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

Estruturas de Dados Cinéticas

Marcos Siolin Martins

Monografia Final

mac 499 — Trabalho de
Formatura Supervisionado

Supervisora: Profª. Drª. Cristina Gomes Fernandes

São Paulo
2022

Agradecimentos

Whatever you do, do it well.
— Walt Disney

Gostaria de colocar aqui os meus agradecimentos a todos aqueles que me apoiaram e contribuíram, de forma direta ou indireta, com a produção deste trabalho.

Gostaria de agradecer à minha família: as minhas irmãs, Melissa e Milena, e os meus pais especialmente, Elke e Renato. Eles têm me guiado e apoiado desde o início e foi com esse apoio e colaboração que as minhas maiores realizações foram possíveis.

Gostaria de agradecer também a todos os amigos e colegas que participaram da realização deste trabalho. Em particular, gostaria de agradecer ao Davi, ao Luciano, ao Willian (Hiroshi) e ao Daniel (Lawand).

Por fim, gostaria de agradecer a todos os meus professores que me ensinaram e me fizeram evoluir nos últimos quatro anos. Em especial, gostaria de agradecer a minha orientadora: a Cris, que me orientou com muita paciência e dedicação, mesmo antes, quando este estudo era ainda uma iniciação científica.

Resumo

Marcos Siolin Martins. **Estruturas de Dados Cinéticas**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2022.

Estruturas de dados cinéticas são um modelo proposto por Basch, Guibas e Hershberger para a resolução de problemas cinéticos de maneira eficiente. Problemas cinéticos são problemas em que os objetos envolvidos estão em movimento contínuo e desejamos saber um determinado atributo destes objetos no instante atual. Por exemplo, consultar qual o par de pontos mais próximo no momento, num conjunto de pontos em movimento.

A interface proposta para as estruturas oferece suporte a três operações: consulta, que retorna o atributo mantido, avançar no tempo, que altera o instante atual para um dado instante no futuro, e alterar a trajetória de objetos, caracterizada pela alteração de uma função cujo nome é plano de vôo. O plano de vôo é necessário para manter o atributo desejado sobre os elementos: ele define a trajetória dos pontos ao longo do tempo e será utilizado para calcular os chamados certificados. Os certificados são objetos que garantem que a organização interna da estrutura está correta até um determinado instante, denominado prazo de validade do certificado. As estruturas serão orientadas a eventos, que são o vencimento de certificados. Os certificados serão mantidos numa fila de prioridades com o seu prazo de validade como prioridade. O vencimento de um certificado significa que a estrutura ficou inválida e é necessário realizar mudanças para que ela volte a se tornar válida, removendo os certificados vencidos e gerando novos certificados no processo.

Com a inclusão da dimensão tempo, a forma tradicional de analisar algoritmos não é adequada para a análise dessas estruturas. Por isso, Basch, Guibas e Hershberger também propuseram outros quatro critérios com o intuito de determinar a eficiência de cada estrutura: responsividade, eficiência, localidade e compacidade.

Neste trabalho, estudaremos quatro problemas: ordenação, máximo, par mais próximo e triangulação de Delaunay, todos num contexto cinético. Também estudaremos as respectivas estruturas utilizadas na solução de cada problema e discutiremos sobre os critérios de eficiência em cada uma delas. Para algumas das estruturas também discutiremos um cenário dinâmico-cinético, ou seja, consideraremos operações de inserção e remoção dentro de um contexto cinético.

Palavras-chave: Estruturas de dados cinéticas. Geometria computacional. Algoritmos.

Abstract

Marcos Siolin Martins. **Kinetic Data Structures**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2022.

Kinetic data structures are a model proposed by Basch, Guibas and Hershberger to efficiently solve kinetic problems. Kinetic problems are problems in which the objects involved are in continuous motion and we want to know a certain attribute about these objects at the current moment. For example, we would like to query the closest pair of points at the moment, in a set of moving points.

The proposed interface for the data structures supports three operations: query, that returns the maintained attribute, advance, that adjusts the current moment to a given instant in the future, and changing the object motion, given by updates on a function called flight plan. The flight plan is necessary to maintain the attribute of interest over the elements, it defines the object motion through time and will be used for computing the so called certificates. The certificates are objects that ensure the internal state of our data structure is correct until a certain instant of time: the certificate's expiration time. Our data structures will be event driven where the events are the expirations of certificates. The certificates will be kept in a priority queue with their expiration time as priority. The expiration of a certificate means the data structure has become invalid and we need to change it in order to be in a correct state again, deleting the expired certificates and generating new ones in the process.

The traditional way of analyzing algorithms does not work well with these structures. Because of that, Basch, Guibas and Hershberger also proposed four criteria to determine the quality of the structures: responsiveness, efficiency, locality, and compactness.

In this work, we address four problems: sorting, maximum, closest pair and Delaunay triangulation, all in a kinetic context. We study the respective kinetic data structures used to solve these problems and discuss the quality criteria for each one of them. For some of the structures we also consider a kinetic-dynamic scenario, where insert and delete operations will also be supported in a kinetic context.

Keywords: Kinetic data structures. Computational geometry. Algorithms.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Sumário

Introdução

Quando desejamos criar algoritmos para resolver problemas com o computador, utilizamos maneiras de organizar os dados, as chamadas estruturas de dados, para que operações de acesso e alteração desses dados possam ser realizadas rapidamente. A forma como serão organizados os dados depende altamente das características do problema em questão.

Neste trabalho estudaremos *estruturas de dados cinéticas* (em inglês, *KDS – Kinetic Data Structures*), propostas por Basch, Guibas e Hershberger [Basch *et al.*, 1999] para a resolução dos chamados problemas *cinéticos*.

Problemas *cinéticos* são problemas em que deseja-se manter um determinado atributo sobre objetos que estão em movimento contínuo. Os objetos nos problemas podem representar entidades do mundo físico: pontos podem representar pessoas, aviões, aparelhos móveis, entre outras coisas, retas podem representar trajetórias. Devido à natureza desses problemas é razoável que estudemos problemas clássicos de geometria computacional, mas dentro de um contexto cinético. Por exemplo, num conjunto dado de pontos em movimento, qual par de pontos possui distância mínima.

Quando se tem dado um conjunto fixo de objetos geométricos, e deseja-se saber informações de um determinado atributo desses objetos (como, por exemplo, em um conjunto dado de pontos, qual par de pontos possui distância mínima), dizemos que esse é um problema *estático*.

O mesmo problema pode ser formulado sobre um conjunto mutável. Por exemplo, pontos poderiam ser inseridos e removidos ao longo do tempo. Queremos calcular o atributo sem ter que resolver do zero a nova instância do problema estático. Chamamos esse tipo de problema de *dinâmico* ou *on-line*.

As *estruturas de dados cinéticas* recebem esse nome para diferenciá-las das estruturas de dados *estáticas* e *dinâmicas*, pois têm como foco manter a descrição combinatória do problema que se altera frequentemente com a passagem de tempo, já que os objetos estão em movimento contínuo.

Essas estruturas nos permitem realizar consultas de um determinado atributo dos objetos, no instante atual. A garantia de que a estrutura permanece correta se dá através do uso de instrumentos chamados *certificados*. Os certificados estabelecem que uma relação entre um objeto da estrutura e outro se mantém verdadeira até o seu vencimento e devem ajudar na manutenção da estrutura para permitir as consultas desejadas. Chamaremos o instante de tempo em que o certificado vence de valor ou *prazo de validade* do certificado.

As estruturas contarão com uma operação ADVANCE, responsável por avançar até o

instante de tempo atual t mantendo a estrutura correta. Para tal, é necessário que nenhum certificado esteja vencido no instante t, ou seja, o certificado de menor prazo de validade expira após o instante t. Sendo assim, estamos interessados nos certificados de menor prazo de validade, para que possamos realizar os ajustes necessários enquanto existir um certificado que expira antes do instante de tempo para o qual desejamos alcançar.

Para identificar o instante de vencimento de certificados manteremos uma fila com prioridades, utilizando como prioridade o prazo de validade do certificado.

Para calcular o prazo de validade dos certificados, utilizaremos o chamado *plano de vôo* dos objetos. O plano de vôo de um objeto é uma função que, dado o instante de tempo atual, determina sua trajetória. Assim como na vida real, o plano de vôo pode sofrer mudanças. Essas mudanças no plano de vôo geram a necessidade de atualização de certificados e de ajustes nas estruturas. A operação CHANGE será utilizada para atualizar o plano de vôo dos objetos e realizar as mudanças necessárias para manter a estrutura correta.

Por fim, a operação QUERY ficará responsável por responder o atributo geométrico que desejamos saber num dado instante.

Uma questão natural a ser feita a respeito destas estruturas é como medir o desempenho delas, já que as formas clássicas de determinar a complexidade de algoritmos não se enquadram muito bem, por conta da adição da dimensão tempo. Basch, Guibas e Hershberger [BASCH *et al.*, 1999] propuseram algumas formas de analisá-las e medi-las. São elas:

- Responsividade: uma estrutura é dita *responsiva* se o custo de processar um certificado, isto é, o custo de atualizar os certificados e as outras estruturas necessárias é pequeno;
- Eficiência: uma estrutura é dita *eficiente* se a razão entre a quantidade total de eventos processados e a quantidade de eventos *externos* é pequena. Um evento diz respeito ao vencimento de um certificado, os eventos chamados *externos* são eventos que geram mudanças na descrição combinatória do atributo, enquanto eventos chamados *internos* não geram mudanças na descrição combinatória do atributo, mas ainda são necessários para manter a estrutura. O total de eventos processados é a soma da quantidade de eventos externos e internos;
- Compacidade: uma estrutura é dita *compacta* se a quantidade máxima de certificados que podem estar na fila com prioridades em um determinado instante é linear;
- Localidade: uma estrutura é dita *local* se a quantidade máxima de certificados na fila que estão relacionados com um determinado objeto é pequena.

O custo de uma operação é dito pequeno se o custo é assintoticamente polilogarítmico no número de objetos ou polinomial para um valor pequeno no expoente.

Neste trabalho estudaremos alguns problema cinéticos e as estruturas utilizadas para resolvê-los. Apesar do modelo proposto por Basch, Guibas e Hershberger [BASCH *et al.*, 1999] funcionar para trajetórias não-lineares, nos restringiremos apenas a trajetórias lineares. Neste caso, cada objeto corresponde a um par (x_0, v), onde x_0 representa o valor inicial do objeto e v é a velocidade atual do objeto.

No Capítulo 1 estudaremos o problema da ordenação cinética. Neste problema, os

pontos movem-se apenas em uma dimensão $y(t) = x_0 + vt$ e desejamos saber qual dos pontos possui o i-ésimo maior valor na coleção em determinado instante t. As estruturas que consideraremos para resolver o problema são a lista ordenada cinética e uma árvore binária balanceada de busca. A árvore binária balanceada de busca além de suportar as operação já citadas também será capaz de realizar operações INSERT e DELETE.

No Capítulo 2 estudaremos o problema do máximo cinético. Assim como no problema anterior, os pontos movem-se apenas em uma dimensão $y(t) = x_0+vt$ e desejamos saber qual dos pontos possui o maior valor no instante atual. Apesar do problema anterior resolver este problema, pois basta buscar pelo primeiro maior valor na coleção, veremos que é possível obter essa resposta de forma mais eficiente utilizando outras estruturas.

No Capítulo 3 estudaremos o problema do par mais próximo cinético, em que utilizaremos estruturas vistas nos dois capítulos anteriores. Diferentemente dos problemas anteriores, os pontos movem-se em duas dimensões de acordo com uma função $\gamma(t) = (x(t), y(t))$, sendo que $x(t) = x_0 + v_x t$ e $y(t) = y_0 + v_y t$. Desejamos saber qual dos pares possíveis de pontos possui distância mínima.

No Capítulo 4 estudaremos o problema da triangulação de Delaunay cinética. Assim como no problema do par mais próximo, os pontos movem-se em duas dimensões de acordo com uma função $\gamma(t) = (x(t), y(t))$, sendo que $x(t) = x_0 + v_x t$ e $y(t) = y_0 + v_y t$. Desejamos manter uma descrição da triangulação de Delaunay dos pontos.

Capítulo 1

Par mais próximo cinético

Considere o seguinte problema cinético. São dados n pontos movendo-se linearmente no plano. Cada ponto é representado por um par $(s_0, \vec{v})$ onde $s_0 = (x_0, y_0)$ é a sua posição inicial e $\vec{v} = (v_x, v_y)$ um vetor velocidade. A posição de um determinado ponto p, num instante arbitrário $t \geq 0$, é $s_p = (x_p, y_p) = (x_0, y_0) + t \cdot \vec{v}$. Queremos saber o par (p, q) cuja distância $d(p, q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$ é mínima, num instante arbitrário $t \geq 0$.

Por exemplo, se tivermos 5 pontos na coleção, representados na figura 1.1: $((1, 0), (2, 1))$, $((5, -1), (-1, 2))$, $((0, 2), (1, -1))$, $((3, 2), (1, -2))$ e $((3, 1), (-1, 0))$.

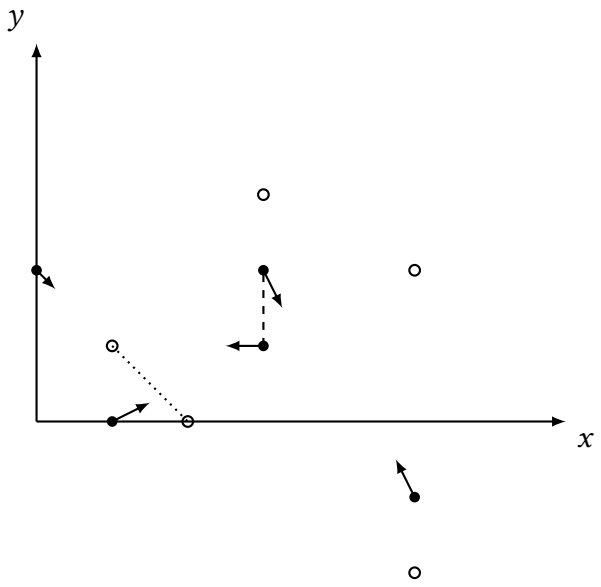

Figura 1.1: *Os pontos preenchidos em preto representam a coleção no instante $t = 0$. Os pontos não preenchidos representam a coleção no instante $t = 2$. As setas representam a direção e sentido do vetor velocidade de cada ponto. A linha tracejada representa a distância entre o par mais próximo no instante $t = 0$, enquanto a linha pontilhada representa a distância entre o par mais próximo no instante $t = 2$.*

Queremos dar suporte às seguintes operações:

- ADVANCE(t) → avança o tempo corrente para t;

- CHANGE($j, \vec{v}$) → altera a velocidade do ponto j para $\vec{v}$;
- QUERY_CLOSEST() → devolve os pontos que formam o par mais próximo no instante atual.

1.1 Algoritmo cinético

1.1.1 Análise de desempenho

As análises de desempenho aqui foram extraídas de **eduardo**.

A estrutura de dados cinética para manter o par de pontos mais próximo é uma estrutura *responsiva*, pois o custo de processar um certificado é $O(\lg n)$. O custo de processar um certificado é o custo de realizar as trocas necessárias nas listas ordenadas que consome tempo $O(\lg n)$, além disso também há o custo de corrigir as árvores *Cands*, $Hits_{low}$, $Hits_{up}$, o torneio e os certificados associados. Mas, essas operações são realizadas em sequência, consumindo um custo também de $O(\lg n)$.

A estrutura é *eficiente*, pois a razão entre o total de eventos e os eventos *externos*, isto é, as trocas de par mais próximo. De acordo com **eduardo**, essa razão é $O(\epsilon \lg n)$, resultando em uma estrutura eficiente.

A estrutura é *compacta*, pois teremos $O(n)$ certificados na fila de prioridades associados a mudanças nas listas ordenadas e $O(n)$ certificados do torneio cinético, resultando em $O(n)$ certificados na fila com prioridades num determinado instante.

A estrutura é *local*, pois um ponto pode estar envolvido em até seis certificados das listas ordenadas, sendo dois para cada uma das ordenações, e pode estar envolvido em até $O(\lg n)$ certificados no torneio.

Capítulo 2

Triangulação de Delaunay cinética

Considere o seguinte problema cinético. São dados n pontos movendo-se linearmente no plano. Cada ponto é representado por um par $(s_0, \vec{v})$ onde $s_0 = (x_0, y_0)$ é a sua posição inicial e $\vec{v} = (v_x, v_y)$ um vetor velocidade. A posição de um determinado ponto p, num instante arbitrário $t \geq 0$, é $s_p = (x_p, y_p) = (x_0, y_0) + t \cdot \vec{v}$. Queremos saber o conjunto de arestas que define a triangulação de Delaunay desses pontos, num instante arbitrário $t \geq 0$.

Por exemplo, se tivermos 8 pontos na coleção, representados na Figura 2.1:

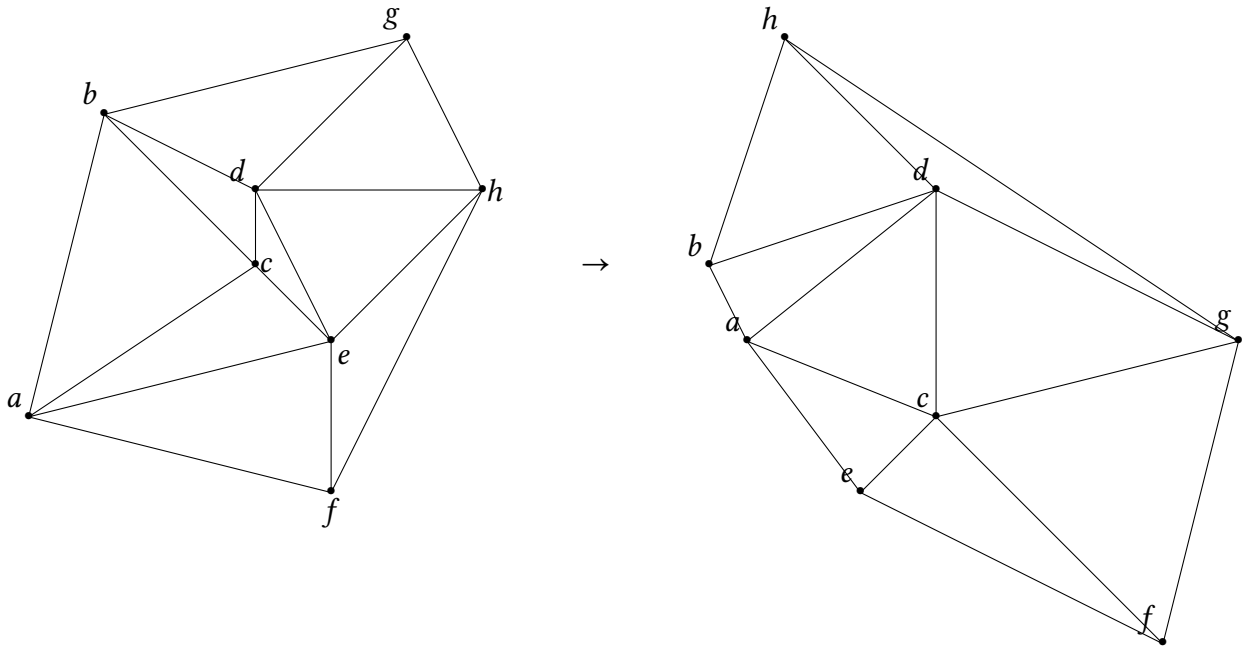

Figura 2.1: *Exemplo.*

Para entender o que é a triangulação de Delaunay de um conjunto de pontos, precisamos antes entender o que é o diagrama de Voronoi.

O diagrama de Voronoi de um conjunto de pontos P é uma partição do plano em regiões que correspondem aos pontos de p. Cada região $R(p)$ compreende todos os pontos mais próximos de p do que qualquer outro ponto do conjunto P. As regiões são polígonos cujas arestas são formadas por pontos que estão equidistantes de dois ou mais pontos de P. Os vértices desse polígono estão equidistantes de três ou mais pontos de P.

A triangulação de Delaunay de um conjunto de pontos P é definida como a triangulação do grafo dual do diagrama de Voronoi para P. Os vértices do grafo dual são os pontos de P e para cada região associada ao ponto no diagrama de Voronoi existe uma aresta entre o vértice associado à região e os vértices associados às regiões adjacentes, veja a Figura **??**. O grafo dual é chamado de grafo de Delaunay e a triangulação desse grafo é chamada de triangulação de Delaunay.

Em geral, o grafo de Delaunay já é uma triangulação. Somente no caso em que existem quatro ou mais pontos co-circulares é possível que existam faces no grafo de Delaunay com quatro ou mais arestas, veja a Figura **??**.

Considere uma aresta e de uma triangulação T de P. Se e não faz parte do fecho convexo de P, então faz parte de dois triângulos adjacentes na triangulação. Se esses triângulos formam um quadrilátero convexo, então os vértices da aresta podem ser trocados pelos outros dois vértices do quadrilátero, gerando uma nova triangulação. Essa troca na aresta gera seis novos ângulos na triangulação, veja a Figura **??**.

Uma aresta é dita ilegal se após a troca dos vértices, o menor dos ângulos do novo quadrilátero formado é maior que o menor dos ângulos do quadrilátero anterior a troca da aresta. Em **computational-geometry** é demonstrado que uma aresta $p_i p_j$ é ilegal se o ponto p_l do quadrilátero convexo $p_i p_k p_j p_l$ está no interior do círculo que passa por $p_i p_k p_j$. Também é demonstrado que uma triangulação que não possui arestas ilegais é uma triangulação de Delaunay.

Isso nos dá um algoritmo para calcular a triangulação de Delaunay, podemos começar com uma triangulação qualquer e trocar arestas que identificarmos como ilegais. Uma possível forma de gerar uma triangulação inicial é utilizando o algoritmo de Graham, veja o Algoritmo **??**.

As rotinas utilizadas no Algoritmo **??** são os testes LEFT e INCIRCLE. O teste LEFT é un teste bem conhecido na geometria computacional e consiste em avaliar o valor do seguinte determinante:

$$\text{LEFT}(p_i, p_j, p_k) = \begin{vmatrix} x_{p_i} & y_{p_i} & 1 \\ x_{p_j} & y_{p_j} & 1 \\ x_{p_k} & y_{p_k} & 1 \end{vmatrix}$$

em que um valor maior que zero significa que p_k está à esquerda da reta orientada que passa por p_i e p_j, zero significa que p_k está sobre a reta e um valor menor que zero significa que p_k está à direita da reta.

Guibas e Stolfi [**guibas-stolfi**] mostram que para determinar se um ponto p_l está dentro

do círculo que passa por $p_i p_j p_k$ podemos avaliar o valor do seguinte determinante:

$$\textsc{incircle}(p_i, p_j, p_k, p_l) = \begin{vmatrix} x_{p_i} & y_{p_i} & x_{p_i}^2 + y_{p_i}^2 & 1 \\ x_{p_j} & y_{p_j} & x_{p_j}^2 + y_{p_j}^2 & 1 \\ x_{p_k} & y_{p_k} & x_{p_k}^2 + y_{p_k}^2 & 1 \\ x_{p_l} & y_{p_l} & x_{p_l}^2 + y_{p_l}^2 & 1 \end{vmatrix}$$

em que um valor maior que zero significa que p_l está dentro do círculo, zero significa que p_l está na borda do círculo e um valor menor que zero significa que p_l está fora do círculo.

Guibas, Mitchell e Roos [**guibas-mitchell-roos**] propuseram uma estrutura de dados cinética para manter a triangulação de Delaunay. Os certificados da estrutura serão os testes de LEFT e INCIRCLE utilizados para assegurar que não existem arestas ilegais. Os momentos em que ocorre uma inversão no valor desses determinantes definem o prazo de validade destes certificados.

Um evento associado ao vencimento de um certificado do tipo INCIRCLE resulta na troca de uma aresta, veja a Figura **??**. Um detalhe que vale ser ressaltado é que a troca só deve ser realizada se o quadrilátero em questão é convexo.

Um evento associado ao vencimento de um certificado do tipo LEFT resulta na adição ou remoção de um vértice ao fecho convexo dos pontos e na remoção ou adição de um novo triângulo na triangulação, veja a Figura **??**.

Na verdade, utilizaremos uma técnica muito comum na geometria computacional para facilitar a computação dos eventos. Adicionaremos um ponto no infinito e uma aresta entre cada ponto no fecho convexo e esse ponto no infinito. Dessa forma, poderemos tratar todos eventos da mesma forma, cada evento consiste na troca de uma aresta do quadrilátero formado pelos quatro vértices que participam do evento. Eventos envolvendo o ponto no infinito ainda devem ter o prazo de validade calculado através da rotina LEFT com os três pontos que não são o infinito.

Além disso, em certos casos degenerados, é possível que quadriláteros adjacentes estejam envolvidos num evento ao mesmo tempo. Esses são os casos em que o grafo de Delaunay não corresponde à triangulação de Delaunay, com faces que são polígonos com mais de três lados. Isso ocorre quando mais de quatro pontos numa vizinhança se tornam co-circulares e são casos que devem ser tratados de forma especial. Esses casos são identificados quando os eventos da fila de prioridades ocorrem no mesmo instante, envolvendo quadriláteros adjacentes. A solução proposta em **aggarwal-guibas-saxe-shor** calcula uma triangulação para o instante $t + \epsilon$ em tempo linear, sendo t o instante de ocorrência do evento. Assim, após o processamento desses eventos a estrutura permanece correta.

A estrutura de dado cinética pode ser construída da seguinte forma:

1. Calcular a triangulação de Delaunay de P para os pontos na posição inicial.
2. Para cada quadrilátero formado por triângulos adjacentes na triangulação calcular os prazos de validade dos certificados e inseri-los numa fila com prioridades. Isso

inclui os quadriláteros que tem o ponto no infinito como vértice, que terão seus prazos de validade calculados de acordo com o zero da função LEFT, e não da função INCIRCLE como os demais.

Note que a estrutura de dados pode ser gerada independentemente do algoritmo utilizado para calcular a triangulação de Delaunay inicial. Isso nos permite utilizar um algoritmo mais eficiente para calcular a triangulação inicial, como o algoritmo incremental que consome tempo $O(n \lg n)$. O segundo passo também consome tempo $O(n \lg n)$, já que calcular os prazos de validade toma tempo constante e a inserção na fila com prioridades $O(\lg n)$.

2.1 Análise de desempenho

As análises de desempenho aqui foram extraídas de **eduardo**.

A estrutura de dados cinética para manter a triangulação de Delaunay é uma estrutura *responsiva*, pois o custo de processar um certificado é $O(\lg n)$. O custo de processar um certificado é o custo de trocar uma aresta e recalcular os certificados dos quatro quadriláteros envolvidos nessa troca. Nos casos degenerados é mais do que quatro quadriláteros serão envolvidos, mas em **citaraqui** é mostrado que o tempo amortizado continua sendo $O(\lg n)$.

A estrutura é *eficiente*, pois todos os eventos processados são eventos *externos*, isto é, todo vencimento de certificado representa uma troca na descrição combinatória da triangulação de Delaunay. De acordo com **eduardo**, no caso de movimentos lineares, o número total de eventos é $O(n^3)$.

A estrutura é *compacta*, pois cada certificado está associado a uma aresta, teremos $O(n)$ certificados na fila de prioridades num determinado instante.

A estrutura não é *local*, pois um ponto pode estar envolvido em até $O(n)$ certificados, veja a Figura **??**.

Referências

[Basch *et al.* 1999] Julien Basch, Leonidas J Guibas e John Hershberger. "Data structures for mobile data". Em: *Journal of Algorithms* 31.1 (1999), pgs. 1–28. issn: 0196-6774. doi: https://doi.org/10.1006/jagm.1998.0988. url: https://www.sciencedirect.com/science/article/pii/S0196677498909889 (citado nas pgs. 1, 2).