DIAGRAMA DE VORONOI

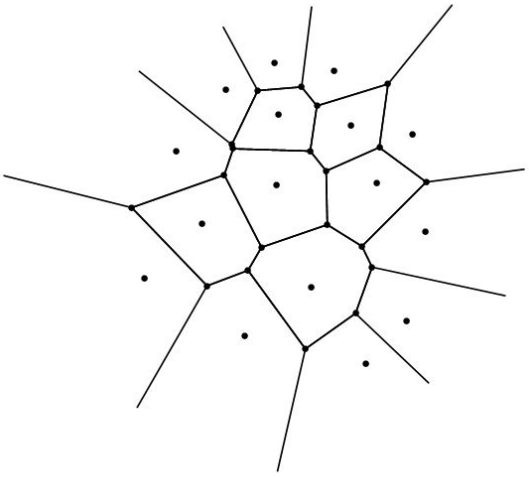
Andrew Ijano Lopes 10297797
Eduardo Rocha Laurentino 8988212

Seja P := {p1, . . . , pn} um conjunto de pontos no plano. Definimos uma célula para cada um destes pontos pi da seguinte forma:

V(pi) := {q : DIST(q, pi) < DIST(q, pj) para todo j ≠ i}.

O diagrama de Voronoi dos pontos de P, Vor(P), é tão somente a subdivisão do plano nas células V(p1), . . . , V(pn).

Um exemplo clássico que pode ser usado para absorver a definição é a situação onde, dados endereços de agências de correio, precisamos determinar qual é a região da cidade que fica mais próxima de cada agência. Cada agência é um ponto de P e a célula de Voronoi por ela determinada compreende toda a região da cidade que ela deve contemplar.



Note que cada célula é delimitada por arestas que são ou segmentos de reta ou semi-retas, e para cada par de pontos pi e pj com células vizinhas, a aresta separando suas células representa uma região do plano igualmente distante de ambos.

Note também que as intersecções entre essas arestas determinam pontos que chamamos de vértices de Voronoi, e estes são equidistantes dos três pontos que determinam as células que o envolvem. Uma propriedade importante é que tal vértice de Voronoi é o centro de uma circunferência que contém esses três pontos.

Neste projeto, dada uma coleção de pontos no plano, construímos o diagrama de Voronoi desta coleção pelo algoritmo de Fortune visto em aula. Assumimos que os pontos estão em posição geral.

CONSTRUÇÃO DO DIAGRAMA DE VORONOI

O algoritmo de Fortune trabalha com a técnica de linha de varredura que nesse caso faremos horizontal, percorrendo os pontos de cima pra baixo e tratando dois tipos de evento: eventos ponto e eventos círculo.

Para a linha de varredura, a estrutura de dados usada é uma Árvore de Busca Binária (ABB) que guarda em suas folhas os pontos já alcançados e nos nós internos pares de pontos vizinhos. Idealmente a ABB deve ser balanceada para garantir o desempenho ótimo com consumo de tempo O(nlog n) propiciado pelo algoritmo de Fortune (onde n é a quantidade de pontos em P). Em nossa implementação a ABB não é balanceada, o que pra efeitos práticos não onera tanto o desempenho do nosso algoritmo pelo fato de que trabalhamos com um conjunto de pontos que além de pequeno está em posição geral - e por está em posição geral a ABB associada tende a ser mais próxima de uma balanceada que de um desequilíbrio.

Além disso, os pontos eventos são mantidos dentro de um heap, de onde são extraídos sob critério de y-ordenação conforme avança a linha de varredura. Para tanto, importamos essa priority queue, *pqdict*, que nos garante buscas em O(1) e remoções e inserções em O(log n).

Com o andamento da varredura, o algoritmo de Fortune trabalha com a construção de uma linha de praia composta por arcos de parábola, decorrente das propriedades geométricas do problema conforme visto em aula.

Para armazenar o diagrama de Voronoi que está sendo construído, utilizamos uma Doubly-Connected Edge List (DCEL), que guarda os vértices do diagrama, suas arestas (um par de meia-arestas gêmeas) e suas células (guardadas como faces na DCEL).

REPRESENTAÇÃO GRÁFICA DO DIAGRAMA DE VORONOI

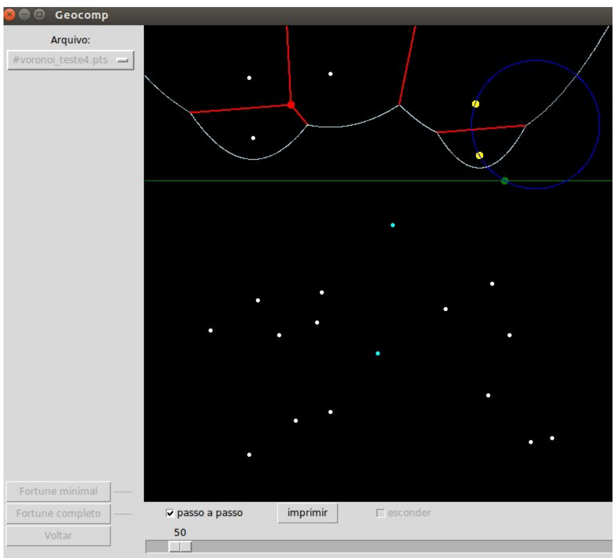
Fazemos a construção gráfica do diagrama contido na DCEL dinamicamente por meio da plataforma Computational Geometry Python Framework, desenvolvida por Alexis Sakurai Landgraf. Nela, adicionamos a funcionalidade de construção do diagrama de Voronoi para os conjuntos de pontos presentes no menu de seleção. No entanto, como restringimos o escopo do nosso projeto para um conjunto de pontos em posição geral, são poucos os casos de exemplo da plataforma em que nosso algoritmo funciona sem nenhum tipo de degeneração. O conjunto de pontos #00 é o que melhor representa uma posição geral para a construção do diagrama de Voronoi com o algoritmo de Fortune.

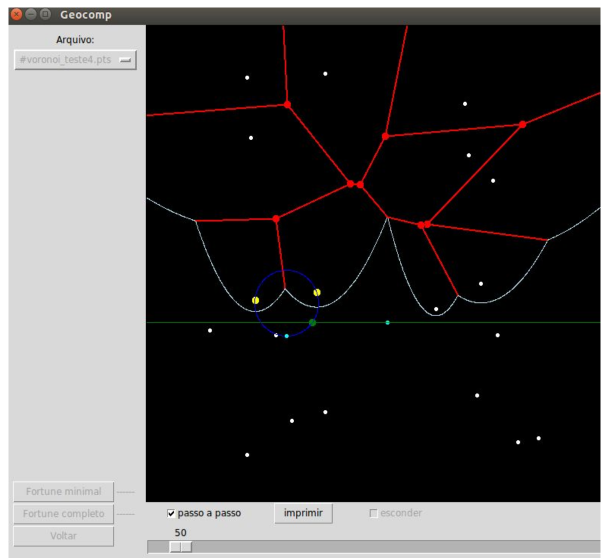
Além disso, a animação da construção do diagrama pode ser feita de duas maneiras, uma mostrando somente o diagrama efetivamente sendo construído e as parábolas da linha de praia e outra incluindo ainda a representação dos eventos círculos. A escolha entre um tipo de

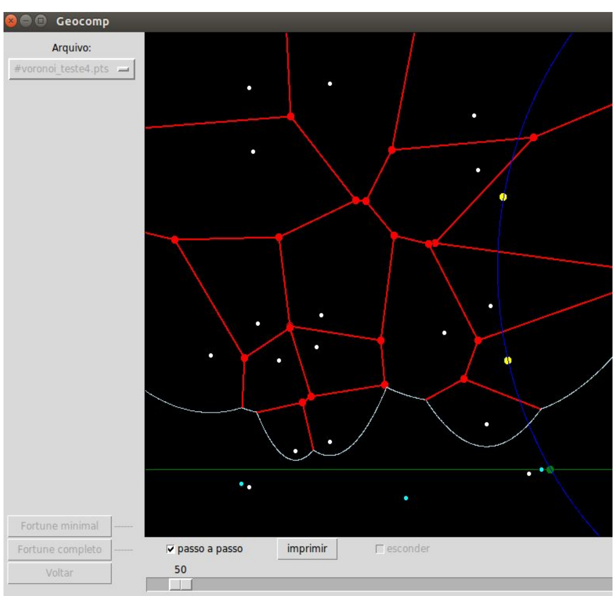
animação e outra é feita por meio de botões que incluímos na interface da própria plataforma: <u>Fortune completo</u> para com os eventos círculos e <u>Fortune minimal</u> para somente a linha de praia.

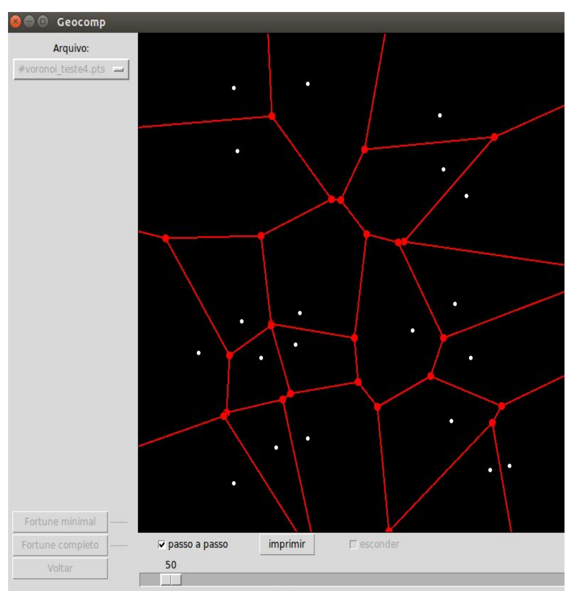
Em ambos os tipos de animação, vale a seguinte lógica de cores:

- Linha vermelha: aresta do diagrama de Voronoi
- Ponto vermelho: vértice do diagrama de Voronoi
- Ponto branco: ponto da coleção de pontos
- Ponto verde: ponto evento
- Circulo azul: evento circulo
- Ponto amarelo: ponto pertencente a um evento círculo
- Linha Verde: linha de varredura









CONSIDERAÇÕES FINAIS

1. Dependências: O código do nosso projeto depende de versões python 3.6.7 em diante, bem como da instalação de pqdict (sudo apt-get install pqdict).
2. De modo a permitir melhor validação, criamos alguns conjuntos de pontos que, assim como o #00, representam uma posição geral boa para o algoritmo de Fortune. Estes conjuntos de pontos estão nomeados como *voronoi_teste##*, onde ## é algum número.