Algoritmos de Otimização para Construção de Tesselações Centroidais de Voronoi

Arthur Gabriel de Santana
Orientador: Ernesto G. Birgin

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

23 de Novembro de 2018

Resumo

Diagramas de Voronoi são particionamentos do $\mathbb{R}^n$ formados por células definidas de acordo com a distância de cada ponto a certos pontos geradores. Tesselações Centroidais de Voronoi (TCVs) são um tipo particular de Diagrama de Voronoi que encontra diversas aplicações em áreas como Pesquisa Operacional, simulação e *clustering*. A construção de TCVs geralmente envolve métodos iterativos e pode ser interpretada como um problema de otimização. Neste trabalho, estudamos propriedades básicas de Diagramas de Voronoi e TCVs, além de algoritmos para a construção destas estruturas.

Palavras-chave: Diagrama de Voronoi, Tesselação Centroidal de Voronoi, Algoritmo de Fortune, Algoritmo de Lloyd, Otimização Não-Linear, Clustering

Abstract

Voronoi Diagrams are partitions of $\mathbb{R}^n$ made of cells that are defined according to the distance between each point and certain generating points. Centroidal Voronoi Tesselations (CVTs) are a particular kind of Voronoi Diagram that has applications in areas such as operations research, simulation and clustering. The generation of CVTs usually involves iterative methods and may be seen as an optimization problem. In this work, we study basic properties of Voronoi Diagrams and CVTs, in addition to algorithms for the construction of such structures.

Keywords: Voronoi Diagrams, Centroidal Voronoi Tesselations, Fortune's algorithm, Lloyd's algorithm, Nonlinear Optimization, Clustering

Agradecimentos

Agradeço ao professor Ernesto G. Birgin, pelo grande esforço e ótima orientação que me foi prestada durante os últimos três semestres. Agradeço também aos professores Pedro da Silva Peixoto, Cristina Gomes Fernandes e Antoine Laurain, que dedicaram tempo precioso me ajudando a entender os problemas de que trata esse trabalho.

Conteúdo

Introdução

Motivação

Tesselações Centroidais de Voronoi (TCVs) são uma construção geométrica que encontra diversas aplicações em áreas como pesquisa operacional, simulação, compressão de imagens e *clustering*, entre outras [7].

Uma aplicação em particular foi a motivação para o estudo que realizamos: a determinação de malhas geodésicas esféricas para a solução de equações diferenciais em modelos atmosféricos globais. Considerações numéricas sugerem alguns critérios de qualidade para tais malhas, como alinhamento das arestas [16], isotropia ou uniformidade de áreas [14].

Os métodos tradicionais para a determinação de malhas não consideram estes critérios explicitamente. Por este motivo, procuramos entender o problema do ponto de vista da Otimização Não-Linear, de forma a podermos futuramente desenvolver métodos que produzam boas malhas diretamente, por exemplo, adicionando um termo que penaliza a falha nos critérios selecionados.

Implementamos, na linguagem Julia, todos os algoritmos estudados. Para tal, não utilizamos nenhuma biblioteca externa, com exceção da biblioteca *Matplotlib*, para vizualização. Fizemos isso porque a implementação direta dos algoritmos, inclusive das estruturas de dados envolvidas, possibilita a utilização de ferramentas de diferenciação automática, o que pode acabar sendo necessário para a incorporação dos critérios na geração das malhas.

Tendo em vista a dificuldade de implementação dos algoritmos para malhas esféricas geodésicas e o curto tempo disponível para o término deste trabalho, simplificamos o problema, limitando-o a malhas em caixas no $\mathbb{R}^2$.

Organização do Texto

No Capítulo 1, introduzimos a ideia de Diagrama de Voronoi e discutimos algumas propriedades geométricas que são fundamentais para a sua construção eficiente, apresentada no capítulo seguinte. Em seguida, definimos o objeto matemático de maior interesse neste trabalho, as Tesselações Centroidais de Voronoi. Ao longo do capítulo, apresentamos algumas aplicações e um breve histórico dos conceitos.

No Capítulo 2, desenvolvemos o algoritmo de Fortune, um método eficiente para a construção de Diagramas de Voronoi no $\mathbb{R}^2$. O leitor poderá, se desejar, ignorar este capítulo e utilizar o procedimento para a construção dos diagramas como uma caixa-preta. No entanto, o algoritmo de Fortune exibe técnicas fundamentais da Geometria Computacional e seu estudo é, portanto, bastante proveitoso.

No Capítulo 3, introduzimos o algoritmo de Lloyd, um dos algoritmos mais usados para a construção de TCVs. Apresentamos o problema que deu origem ao algoritmo: a quantização ótima de sinais analógicos. A partir deste problema, generalizamos o tratamento para um número arbitrário de dimensões, interpretando a construção de TCVs como um problema de Otimização Não-Linear.

No Capítulo 4, apresentamos nossa implementação dos algoritmos discutidos e comparamos seus desempenhos.

Finalmente, apontamos nossas conclusões e próximos passos no capítulo final.

Notação e Definições Preliminares

Definição (Fecho)**.** *Seja $X \subset \mathbb{R}^n$. O* fecho *de X, denotado $\overline{X}$, é o menor conjunto fechado que contém X.*

Definição (Tesselação)**.** *Seja Ω um conjunto aberto no $\mathbb{R}^n$. Um conjunto $\{V_i\}_{i=1}^m$ é uma tesselação de Ω se $\bigcup_{i=1}^m \overline{V_i} = \overline{\Omega}$.*

Definição (Norma L_p)**.** *Sejam $n \in \mathbb{N}$ e $x \in \mathbb{R}^n$. Definimos a norma L_p de x, denotada $\|x\|_p$, por:*

$$\|x\|_p = \Big(\sum_{i=1}^{n} |x_i|^p\Big)^{\frac{1}{p}}$$

Definição (Norma L_∞)**.** *Seja $x \in \mathbb{R}^n$. Definimos a norma L_∞ de x, denotada $\|x\|_\infty$, por:*

$$\|x\|_\infty = \max_{i=1}^{n} |x_i|$$

Capítulo 1

Diagramas de Voronoi

1.1 Contexto e Definição

Há diversos problemas em que se deseja classificar os pontos de um conjunto de acordo com a distância a certos representantes. Quando o domínio é contínuo, chegamos à ideia de Diagramas de Voronoi:

Definição 1.1 (Diagrama de Voronoi)**.** *Dados um conjunto aberto* $\Omega \subseteq \mathbb{R}^n$ *e um conjunto de pontos não coincidentes* $Z = \{z_i\}_{i=1}^m$ *em* $\overline{\Omega}$*, a* região de Voronoi *associada ao* ponto gerador z_i *é definida como o conjunto*

$$V_i(Z) = \{x \in \Omega \mid \|x - z_i\| < \|x - z_j\|, \forall j \neq i\}.$$

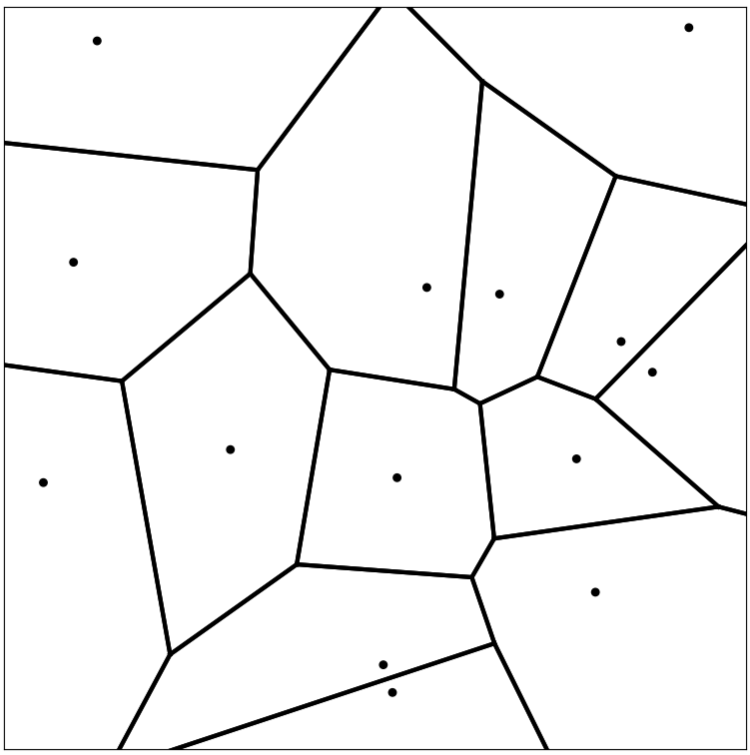
Chamamos o conjunto $V(Z) = \{V_i(Z)\}_{i=1}^m$ *de* diagrama de Voronoi.

Neste trabalho, utilizaremos a norma L_2 (norma euclidiana), mas grande parte da teoria pode ser estendida para normas diferentes [7].

Uma breve história do conceito é apresentada por [11]. Citamos aqui um resumo dos principais desenvolvimentos. O primeiro estudo sobre diagramas de Voronoi foi feito por Dirichlet [4], em 1850, associado a polinômios multivariados de grau dois, em duas ou três dimensões. Em 1907, Voronoi [19] generalizou o estudo para um número de dimensões arbitrárias. Em 1928, Delaunay [3] definiu o que hoje conhecemos como *triangulação de Delaunay*, um diagrama intimamente relacionado ao de Voronoi, que encontra diversas aplicações, por exmplo, na geração de malhas.

Os diagramas de Voronoi foram redescobertos diversas vezes através do século XX. Por este motivo, o conceito é conhecido por diferentes termos, de acordo com a área de aplicação. Quando aplicados à estimativa de precipitação de chuvas, as regiões de Voronoi são chamadas *polígonos de Thiessen*. Para análise de distribuição de minerais, são conhecidas como *polígonos de área de influência*. Em Cristalografia, casos particulares são conhecidos como *regiões de Wigner-Seitz* e *zonas de Brillouin*. Em Teoria da Informação, as

Figura 1.1: Um diagrama de Voronoi em $\mathbb{R}^2$, com 14 pontos geradores, escolhidos aleatoriamente. Os conjuntos mais externos se estendem para fora da figura e vários são ilimitados.



regiões de máxima verossimilhança são regiões de Voronoi. A Ecologia define o conceito relacionado de *área potencialmente disponível*. Na Anatomia, *domínio capilar* se refere à região de Voronoi definida em um tecido pelo centro de um capilar.

A primeira aplicação documentada de diagramas de Voronoi foi feita pelo epidemiologista John Snow em 1855 [11], à modelagem da epidemia de cólera de 1854, em Londres.

1.2 Caracterização Geométrica

Dados dois pontos geradores z_i e z_j, o hiperplano dos pontos em $\mathbb{R}^n$ equidistantes a z_i e z_j divide Ω em dois conjuntos: um contendo z_i e outro contendo z_j. Seguindo [1], denotemos o primeiro por $h(z_i, z_j)$:

$$h(z_i, z_j) := \{x \in \Omega \mid \|x - z_i\| < \|x - z_j\|\}$$

Podemos, então, descrever a região de Voronoi associada a z_i na forma de uma intersecção de conjuntos:

$$V_i(Z) = \bigcap_{\substack{j=1 \\ j \neq i}}^{m} h(z_i, z_j) \tag{1.1}$$

Quando Ω é convexo, cada $h(z_i, z_j)$ é um conjunto convexo. Portanto, segue imediatamente de (1.1) que as regiões de Voronoi são conjuntos convexos.

Consideremos agora apenas diagramas no $\mathbb{R}^2$. Dadas duas regiões de Voronoi $V_i(Z)$ e $V_j(Z)$ adjacentes, a aresta que separa $V_i(Z)$ e $V_j(Z)$ está contida na reta $\{x \mid \|x - z_i\| = \|x - z_j\|\}$.

Analogamente, um vértice v que separa três ou mais regiões $\{V_k(Z)\}_{k \in K}$ tem a propriedade de que, para algum $r > 0$, $\|v - z_k\| = r, \forall k \in K$. Em outras palavras, v é o centro da circunferência que contém os pontos geradores $\{z_k\}_{k \in K}$.

Para formalizarmos as afirmações acima, é necessário definirmos dois conceitos:

Definição 1.2 (Círculo vazio)**.** *Um círculo que não contém nenhum ponto gerador em seu interior é chamado* círculo vazio.

Definição 1.3 (Reta bissetora)**.** *O conjunto dos pontos equidistantes entre dois pontos geradores z_i e z_j é chamado* reta bissetora entre z_i e z_j.

Teorema 1.1 (Caracterização de vértices e arestas)**.** *Seja $V(Z)$ um diagrama de Voronoi. Então, valem:*

1. *Um ponto p é um vértice de $V(Z)$ se e somente se o maior círculo vazio com centro em p contém três ou mais pontos geradores em sua borda.*

2. *A reta bissetora r entre dois pontos geradores z_i e z_j define uma aresta em V se e somente se existe um ponto p em r tal que o maior círculo vazio com centro em p contém z_i e z_j, mas nenhum outro ponto gerador, em sua borda.*

Demonstração. [1], página 150. □

1.3 Tesselações Centroidais de Voronoi

Definição 1.4 (Centróide)**.** *Dados um conjunto aberto $\Omega \subseteq \mathbb{R}^n$, um conjunto de pontos não coincidentes $Z = \{z_i\}_{i=1}^m$ em $\overline{\Omega}$ e uma função de densidade de probabilidade ρ, a centróide de uma região de Voronoi V_i é dada por*

$$C_i(Z) = \frac{\int_{V_i(Z)} x\rho(x)\ \mathrm{d}x}{\int_{V_i(Z)} \rho(x)\ \mathrm{d}x}$$

Por conveniência, definimos também $C(Z) = \{C_i(Z)\}_{i=1}^m$.

A centróide, também chamada de *centro de massa*, pode ser interpretada como a média dos pontos de uma região, ponderados pela *função de densidade* $\rho(x)$. O denominador pode ser visto como a *massa total* de V_i. Quando $\rho(x) = 1$, isso corresponde à área de V_i.

A centróide de uma região não coincide necessariamente com o ponto gerador. Na verdade, esta situação raramente ocorre com pontos geradores aleatórios. Por este motivo, damos um nome especial para este caso:

Definição 1.5 (Tesselação Centroidal de Voronoi)**.** *Seja $V(Z)$ um diagrama de Voronoi. Se todos os pontos geradores z_i coincidem com as centróides $C_i(Z)$, $V(Z)$ é chamado de* Tesselação Centroidal de Voronoi *(TCV).*

Neste trabalho, será conveniente termos em mente o problema de *quantização*, ou *clustering*: queremos escolher um número finito de representantes para um domínio infinito (e normalmente ilimitado), de forma a obedecer um determinado critério de qualidade, que dependerá da aplicação em questão. Veremos mais adiante que os pontos geradores de uma TCV são pontos estacionários do seguinte critério bastante natural:

$$\mathcal{G}(Z) = \sum_{i=1}^{m} \int_{V_i(Z)} \|x - z_i\|^2\ \rho(x)\ \mathrm{d}x$$

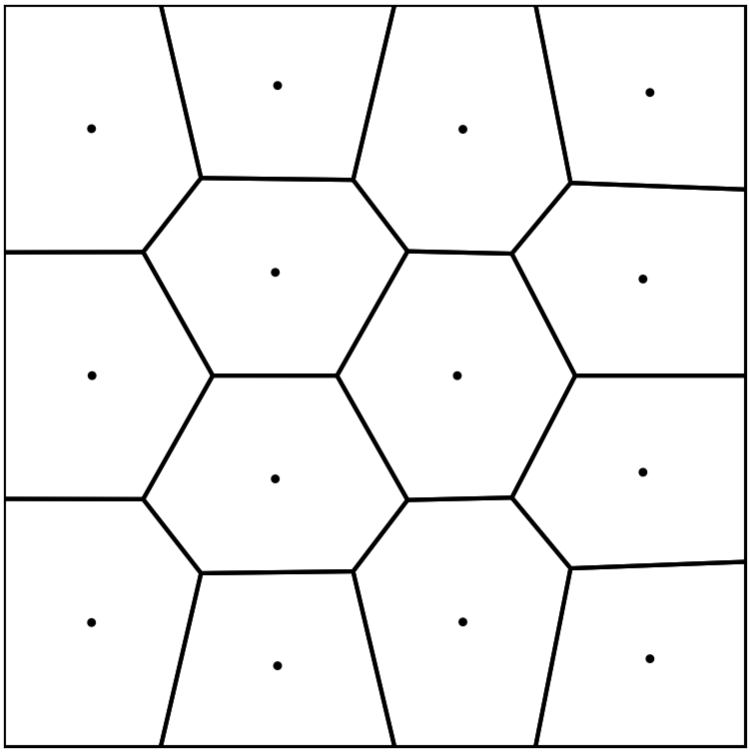
[7] apresenta uma série de aplicações de TCVs. Discutiremos brevemente três destas, que podem ser interpretadas como casos particulares de quantização ótima, ou seja, as soluções são pontos estacionários de $\mathcal{G}$.

1.3.1 Compressão de Imagens

Suponha que tenhamos uma imagem armazenada como uma matriz de pixels, onde cada pixel é uma tripla de números no intervalo $[0, 1]$, representando uma combinação de vermelho, verde e azul.

Se quisermos comprimir a imagem (com perdas), uma ideia seria dividir o espaço das cores, $[0, 1]^3$, em um número finito m de conjuntos $\{Q_i\}_{i=1}^m$, cada um com uma cor representante z_i, e armazenar para cada pixel apenas

Figura 1.2: Uma Tesselação Centroidal de Voronoi em $\mathbb{R}^2$, com 14 pontos geradores.

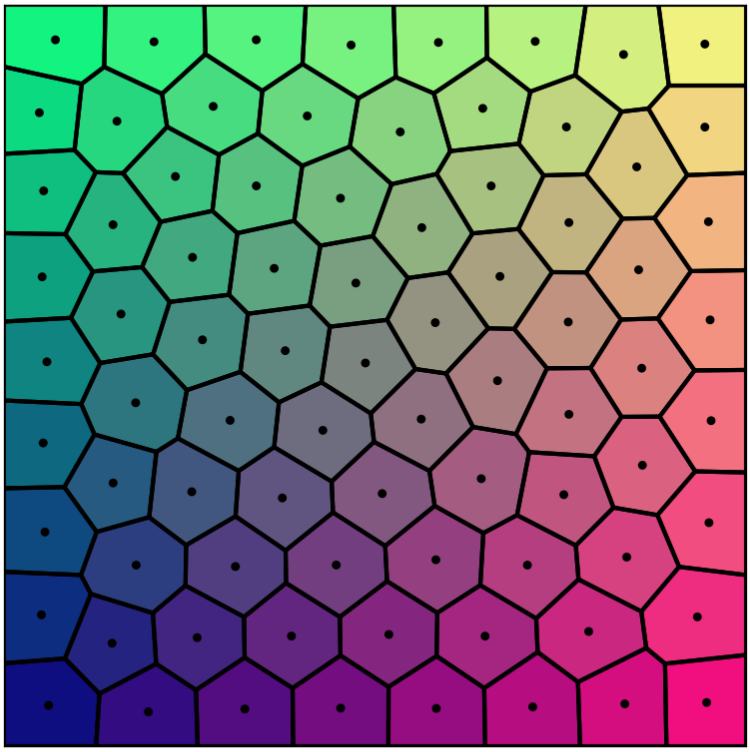


o índice da cor representante do conjunto a que pertence a cor do pixel na imagem original.

Quais são os conjuntos Z e Q que fazem a imagem comprimida melhor aproximar a imagem original? Se calcularmos a frequência ρ de cada cor na imagem, é natural querermos minimizar o quadrado das distâncias (no espaço das cores) entre cada pixel e sua versão comprimida. Ou seja, se definirmos $\gamma(x)$ como o índice i do conjunto Q_i a que pertence a cor do pixel x, queremos um minimizador da função

$$\mathcal{E}(Z) = \sum_{i=1}^{m} \sum_{x \in Q_i} \|x - z_{\gamma(x)}\|^2 \ \rho(x) \ .$$

Figura 1.3: Espaço das cores (limitado a um valor fixo na coordenada correspondente à cor azul) quantizado em 80 classes. Cada região foi preenchida com a cor correspondente ao seu ponto representante.



Claramente, trata-se da versão discretizada de $\mathcal{G}$. Aproximando $\mathcal{E}$ por $\mathcal{G}$, os Z e Q ótimos formarão uma TCV, com $Q = V(Z)$.

1.3.2 Alocação Ótima de Recursos

Suponha que m terminais bancários estejam distribuídos em uma região, de acordo com o conjunto de posições $Z = \{z_i\}_{i=1}^{m}$. Em geral, cada cliente procurará se dirigir ao terminal mais próximo. Assim, a área servida por cada terminal i é dada por $V_i(Z)$.

Como escolher a melhor localização para os terminais? Vários critérios

poderiam ser razoáveis. Por exemplo, poderíamos minimizar a maior distância percorrida por um cliente. Pode se argumentar, porém, que isso valoriza demais este cliente em detrimento dos outros.

Novamente, uma solução natural seria minimizar a soma do quadrado das distâncias percorridas. Utilizando uma aproximação contínua e conhecendo uma função ρ que exprime a densidade populacional em cada local da região, queremos minimizar

$$\mathcal{G}(Z) = \sum_{i=1}^{m} \int_{V_i(Z)} \|x - z_i\|^2 \ \rho(x) \ \mathrm{d}x$$

Como argumentamos anteriormente, a solução será um conjunto de pontos geradores de uma TCV.

Vale notar que a escolha da norma que fizemos inicialmente pode ser problemática: a distância percorrida pode ser mal aproximada pela norma euclidiana se a região não for convexa ou se houver obstáculos grandes, como lagos ou montanhas.

1.3.3 Integração Numérica

Dados um conjunto aberto $\Omega \in \mathbb{R}^n$, uma função f em Ω, Lipschitz contínua com constante de Lipschitz L, e $m \in \mathbb{N}$, queremos determinar $Z \in \overline{\Omega}^m$ e uma tesselação Q de Ω que minimize o erro da aproximação

$$\int_{\Omega} f(x) \ \mathrm{d}x \approx \sum_{i=1}^{m} A_i f(z_i) \ ,$$

onde A_i é a área de Q_i. Então, queremos minimizar

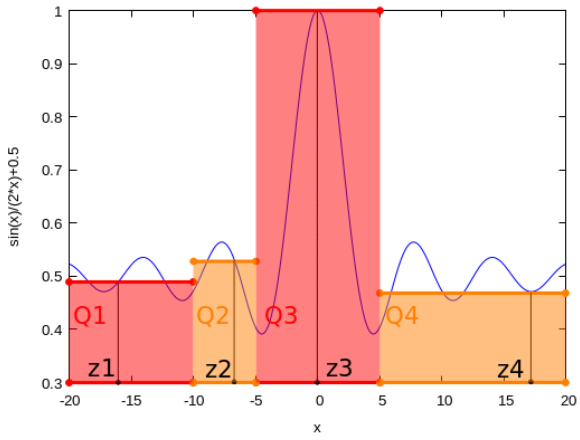
$$\mathcal{E}(Z,Q) = \left| \int_{\Omega} f(x) \ \mathrm{d}x - \sum_{i=1}^{m} A_i f(z_i) \right| = \left| \sum_{i=1}^{m} \left(\int_{Q_i} f(x) \ \mathrm{d}x - \int_{Q_i} 1 \ f(x) \ \mathrm{d}x \right) \right| =$$

$$\left| \sum_{i=1}^{m} \int_{Q_i} (f(x) - f(z_i)) \ \mathrm{d}x \right| \leq L \sum_{i=1}^{m} \int_{Q_i} |x - z_i| \ \mathrm{d}x$$

De forma análoga aos exemplos anteriores, o limitante superior acima é minimizado quando $V(Z)$ é uma TCV. Porém, neste caso, como não há um quadrado na expressão da norma, consideramos a centróide definida pela norma L_1.

É possível modificar o método de aproximação da integral para utilizar informação das primeiras $p-1$ derivadas, o que leva a TCVs com centróides definidas pela respectiva norma L_p [7].

Figura 1.4: Aproximação da integral definida de uma função, utilizando apenas 4 avaliações. Garantimos um valor ótimo de um certo limitante superior se escolhermos os valores $\{z_i\}_{i=1}^{m}$ de forma que $V(Z)$ seja uma TCV, com $Q_i = V_i(Z)$.



Capítulo 2

O Algoritmo de Fortune

2.1 Ideia Geral

No capítulo anterior, exibimos uma expressão simples que define um conjunto de Voronoi no $\mathbb{R}^2$:

$$V_i(Z) = \bigcap_{\substack{j=1 \\ j \neq i}}^{m} h(z_i, z_j) \tag{2.1}$$

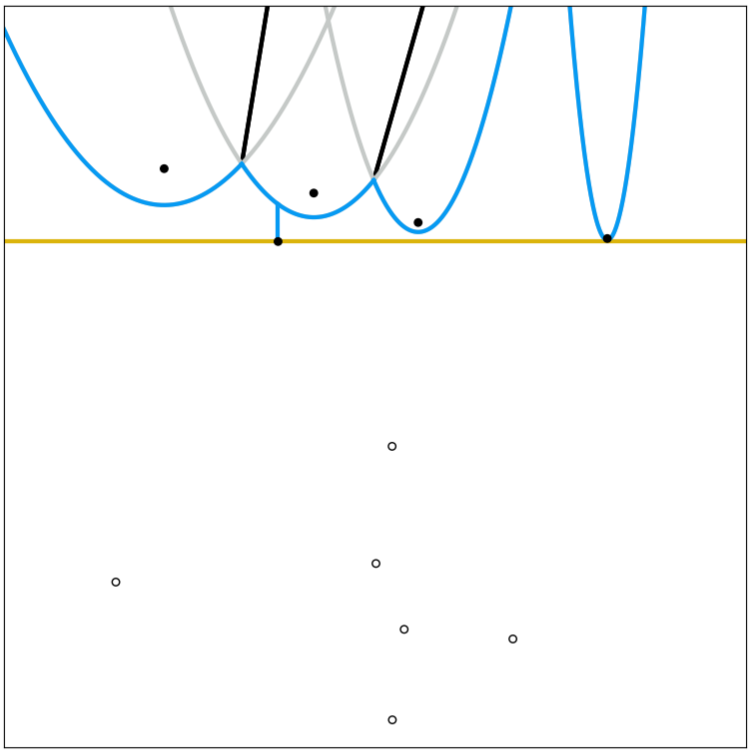
A expressão acima sugere um algoritmo natural para a construção de um diagrama de Voronoi: para cada ponto gerador z_i, calculamos iterativamente a intersecção entre os semiplanos $h(z_i, z_j)$. Apesar de simples e intuitivo, este algoritmo possui complexidade em tempo da ordem de $n^2 \log n$ [1], o que pode ser insatisfatório.

Neste capítulo, apresentaremos um algoritmo para a construção de diagramas de Voronoi no $\mathbb{R}^2$, publicado por Steven Fortune em 1986 [9], cuja complexidade é $\mathcal{O}(n \log n)$. Diversas fontes apresentam demonstrações de que não é possível atingir complexidade inferior a esta [17, 20, 5]. Nosso tratamento será fortemente baseado no livro [1].

O algoritmo de Fortune é um algoritmo de linha de varredura, ou seja, utiliza o artifício teórico de uma linha reta que se move pelo domínio, fazendo paradas em um número finito de pontos de interesse. Convencionaremos daqui em diante que a linha de varredura é horizontal, e se movimenta de cima pra baixo.

Para cada ponto gerador que está acima da linha de varredura, é determinada uma parábola: o conjunto dos pontos equidistantes entre a linha de varredura e o ponto. Chamamos a borda da intersecção dos conjuntos acima de tais parábolas de *linha da praia*. A linha da praia é o objeto matemático central do algoritmo de Fortune. Sua utilidade se dá principalmente a partir de duas propriedades. Em primeiro lugar, é fácil verificar que nenhum ponto acima da linha da praia pode estar mais próximo de um ponto gerador abaixo

Figura 2.1: Uma iteração do algoritmo de Fortune. Em amarelo, temos a linha de varredura. Em azul, a linha da praia. O ponto gerador que está sobre a linha de varredura define uma parábola degenerada, que corresponde a um segmento de reta. As arestas do diagrama estão em negrito, sendo construídas pelas intersecções entre as parábolas. Estas arestas são exibidas apenas para fins didáticos; no algoritmo real, as coordenadas só são calculadas quando um vértice é determinado.



da linha de varredura que de todos os pontos geradores acima da linha de varredura. Isso significa que, acima da linha da praia, o diagrama de Voronoi está determinado apenas pelos pontos geradores que já foram visitados pela linha de varredura. Em segundo lugar, as intersecções entre duas parábolas adjacentes equidistam dos dois pontos geradores relacionados. Por isso, essas intersecções desenham as arestas do diagrama de Voronoi.

O algoritmo de Fortune consiste, em linhas gerais, em calcular as mudanças da estrutura da linha da praia à medida que a linha de varredura avança. Quando um novo ponto é encontrado, uma nova parábola é adicionada à linha da praia, definindo uma nova aresta do diagrama. Chamaremos este caso de *evento ponto*. A outra situação de interesse é quando uma parábola passa a estar estritamente acima da linha da praia. Isso ocorre quando a linha da praia chega no centro de algum círculo vazio que contenha três ou mais pontos geradores em sua borda, definindo um vértice do diagrama. A este caso damos o nome de *evento círculo*.

Detalharemos o algoritmo mais adiante. Antes disso, é conveniente entendermos como os dados ficarão armazenados na memória.

2.2 Estruturas de Dados

Para que o algoritmo tenha a eficiência desejada, utiliza-se algumas estruturas de dados específicas. As arestas do diagrama são guardadas em uma estrutura tradicional em Geometria Computacional, chamada *Doubly-Connected Edge List*, ou DCEL. A linha da praia é representada por uma árvore binária balanceada, de forma que seja possível encontrar, em tempo $\mathcal{O}(\log n)$, abaixo de qual parábola está um ponto que acabou de ser encontrado pela linha de varredura. Finalmente, uma fila de prioridade é utilizada para ordenar os eventos ponto e eventos círculo que são encontrados à medida que a linha de varredura se move.

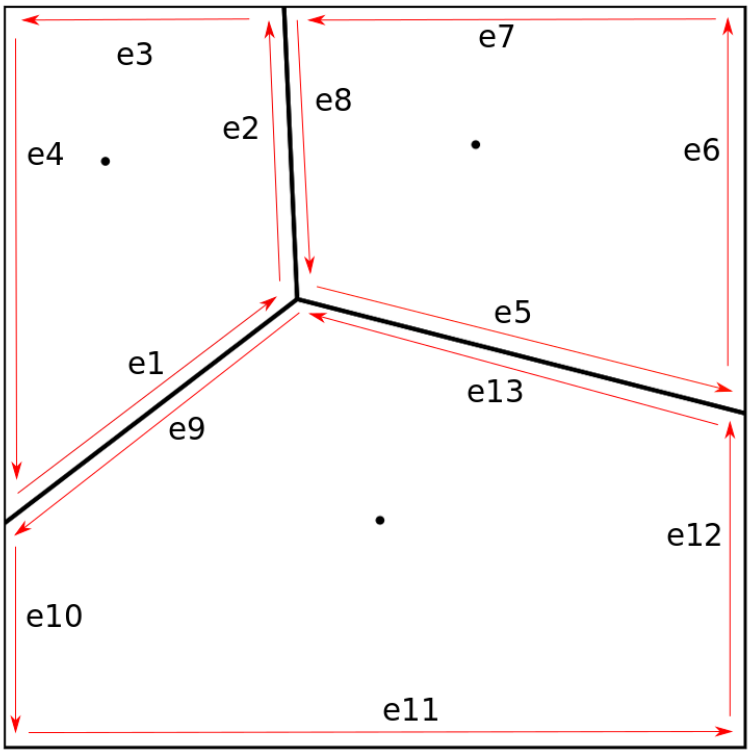
2.2.1 Doubly-Connected Edge List

Uma DCEL é uma representação conveniente de uma tesselação poligonal do plano, que consiste em diversas listas duplamente ligadas de objetos que chamamos de *meia aresta*. Cada lista percorre as arestas que definem uma região, na direção anti-horária. Como cada aresta é adjacente a duas regiões, temos duas meias arestas associadas a cada aresta. Duas meias arestas associadas à mesma aresta são chamadas *arestas gêmeas*. Como Diagramas de Voronoi são conexos e convexos, omitimos aqui alguns detalhes que seriam necessários para o caso mais geral.

2.2.2 Fila de Eventos

Para a fila de eventos, basta utilizarmos uma fila de prioridade que permita a remoção do elemento mais prioritário em $\mathcal{O}(\log n)$. Seria suficiente, por exemplo, um *heap*. A fila armazenará elementos de dois tipos diferentes, representando eventos pontos e eventos círculo. Para eventos ponto, serão guardadas as coordenadas do ponto a ser adicionado. Para eventos círculo, serão guardadas as coordenadas do ponto mais baixo do circuncírculo, pois quando a linha de varredura atravessa esse ponto, a distância entre a linha de

Figura 2.2: Representação de uma DCEL de um diagrama de Voronoi. Neste caso, os conjuntos $\{e_1, e_9\}$, $\{e_5, e_13\}$ e $\{e_2, e_8\}$ são pares de arestas gêmeas.



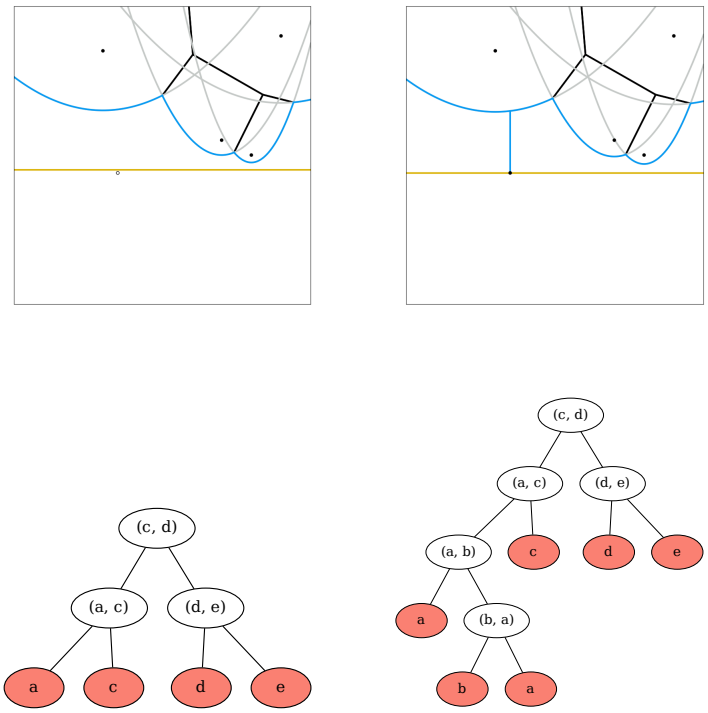
varredura e o centro é igual à distância entre o centro e três pontos geradores de parábolas adjacentes, definindo assim um vértice do diagrama. O objetivo da fila de eventos é obter os pontos na ordem em que estes são encontrados pela linha de varredura. Por isso, o critério de ordenação será a coordenada y de cada ponto. No caso de empate, qualquer escolha de ordenação constrói o diagrama corretamente.

2.2.3 Linha da Praia

Para a linha da praia, utilizaremos uma árvore binária balanceada, ordenada pela coordenada x dos pontos, com uma estrutura especial: teremos os

dados referentes a cada parábola nas folhas da árvore, enquanto os outros nós representarão *pontos de quebra*. Por exemplo, se a linha de varredura atravessou apenas dois pontos, a árvore terá três nós: um para cada ponto e um representando o ponto onde as duas parábolas se intersectam. Quando um novo ponto for encontrado, teremos um novo ponto de quebra, que terá como sub-árvores filhas a árvore anterior e o novo ponto.

Figura 2.3: Inserção de um novo elemento na árvore, ocorrendo num evento ponto.



Para os pontos de quebra, são armazenadas as coordenadas dos focos que definem as parábolas adjacentes. Em relação à estrutura de dados, essas parábolas correspondem ao nó mais à direita da sub-árvore esquerda e ao nó mais à esquerda da sub-árvore direita. Não seria suficiente calcular uma única coordenada x do ponto de quebra, pois esse valor varia de acordo com a posição da linha de varredura.

Quando um evento círculo for avaliado, uma parábola será removida da linha da praia. O processo é semelhante à inserção, revertida. Porém, há mais um detalhe: em alguns casos, é necessário corrigir os valores armazenados nos pontos de quebra acima do nó removido. Isso pode ser feito, por exemplo, com uma construção recursiva ou com ponteiros adicionais,

sem nenhuma penalidade à ordem de complexidade da remoção da árvore binária.

Finalmente, é fácil notar que rotações não alteram as propriedades até aqui discutidas. Por isso, podemos manter a árvore balanceada (utilizando, por exemplo, uma estrutura *red-black* ou *AVL*), de forma a garantir altura de ordem logarítmica no número de pontos geradores.

2.3 Descrição do Algoritmo

Podemos, finalmente, listar o algoritmo, de forma simplificada:

Algoritmo 1 Algoritmo de Fortune

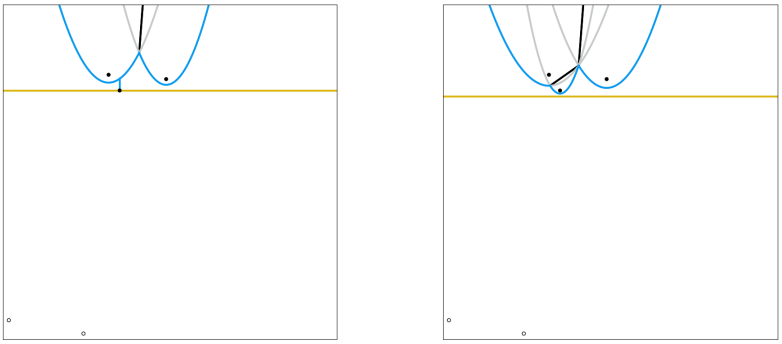
```
função FORTUNE(Z)
    V ← DCEL vazia
    Q ← Fila de eventos vazia
    T ← Árvore binária balanceada vazia

    para cada z_i ∈ Z faça
        Insira z_i como evento ponto em Q
    fim para

    enquanto Q não está vazia faça
        Remova o evento q_i mais prioritário de Q
        se q_i for um evento ponto então
            EVENTOPONTO(V, T, Q, q_i)
        senão
            EVENTOCÍRCULO(V, T, Q, q_i)
        fim se
    fim enquanto

    devolva V
fim função
```

Por brevidade, não listaremos o tratamento dos eventos ponto e círculo. Em vez disso, trazemos uma descrição informal de cada procedimento.

O tratamento de um evento ponto consiste, basicamente, de três tarefas: atualizar a linha da praia com a nova parábola, atualizar o diagrama com a nova aresta, entre os pontos de quebra que foram criados, e procurar por possíveis eventos círculo, que podem ter aparecido quando o novo ponto foi atravessado pela linha de varredura.

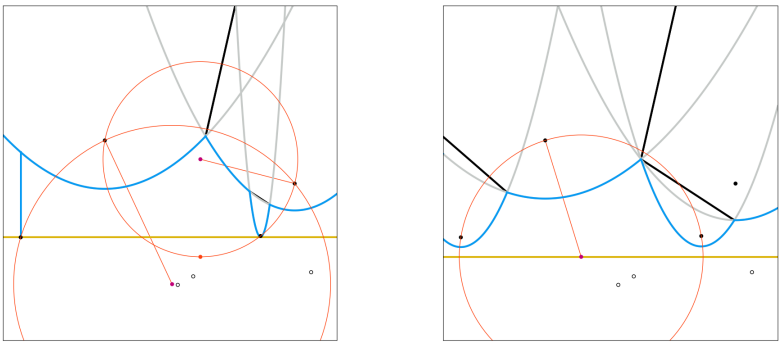
O tratamento de um evento círculo tem estrutura similar: é necessário atualizar a linha da praia removendo a parábola que foi ultrapassada, atualizar o diagrama com o novo vértice que foi definido, no centro da circun-

Figura 2.4: Evento ponto. As arestas do diagrama são criadas com pontas nos pontos de quebra entre a nova parábola e a parábola adjacente, que está logo acima do ponto visitado.



ferência que contém os três pontos que definem o evento círculo, e procurar por possíveis novos eventos círculo, que podem ter aparecido porque alguns arcos de parábola passaram a ser adjacentes com a remoção de um arco da linha da praia.

Figura 2.5: Evento círculo. O ponto armazenado na fila de eventos, em vermelho, é o ponto mais baixo do círculo que contém na borda os pontos geradores das regiões que se encontram em seu centro (exibido em magenta), onde haverá um vértice do diagrama.



Vimos anteriormente que as arestas são definidas pelos pontos de quebra entre os arcos de parábola da linha da praia. Por isso, o lema seguinte garante que o algoritmo encontra todas as arestas do diagrama:

Lema 2.1. *Novos arcos de parábola podem aparecer na linha da praia apenas através de eventos ponto.*

Demonstração. [1], página 153. □

Analogamente, os dois lemas a seguir garantem que o algoritmo encontra todos os vértices do diagrama:

Lema 2.2. *Arcos de parábola podem ser removidos da linha da praia apenas através de eventos círculo.*

Demonstração. [1], página 155. □

Lema 2.3. *Todo vértice do diagrama de Voronoi é detectado por um evento círculo.*

Demonstração. [1], página 157. □

2.4 Intersecção com o Domínio

A construção de TCVs, que faremos no capítulo seguinte, depende do cálculo do centro de massa, que é muito mais simples quando o domínio é um conjunto limitado. Por este motivo, escolheremos Ω sendo uma caixa no plano: $\Omega_\square = \{(x, y) \mid 0 \leq x \leq W \ e \ 0 \ \leq y \leq H\}$, com W e H constantes pré-determinadas.

O algoritmo até aqui constrói um diagrama de Voronoi no $\mathbb{R}^2$. Para adaptarmos o algoritmo de Fortune para $\Omega_\square$, será necessário calcular a intersecção da DCEL resultante do algoritmo de Fortune aplicado ao plano com a caixa $\Omega_\square$. Isto pode ser feito em tempo linear no número de pontos. Para isto, operamos como descrito a seguir, para cada região R.

Primeiro, encontramos, se existe, um segmento que intersecta a borda da caixa. Em seguida, acompanhamos, a partir do ponto de intersecção, os vértices da caixa em sentido anti-horário, até que encontremos mais um segmento de R, eliminando assim uma seção ilimitada da região.

Continuamos percorrendo as meia-arestas até encontrarmos, se houver, outra que intersecta a borda da caixa e repetimos o corte da região externa à caixa. Iteramos até que encontremos novamente o segmento inicial. Ao fim do processo, teremos um polígono limitado e contido na caixa, como desejávamos.

Há alguns casos especiais: por exemplo, um segmento pode cortar a borda da caixa em mais de um ponto. Porém, estes casos são de fácil tratamento e serão omitidos, por brevidade.

Capítulo 3

Construção de Tesselações Centroidais de Voronoi

3.1 O Algoritmo de Lloyd

Uma ideia natural para calcular uma Tesselação Centroidal de Voronoi é: começando com um Diagrama de Voronoi qualquer, substituimos cada ponto gerador pela centróide da respectiva região de Voronoi. Isto define um algoritmo iterativo; esperamos que a cada novo conjunto de pontos geradores, estes estejam mais próximos das centróides correspondentes.

Este é o procedimento que define o *algoritmo de Lloyd*, criado por Stuart Lloyd, em 1957, na Bell Labs.

Definição 3.1 (Mapeamento de Lloyd)**.** *Dados um conjunto aberto $\Omega \subseteq \mathbb{R}^n$ e um conjunto de pontos não coincidentes $Z = \{z_i\}_{i=1}^m$ em $\overline{\Omega}$, chamamos de* mapeamento de Lloyd *a função $T(Z) = \{C_i(Z)\}_{i=1}^m$, ou seja, a função que leva do conjunto de pontos geradores no conjunto de centróides das regiões de Voronoi por eles determinadas.*

Claramente, um conjunto Z é ponto fixo de T se e somente se $V(Z)$ é uma Tesselação Centroidal de Voronoi. O algoritmo de Lloyd consiste em escolher um conjunto Z inicial qualquer e calcular a sequência $\{T^k(Z)\}$ até que $T^k(Z)$ seja suficientemente próximo de $T^{k-1}(Z)$, para algum critério de proximidade relevante à aplicação em questão.

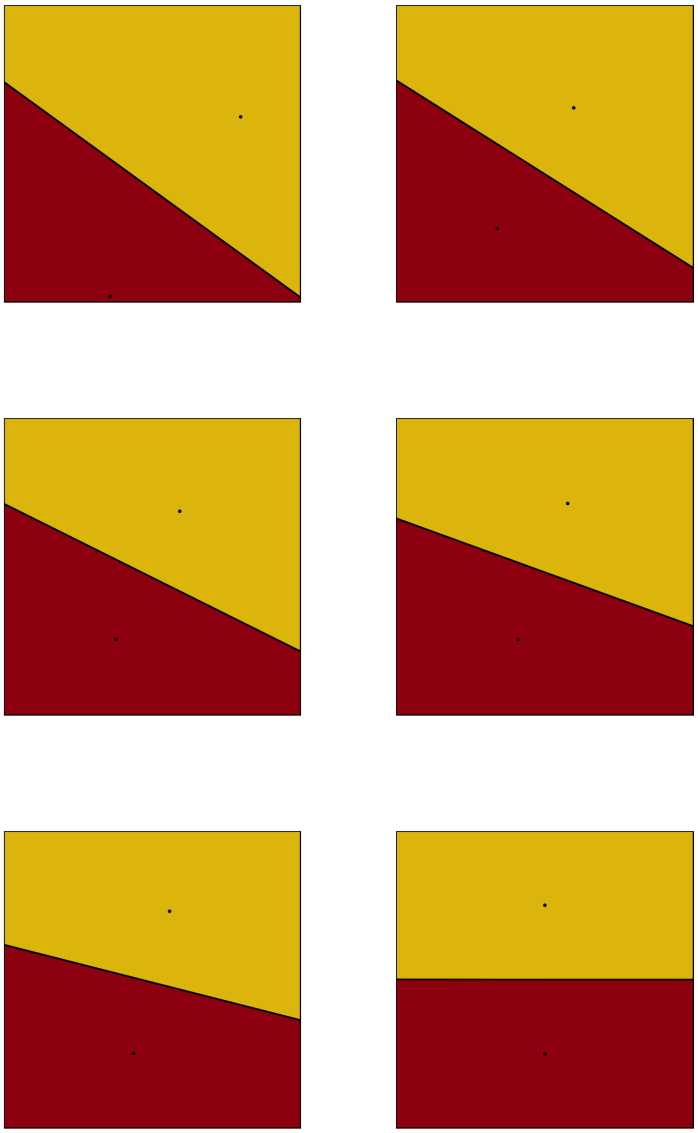
Podemos definir exatamente o mesmo procedimento quando o domínio Ω for discreto. Neste contexto, o algoritmo de Lloyd é mais frequentemente conhecido como *k-means clustering* [7].

A convergência do algoritmo de Lloyd é conhecida para alguns casos particulares. Destacamos os seguintes resultados, retirados de [6]:

Teorema 3.1 (Convergência global)**.** *Se T tem um único ponto fixo, o algoritmo de Lloyd converge globalmente.*

Teorema 3.2 (Convergência global em $\mathbb{R}$)**.** *Em uma dimensão, o algoritmo de Lloyd converge globalmente, para qualquer função de densidade positiva e suave.*

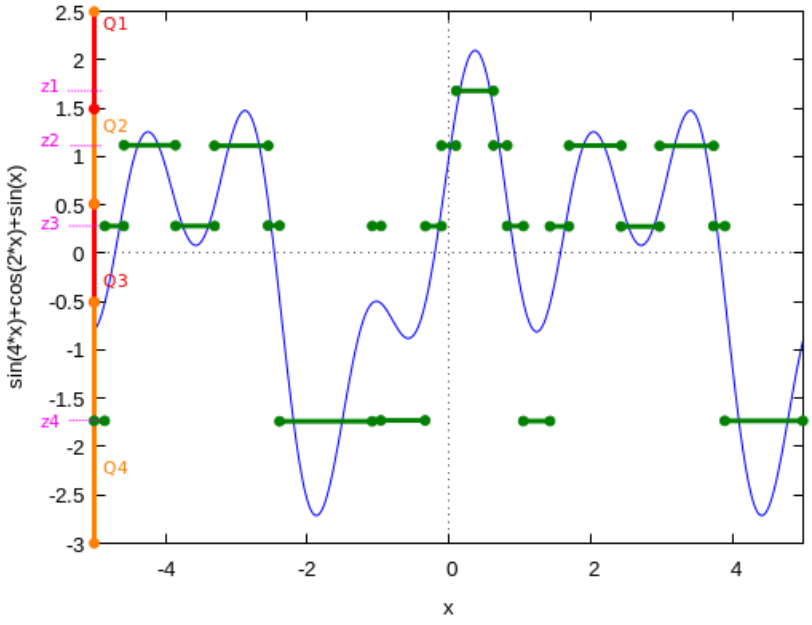
Figura 3.1: Iterações 1, 2, 3, 4, 5 e 20 do algoritmo de Lloyd em duas dimensões, com 2 pontos geradores.



3.2 Quantização Ótima em Uma Dimensão

O algoritmo de Lloyd foi criado para a quantização ótima de sinais analógicos, isto é, para escolher um conjunto finito de valores que representam um domínio contínuo de valores possíveis, de forma a minimizar o erro esperado de um sinal aleatório em relação à sua versão *quantizada*, ou seja, a versão que contém apenas o representante de cada valor. É importante notar que aqui nos referimos apenas à quantização do domínio de valores, não do sinal no tempo.

Figura 3.2: Sinal analógico (em azul) e versão quantizada (em verde). Cada conjunto Q_i tem seu valor representante z_i. Note que estamos ignorando a quantização no tempo (coordenada x), que será feita através do Teorema de Shannon-Nyquist.



Vamos analisar brevemente o problema da quantização ótima em uma dimensão, seguindo sua formulação no trabalho original de Lloyd [13]. Além de ser uma aplicação importante de TCVs, veremos adiante que este problema leva naturalmente à formulação da construção de TCVs como um problema de otimização.

O teorema da amostragem de Shannon-Nyquist [18] afirma que um sinal $s\colon \mathbb{R} \to \mathbb{R}$ que possua apenas componentes de frequência menor que W pode ser recuperado perfeitamente a partir de um conjunto de amostras $s(t_j)$, onde $t_j = \frac{j}{2W}$, $j \in \mathbb{Z}$, através da fórmula:

$$s(t) = \sum_j s(t_j) K(t - t_j)$$

com $K(t) = \frac{\sin 2\pi W t}{2\pi W t}$.

Assumimos, a partir de agora, que temos um sinal que pode ser representado pela frequência de amostragem utilizada.

Considere agora uma tesselação finita $Q = \{Q_i\}_{i=1}^m$ de intervalos em $\mathbb{R}$ e um conjunto de representantes $z = \{z_i\}_{i=1}^m$. Seja γ a função que associa cada $x \in \mathbb{R}$ ao índice i do conjunto $Q_i \ni x$.

Na modulação por código de pulsos (do inglês *pulse-code modulation*, ou *PCM*), recuperamos o sinal utilizando os representantes de cada classe, transmitindo apenas $\gamma(s(t_j))$, $j \in \mathbb{Z}$:

$$r(t) = \sum_j z_{\gamma(s(t_j))} K(t - t_j)$$

Definindo o sinal de ruído $n(t) = s(t) - r(t)$, estamos interessados em escolher Q e Z de forma a minimizar a *potência do ruído*, definida como a esperança de n^2. Ou seja, queremos minimizar

$$N(Z, Q) = \sum_{i=1}^{m} \int_{Q_i} n(t)^2 \, \rho(t) \, \mathrm{d}t \ ,$$

onde a esperança é tomada em relação ao processo estocástico estacionário que gerou o sinal $s(t)$, ou seja, ρ é a função de densidade de probabilidade de cada valor em um instante aleatório do sinal.

A quantia $E[n^2]$, restrita a cada Q_i, é o *segundo momento* em torno de z_i. Fixando Q_i, é um resultado clássico da Teoria de Probabilidades que o segundo momento é minimizado quando é tomado em torno da média, ou seja, quando z_i é o centro de massa de Q_i [2]. Neste caso, N corresponde a *variância* de n.

Por outro lado, é fácil ver que, fixados os pontos $\{z_i\}_{i=1}^m$, os conjuntos Q_i que minimizam a potência do ruído são aqueles com limites nos pontos médios entre os representantes [13], ou seja, os conjuntos dados por:

$$Q_i = \{x \in \mathbb{R} \mid |x - z_i| < |x - z_j|, \forall j \neq i\}$$

Assim, temos duas condições necessárias para a otimalidade de uma determinada quantização. Condições suficientes foram encontradas por Fleischer em 1964 [8].

3.3 Construção de TCVs como um Problema de Otimização

Podemos nos inspirar na discussão anterior para modelar a construção de TCVs como a minimização de uma função equivalente à potência do ruído,

seguindo [7]:

Definição 3.2 (Função objetivo)**.** *Dados um conjunto aberto $\Omega \subseteq \mathbb{R}^n$, uma distribuição de probabilidade ρ em Ω, uma tesselação $Q = \{Q_i\}_{i=1}^m$ de Ω e um conjunto de pontos não coincidentes $Z = \{z_i\}_{i=1}^m$ em $\overline{\Omega}$, definimos:*

$$\mathcal{H}(Z, Q) = \sum_{i=1}^{m} \int_{Q_i} \|x - z_i\|^2 \, \rho(x) \, \mathrm{d}x$$

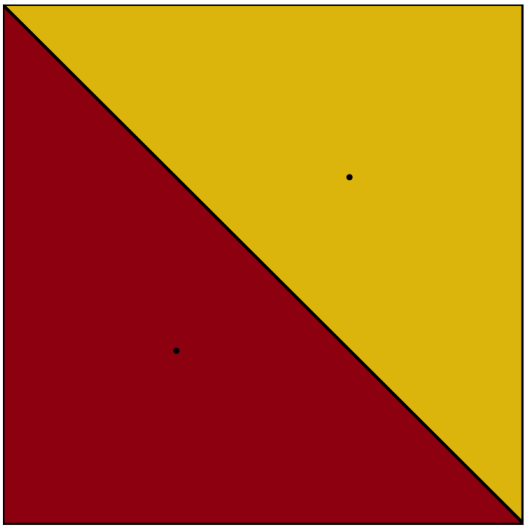
e

$$\mathcal{G}(Z) = \mathcal{H}(Z, C(Z))$$

Analogamente ao que fizemos na seção anterior, é possível mostrar que Z é ponto estacionário de $\mathcal{G}(Z)$ somente se $Z = T(Z)$, ou seja, se $V(Z)$ é uma Tesselação Centroidal de Voronoi.

Assim, reduzimos o problema de construir uma TCV, com m pontos geradores, em n dimensões, ao problema de minimizar $\mathcal{G}(Z)$ com $Z \in \mathbb{R}^{mn}$. Note que isto não garante otimalidade da função objetivo; é fácil encontrar pontos de sela em $\mathcal{G}$. No entanto, todo ponto estacionário de $\mathcal{G}$ corresponde a uma TCV.

Figura 3.3: Um ponto de sela de $\mathcal{G}$. Qualquer translação dos pontos aumenta a função objetivo. Uma rotação em torno do centro da figura diminui a função objetivo. Exemplo retirado de [7].



Para utilizar os algoritmos clássicos de Otimização Não-Linear, precisamos calcular o gradiente de $\mathcal{G}$.

Seja $M(Q)$ a matriz diagonal com $M(Q)_{ii} = \int_{Q_i} \rho(x)\,\mathrm{d}x$, ou seja, a matriz determinada pelas massas totais de probabilidade de cada região da tesselação. Temos então:

Teorema 3.3 (Gradiente de $\mathcal{G}$)**.** $\nabla\mathcal{G}(Z) = 2M(V(Z))(Z - T(Z))$

Demonstração. [7], página 659. □

Podemos agora utilizar o *método do gradiente*, um dos algoritmos mais simples da Otimização Não-Linear, para encontrar pontos estacionários de $\mathcal{G}$. Este método consiste em, a partir de um ponto Z^k, fixar a direção $d = -\nabla\mathcal{G}(Z^k)$ e procurar $\lambda > 0$ tal que, para $Z^{k+1} = Z^k + \lambda d$,

$$\mathcal{G}(Z^{k+1}) < \mathcal{G}(Z^k) - \alpha\lambda d^t d\ , \tag{3.1}$$

para algum $\alpha \in (0, 1)$. É garantido que exista tal λ, que pode ser encontrado por um processo chamado *backtracking*: começamos com um λ^0 arbitrário e fazemos $\lambda^{i+1} = \lambda^i * \mu$, com $\mu \in (0, 1)$, até que, para algum i, a Equação 3.1 seja satisfeita com $\lambda = \lambda_i$.

Uma descrição mais completa do método do gradiente pode ser encontrada em livros clássicos de Otimização Não-Linear [10, 15].

O método do gradiente, assim como o algoritmo de Lloyd, possui ordem de convergência linear. [12] apresenta a aplicação de algoritmos mais avançados ao cálculo de TCVs, como o *método de Newton* ou o *algoritmo L-BFGS*, que possuem ordem de convergência quadrática e superlinear, respectivamente.

Capítulo 4

Testes

Implementamos, na linguagem Julia (versão 1.0.2), o algoritmo de Fortune, o algoritmo de Lloyd e o método do gradiente para o cálculo de Tesselações Centroidais de Voronoi. Todo o código está disponível online.

Todos os testes foram realizados em ambiente Linux, utilizando um processador Intel Core i5-7600, 3.8GHz, com 16GB de memória RAM disponíveis.

Para a definição das centróides, utilizamos sempre a norma L_2 e a função de densidade constante. A utilização de normas diferentes implicaria uma modificação total do algoritmo de Fortune, pois os elemento constituintes da linha da praia não seriam mais parábolas. Já a utilização de outra função de densidade exigiria poucas modificações, visto que esta é usada apenas para o cálculo das centróides. Alguma dificuldade poderia ocorrer na avaliação da função objetivo, que é calculada através de uma integral envolvendo a densidade.

4.1 Construção de Diagramas de Voronoi

Implementamos o algoritmo de Fortune para a construção de Diagramas de Voronoi no plano. Uma rotina separada foi desenvolvida para determinar a intersecção do diagrama com uma caixa, como descrito na Seção 2.4.

Nossa implementação possui algumas limitações conhecidas. Primeiramente, alguns casos especiais não foram considerados; por exemplo, caso os primeiros dois pontos tenham a mesma coordenada y, é necessário um tratamento especial, que não foi feito. Além disso, não implementamos o balanceamento da árvore. Como esperado, não observamos piora na complexidade, já que o tempo de busca em uma árvore não balanceada continua sendo $\mathcal{O}(\log n)$ para uma entrada aleatória.

Figura 4.1: Evolução do algoritmo de Fortune com 10 pontos geradores.

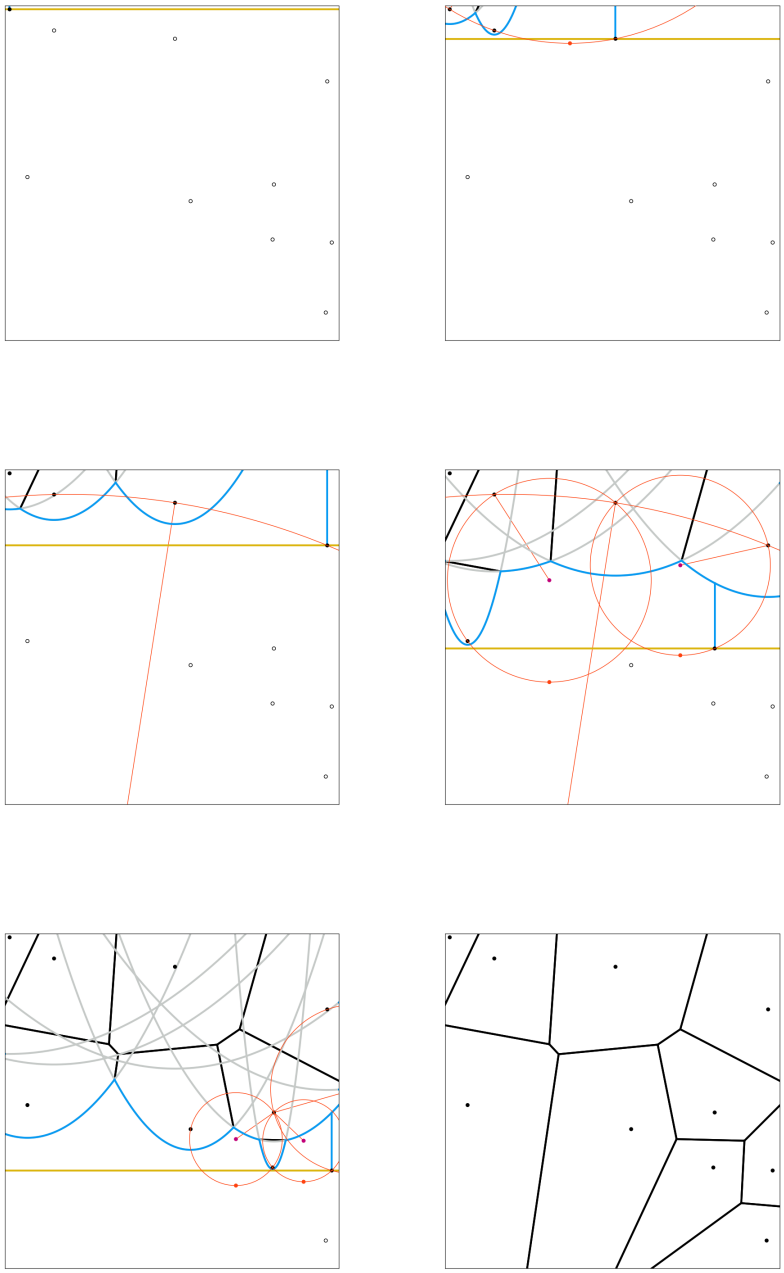


Figura 4.2: Evolução do algoritmo de Fortune com 50 pontos geradores.

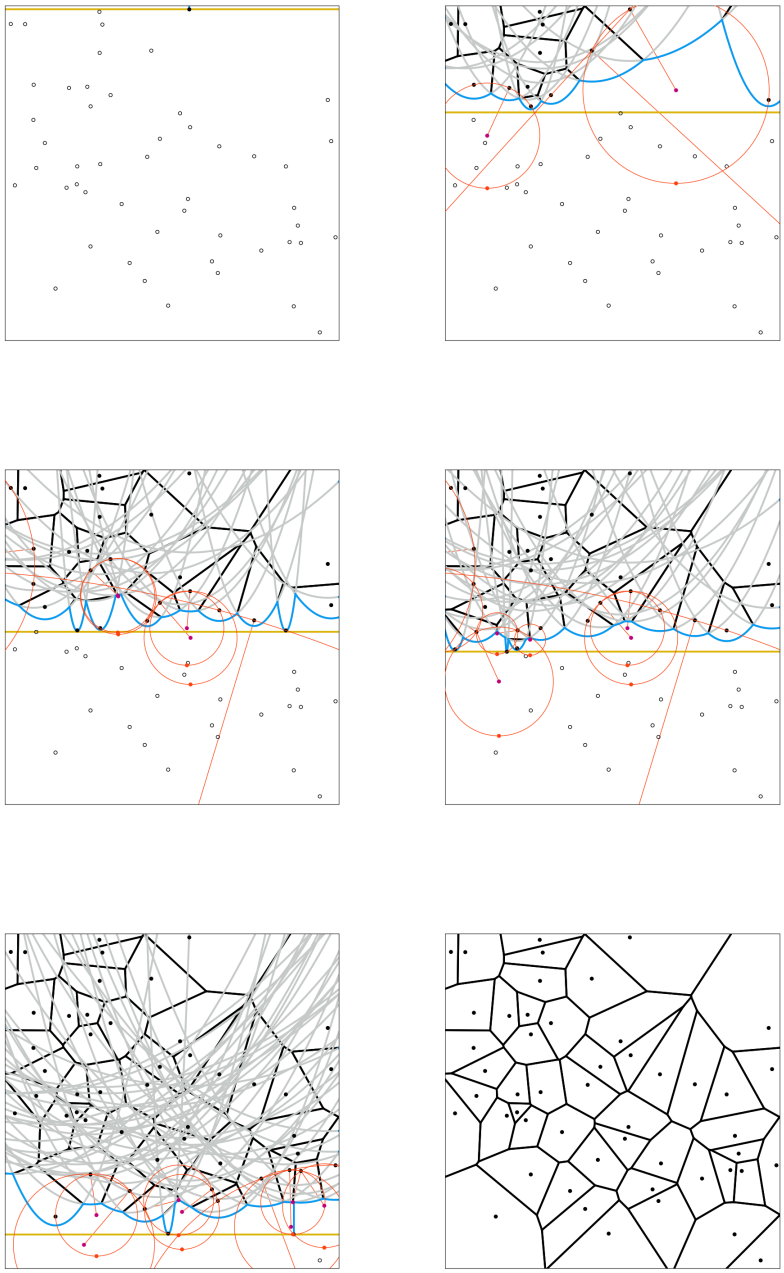
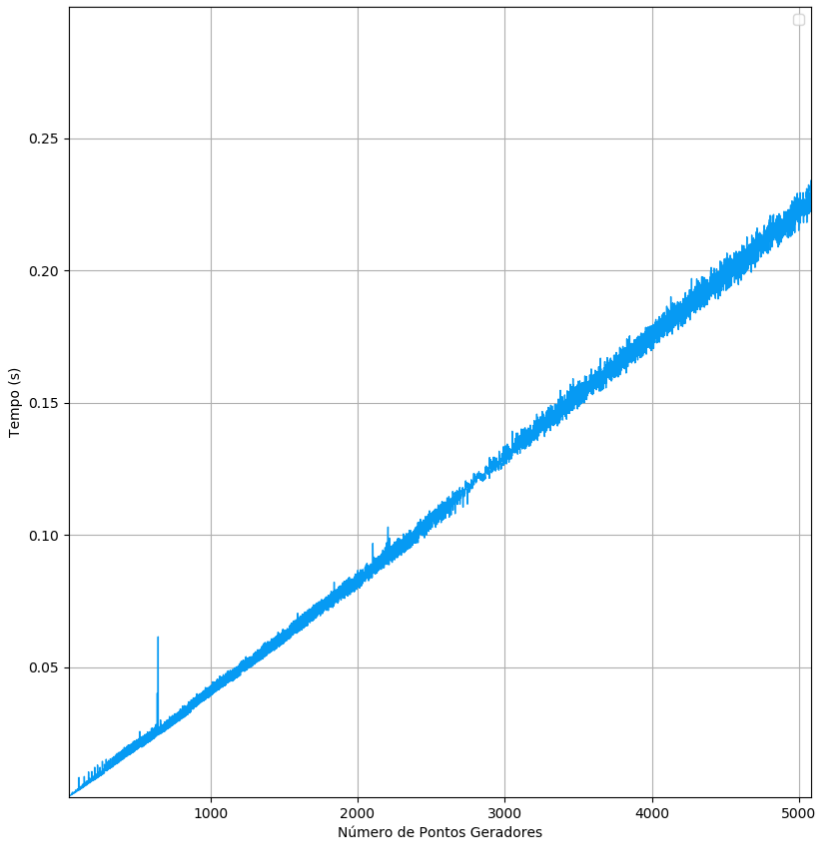


Figura 4.3: Tempo de Execução do algoritmo de Fortune



4.2 Cálculo de Tesselações Centroidais de Voronoi

Implementamos o algoritmo de Lloyd e o método do gradiente para o cálculo de TCVs em uma caixa do $\mathbb{R}^2$.

Observamos convergência linear em ambos, com o algoritmo de Lloyd de forma geral convergindo mais rapidamente. Porém, nossa implementação do método do gradiente foi bastante simplificada e consideramos que a utilização de técnicas tradicionais como interpolação quadrática poderiam trazer um grande ganho de desempenho.

Para avaliar o desempenho dos métodos, foi necessário computar explicitamente o valor da função objetivo a cada iteração. Para isso, encontramos

uma solução analítica para o seu valor em cada região de Voronoi. Primeiramente, notamos que qualquer região, sendo um polígono convexo, pode ser divida em triângulos formados pelo ponto gerador e dois pontos de cada aresta. Assim, bastou, dados pontos z_i, a e b, encontrar o valor da função objetivo no triângulo formado por eles. Ou seja, queremos calcular:

$$\mathcal{G}(z_i,\ a,\ b)_{\triangle} = \int_{\triangle} f(x,y)\ \mathrm{d}x\mathrm{d}y,$$

com $f(x,y) = x^2 + y^2$. Definindo $p = (a - z)$ e $q = (b - z)$ e utilizando a mudança de variáveis

$$[x\ y]^t = T([u\ v]^t) = up + vq,$$

temos que:

$$\mathcal{G}'(p,\ q)_{\triangle} = \int_0^1 \int_0^{1-v} f(x(u,v), y(u,v))\ |Jt|\ \mathrm{d}u\mathrm{d}v =$$

$$\frac{1}{12}\ |p_1q_2 - p_2q_1|\ p_1^2 + q_1^2 + p_2^2 + q_2^2 + p_1q_1 + p_2q_2\ .$$

Figura 4.4: Algumas iterações do algoritmo de Lloyd com 5 pontos geradores.

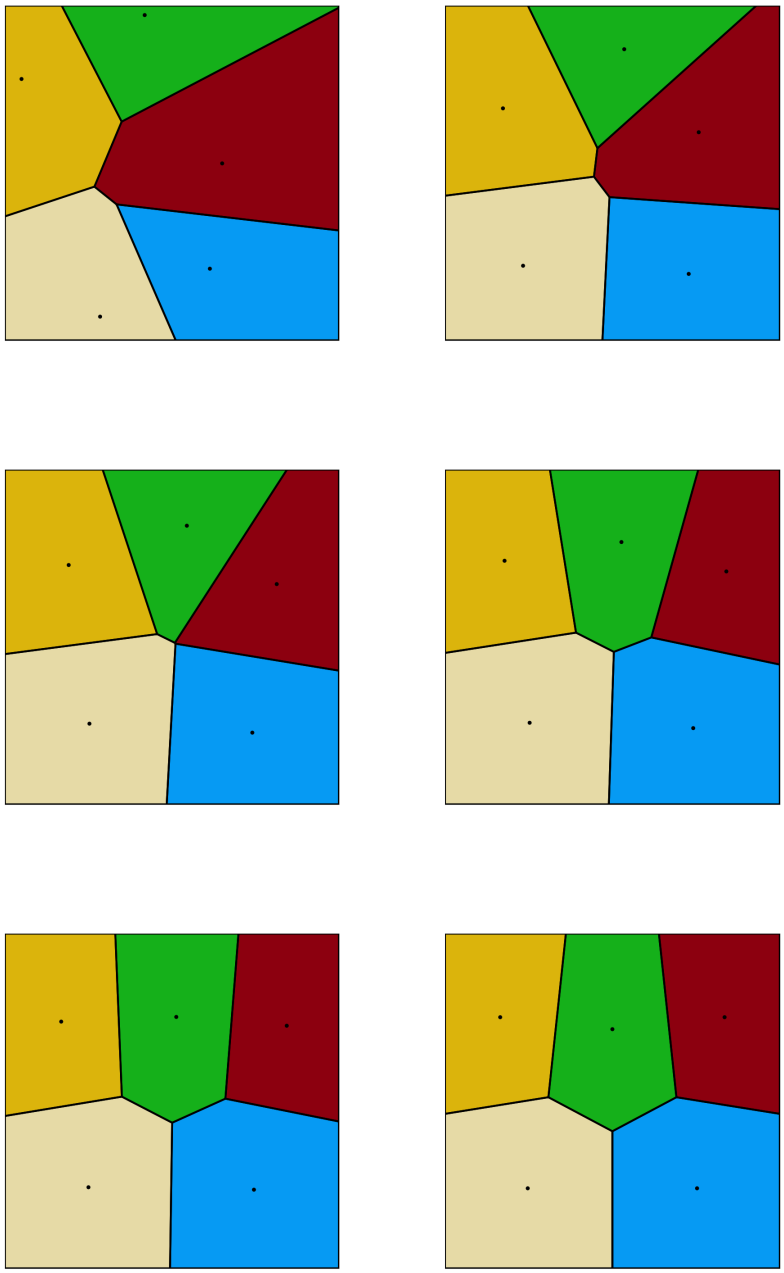


Figura 4.5: Algumas iterações do algoritmo de Lloyd com 8 pontos geradores.

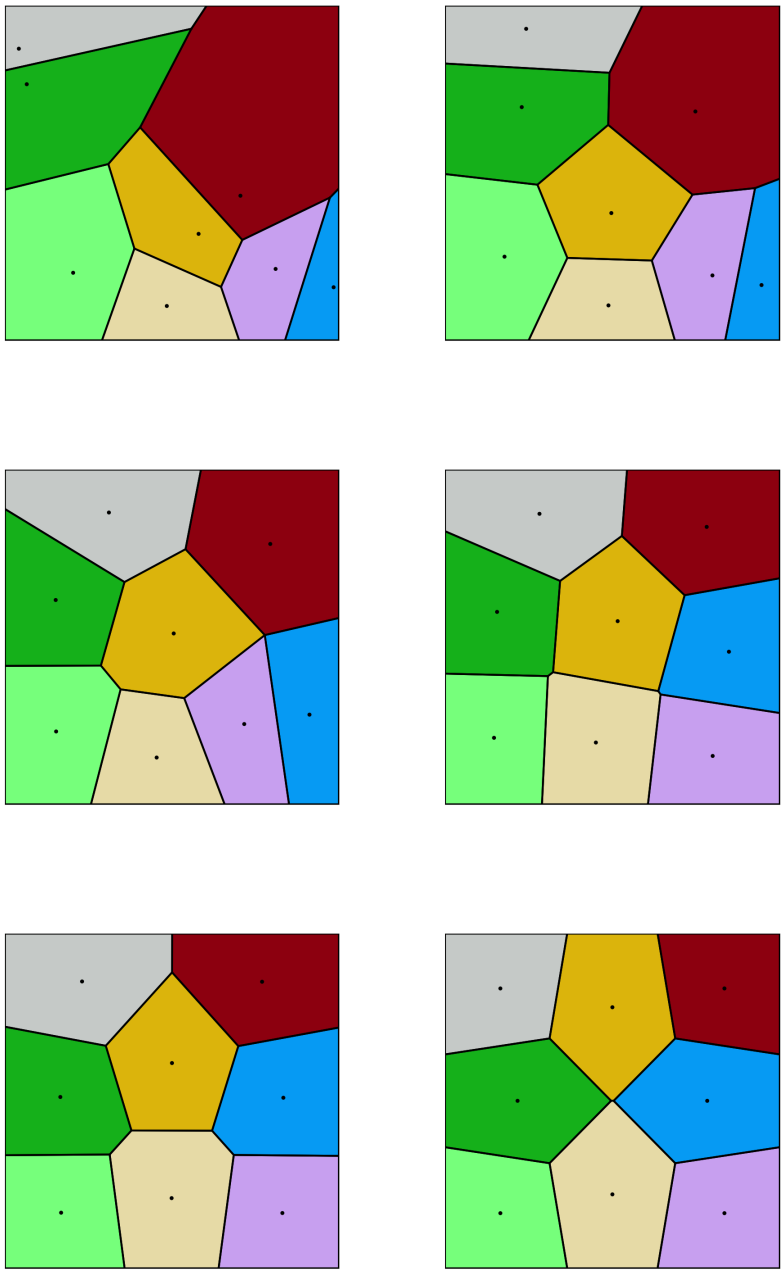


Figura 4.6: TCVs com 3, 5, 6, 7, 8 e 12 pontos geradores, obtidas com o método do gradiente.

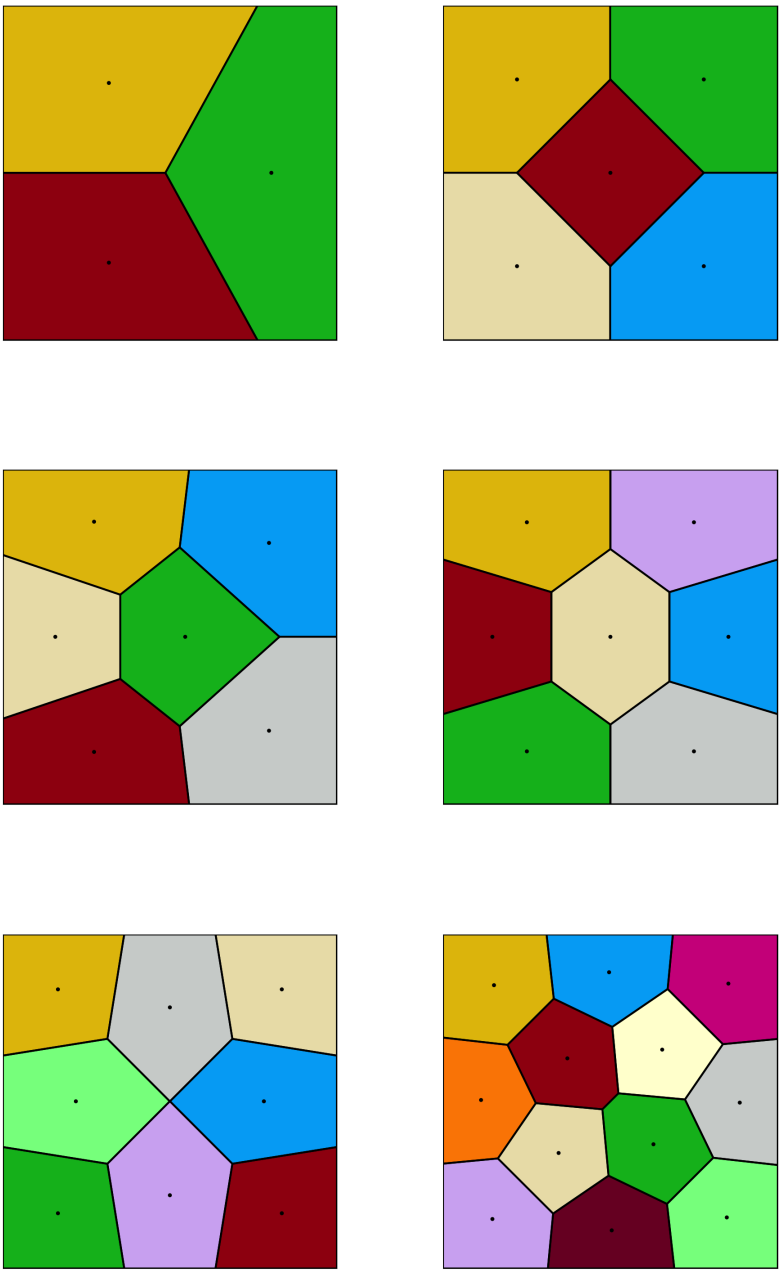


Figura 4.7: Três TCVs diferentes com 17 pontos geradores. De forma geral, a função objetivo não tem um único ponto estacionário.

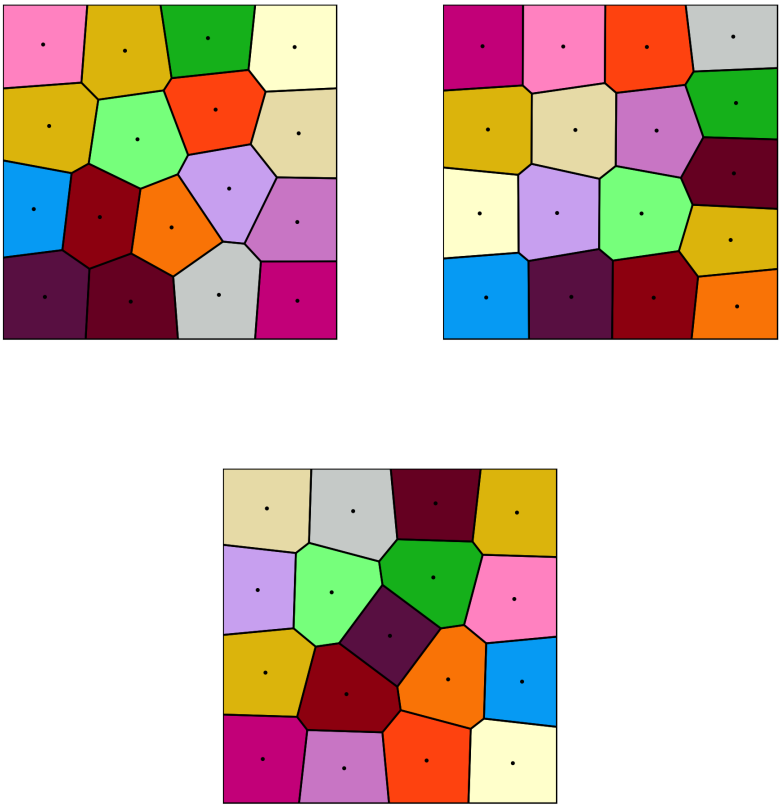


Figura 4.8: TCVs para 50, 100, 200, 300, 500 e 1000 pontos geradores. Quando o número de pontos geradores cresce, as células do interior da tesselação tendem a hexágonos regulares.

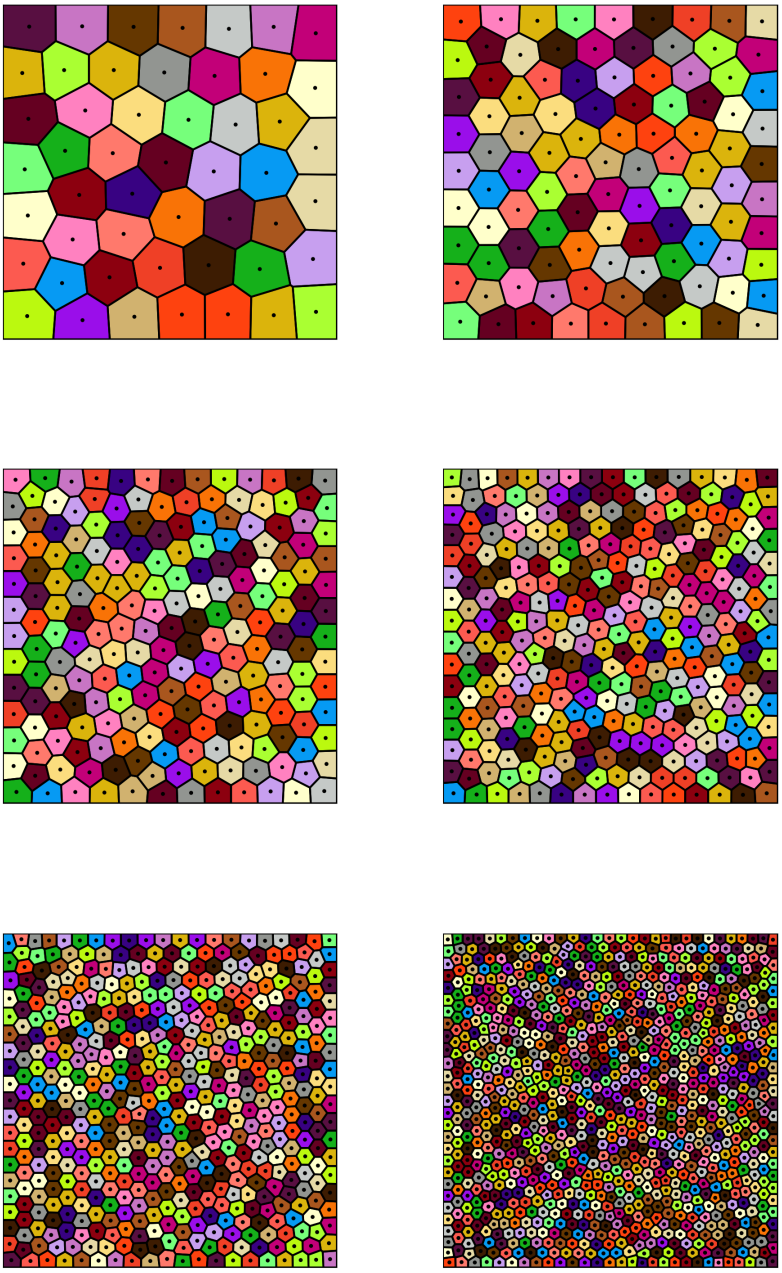
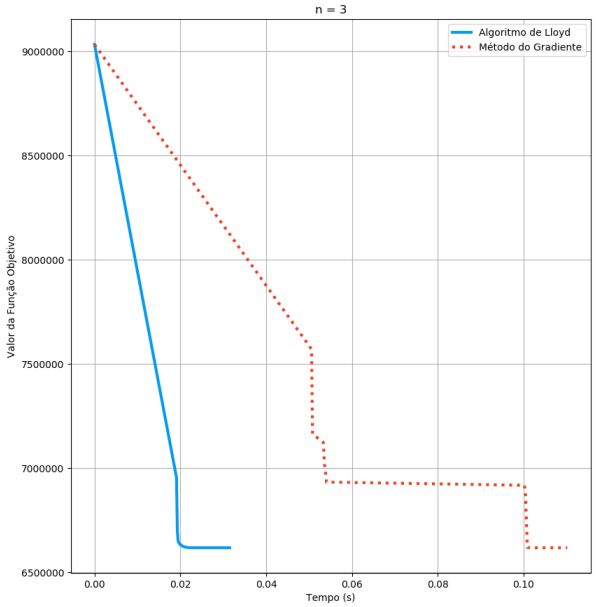


Figura 4.9: Comparação entre o algoritmo de Lloyd e o método do gradiente, para 3 pontos geradores. Intervalos maiores de tempo entre as iterações acontecem com frequência no método do gradiente, causados pelo uso de *backtracking* na busca linear.



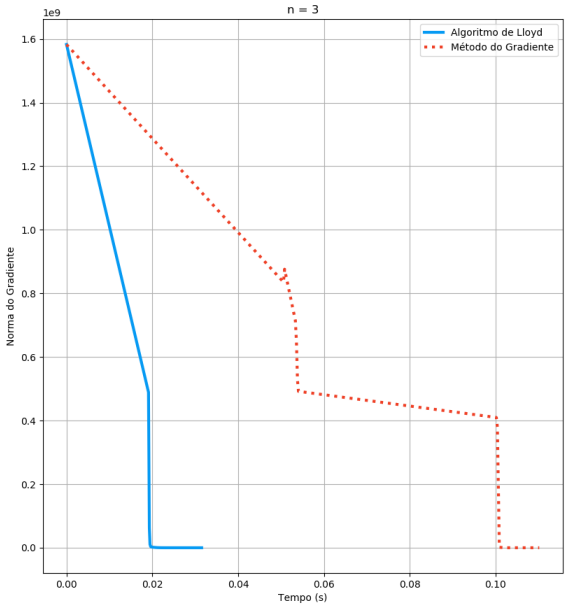


Figura 4.10: Comparação entre o algoritmo de Lloyd e o método do gradiente, para 30 pontos geradores.

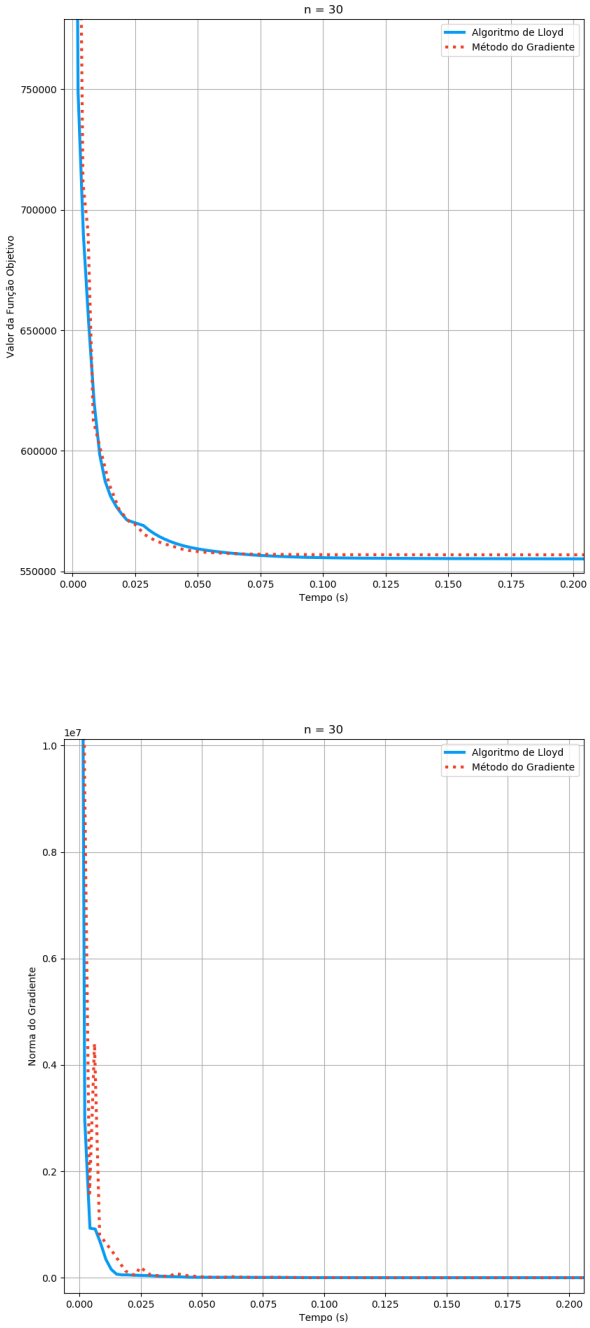


Figura 4.11: Comparação entre o algoritmo de Lloyd e o método do gradiente, para 100 pontos geradores.

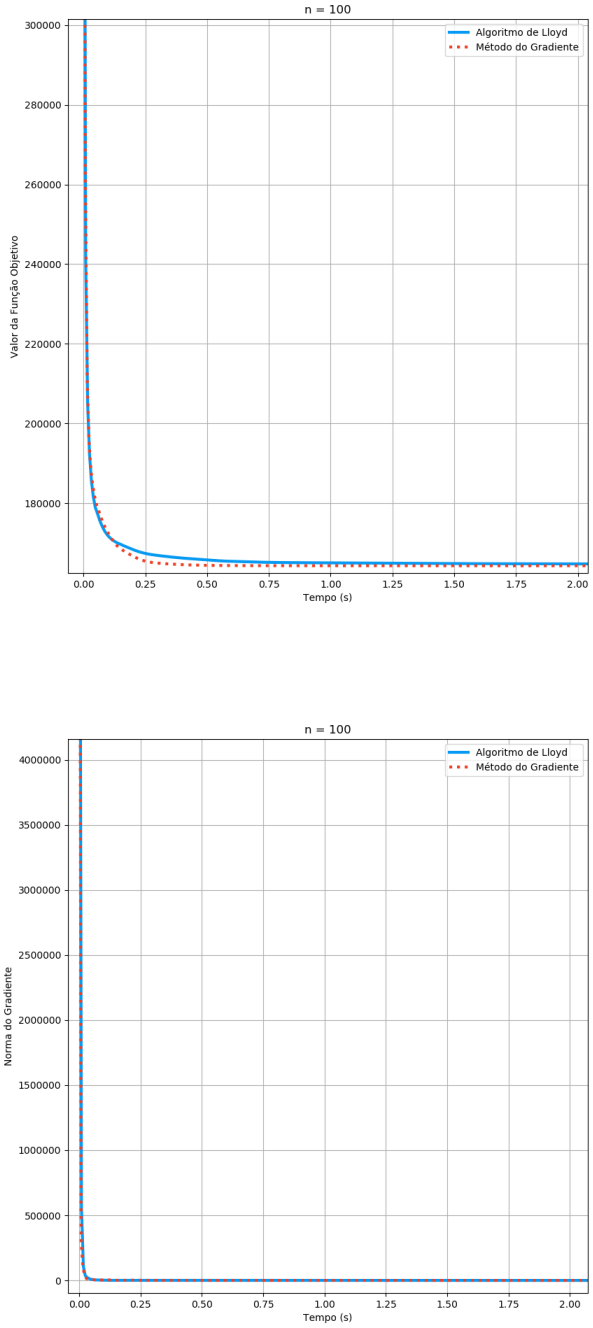


Figura 4.12: Comparação entre o algoritmo de Lloyd e o método do gradiente, para 300 pontos geradores.

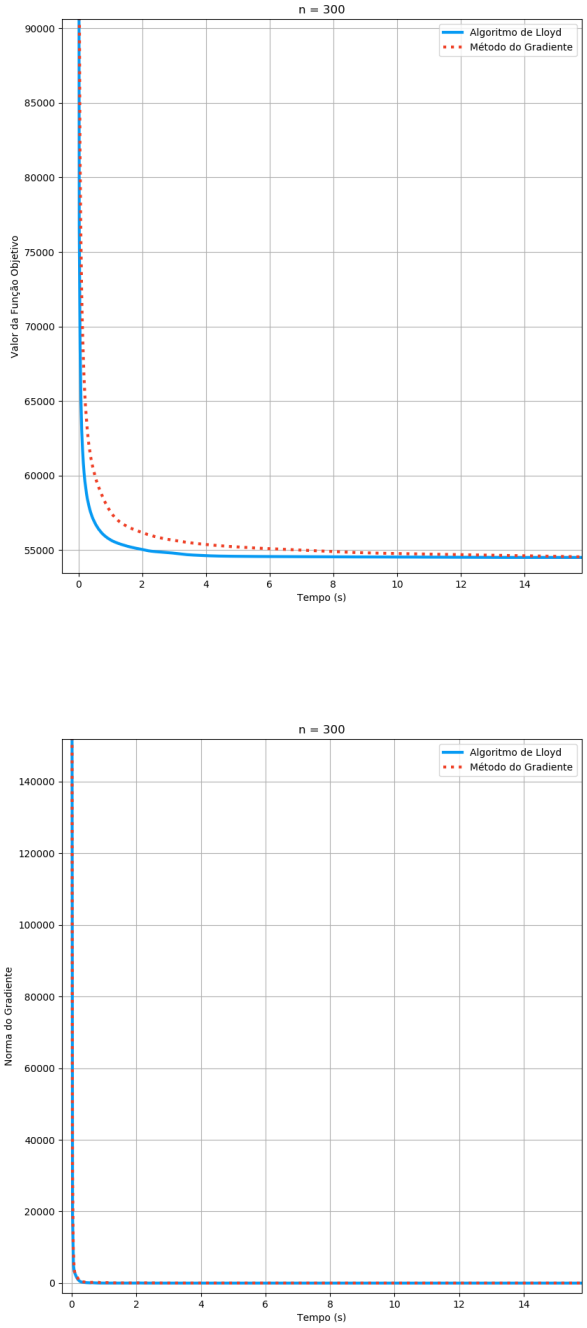
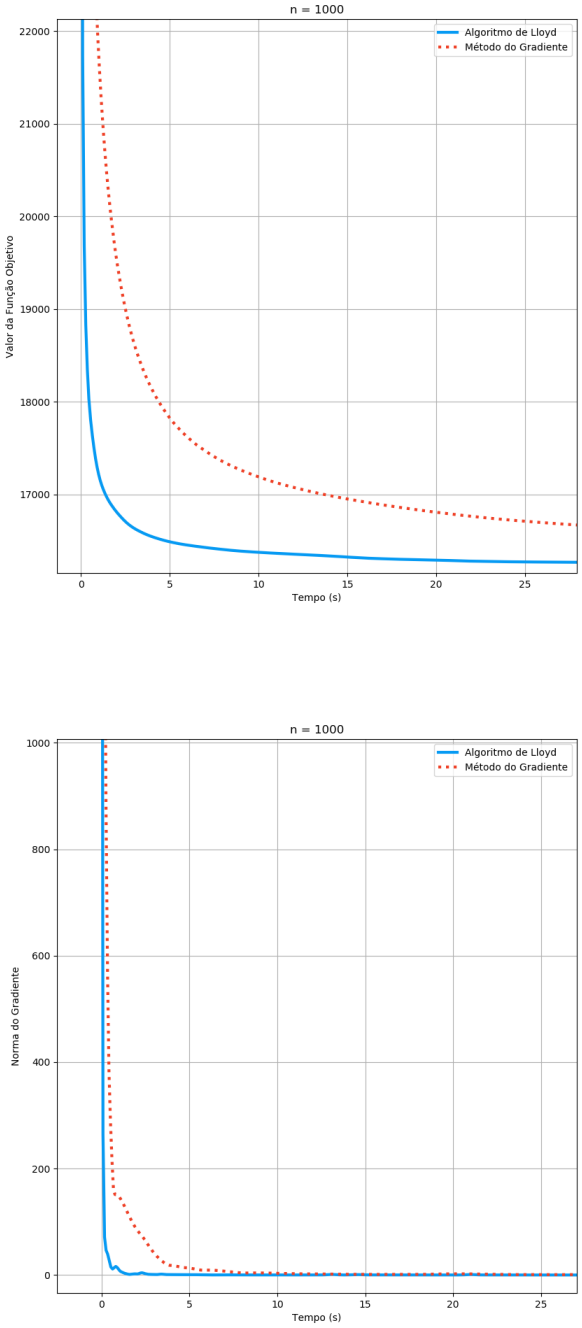


Figura 4.13: Comparação entre o algoritmo de Lloyd e o método do gradiente, para 1000 pontos geradores.



Conclusão

Neste trabalho, estudamos Diagramas de Voronoi e Tesselações Centroidais de Voronoi. Apresentamos propriedades geométricas destas estruturas e algumas aplicações.

Analisamos um algoritmo eficiente para a construção de Diagramas de Voronoi no plano, o algoritmo de Fortune.

Estudamos o algoritmo de Lloyd, um algoritmo iterativo para a construção de Tesselações Centroidais de Voronoi, explicando o seu problema de origem, a quantização ótima. A partir deste problema, interpretamos a construção de TCVs como problema de otimização e analisamos a sua solução com o método do gradiente.

Ao longo do trabalho, estudamos parte da literatura relevante e listamos neste documento alguns dos resultados mais importantes.

Implementamos os algoritmos descritos e testamos os seus desempenhos. Procuramos desenvolver todos os programas incluindo vizualizações claras e interativas do funcionamento dos algoritmos.

Para o futuro próximo, planejamos utilizar nossa compreensão atual do assunto para aplicarmos algoritmos clássicos de Otimização Não-Linear com funções objetivo estendidas, incorporando critérios de qualidade para a geração de malhas.

Bibliografia

[1] BERG, M. et al. Computational Geometry. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2008.

[2] CRAMÉR, H. Mathematical Method of Statistics. Princeton, NJ: Princeton University, 1951.

[3] DELAUNAY, B. Sur la sphère vide. Em: Proceedings of the International Mathematical Congress, p. 695–700, 1928.

[4] DIRICHLET, G. Über die Reduction der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen. Journal für die reine und angewandte Mathematik (Crelles Journal), v. 1850, n. 40, p. 209-227, 1850.

[5] DJIDJEV H.; LINGAS A. On computing the voronoi diagram for restricted planar figures. Em: Dehne F., Sack JR., Santoro N. (eds) Algorithms and Data Structures. Lecture Notes in Computer Science, vol 519. Springer, Berlin, Heidelberg, 1991.

[6] DU, Q.; EMELIANENKO, M.; JU, L. Convergence of the Lloyd Algorithm for Computing Centroidal Voronoi Tessellations. SIAM Journal on Numerical Analysis, v. 44, n. 1, p. 102-119, 2006.

[7] DU, Q.; FABER, V.; GUNZBURGER, M. Centroidal Voronoi Tessellations: Applications and Algorithms. SIAM Review, v. 41, n. 4, p. 637-676, 1999.

[8] FLEISCHER, P. Sufficient conditions for achieving minimum distortion in quantizer. IEEE Int. Convention Record, part I, vol. 12, pp. 104-111, 1964.

[9] FORTUNE, S. A sweepline algorithm for Voronoi diagrams. Algorithmica, v. 2, n. 1-4, p. 153-174, 1987.

[10] FRIEDLANDER, A. Elementos de Programação Não-Linear, Editora da Unicamp, 1994.

[11] JU, L.; RINGLER, T.; GUNZBURGER, M. Voronoi Tessellations and Their Application to Climate and Global Modeling. Washington, D.C: United States. Dept. of Energy, 2011.

[12] LIU, Y. et al. On centroidal voronoi tessellation—energy smoothness and fast computation. ACM Transactions on Graphics, v. 28, n. 4, p. 1-17, 2009.

[13] LLOYD, S. Least squares quantization in PCM. IEEE Transactions on Information Theory, v. 28, n. 2, p. 129-137, 1982.

[14] MIURA, H.; KIMOTO, M. A Comparison of Grid Quality of Optimized Spherical Hexagonal–Pentagonal Geodesic Grids. Monthly Weather Review, v. 133, n. 10, p. 2817-2833, 2005.

[15] NOCEDAL, J.; WRIGHT, S. Numerical optimization. Traducao . New York: Springer, 2006.

[16] PEIXOTO, P.; BARROS, S. Analysis of grid imprinting on geodesic spherical icosahedral grids. Journal of Computational Physics, v. 237, p. 61-78, 2013.

[17] SHAMOS, M. Computational Geometry. Tese de Doutorado—[s.l.] Yale Univesity, 1978.

[18] SHANNON, C. Communication in the Presence of Noise. Proceedings of the IRE, v. 37, n. 1, p. 10-21, 1949.

[19] VORONOI, G. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Journal für die reine und angewandte Mathematik (Crelle's Journal), v. 1908, n. 133, 1907.

[20] ZHU, B.; MIRZAIAN, A. Sorting does not always help in computational geometry. In Proc. 3rd Canad. Conf. Comput. Geom., p. 239242, 1991.