9. Aplicações

O diagramas de Voronoi e Delaunay podem ser usados para resolver diversos problemas. Nesta seção apresentaremos alguns destas aplicações.

Observação. O material desta seção foi extraído do: Capítulo 4 de [5]; Capítulos 5 e 6 de [12]; e Capítulo 5 de [11].

9.1. O Problema do Par Mais Próximo. O *Problema do Par Mais Próximo* consiste em: *dado*: um conjunto S de n pontos em E^2; *encontrar:* um par de pontos de S cuja distância entre eles é mínima. Um algoritmo ingênuo resolveria este problema em tempo $O(n^2)$. Como já vimos este problema pode ser resolvido em tempo ótimo $O(n \log n)$ por um algoritmo de divisão-e-conquista. Pelo Teorema 5 temos que cada par mais próximo corresponde a uma aresta de $\mathrm{Vor}(S)$ e a uma aresta de $\mathrm{Del}(S)$. Para encontrarmos um par mais próximo entre os pontos de S basta então examinarmos todas as arestas de $\mathrm{Vor}(S)$ ou $\mathrm{Del}(S)$. Este algoritmo gasta tempo $O(n)$ já que número de arestas em ambos os diagramas é $O(n)$ (cf. Corolário 1). Se combinarmos este algoritmo com um algoritmo que constroi $\mathrm{Vor}(S)$ ou $\mathrm{Del}(S)$ em $O(n \log n)$ teremos um algoritmo ótimo para o Problema do Par Mais Próximo.

Teorema 19. *O Problema do Par Mais Próximo pode ser resolvido em tempo $O(n \log n)$.* ∎

9.2. Todos os vizinhos mais próximos. O *Problema de Todos os Vizinhos Mais Próximos* (*All Nearest Neighbors*) consiste em: *dado*: um conjunto S de n pontos em E^2; *encontrar*: para cada ponto $p \in S$ um ponto em S que seja mais-próximo de p. Novamente, devido ao Teorema 5, para encontrarmos um (ou todos) os pontos mais próximos de um ponto $p \in S$ basta procurarmos entres seus vizinhos em $\mathrm{Del}(S)$ (ou, equivalentemente, examinar as arestas da região de Voronoi $V(p)$). Como o número de arestas é $O(n)$, a determinação de todos os vizinhos mais próximos é feita em $O(n)$. (Note que aqui estamos fazendo análise amortizada.)

Teorema 20. *O Problema de Todos os Vizinhos Mais Próximos pode ser resolvido em tempo $O(n \log n)$.* ∎

[Uma maneira de capiturar a essência do algoritmo para o Problema de Todos os Vizinhos Mais Próximos é através do *grafo dos vizinhos mais próximos* (*Nearest Neighbor Graph*) de S, denotado por $\mathrm{NNG}(S)$. Este grafo tem como vértices os pontos de S e tem uma aresta entre dois pontos se um dos pontos é o vizinho mais próximo do outro. Devido ao Teorema 5 podemos verificar que este grafo pode ser representado como um diagrama planar que esta contido em $\mathrm{Del}(S)$.]

9.3. Triangularização maximizando o ângulo mínimo. Uma *seqüência de ângulos de uma triangularização T* é uma seqüência $(\alpha_1, \alpha_2, \ldots, \alpha_{3t})$ dos ângulos dos triângulos que fazem parte de T em ordem crescente. Sejam T e T' triangularizações de um conjunto de pontos S, diremos que $T \succeq T'$ se a seqüência de ângulos de T é lexicograficamente maior ou igual e seqüência de ângulos de T': ou $\alpha_1 > \alpha'_1$ ou $\alpha_1 = \alpha'_1$ e $\alpha_2 > \alpha'_2$ ou $\alpha_1 = \alpha'_1$ e $\alpha_2 = \alpha'_2$ e

$\alpha_3 > \alpha'_3$, ou ... Em Edelsbrunner [4] páginas 302 e 303 o seguinte teorema é provado.

Teorema 21. *A triangularização de Delaunay $T := Del(S)$ é maximal em relação a ordem $\succeq$ das triangularizações de um conjunto S de pontos de E^2.* ∎

Em particular o teorema acima implica que a triangularização de Delaunay maximiza o menor ângulo.

Teorema 22. *O problema de encontrar uma triangularização T de um conjunto S de pontos tal que o menor ângulo de um triângulo em T seja máximo pode ser resolvido em tempo $O(n \log n)$.* ∎

9.4. **Maior círculo vazio.** O *Problema do Maior Círculo Vazio* (*Largest Empty Cycle*) consiste em: *dado*: um conjunto S de n pontos (sítios) em E^2; *encontrar:* o círculo com maior raio que tem o seu centro no fecho convexo de S e não contém nenhum ponto de S em seu interior.

Para cada ponto $p \in \text{conv}(S)$ denotaremos por $f(p)$ o valor do maior raio de um círculo com centro em p que não contenha nenhum sítio de S em seu interior. (Existe um número potencialmente infinito de pontos em $\text{conv}(S)$ que são candidatos a maximizar a função $f(p)$.) Chamaremos de *círculo vazio* a todo círculo que não contenha em seu interior nenhum ponto do conjunto dado S.

9.4.1. *Centro no interior de conv(S).* Seja p no interior de $\text{conv}(S)$ e seja $C(p)$ o círculo (vazio) de centro p e raio $f(p)$. Suponha primeiro que $C(p)$ contém exatamente um ponto $q \in S$. Se movermos o ponto p para um ponto p' um pouco mais distante de q na direção do vetor (p, q) teremos que $f(p') > f(p)$. Logo, se p é um ponto de $\text{conv}(S)$ que maximiza $f(p)$ então $C(p)$ não pode conter um único ponto de S. (Veja Figura 24.) Suponha agora que $C(p)$ contém dois pontos q e r de S. Se movermos o ponto p para o ponto p', mais distante do segmento qr ao longo da mediana do segmento qr teremos que $f(p') > f(p)$. (Veja Figura 24.) Assim, se p é um ponto de S que maximiza $f(p)$ então $C(p)$ não pode conter somente dois pontos de S. Agora se $C(p)$ contém três pontos de S não é possível movermos p um pouco e obter um ponto p' tal que $f(p') > f(p)$. Portanto, devido ao Teorema 3 podemos concluir o seguinte:

Lema 2. *Seja o p é um ponto no interior de conv(S) tal que o maior raio $f(p)$ (o maior raio de um círculo vazio com centro em p) é máximo. Então p é um vértice de Voronoi de Vor(S).* ∎

9.4.2. *Centro na fronteira de conv(S).* Seja p um ponto na fronteira de $\text{conv}(S)$ e seja $C(p)$ o círculo de centro p e raio $f(p)$. Analogamente ao que foi feito acima podemos concluir que se p é um ponto da fronteira de $\text{conv}(S)$ que maximiza $f(p)$ então $C(p)$ não pode conter um único ponto de S. (Veja Figura 25.) O raciocínio que usamos para mostrar que $C(p)$ deve ter mais do que dois pontos de S não funciona para um ponto p na fronteira de $\text{conv}(S)$ (pois ao movermos p para p' podemos obter um ponto p' fora de $\text{conv}(S)$).

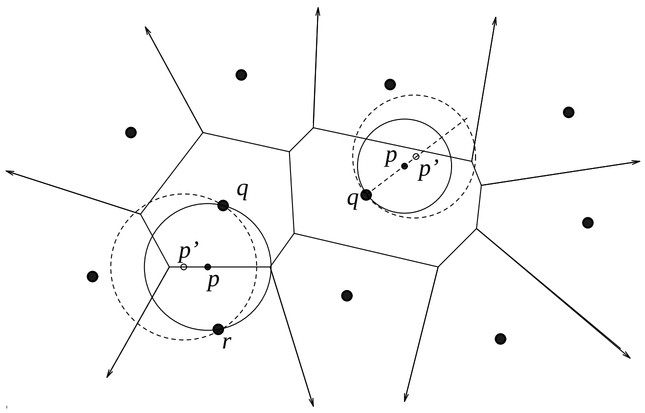

FIGURA 24. Círculos de centro p no interior de conv(S): círculo contendo exatamente um sítio e círculo contendo exatamente dois sítios.

Lema 3. *Seja p um ponto da fronteira do fecho convexo tal que $f(p)$ (o maior raio de um círculo vazio com centro em p) é máximo. Então p pertence a uma aresta de Voronoi.* ∎

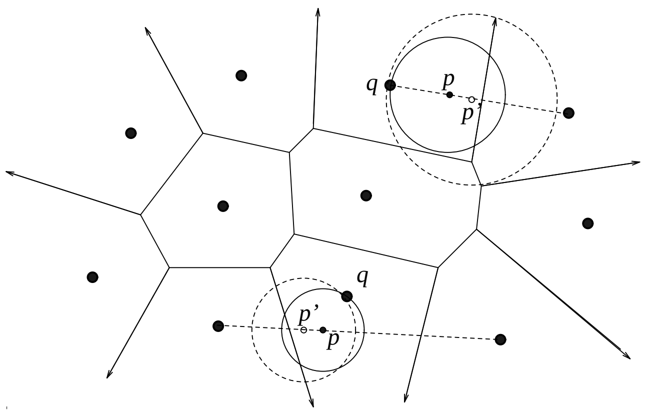

FIGURA 25. Círculo de centro p na fronteira de conv(S): círculo contendo exatamente um sítio.

Devido aos lemas anteriores vemos que é possível resolver o Problema do Maior Círculo Vazio em tempo $O(n \log n)$ se numa fase de pré-processamento construirmos Vor(S). O algoritmo está descrito abaixo.

Algoritmo `Maior-Circulo-Vazio`(S)
`Entrada`: Um conjunto S de n pontos em E^2.
`Saída`: Um ponto $p \in$ conv(S) tal que o raio $f(p)$ do círculo vazio de centro p é máximo.
1. Construa o diagrama de Voronoi de S. [Custa tempo $O(n \log n)$.]
2. $max \leftarrow 0$; $p \leftarrow$ `null`;
3. **for** cada vértice de voronoi v de Vor(S) **do**
4.   **if** v está em P

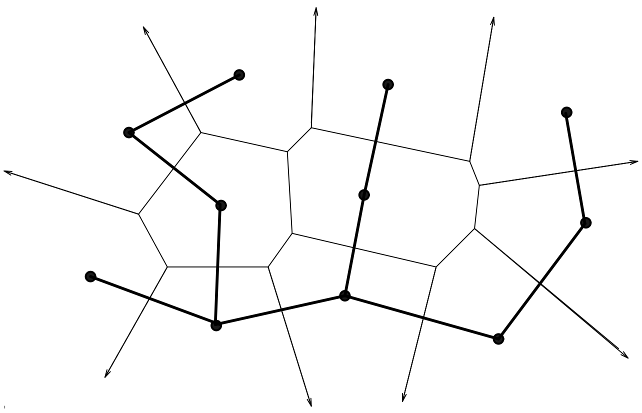

FIGURA 26. Uma árvore geradora mínima dos pontos de S

5. **then** Compute o maior círculo vazio de centro v e atualize p e max (se necessario).
6. **for** cada aresta de Voronoi e de Vor(S) **do**
7. Calcule a intersecção q de e com a fronteira de P.
8. Calcule $f(q)$ o raio do círculo vazio com centro em q e atualize p e max (se necessario).
9. **return** p e max.

Teorema 23. *O Problema do Maior Círculo Vazio pode ser resolvido em tempo $O(n \log n)$.* ∎

9.5. **Árvore geradora Euclidiana mínima.** Uma *árvore geradora Euclidiana mínima* (*Euclidean minimum spanning tree*) de um conjunto S de pontos em E^2 é uma árvore que tem como vértices os ponto de S, tem como arestas segmentos de reta com extremos em S e que entre as árvores que interligam todos os pontos de S e uma cujo comprimento total de suas aresta é mínimo. Um exemplo de árvore geradora mínima e mostrado na Figura 26.

Consideraremos aqui o *Problema da Árvore Geradora (Euclidiana) Mínima*: *dados:* uma conjunto S de n pontos em E^2; *encontrar:* uma árvore gerado de S cujo comprimento de suas arestas é mínimo. Um algoritmo conhecido para resolver este problema é o seguinte Algoritmo de `Kruskal` (que resolve um problema mais geral: *dados*: um grafo $G = (V, E)$ e uma função 'comprimento' $l : E \to \mathbb{R}$; *encontrar*: uma árvore geradora T de G de comprimento mínimo (i.e. $\sum_{e \in T} l(e)$ é mínimo)):

Algoritmo `Kruskal`(S)
`Entrada:` Um conjunto S de pontos em E^2.
`Saída:` Uma árvore geradora T de S de comprimento mínimo.
1. Sejam todos os segmentos $e_1, e_2, \ldots, e_k$ com extremidades em S em ordem crescente dos seus comprimentos.
2. $T \leftarrow \emptyset$; $i \leftarrow 1$;
3. **while** T não é uma árvore geradora **do**
4. **if** $T + e_i$ é acíclico

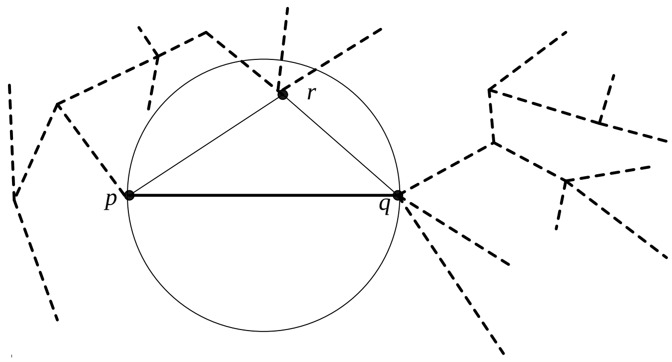

FIGURA 27. Ilustração da prova do Teorema 24.

5.   **then** $T \leftarrow T + e_i$
6. $i \leftarrow i + 1$;
7. **return** T.

A complexidade de tempo do algoritmo acima é $O(n^2 \log n)$. O tempo para ordenar os $\binom{n}{2}$ segmentos com extremidades no pontos de S. O tempo total gasto pelas Linhas 3–6 também é $O(n \log n)$). Como observaremos a seguir, com o auxílio do diagrama de Delaunay para S, o algoritmo acima pode ser acelerado para uma complexidade de tempo de $O(n \log n)$.
[Note a semelhança do algoritmo acima e do Algoritmo `Guloso` (da Lista de Exercícios 5) para encontrar uma triangularização de peso mínimo de um dado conjunto S de pontos de E^2. O Algoritmo `Kruskal` encontra uma árvore geradora (Euclidiana) mínima de S enquanto o Algoritmo `Guloso` falha na sua missão de encontrar um triangularização de S de peso mínimo.]

A observação chave para obtermos um algoritmo mais rápido para o Problema da Árvore Geradora (Euclidiana) Mínima é o seguinte teorema

Teorema 24. *Seja T uma árvore geradora mínima de um conjunto S de pontos em E^2. Então as arestas T são arestas do diagrama de Delaunay de S.*

Prova. Seja $e = pq$ uma aresta de uma árvore geradora mínima de S. Se pq não pertence a Del(S) então, pelo Teorema 4, o círculo que tem pq como diâmetro deve conter algum ponto de S em seu interior. Seja r um tal ponto. Quando removemos e de T o conjunto S é particionado em duas componentes conexas que podem ser ligadas pela aresta pr ou qr. Como ambas estas arestas são mais curtas que a aresta e nós temos uma contradição. (Veja Figura 27.) ∎

O teorema acima mostra como podemos melhorar o algoritmo de Kruskal. Primeiro, em um pré-processamento, construirmos em $O(n \log n)$ o diagrama de Delaunay de S e ordenamos as $O(n)$ arestas do diagrama (cf. Corolário 1) em tempo $O(n \log n)$. Agora, as Linhas 2–7 do algoritmo de Kruskal podem ser aplicadas nesta arestas ordenadas de Del(S) para obtermos uma árvore geradora mínima de S em tempo $O(n \log n)$ (veja, por exemplo, o Capítulo 24 de [1]).

Teorema 25. *O Problema da Árvore Geradora Mínima pode ser resolvido em tempo* $O(n \log n)$. ∎