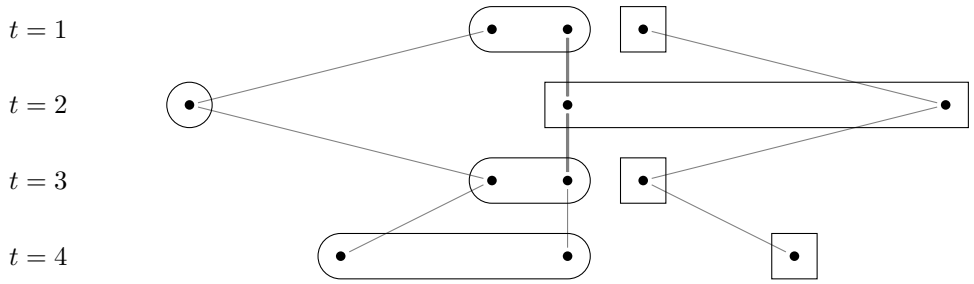


Figura 6.6: Trajetórias construídas a partir de $A = \{1, 5, 1, 3\}$. O 2-clustering dinâmico composto pelos clusters arredondado e retangular é ótimo e cada cluster tem diâmetro total 5.

A partir de $\mathscr{C}^*$, particione A em A_1 e A_2 da seguinte forma. Para cada t em $[T]$, insira a_t em A_1 se $\text{diam}(C_1^t) > 0$, isto é, se C_1^t contém a trajetória 2. Caso contrário, insira a_t em A_2. Note que $\text{diam}(C_1) = \sum_{a \in A_1} a$ e $\text{diam}(C_2) = \sum_{a \in A_2} a$. Se $\text{diam}(C_1) = \text{diam}(C_2)$, a resposta do Partição é positiva para A, pois $\sum_{a \in A_1} a = \sum_{a \in A_2} a$.

Agora, suponha que existe uma partição $\{B_1, B_2\}$ de A tal que $\sum_{b \in B_1} b = \sum_{b \in B_2} b$, ou seja, a resposta do Partição para A é positiva. Por um processo inverso ao utilizado para construir A_1 e A_2, podemos obter um 2-clustering dinâmico $\mathscr{C}'$ com clusters D_1 e D_2. Note que $\text{diam}(D_1) = \text{diam}(D_2)$, implicando que $\mathscr{C}'$ é ótimo para o kCD-M.

Logo, o Partição tem resposta positiva para A se e só se custo-m$_\gamma(\mathscr{C}^*) = \sum_{a \in A} a/2$. Portanto, kCD-M com $\gamma = 0$ é tão difícil quanto o Partição. □

Teorema 6.11. O kCD-M é NP-difícil para $\gamma = \infty$.

Demonstração. Considere uma instância do Partição que consiste no multiconjunto de números inteiros positivos $A = \{a_1, a_2, \ldots, a_m\}$, com $m \geqslant 3$. Mostraremos como construir uma instância do kCD-M com $\gamma = \infty$ a partir dela. Tome $n = m$, $k = 2$ e $T = m$. Para cada $1 \leqslant i \leqslant n$ e $1 \leqslant t \leqslant T$, a posição da trajetória i no instante t é $x_i^t = a_t$ se $t = i$, e $x_i^t = 0$ caso contrário. Essa construção claramente é polinomial. A Figura 6.7 mostra um exemplo da construção.

Seja $\mathscr{C}^* = \{C_1, C_2\}$ um 2-clustering dinâmico ótimo para o kCD-M dessas n trajetórias. Como $\gamma = \infty$, uma trajetória nunca muda de cluster em $\mathscr{C}^*$. A partir de $\mathscr{C}^*$, particione A em $A_1 = \{a_i \in A \mid i \in C_1\}$ e $A_2 = \{a_i \in A \mid i \in C_2\}$. Note que $\text{diam}(C_1) = \sum_{a \in A_1} a$ se $|C_1| \geqslant 2$, caso contrário $\text{diam}(C_1) = 0$. O análogo vale para C_2 e A_2.

Seja $M = \sum_{i=1}^n a_i$. Suponha, sem perda de generalidade, que $\text{diam}(C_1) \geqslant \text{diam}(C_2)$. Como $n \geqslant 3$, temos que $|C_1| \geqslant 2$. Logo, $\text{diam}(C_1) \geqslant M/2$. Se $\text{diam}(C_1) = M/2$, não é difícil ver que a resposta do Partição é positiva para A.

Suponha que existe uma partição $\{B_1, B_2\}$ de A tal que $\sum_{b \in B_1} b = \sum_{b \in B_2} b = M/2$. Por um processo inverso ao utilizado para construir A_1 e A_2, podemos obter um 2-

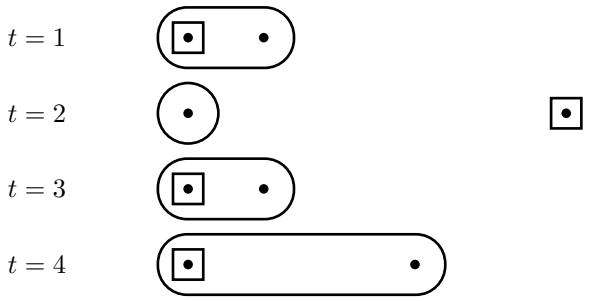


Figura 6.7: Trajetórias construídas a partir de $A = \{1, 5, 1, 3\}$. O 2-clustering dinâmico composto pelos clusters arredondado e retangular é ótimo e cada cluster dinâmico tem diâmetro 5 e 0, respectivamente. Nos quatro instantes, as três trajetórias que coincidem na posição zero estão representadas por um único ponto.

clustering dinâmico $\mathscr{C}' = \{D_1, D_2\}$ tal que $\max(\text{diam}(D_1), \text{diam}(D_2)) = M/2$, implicando que $\mathscr{C}'$ é ótimo para o kCD-M.

Portanto, o Partição tem resposta positiva para A se e só se custo-m$_\gamma(\mathscr{C}^*) = M/2$. Logo, kCD-M com $\gamma = \infty$ é tão difícil quanto o Partição. □

6.4.1 Uma 2-aproximação para $\gamma = 0$

Como mostrado pelo Teorema 6.10, para o caso $\gamma = 0$, diferentemente do kCD-S, não existe algoritmo polinomial para o kCD-M, supondo que P≠NP. Assim, o melhor que conseguimos fazer é desenvolver algoritmos de aproximação. Apresentaremos uma 2-aproximação para o kCD-M para $\gamma = 0$.

Para apresentar essa 2-aproximação, vamos supor que conhecemos o valor D do maior diâmetro de um cluster, em algum instante, de um k-clustering dinâmico ótimo. Logo, para todo instante, todo cluster tem diâmetro no máximo D. O valor de D não é trivial de se calcular, mas podemos testar todos os possíveis valores. Para cada instante, existem $\binom{n}{2}$ possibilidades de extremos de um cluster. Logo, temos no máximo $T\binom{n}{2} = Tn(n-1)/2$ candidatos a valor de D.

Nosso algoritmo de aproximação pode ser dividido em duas partes:

(I) escolha de um k-clustering das trajetórias para cada instante;

(II) ordenação dos clusters para definir o k-clustering dinâmico final.

Na parte (I), para cada instante t em $[T]$, encontramos um k-clustering $\mathcal{C}^t$ das trajetórias que minimiza $\text{sd}(\mathcal{C}^t)$ sujeito a $\text{diam}(C_j^t) \leqslant D$ para todo cluster C_j^t em $\mathcal{C}^t$. Seja $\mathscr{C} = \{\mathcal{C}^1, \mathcal{C}^2, \ldots, \mathcal{C}^T\}$ o k-clustering dinâmico obtido de uma combinação dos k-clusterings encontrados. Temos que

$$\sum_{t=1}^{T} \text{sd}(\mathcal{C}^t) = \sum_{t=1}^{T} \sum_{j=1}^{k} \text{diam}(C_j^t) = \sum_{j=1}^{k} \sum_{t=1}^{T} \text{diam}(C_j^t) = \sum_{j=1}^{k} \text{diam}(C_j).$$

Seja $\mathscr{C}^* = \{\mathcal{C}^{*1}, \mathcal{C}^{*2}, \ldots, \mathcal{C}^{*T}\}$ um k-clustering dinâmico ótimo para o kCD-M para $\gamma = 0$. Como em $\mathscr{C}^*$ todo cluster tem diâmetro no máximo D em qualquer instante, temos que

$$\sum_{t=1}^{T} \mathrm{sd}(\mathcal{C}^t) \leqslant \sum_{t=1}^{T} \mathrm{sd}(\mathcal{C}^{*t}) \quad \Longrightarrow \quad \sum_{j=1}^{k} \mathrm{diam}(C_j) \leqslant \sum_{j=1}^{k} \mathrm{diam}(C_j^*). \tag{6.4}$$

Um k-clustering $\mathcal{C}^t$ que apenas minimiza $\mathrm{sd}(\mathcal{C}^t)$ pode ser encontrado por um algoritmo guloso, como mencionado no Capítulo 3. Não podemos utilizar o mesmo algoritmo para encontrar os k-clusterings de $\mathscr{C}$, pois um dos clusters pode acabar com diâmetro maior que D. No entanto, não é difícil notar que, em um k-clustering da forma desejada, ainda vale a propriedade de que um extremo de um cluster nunca está posicionado entre os extremos de outro cluster. Ou seja, não existe sobreposição de clusters.

Uma maneira de encontrar os k-clusterings de $\mathscr{C}$ é por programação dinâmica. Fixe um instante t em $[T]$. Considere que as n trajetórias estão ordenadas de forma que $x_1^t \leqslant x_2^t \leqslant \cdots \leqslant x_n^t$. Denote por $\mathrm{sd}_{D,t}(n,k)$ o valor mínimo da soma dos diâmetros dos clusters de um k-clustering das n primeiras trajetórias, considerando duração t, em que nenhum dos clusters tenha diâmetro maior que D. Logo,

$$\mathrm{sd}_{D,t}(n,k) = \begin{cases} \infty, & \text{se } k = 1 \text{ e } x_n^t - x_1^t > D \\ x_n^t - x_1^t, & \text{se } k = 1 \text{ e } x_n^t - x_1^t \leqslant D \\ 0, & \text{se } k \geqslant n \text{ ou } n \leqslant 0 \\ \min\limits_{i \in [n]} \{\, \mathrm{sd}_{D,t}(i-1, k-1) + (x_n^t - x_i^t) \,\}, & \text{caso contrário.} \end{cases} \tag{6.5}$$

Um algoritmo de programação dinâmica que calcula $\mathrm{sd}_{D,t}(n,k)$ pode ser implementado de maneira a consumir tempo $\mathrm{O}(n^2 k)$. Se $\mathrm{sd}_{D,t}(n,k) = \infty$, então não existe k-clustering, no instante t, em que todo cluster tenha diâmetro no máximo D. Ou seja, o valor de D é menor do que deveria ser e precisa ser aumentado.

Na parte (II), a ordenação dos clusters encontrados na parte (I) pode ser feita pelo seguinte procedimento guloso. Escolha uma ordem arbitrária para os clusters de $\mathcal{C}^1$. Para cada $t \geqslant 2$ em ordem, suponha, sem perda de generalidade, que após estabelecidas as ordens nos k-clusterings anteriores, temos que

$$\sum_{t'=1}^{t-1} \mathrm{diam}(C_1^{t'}) \leqslant \sum_{t'=1}^{t-1} \mathrm{diam}(C_2^{t'}) \leqslant \cdots \leqslant \sum_{t'=1}^{t-1} \mathrm{diam}(C_k^{t'}).$$

Dessa forma, ordenamos os clusters de $\mathcal{C}^t$ em ordem crescente de diâmetro. Para ficar mais claro, descrevemos o algoritmo Ordenação, que recebe $\mathscr{C} = \{\mathcal{C}^1, \mathcal{C}^2, \ldots, \mathcal{C}^T\}$ e devolve uma função $r\colon [T] \times [k] \to [k]$ indicando a nova ordem dos clusters.

ORDENAÇÃO($\mathscr{C}$)
  para $j \leftarrow 1$ **até** k **faça**
    $r(1, j) \leftarrow j$
    $d(j) \leftarrow \text{diam}(C_j^1)$
  para $t \leftarrow 2$ **até** T **faça**
    rearranje os clusters de $\mathcal{C}^t$ tal que $\text{diam}(C_1^t) \geqslant \cdots \geqslant \text{diam}(C_k^t)$
    seja π uma permutação de $[k]$ tal que $d(\pi(1)) \leqslant \cdots \leqslant d(\pi(k))$
    para $p \leftarrow 1$ **até** k **faça**
      $r(t, p) \leftarrow \pi(p)$
      $d(\pi(p)) \leftarrow d(\pi(p)) + \text{diam}(C_p^t)$
  devolva r

O algoritmo ORDENAÇÃO consome tempo $O(Tk \log k)$. O fator $k \log k$ é devido ao cálculo da permutação π. Após a ordenação dos clusters, $\mathscr{C}$ é o k-clustering dinâmico que nossa 2-aproximação devolve. Os lemas a seguir serão úteis para deduzirmos delimitações para o custo de $\mathscr{C}$. Um k-clustering dinâmico ótimo para o kCD-M para $\gamma = 0$ será denotado por $\mathscr{C}^* = (C_j^*)_{j=1}^k$.

Lema 6.12. Considere duas sequências de números reais positivos $a_1 \geqslant a_2 \geqslant \cdots \geqslant a_n$ e $b_1 \leqslant b_2 \leqslant \cdots \leqslant b_n$ tais que $a_1 - a_n \leqslant D$ e $b_n - b_1 \leqslant D$, para algum $D > 0$. Vale que

$$\max_{i \in [n]}(a_i + b_i) - \min_{i \in [n]}(a_i + b_i) \leqslant D.$$

Demonstração. Sejam p e q em $[n]$ tais que $a_p + b_p$ é máximo e $a_q + b_q$ é mínimo. Suponha que $p < q$. Logo,

$$\max_{i \in [n]}(a_i + b_i) - \min_{i \in [n]}(a_i + b_i) = a_p + b_p - a_q - b_q = a_p - a_q + b_p - b_q \leqslant D.$$

A desigualdade segue pelo fato de que $a_p - a_q \leqslant a_1 - a_n \leqslant D$ e $b_p - b_q \leqslant 0$, pois $p < q$. O caso $q > p$ é análogo. □

Lema 6.13. Seja $\mathscr{C} = (C_j)_{j=1}^k$ o k-clustering dinâmico obtido após a execução das partes (I) e (II). Vale que

$$\max_{j \in [k]} \text{diam}(C_j) \leqslant \sum_{j \in [k]} \frac{\text{diam}(C_j^*)}{k} + D.$$

Demonstração. Como a média de um conjunto de valores nunca é menor que o mínimo desses valores, temos que

$$\min_{j \in [k]} \text{diam}(C_j) \leqslant \sum_{j \in [k]} \frac{\text{diam}(C_j)}{k} \leqslant \sum_{j \in [k]} \frac{\text{diam}(C_j^*)}{k}.$$

A última desigualdade segue de (6.4).

Pelo Lema 6.12 e pelo procedimento guloso utilizado na parte (II), temos que, para cada instante t em $[T]$,

$$\max_{j\in[k]}\left(\sum_{t'=1}^{t}\operatorname{diam}(C_j^{t'})\right)-\min_{j\in[k]}\left(\sum_{t'=1}^{t}\operatorname{diam}(C_j^{t'})\right)\leqslant D.$$

Logo,

$$\max_{j\in[k]}\operatorname{diam}(C_j)\leqslant\min_{j\in[k]}\operatorname{diam}(C_j)+D\leqslant\sum_{j\in[k]}\frac{\operatorname{diam}(C_j^*)}{k}+D. \qquad \square$$

Teorema 6.14. O algoritmo resultante da combinação dos algoritmos para as partes (I) e (II) é uma 2-aproximação para o kCD-M quando $\gamma = 0$.

Demonstração. Seja $\mathscr{C} = (C_j)_{j=1}^k$ o k-clustering dinâmico obtido após execução das partes (I) e (II). Dado que $\gamma = 0$, temos que custo-m$_\gamma(\mathscr{C}) = \max_{j\in[k]}\operatorname{diam}_T(\mathscr{C}, j)$.

Como a média de um conjunto de valores nunca é maior que o máximo desses valores, temos que custo-m$_\gamma(\mathscr{C}^*) \geqslant \sum_{j\in[k]}\operatorname{diam}(C_j^*)/k$. Pela definição do valor D, também temos que custo-m$_\gamma(\mathscr{C}^*) \geqslant D$. Logo, pelo Lema 6.13,

$$\text{custo-m}_\gamma(\mathscr{C})\leqslant\sum_{j\in[k]}\frac{\operatorname{diam}(C_j^*)}{k}+D\leqslant 2\,\text{custo-m}_\gamma(\mathscr{C}^*).$$

Note que o tempo de processamento da parte (I) não interfere no tempo de processamento da parte (II). Logo, o algoritmo resultante da combinação das duas partes consome tempo $\mathrm{O}(Tn^2k + Tk\log k)$, para um dado D. Lembrando que temos que testar os $T\binom{n}{2}$ candidatos a valores de D e escolher o melhor k-clustering dinâmico entre os encontrados, o consumo de tempo do algoritmo é $\mathrm{O}((Tn(n-1)/2)(Tn^2k+Tk\log k)) = \mathrm{O}(T^2n^4k)$. $\square$

Capítulo 7

Considerações finais

Problemas de clustering são bem conhecidos e estudados, tanto no contexto estático quanto dinâmico. Nesta tese abordamos problemas de clustering novos, ou, pelo menos, pouco conhecidos. Classificamos os problemas por "clustering cinético" e "clustering dinâmico" para diferenciá-los, chamando a atenção para o fato dos adjetivos cinético e dinâmico serem utilizados com significados diferentes do usual.

No clustering cinético, o diferencial para os problemas já estudados da literatura é o fato dos problemas tratados não serem *on-line* e se exigir um k-clustering que considere todo o intervalo de tempo na função objetivo. No clustering dinâmico, o diferencial é a introdução da instabilidade, que permite um controle na mudança dos clusters.

A tabela 7.1 resume os principais algoritmos que desenvolvemos para os problemas de clustering cinético. Para o caso particular de trajetórias lineares por partes, apresentamos também um algoritmo que encontra um ε-representante linear, com ε ótimo. Esse algoritmo tem complexidade de tempo linear no número de trechos lineares da trajetória. Com o ε ótimo obtemos a melhor limitação possível para o erro aditivo dos algoritmos para trajetórias não lineares.

Variante	k**CC1D-S**	k**CC1D-M**
Geral	$\mathrm{O}((k-1)!\, n^{2k-1} \log n)$	$2(2+\sqrt{2})$-aproximação $((2+\sqrt{2}/2)+\varepsilon)$-aproximação
Bem-separado	$\mathrm{O}(n^6 \log n)$	$\mathrm{O}(n^6 \log n)$
$k = 3$	2-aproximação	—
Trajetórias não lineares	erro aditivo de até $2k\varepsilon$	erro aditivo de até 2ε NP-difícil
Movimento aleatório	erro aditivo de até $2k\varepsilon$	erro aditivo de até 2ε

Tabela 7.1: Resumo dos algoritmos e resultados para clustering cinético

Para os problemas de clustering dinâmico, além de alguns resultados sobre a estrutura de um k-clustering ótimo, apresentamos algoritmos e resultados de complexidade para variantes dos problemas. A tabela 7.2 apresenta um resumo.

Variante	**kCD-S**	**kCD-M**
Geral	$O(2^{kT}n^{2kT+1}k^3T/k!)$	$O(2^{kT}n^{2kT+1}k^3T/k!)$
$\gamma = 0$	$O(Tn \log n)$	2-aproximação NP-difícil
$\gamma = \infty$	—	NP-difícil
$\gamma = \infty$, sem cruzamentos	$O(Tn + n \log n)$	—

Tabela 7.2: Resumo dos resultados para clustering dinâmico

Não conhecemos provas de dificuldade para os problemas kCC1D-S, kCC1D-M e kCD-S, nem para o problema kCD-M quando $0 < \gamma < \infty$. Sabemos que as versões estáticas do kCC1D-S e kCC1D-M restritas ao caso unidimensional podem ser resolvidas em tempo polinomial.

A maioria dos resultados dos Capítulos 3 e 4 são originais. Os resultados já conhecidos são devidamente referenciados, com exceção da Proposição 3.2, que é um resultado bem conhecido, mas não conseguimos encontrar sua origem. Os resultados do Capítulo 5 foram obtidos em conversas com Cristiane M. Sato e Marcel K. C. Silva. O Capítulo 6 é o resultado de uma colaboração com Cristina G. Fernandes, Damien Regnault e Nicolas Shabanel.

Com isso, contribuímos com o estudo teórico de problemas de clustering de pontos em movimento. Apesar de restringirmos o movimento ao espaço unidimensional ($\mathbb{R}$), os problemas se mostraram desafiadores e nada triviais.

Além dos resultados apresentados nessa tese, durante o doutorado obtivemos alguns outros resultados em outros temas. Um deles foi sobre um problema de escalonamento chamado de *job shop* flexível [BFF+14]. Outros foram sobre decomposição de grafos [BMOW15a, BMOW15b].

Apêndice A

Discos no espaço métrico do diâmetro

Denotamos por $\mathcal{S}$ o conjunto de todas as trajetórias (lineares). Consideraremos, aqui, a função diam restrita apenas a conjuntos com no máximo duas trajetórias. Assim, para um conjunto de trajetórias $\{s_1, s_2\}$, escrevemos apenas $\text{diam}(s_1, s_2)$ e para um conjunto $\{s_1\}$ escrevemos $\text{diam}(s_1, s_1)$.

Podemos provar que $(\mathcal{S}, \text{diam})$ é um espaço métrico, isto é, para quaisquer s_1, s_2 e s_3 em $\mathcal{S}$, vale o seguinte:

1. $\text{diam}(s_1, s_2) \geqslant 0$;
2. $\text{diam}(s_1, s_2) = 0 \iff s_1 = s_2$;
3. $\text{diam}(s_1, s_2) = \text{diam}(s_2, s_1)$; e
4. $\text{diam}(s_1, s_2) + \text{diam}(s_2, s_3) \geqslant \text{diam}(s_1, s_3)$.

As três primeiras condições seguem trivialmente do fato de diam corresponder ao valor da área de um polígono que tem s_1 e s_2 entre suas arestas. A quarta condição segue do Lema 4.3.

Seja s uma trajetória em $\mathcal{S}$. O **disco** de raio r centrado em s é o subconjunto $B_2(s, r)$ de $\mathcal{S}$ que consiste de toda trajetória s' em $\mathcal{S}$ tal que $\text{diam}(s, s') \leqslant r$. No Lema 4.5, mostramos que a área da região$(B_2(s, r))$ é no máximo $(2 + \sqrt{2})r$. No entanto, como sugerido na Figura A.1, essa limitação superior não é justa. Note que as bordas laterais das figuras são levemente curvadas.

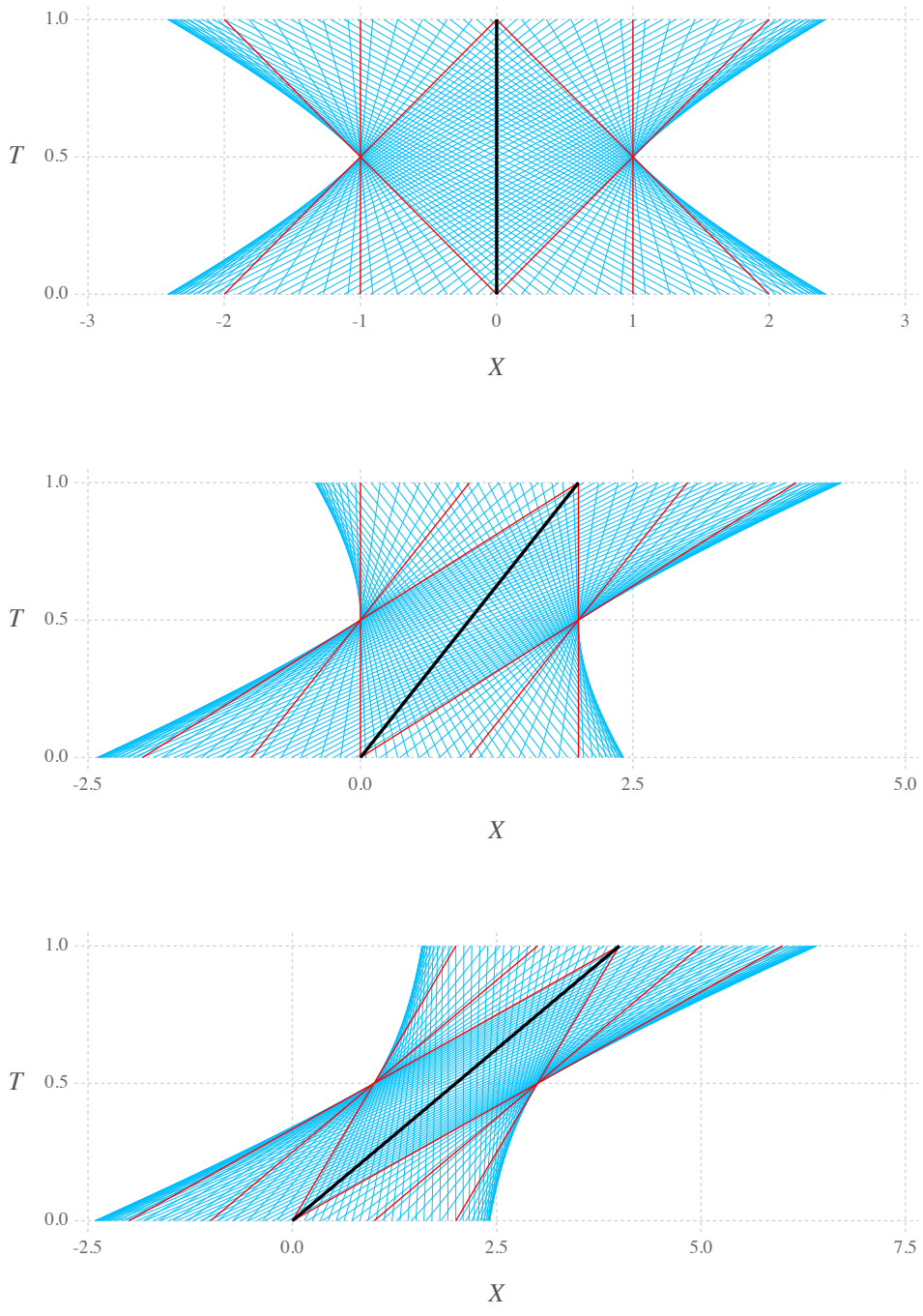


Figura A.1: Discos de raio 1 para três trajetórias centrais iniciando na origem e terminando nos pontos $(0, 1)$, $(2, 1)$ e $(4, 1)$. Segmentos azuis e vermelhos representam trajetórias com distância 1 da trajetória central. A cor vermelha serve apenas como referência.

Apêndice B

Programas lineares inteiros para clustering cinético e dinâmico

Apresentaremos modelos de programas lineares inteiros para os problemas kCC1D-S, kCC1D-M, kCD-S e kCD-M. Alguns desses modelos foram implementados utilizando a biblioteca de *solvers* conhecidos. O modelo para o kCC1D-S foi implementado com o Gurobi e o modelo para o kCD-M foi implementado com o CPLEX. O uso de bibliotecas diferentes está exclusivamente relacionado à disponibilidade de licença de uso na época em que cada modelo foi implementado.

Implementamos os dois modelos citados apenas para ajudar no entendimento dos problemas. Em especial, eles foram utilizados apenas para resolver instâncias pequenas dos problemas, como a instância descrita no Apêndice C e no exemplo da Figura 6.4, por exemplo. Para instâncias pequenas,a solução ótima era obtida rapidamente. O estudo dos modelos em si não foi o foco da tese. Logo, não temos como afirmar qualquer coisa sobre a qualidade desses modelos.

Seja S um conjunto de n trajetórias e k um número inteiro em $[n]$. Lembre que B denota o conjunto de buracos de S. Considere as seguintes variáveis:

- $x_{ij} \in \{0, 1\}$, para todo $i \in S$ e todo $j \in [k]$;
- $y_{bj} \in \{0, 1\}$, para todo $b \in B$ e todo $j \in [k]$;
- $w_{bj} \in \{0, 1\}$, para todo $b \in B$ e todo $j \in [k]$;
- $z_{bj} \in \{0, 1\}$, para todo $b \in B$ e todo $j \in [k]$.

O valor de uma variável x_{ij} indica se a trajetória i pertence ao cluster C_j ou não. O valor da variável y_{bj} indica se o buraco b está contido em região(C_j). As variáveis w_{bj} e z_{bj} indicam, respectivamente, se $C_j \cap S_e(b) \neq \emptyset$ e se $C_j \cap S_d(b) \neq \emptyset$. Com essas variáveis, podemos modelar o kCC1D-S como programa linear inteiro da seguinte forma, com as restrições de variáveis omitidas, pois são as mesmas descritas anteriormente.

$$\text{Minimizar} \quad \sum_{j\in[k]} \sum_{b\in B} \text{área}(b)\, y_{bj} \tag{B.1}$$

$$\text{s. a} \quad \sum_{j\in[k]} x_{ij} = 1, \qquad \text{para todo } i \in S \tag{B.2}$$

$$y_{bj} - w_{bj} - z_{bj} \geqslant -1, \qquad \text{para todo } b \in S \text{ e todo } j \in [k] \tag{B.3}$$

$$|S_e(b)| w_{bj} - \sum_{i\in S_e(b)} x_{ij} \geqslant 0, \text{ para todo } b \in S \text{ e todo } j \in [k] \tag{B.4}$$

$$|S_d(b)| z_{bj} - \sum_{i\in S_d(b)} x_{ij} \geqslant 0, \text{ para todo } b \in S \text{ e todo } j \in [k]. \tag{B.5}$$

Na função objetivo (B.1), o diâmetro de um cluster C_j é calculado como a soma das áreas dos buracos contidos em região(C_j). Um buraco b está contido em região(C_j) se e somente se existem uma trajetória de C_j à esquerda de b e uma trajetória de C_j à direita de b, isto é, se e somente se $C_j \cap S_e(b)$ e $C_j \cap S_d(b)$ não são vazios. Isso é capturado pela restrição (B.3) juntamente com a minimização da função objetivo, pois $y_{bj} = 1$ se $w_{bj} = z_{bj} = 1$, caso contrário $y_{bj} = 0$ satisfaz (B.3) e é melhor para a função objetivo. A restrição (B.4) garante que $w_{bj} = 1$ se $C_j \cap S_e(b) \neq \emptyset$ e, de forma análoga, por (B.5), temos que $z_{bj} = 1$ se $C_j \cap S_d(b) \neq \emptyset$.

Pela restrição (B.2), cada trajetória pertencerá a exatamente um cluster. Logo, as variáveis x definem um k-clustering e em uma solução ótima são escolhidas de forma a minimizar a soma dos diâmetros de cada cluster.

O kCC1D-M pode ser modelado de maneira semelhante, bastando alterar a função objetivo. Para isso, adicionamos uma variável real $f \geqslant 0$ que representará o valor do diâmetro máximo de um cluster no k-clustering descrito pelas variáveis x.

$$\text{Minimizar} \quad f \tag{B.6}$$

$$\text{s. a} \quad f - \sum_{b\in B} \text{área}(b)\, y_{bj} \geqslant 0, \qquad \text{para todo } j \in [k] \tag{B.7}$$

$$\sum_{j\in[k]} x_{ij} = 1, \qquad \text{para todo } i \in S \tag{B.8}$$

$$y_{bj} - w_{bj} - z_{bj} \geqslant -1, \qquad \text{para todo } b \in S \text{ e todo } j \in [k] \tag{B.9}$$

$$|S_e(b)| w_{bj} - \sum_{i\in S_e(b)} x_{ij} \geqslant 0, \text{para todo } b \in S \text{ e todo } j \in [k] \tag{B.10}$$

$$|S_d(b)| z_{bj} - \sum_{i\in S_d(b)} x_{ij} \geqslant 0, \text{ para todo } b \in S \text{ e todo } j \in [k]. \tag{B.11}$$

Agora, para o problema kCD-M, utilizaremos os conjuntos $[k]$ e $[T]$ para denotar os conjuntos de clusters dinâmicos e o conjunto de instantes, respectivamente. Como parâmetros temos $\gamma \geqslant 0$ para controlar a instabilidade e p_{it} indicando a posição da trajetória i no instante t. Considere as seguintes variáveis:

- $x_{ijt} \in \{0, 1\}$, para todo $i \in S$, todo $j \in [k]$ e todo $t \in [T]$;
- $d_{jt} \in \mathbb{R}$, para todo $j \in [k]$ e todo $t \in [T]$;
- $s_{it} \in \{0, 1\}$, para todo $i \in S$ e todo $t \in [T-1]$;
- $f \in \mathbb{R}$.

O valor de uma variável x_{ijt} indica se a trajetória i pertence ao cluster C_j^t ou não. O valor da variável d_{jt} armazena o valor do diâmetro do cluster C_j^t. Cada variável s_{it} indica se a trajetória i mudou de cluster entre os instantes t e $t+1$. Com essas variáveis, podemos modelar o kCD-M como programa linear inteiro da seguinte forma, com as restrições de variáveis omitidas, pois são as mesmas descritas anteriormente.

$$\text{Minimizar} \quad f + \gamma \sum_{t \in [T-1]} s_{it} \tag{B.12}$$

$$\text{s. a} \quad f - \sum_{t \in [T]} d_{jt} \geqslant 0, \qquad \forall j \in [k] \tag{B.13}$$

$$\sum_{j \in [k]} x_{ijt} = 1, \qquad \forall i \in S \text{ e } \forall t \in [T] \tag{B.14}$$

$$d_{jt} - p_{it}x_{ijt} + p_{\ell t}x_{\ell jt} + p_{it}(1 - x_{\ell jt}) \geqslant 0, \quad \forall i, \ell \in S, \forall j \in [k] \text{ e } \forall t \in [T] \tag{B.15}$$

$$s_{it} - x_{ijt} + x_{ij(t=1)} \geqslant 0, \qquad \forall i \in S, \forall j \in [k] \text{ e } \forall t \in [T-1]. \tag{B.16}$$

A restrição (B.14) garante que, em cada instante, um ponto pertença a um único clusters. A restrição (B.15) impede que o valor da variável d_{jt} seja menor que a distância entre as trajetórias do cluster C_j^t mais distantes. Como queremos minimizar o valor de d_{jt}, no final, seu valor corresponderá ao diâmetro do cluster C_j^t.

O kCD-S pode ser modelado de maneira semelhante. Para isso, removemos a restrição (B.13) e a variável f por $\sum_{j \in [k]} \sum_{t \in [T]} d_{jt}$ na função objetivo.

Apêndice C

Exemplo não bem-separado para kCC1D-S

A seguir mostraremos um exemplo onde o único 3-clustering ótimo para o kCC1D-S não é bem-separado. Uma trajetória que começa em um ponto $(x_0, 0)$ e termina em um ponto $(x_1, 1)$ será denotada por $x_0 \to x_1$.

Considere os seguintes conjuntos de trajetórias, onde a é uma variável inteira:

- $D = \{0{,}0 \to 1{,}02\}$;
- $E_1 = \{0{,}005a \to 0{,}75 + 0{,}005a \mid 0 \leqslant a \leqslant 5\}$;
- $E_2 = \{0{,}01a \to 0{,}75 + 0{,}01a \mid 3 \leqslant a \leqslant 6 \text{ ou } 19 \leqslant a \leqslant 20\}$;
- $E_3 = \{0{,}02a \to 0{,}75 + 0{,}02a \mid 4 \leqslant a \leqslant 9\}$;
- $F = \{0{,}75 + 0{,}004a \to 0{,}75 + 0{,}004a \mid 0 \leqslant a \leqslant 67\} \cup \{1{,}02 \to 1{,}02\}$;
- $S = D \cup E_1 \cup E_2 \cup E_3 \cup F$.

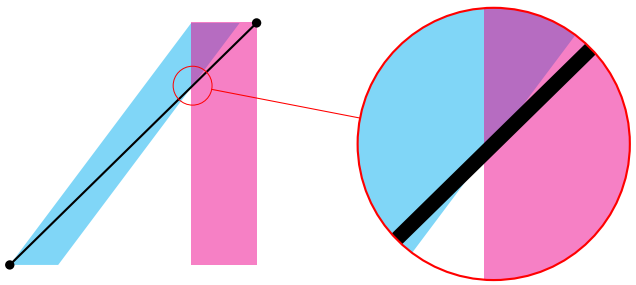


Figura C.1: A única trajetória em D está destacada. As trajetórias em $E_1 \cup E_2 \cup E_3$ estão dentro da região azul. As trajetórias em F estão dentro da região magenta.

O conjunto de trajetórias S, ilustrado na Figura C.1, contém 88 trajetórias. Todas as trajetórias em $E_1 \cup E_2 \cup E_3$ são paralelas e todas as trajetórias em F também são paralelas.

O único 3-clustering ótimo de S para o kCC1D-S é $\mathcal{C}^* = \{D, E_1 \cup E_2 \cup E_3, F\}$, cuja soma dos diâmetros de seus clusters é $\text{sd}(\mathcal{C}^*) = 0{,}47$. É possível verificar que o ponto de intersecção da trajetória em D com a trajetória de E mais à direita fica dentro de região(F). Logo, existe um único buraco descoberto por $\mathcal{C}^*$, que separa $D \cup E_1 \cup E_2 \cup E_3$ de F. Qualquer buraco que separa D de $E_1 \cup E_2 \cup E_3$ está coberto por F.

O fato de $\mathcal{C}^*$ ser o único 3-clustering ótimo para S foi verificado experimentalmente resolvendo o programa linear inteiro para o kCC1D-S descrito no apêndice B. Para isso foi utilizada a biblioteca do Gurobi Optimizer versão 6.0.

Referências Bibliográficas

[And79] Alex M. Andrew, *Another efficient algorithm for convex hulls in two dimensions*, Information Processing Letters **9** (1979), número 5, 216–219.

[Ata85] Mikhail Jibrayil Atallah, *Some dynamic computational geometry problems*, Computers & Mathematics with Applications **11** (1985), número 12, 1171–1181.

[BFF+14] Ernesto Julián Goldberg Birgin, Paulo Feofiloff, Cristina Gomes Fernandes, Everton Luiz de Melo, Marcio Takashi Iura Oshiro e Débora Pretti Ronconi, *A MILP model for an extended version of the Flexible Job Shop Problem*, Optimization Letters **8** (2014), número 4, 1417–1431.

[BGH99] Julien Basch, Leonidas John Guibas e John Hershberger, *Data Structures for Mobile Data*, Journal of Algorithms **31** (1999), 1–28.

[BMOW15a] Fabio Botler, Guilherme Oliveira Mota, Marcio Takashi Iura Oshiro e Yoshiko Wakabayashi, *Decompositions of highly connected graphs into paths of length five*, ArXiv e-prints (2015).

[BMOW15b] ———, *Decomposing highly edge-connected graphs into paths of any given length*, ArXiv e-prints (2015).

[BO79] Jon Louis Bentley e Thomas A. Ottmann, *Algorithms for Reporting and Counting Geometric Intersections*, IEEE Transactions on Computers **C-28** (1979), número 9, 643–647.

[Bru78] Peter Brucker, *On the Complexity of Clustering Problems*, Optimization and Operations Research (Rudolf Henn, Bernhard Korte e Werner Oettli, eds.), Lecture Notes in Economics and Mathematical Systems, volume 157, Springer Berlin Heidelberg, 1978, páginas 45–54.

[Cat38] Eugène Charles Catalan, *Note sur une équation aux différences finies*, Journal de Mathématiques Pures et Appliquées **3** (1838), 508–516.

[Cor71] Richard Melville Cormack, *A review of classification*, Journal of the Royal Statistical Society. Series A (General) **134** (1971), número 3, 321–367.

[CP04] Moses Charikar e Rina Panigrahy, *Clustering to Minimize the Sum of Cluster Diameters*, Journal of Computer System Sciences **68** (2004), número 2, 417–441.

[CW14] Danny Ziyi Chen e Haitao Wang, *New Algorithms for Facility Location Problems on the Real Line*, Algorithmica **69** (2014), número 2, 370–383.

[Das07] Sanjoy Dasgupta, *The hardness of k-means clustering*, Relatório Técnico CS2007-0890, Department of Computer Science and Engineering, University of California, San Diego, California, 2007.

[DGL10] Bastian Degener, Joachim Gehweiler e Christiane Lammersen, *Kinetic Facility Location*, Algorithmica **57** (2010), número 3, 562–584.

[DLS12] Amit Daniely, Nati Linial e Michael Ezra Saks, *Clustering is difficult only when it does not matter*, ArXiv e-prints (2012).

[Fre00] Eduardo Garcia de Freitas, *Problemas Cinéticos em Geometria Computacional*, Dissertação de Mestrado, Universidade de São Paulo, 2000.

[GGH+03] Jie Gao, Leonidas John Guibas, John Hershberger, Li Zhang e An Zhu, *Discrete Mobile Centers*, Discrete & Computational Geometry **30** (2003), número 1, 45–63.

[GJS76] Michael Randolph Garey, David Stifler Johnson e Larry Joseph Stockmeyer, *Some simplified NP-complete graph problems*, Theoretical Computer Science **1** (1976), número 3, 237–267.

[Gon85] Teofilo Francisco Gonzalez, *Clustering to minimize the maximum intercluster distance*, Theoretical Computer Science **38** (1985), 293–306.

[Gra72] Ronald Lewis Graham, *An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set*, Information Processing Letters **1** (1972), número 4, 132–133.

[HKW14] Tanja Hartmann, Andrea Kappes e Dorothea Wagner, *Clustering Evolving Networks*, ArXiv e-prints (2014).

[HN79] Wen-Lian Hsu e George Lann Nemhauser, *Easy and hard bottleneck location problems*, Discrete Applied Mathematics **1** (1979), número 3, 209–215.

[Hoe63] Wassily Hoeffding, *Probability Inequalities for Sums of Bounded Random Variables*, Journal of the American Statistical Association **58** (1963), número 301, 13–30.

[HP04] Sariel Har-Peled, *Clustering Motion*, Discrete & Computational Geometry **31** (2004), número 4, 545–565.

[HS86] Dorit Simona Rotner Hochbaum e David Bernard Shmoys, *A Unified Approach to Approximation Algorithms for Bottleneck Problems*, Journal of the ACM **33** (1986), número 3, 533–550.

[HT91] Refael Hassin e Arie Tamir, *Improved complexity bounds for location problems on the real line*, Operations Research Letters **10** (1991), número 7, 395–402.

[JRS07] Laura E. Jackson, George N. Rouskas e Matthias F.M. Stallmann, *The directional p-median problem: Definition, complexity, and algorithms*, European Journal of Operational Research **179** (2007), número 3, 1097–1108.

[JV01] Kamal Jain e Vijay V. Vazirani, *Approximation Algorithms for Metric Facility Location and k-Median Problems Using the Primal-dual Schema and Lagrangian Relaxation*, Journal of the ACM **48** (2001), número 2, 274–296.

[Kar72] Richard Manning Karp, *Reducibility among Combinatorial Problems*, Complexity of Computer Computations (Raymond E. Miller, James W. Thatcher e Jean D. Bohlinger, eds.), The IBM Research Symposia Series, Springer US, 1972, páginas 85–103.

[KH79a] Oded Kariv e Seifollah Louis Hakimi, *An Algorithmic Approach to Network Location Problems. I: The p-Centers*, SIAM Journal on Applied Mathematics **37** (1979), número 3, 513–538.

[KH79b] ———, *An Algorithmic Approach to Network Location Problems. II: The p-Medians*, SIAM Journal on Applied Mathematics **37** (1979), número 3, 539–560.

[KLS13] Marek Karpinski, Andrzej Lingas e Dzmitry Sledneu, *Optimal cuts and partitions in tree metrics in polynomial time*, Information Processing Letters **113** (2013), número 12, 447–451.

[LHW07] Jae-Gil Lee, Jiawei Han e Kyu-Young Whang, *Trajectory Clustering: A Partition-and-group Framework*, Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (New York, NY, USA), SIGMOD '07, ACM, 2007, páginas 593–604.

[MNV09] Meena Mahajan, Prajakta Nimbhorkar e Kasturi Varadarajan, *The Planar k-Means Problem is NP-Hard*, WALCOM: Algorithms and Computation (Sandip Das e Ryuhei Uehara, eds.), Lecture Notes in Computer Science, volume 5431, Springer Berlin Heidelberg, 2009, páginas 274–285.

[MS84] Nimrod Megiddo e Kenneth Jay Supowit, *On the Complexity of Some Common Geometric Location Problems*, SIAM Journal on Computing **13** (1984), número 1, 182–196.

[MT82] Nimrod Megiddo e Arie Tamir, *On the complexity of locating linear facilities in the plane*, Operations Research Letters **1** (1982), número 5, 194–197.

[MT83] ———, *Finding Least-Distances Lines*, SIAM Journal on Algebraic Discrete Methods **4** (1983), número 2, 207–211.

[O'R98] Joseph O'Rourke, *Computational Geometry in C*, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1998.

[Pun91] Abraham P. Punnen, *A linear time algorithm for the maximum capacity path problem*, European Journal of Operational Research **53** (1991), número 3, 402–404.

[RD69] Basil Cameron Rennie e Annette Jane Dobson, *On Stirling Numbers of the Second Kind*, Journal of Combinatorial Theory **7** (1969), número 2, 116–121.

[SH76] Michael Ian Shamos e Dan Hoey, *Geometric intersection problems*, Foundations of Computer Science, 1976., 17th Annual Symposium on, Oct 1976, páginas 208–215.