UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

IMPLEMENTAÇÕES DO ALGORITMO DE FORTUNE PARA VARIANTES DO DIAGRAMA DE VORONOI

MATHEUS ANTONIO VENANCIO DALL'ROSA

CHAPECÓ
2017

MATHEUS ANTONIO VENANCIO DALL'ROSA

IMPLEMENTAÇÕES DO ALGORITMO DE FORTUNE PARA VARIANTES DO DIAGRAMA DE VORONOI

Trabalho de conclusão de curso de graduação apresentado como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.
Orientador: Prof. Dr. Emílio Wuerges

Co-orientador: Prof. Dr. André Luiz Pires Guedes - UFPR

CHAPECÓ
2017

MATHEUS ANTONIO VENANCIO DALL'ROSA

IMPLEMENTAÇÕES DO ALGORITMO DE FORTUNE PARA VARIANTES DO DIAGRAMA DE VORONOI

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Prof. Dr. Emílio Wuerges

Aprovado em: 21 / 07 / 2014

BANCA EXAMINADORA:

Dr. Emílio Wuerges - UFFS

Dr. André Luiz Pires Guedes - UFPR por vídeo

Dr. Braulio Adriano de Mello - UFFS

Me. Leandro Miranda Zatesko - UFFS por vídeo

RESUMO

Dado um conjunto S com n pontos no plano, o diagrama de Voronoi é uma subdivisão do plano em n regiões, uma região para cada ponto em S. A construção do diagrama de Voronoi pode ser feita através de algoritmos que se utilizam de técnicas como divisão e conquista, construção aleatória incremental e varredura do plano. A literatura aponta que os algoritmos para construção do diagrama possuem considerável complexidade de implementação. Este trabalho descreve formas de implementação de alguns passos do algoritmo de Fortune para as variações do diagrama de Voronoi em que o conjunto S pode ser formado por pontos com ou sem peso. Além disso, este trabalho propõe uma implementação do algoritmo de Fortune para estas duas variantes do diagrama.

Palavras-chave: Geometria Computacional. Diagrama de Voronoi. Algoritmo de Fortune.

ABSTRACT

Given a set S of points on the plane, the Voronoi diagram is a subdivision of the plane in n regions, one region for each point in S. The construction of the Voronoi diagram can be made through algorithms which are based on techniques like Divide and Conquer, Randomized Incremental Construction and Plane Sweep. Some important works on the construction of the diagram point out that these algorithms have high implementation complexity. This work describes ways of implementation of some steps of Fortune's algorithm for the variations of Voronoi diagram wherein the set S can contain weighted or unweighted points. Furthermore, this work proposes an implementation of Fortune's algorithm for these two variants of the diagram.

Keywords: Computational Geometry. Voronoi Diagram. Fortune's Algorithm.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ES	Evento de Site
EV	Evento de Vértice
BST	Árvore Binária Balanceada

SUMÁRIO

1 INTRODUÇÃO

Dado um conjunto S com N pontos no plano, o diagrama de Voronoi é uma subdivisão do plano em N regiões, uma região para cada ponto em S. A região de cada ponto $i \in S$ é formada por todos os pontos mais próximos de i do que qualquer outro ponto em S. O diagrama de Voronoi possui propriedades interessantes sobre à distância euclidiana entre pontos no plano e também possui relações com outras estruturas geométricas importantes, como a Triangulação de Delaunay [1] e o Fecho Convexo [2], o diagrama possui muitas aplicações em diversas áreas do conhecimento [1].

1.1 Contextualização

Em 1975 o diagrama de Voronoi foi introduzido na Geometria Computacional por Shamos e Hoey. O diagrama foi utilizado em [10] para prover as soluções de sete problemas computacionais de natureza geométrica. No mesmo trabalho foi apresentado um algoritmo que se utiliza da técnica de divisão e conquista para construir o diagrama de Voronoi em tempo $O(N \log N)$ e complexidade de espaço $O(N)$, uma vez tendo o diagrama construído, as soluções apresentadas em [10] para cinco dos sete problemas passam a ter solução com complexidade de tempo $O(N \log N)$, e os outros dois problemas passam a ter solução com complexidade de tempo $O(N)$. Tendo em vista que o diagrama pode ser construído em tempo $O(N \log N)$, antes de [10] todos os problemas possuíam somente soluções com complexidade de tempo $\Omega(N^2)$.

Todos os sete problemas solucionados em [10] possuem um conjunto S de N pontos no plano como entrada, e estão relacionados a distância entre os pontos deste conjunto. Segue a lista dos sete problemas:

- Para cada ponto $p_i \in S$ encontrar o ponto $p_j \in S$ diferente de p_i e que esteja mais próximo de p_i do que qualquer outro ponto em S.
- Encontrar uma árvore geradora mínima cujo os vértices são os pontos em S.
- Encontrar uma triangulação T tal que todos os pontos em S sejam utilizados como vértices dos triângulos em T, e que a soma do comprimento de todas as arestas dos triângulos

[1] A relação com a Triangulação de Delaunay pode ser encontrada nos Capítulos 7 e 9 de [2].
[2] A relação com o Fecho Convexo pode ser encontrada em [10] e no Capítulo 11 de [2].

em T seja mínima.

- Encontrar o Fecho Convexo de S.
- Encontrar o maior círculo C que não contém nenhum ponto de S em seu interior e cujo o centro de C está contido no interior do Fecho Convexo de S ou está contido no Fecho Convexo.
- Encontrar os k pontos em S mais próximos de um ponto $p \notin S$.
- Encontrar o menor círculo que contém todos os pontos de S.

Uma vez construído o diagrama para um conjunto S com N pontos do plano, chamados *sites*, o plano será particionado em N regiões, uma região para cada *site* em S. A Figura 1.1 ilustra o diagrama de Voronoi para o conjunto $S = \{a,\ b,\ c,\ d,\ e,\ f\}$.

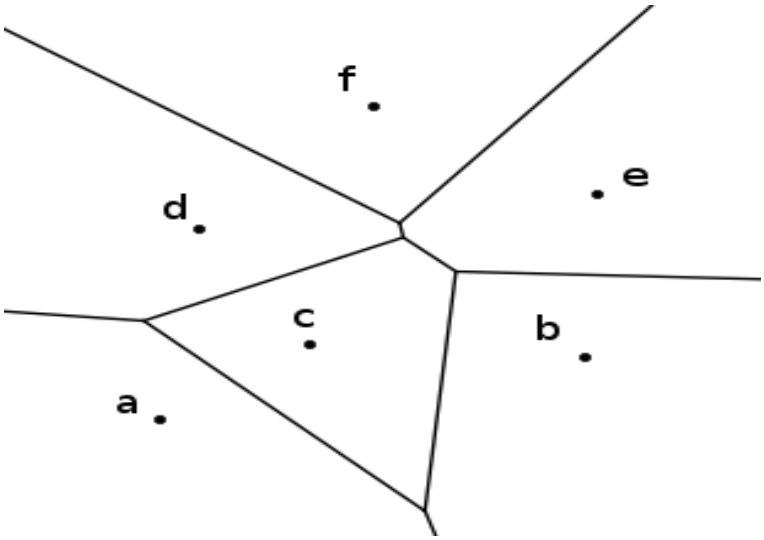

Figura 1.1 – Diagrama de Voronoi para seis *sites*

Na Figura 1.1 cada ponto do plano contido na região do *site* c está mais próximo de c do que qualquer outro *site* em S, esta propriedade também é válida para todos os outros *sites* em S, neste exemplo as regiões são limitadas por semirretas e segmentos de retas, porém se todos os *sites* forem colineares as regiões devem ser limitadas por retas.

A variante apresentada na Figura 1.1 é a variante para pontos do diagrama de Voronoi. Existem outras variações do diagrama. A variante para pontos com peso [3] também é apresentada neste trabalho, nesta segunda variante cada *site* terá um peso não-negativo associado, este peso

[3] Como é apontado em [3] o diagrama para pontos com peso é equivalente ao diagrama para círculos. Em [11] são apresentadas várias propriedades do diagrama para círculos.

será utilizado no cálculo da distância de um ponto no plano para o respectivo *site*, ou seja, a métrica de distância deixa de ser apenas a distância euclidiana, e passa a ser a distância euclidiana somada ao peso do respectivo *site*. Além destas duas variantes existem outras com diferentes propriedades, diferentes formas geométricas de *sites* e diferentes métricas de distância.

O diagrama de Voronoi também pode ser construído através de outros algoritmos, como o algoritmo de Fortune, a primeira versão deste algoritmo publicada em [3] se utiliza da técnica de varredura do plano, para ser possível a utilização da técnica, o algoritmo realiza uma transformação geométrica no diagrama e só então computa o diagrama do conjunto de *sites*, a transformação é importante, pois o diagrama após a transformação possui algumas propriedades que facilitam a sua construção. Segundo Fortune seu algoritmo pode ser adaptado para os casos onde o conjunto de *sites* S é formado somente por pontos, somente por pontos com peso ou somente por segmentos de reta. A segunda versão publicada em [2] também utiliza a técnica de varredura do plano, porém ao invés da transformação geométrica a segunda versão do algoritmo mantém uma *Beach Line* [4], a *Beach Line* garante algumas propriedades durante a varredura do plano que são importantes para a construção do diagrama. As duas versões do algoritmo de Fortune possuem complexidade de tempo no pior caso de $O(N \log N)$ e $O(N)$ de espaço.

Existem outros algoritmos que se utilizam da técnica de construção aleatória incremental, alguns destes algoritmos possuem complexidade de tempo esperada de $O(N\ log\ N)$, um exemplo é o algoritmo apresentado em [5], o algoritmo consiste na escolha aleatória do próximo *site* a ser colocado no diagrama.

Este trabalho apresenta formas de implementação para alguns passos da primeira versão do algoritmo de Fortune, que não foram relatados por Fortune. Além disso, é proposto uma implementação deste algoritmo para a construção do diagrama de Voronoi quando os *sites* não possuem peso, e outra implementação para o caso em que os *sites* possuem peso. Neste trabalho foi escolhido a primeira versão do algoritmo de Fortune, pois em [3] foi mostrado como utilizar o algoritmo para pontos e pontos com peso, já a apresentação da segunda versão do algoritmo em [2] não possui a descrição de uma forma de adaptação do algoritmo para pontos com peso.

[4] *Beach Line* é uma sequência alternada de arcos parabólicos e pontos do plano.

1.2 Problemas e Resultados

Fortune propôs o uso de uma transformação geométrica para fazer possível a utilização da técnica de varredura do plano, para a construção do diagrama Voronoi. Em [3] é esclarecido como se realiza a transformação geométrica na estrutura do diagrama para o caso em que os *sites* não possuem peso, porém o algoritmo necessita de comparações entre os elementos do diagrama após a transformação. O trabalho [3] explica que é necessário realizar comparações, mas não recomenda uma forma com que essas comparações sejam feitas. Já no caso em que os *sites* possuem um peso não-negativo associado, Fortune não expôs uma forma de computar a transformação geométrica nos elementos do diagrama, e também não mostrou uma maneira de efetuar as comparações necessárias.

Para a variante em que os *sites* possuem um peso não-negativo associado, a computação da transformação geométrica é mais complexa devido às equações envolvidas no processo, por este mesmo motivo, computar os vértices do diagrama não é uma tarefa fácil. Já às comparações necessárias para esta variante podem ser reduzidas as comparações feitas no caso em que os sites não possuem peso.

Como resultado deste trabalho, primeiramente no Capítulo 2 são apresentadas algumas propriedades sobre a estrutura do diagrama de Voronoi, já no Capítulo 3 é mostrado o motivo pelo qual se deve utilizar a transformação geométrica no diagrama, além disso, é apresentado o algoritmo de Fortune. No Capítulo 4 é relatado um método para realizar as comparações entre os elementos do diagrama para o caso em que os *sites* possuem peso ou não. A estrutura do diagrama de Voronoi para pontos com peso é exibida no Capítulo 5. No Capítulo 6 é mostrado como a técnica para construir cônicas apresentada em [8] pode ser utilizada para computar a transformação geométrica dos elementos do diagrama, para o caso em que os *sites* possuem um peso não-negativo. Também é comentada uma possível forma de se computarem os vértices do diagrama a partir do resultado do trabalho [6].

2 O DIAGRAMA DE VORONOI PARA PONTOS

Na Seção 2.1 são apresentadas algumas notações básicas que são utilizadas no restante do trabalho, ao mesmo tempo são mostradas algumas características da estrutura do diagrama de Voronoi utilizando exemplos do diagrama construído para um, dois e três pontos, na mesma Seção são realizadas algumas definições e também é provada a existência de algumas propriedades do diagrama.

2.1 Estrutura do Diagrama de Voronoi

Um ponto no plano é um par ordenado (x, y) sendo que $x, y \in \mathbb{R}$, as coordenadas x e y de um ponto p são denotadas respectivamente por p_x e p_y. No diagrama de Voronoi para pontos no plano a métrica de distância utilizada será a distância euclidiana. Denotamos a distância euclidiana entre dois pontos no plano p e q por $dist(p, q)$ onde $dist(p, q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$.

Seja S um conjunto de N pontos no plano, chamados de *sites*, o diagrama de Voronoi de S denotado por $V(S)$ é uma subdivisão do plano em N regiões uma para cada *site* em S, denotamos a região de um *site* $p \in S$ por R_p, os pontos que estão contidos na região R_p são mais próximos de p do que de qualquer outro *site* em S.

Caso $N = 1$, o conjunto S possui somente um *site* p. A região R_p será formada por todo o $\mathbb{R}^2$. Caso $N = 2$, o conjunto S contém dois *sites*, p e q. Neste caso, o diagrama vai possuir duas regiões, sendo elas: R_p e R_q, onde $R_p = \{z \in \mathbb{R}^2 : dist(z, p) \leq dist(z, q)\}$ e $R_q = \{z \in \mathbb{R}^2 : dist(z, q) \leq dist(z, p)\}$. A fronteira entre as duas regiões é dada por $\{z \in \mathbb{R}^2 : dist(z, q) = dist(z, p)\}$, logo a fronteira entre as duas regiões é o bissetor perpendicular do segmento $\overline{pq}$, que é denotado por B_{pq}. O bissetor B_{pq} será uma aresta do diagrama e vai repartir o plano em dois semiplanos. Um semiplano contém p e é denotado por R_{pq} e o outro semiplano contém q e é denotado por R_{qp}. A Figura 2.1 ilustra o diagrama para os *sites* p e q. Somente quando $N = 2$ é verdade que $R_p = R_{pq}$ e $R_q = R_{qp}$. Considerando $N = 3$ e $S = \{p, q, r\}$, o semiplano $R_{pq} = \{z \in \mathbb{R}^2 : dist(z, p) \leq dist(z, q)\}$ se difere da região $R_p = \{z \in \mathbb{R}^2 : dist(z, p) \leq dist(z, q)\} \cap \{z \in \mathbb{R}^2 : dist(z, p) \leq dist(z, r)\}$. Considera-se que as arestas e vértices na fronteira de uma região R_p com outras regiões também fazem parte de R_p, porém não do interior de R_p, com isto pode-se dizer que R_p é formada pelo seu interior e pela fronteira com outras regiões.

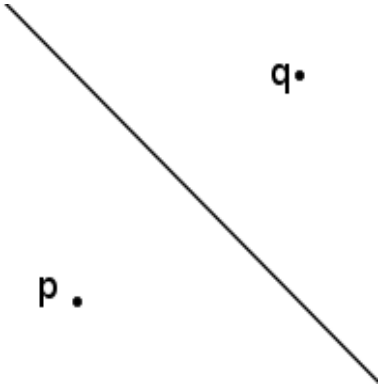

Figura 2.1 – Diagrama de Voronoi para dois *sites*

A Figura 2.1 ilustra que para todo ponto z em R_{pq} é verdade que $dist(z,p) < dist(z,q)$, a observação é análoga para o semiplano R_{qp}.

Uma vez que um novo *site* t seja adicionado ao diagrama a nova região R_t deve ser construída, como R_t deve ser formada pelos pontos mais próximos de t do que p ou q, nota-se que $R_t = \{z \in \mathbb{R}^2 : dist(z,t) \leq dist(z,p)\} \cap \{z \in \mathbb{R}^2 : dist(z,t) \leq dist(z,q)\} = R_{tp} \cap R_{tq}$, as Figuras 2.2, 2.3 e 2.4 ilustram esta igualdade. Além da criação de R_t, as outras duas regiões R_p e R_q devem ser alteradas, pois neste novo diagrama $R_p = R_{pt} \cap R_{pq}$, e $R_q = R_{qt} \cap R_{qp}$.

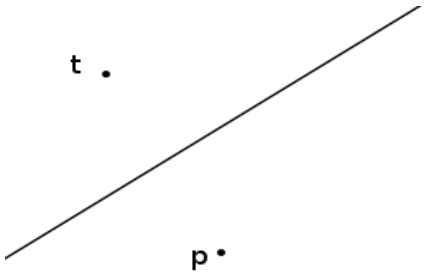

Figura 2.2 – Semiplanos R_{tp} e R_{pt}

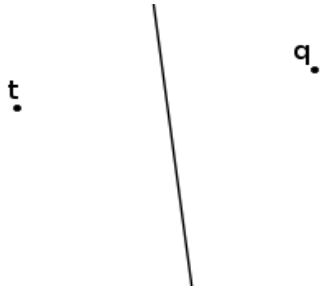

Figura 2.3 – Semiplanos R_{tq} e R_{qt}

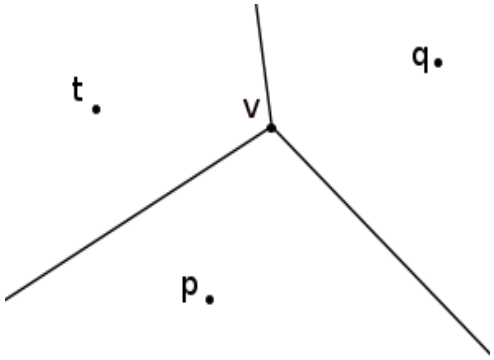

Figura 2.4 – Região $R_t = R_{tp} \cap R_{tq}$ no diagrama para t, p e q

Na Figura 2.4 os três bissetores B_{tq}, B_{tp} e B_{pq} não foram utilizados por completo, na realidade apenas semirretas contidas nos bissetores foram utilizadas para realizar a partição do plano. O bissetor B_{tq} não está sendo utilizado por inteiro, pois um trecho pertence a região do *site* p. A intersecção dos bissetores B_{tq} e B_{tp} é chamada de v, o trecho de B_{tq} que não está sendo utilizado é exatamente a semirreta de B_{tq} que está abaixo de v, analogamente trechos dos outros bissetores também não são utilizados. Pelo fato de que v pertence aos três bissetores ele é equidistante aos três *sites*, o ponto v é chamado de vértice do diagrama. As arestas do diagrama que são semirretas possuem seu extremo em um vértice de $V(S)$. As arestas ainda podem ser representadas por um segmento de reta contido em um bissetor e com seus seus dois extremos em dois vértices, como ilustra a Figura 2.5. Independente de ser reta, semirreta ou segmento de reta uma aresta contida em um bissetor B_{pq} é denotada por e_{pq}.

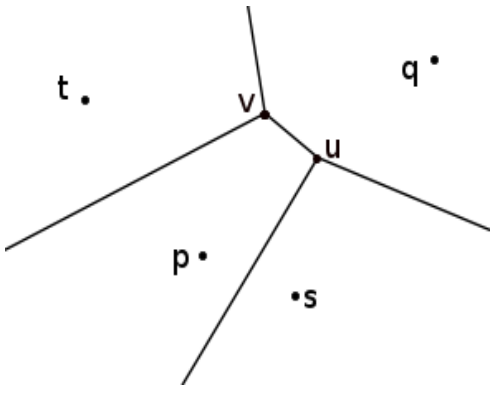

Figura 2.5 – Diagrama para quatro *sites* com segmento de reta como aresta

Apenas um segmento de reta foi necessário para separar as regiões R_p e R_q, pois os pontos do bissetor B_{pq} acima do vértice v pertencem a região R_t, e os pontos do bissetor B_{pq} abaixo do vértice u pertencem a região R_s.

Todos os pontos no plano podem ser classificados de três formas. Seja z um ponto no plano, a função de distância é definida por $d : \mathbb{R}^2 \to \mathbb{R}$ entre z e o *site* mais próximo a z em S por $d(z) = \min_{p \in S} dist(p, z)$.

Se existir apenas um *site* $p \in S$ tal que $d(z) = dist(z, p)$ então z está contido em R_p, se existirem dois *sites* p e q em S tal que $d(z) = dist(z, p) = dist(z, q)$ logo z está contido na aresta e_{pq} que representa a fronteira entre as regiões R_p e R_q, se existir um conjunto $C \subseteq S$ contendo três ou mais *sites* tal que para todo $p \in C$ seja verdade que $d(z) = dist(z, p)$, então z é um vértice de $V(S)$. Uma observação importante a ser feita é que caso z seja um vértice ele é também o centro de um círculo, este círculo vai possuir todos os *sites* de C em sua circunferência, como todos os pontos de uma aresta são equidistantes a dois *sites* p e q, cada aresta deve possuir um ponto c que é o centro de um círculo que possui p e q na sua circunferência, o interior dos círculos com o centro nos pontos c e z é vazio, pois seus raios são definidos por $d(c)$ e $d(z)$ respectivamente. O maior círculo vazio de um ponto c qualquer no plano é denotado por $C_S(c)$, e é útil nas próximas seções.

2.2 Definições e Propriedades

Definição 2.1. *Seja p um site de S, a região de p em $V(S)$ é dada por:* $R_p = \bigcap_{q \in S - \{p\}} R_{pq}$

Como aponta [10] pelo fato de que uma região é formada pela intersecção de semiplanos, toda região do diagrama de Voronoi é convexa, cada semiplano R_{pq} é um conjunto convexo, consequentemente a intersecção de todos estes semiplanos também vai ser um conjunto convexo. Como já foi apresentado na Figura 2.5 uma região pode não ser limitada, por definição sempre existem regiões não limitadas, pois para que uma região R_p seja limitada deve existir um conjunto de *sites* $K \subseteq S$ ao redor de R_p, para que então bissetores fossem utilizados para separar R_p das regiões dos *sites* em K, porém como S é um conjunto finito, alguns *sites* em K pode não ter *sites* ao seu redor. As regiões não limitadas possuem pelo menos duas semirretas como fronteira, já as regiões limitadas devem possuir somente segmentos de reta como fronteira. O Teorema em seguida foi retirado de [2] e mostra que semirretas e segmentos de reta são utilizados como arestas.

Teorema 2.1. *Seja S um conjunto de N sites no plano. Se todos os sites são colineares então $V(S)$ consiste de $N - 1$ linhas paralelas. Caso contrário, $V(S)$ é conectado e suas arestas são segmentos de retas ou semirretas.*

Demonstração. Primeiramente será mostrado que se todos os *sites* forem colineares então $V(S)$ possuirá $N-1$ linhas paralelas como aresta. Seja $r = a + t * u$ onde a é um ponto, t é um parâmetro e u é um vetor, a linha que contém todos os *sites* do conjunto S. Podemos assumir sem perda de generalidade que u possui apenas um vetor perpendicular chamado v, pois os vetores paralelos a v podem ser encontrados através da multiplicação por um escalar. O vetor do bissetor de qualquer par de *sites* (p_i, p_j) com $i \neq j$ onde $p_i, p_j \in S$ será perpendicular ao vetor u, visto que o segmento $\overline{p_i q_j}$ está contido em r, portanto o vetor do bissetor de qualquer par de *sites* deve ser v ou αv onde $\alpha \in \mathbb{R}$, logo os bissetores são paralelos.

A demonstração da segunda parte do Teorema foi retirada de [2]. Primeiramente será mostrado que as arestas de $V(S)$ serão segmentos ou semirretas, ou seja, nunca será uma linha inteira. Suponha por contradição que existe uma aresta e de $V(S)$ que é uma linha inteira. Deixe e estar na fronteira de duas regiões R_{p_i} e R_{p_j}. Deixe $p_k \in S$ ser um *site* não colinear aos *sites* p_i e p_j. O bissetor dos sites p_j e p_k não é paralelo a e, logo intercepta e. Então a parte de e que está contida em R_{p_k,p_j} não pode estar na fronteira de R_{p_j}, pois está mais próxima de p_k do que p_j, uma contradição.

Prova de que $V(S)$ é conectado. Caso o diagrama não fosse conectado existiria uma célula R_{p_i} separando o plano. Dado o fato de que uma célula de Voronoi é convexa, R_{p_i} seria limitada por duas linhas paralelas e infinitas, isso é impossível, visto que foi provado que não existem linhas infinitas. □

Uma vez que cada semiplano é gerado por um bissetor e pela definição 2.1 uma região é formada pela intersecção de $N-1$ semiplanos, todos os semiplanos utilizados para formar uma região são gerados por um bissetor diferente, levando adiante este argumento poderia ser concluído que são necessários $N-1$ bissetores para formar uma região, e consequentemente $\frac{N*(N-1)}{2} = \theta(N^2)$ bissetores para formar o diagrama, porém nem todos os $N-1$ semiplanos utilizados na definição são realmente necessários para formar uma região, a Figura 2.5 mostra que o bissetor B_{ts} não é necessário para formar a região R_t pois $R_{tp} \cap R_{tq} = R_{tp} \cap R_{tq} \cap R_{ts}$, ou seja, o semiplano R_{ts} gerado por B_{ts} não é realmente útil neste caso. O próximo Teorema mostra que $V(S)$ possui somente $O(N)$ arestas e $O(N)$ vértices. O Teorema 2.2 e sua demonstração foram retirados de [2].

Teorema 2.2. *Para $N \geq 3$, o número de vértices no Diagrama de Voronoi de um conjunto com N sites no plano é de no máximo $2N-5$ e o número de arestas é no máximo $3N-6$.*

Demonstração. A demonstração utiliza a característica de Euler, porém o diagrama de Voronoi não é exatamente um grafo planar, pois existem algumas arestas que possuem vértice somente em um extremo. Para contornar isso será criado um vértice extra V_∞ "no infinito", e todos extremos de arestas que não possuem vértice serão ligados a este vértice.

Seja $V(S)$ o diagrama de Voronoi para um conjunto S com N *sites*, possuindo N_v vértices, N_e arestas e N *sites*, como o número de faces do diagrama é exatamente o número de *sites*, e adicionamos o novo vértice V_∞, da característica de Euler temos que:

$$(N_v + 1) - N_e + N = 2$$

$$N_e = N + N_v - 1 \tag{2.1}$$

Levando em consideração o diagrama com o novo vértice V_∞ cada aresta está ligada a extamente 2 vértices. Como cada vértice possui grau de no minímo 3, sabendo que a soma dos graus é igual a duas vezes o número de arestas e que portanto a soma dos graus é no mínimo $3(N_v + 1)$, temos:

$$2N_e \geq 3(N_v + 1)$$

Substituindo a equação 2.1 temos:

$$2(N + N_v - 1) \geq 3(N_v + 1)$$

De onde tiramos o limite para o número de vértices:

$$N_v \leq 2N - 5 \tag{2.2}$$

Da equação 2.2 temos que o pior caso para o número de vértices é quando $N_v = 2N - 5$, levaremos em consideração o pior caso e substituindo-o na equação 2.1 temos que:

$$N_e = N + 2N - 5 - 1 = 3N - 6 \tag{2.3}$$

Porém a equação 2.3 nos da o pior caso para o número de arestas, ou seja, quando o número de vértices também é igual ao pior caso, na realidade o número de arestas é limitado por:

$$N_e \leq 3N - 6 \tag{2.4}$$

□

Como já foi mostrado a complexidade de espaço de $V(S)$ é linear, e portanto não necessariamente todos os bissetores são realmente necessários para a formação da região de um *site*, o próximo Teorema e sua demonstração retirados de [2] apontam quais bissetores são utilizados e quais intersecções dos bissetores são vértices do diagrama.

Teorema 2.3. *Para o diagrama de Voronoi $V(S)$ as seguintes afirmações são validas:*

1. *Um ponto v é um vértice de $V(S)$ se e somente se $C_S(v)$ contém três ou mais sites na sua circunferência.*

2. *O bissetor dos sites p_i e p_j define uma aresta de $V(S)$, se e somente se existe um ponto q que pertence ao bissetor de p_i e p_j, tal que $C_S(q)$ contém exatamente p_i e p_j na sua circunferência e nem um outro site.*

Demonstração.

1. Seja v um ponto do plano onde $C_S(v)$ contém três ou mais *sites* na sua circunferência, deixe p_i, p_j e p_k serem três desses sites, como o interior de $C_S(v)$ é vazio então v deve estar na fronteira das regiões R_{p_i},R_{p_j} e R_{p_k}, logo v deve ser vértice de $V(S)$.

   Todo vértice v de $V(S)$ está no extremo de no minímo três arestas e consequentemente na fronteira de no minímo três regiões R_{p_i},R_{p_j} e R_{p_k}. Logo v deve ser equidistante aos *sites* p_i, p_j e p_k e não pode haver nenhum *site* mais perto de v, pois se isso ocorresse R_{p_i},R_{p_j} e R_{p_k} não se encontrariam em v, então o interior do círculo com p_i, p_j e p_k na sua circunferência não contém nenhum outro site.

2. Suponha que exista um ponto q com a propriedade descrita no segundo caso do Teorema. Uma vez que $C_S(q)$ não contém nenhum *site* no seu interior e apenas p_i e p_j estão em sua circunferência, temos que $dist(q, p_i) = dist(q, p_j) \leq dist(q, p_k)$ para todo $p_k \in S$. Portanto q está em uma aresta ou em um vértice de $V(S)$. O primeiro caso do Teorema implica que q não pode ser um vértice. Consequentemente, q está contido em uma aresta de $V(S)$ definida por p_i e p_j.

   Deixe o bissetor entre p_i e p_j definir uma aresta e do diagrama, visto que e é fronteira de apenas duas regiões diferentes, e todos os pontos em e são equidistantes a p_i e p_j, então todo ponto $q \in e$ possui $C_S(q)$ com p_i e p_j na sua circunferência.

□

A definição do diagrama de Voronoi utilizada neste trabalho é a definição sucinta apresentada em [3], seja S um conjunto de *sites* $V(S) = \{z \in \mathbb{R}^2, p \text{ e } q \in S, p \neq q : d(z) = dist(z,p) = dist(z,q)\}$.

A grande maioria das notações utilizadas nesta seção foram retiradas de [3], porém também foram utilizadas notações de [2].

3 O ALGORITMO DE FORTUNE

O algoritmo escolhido para este trabalho para realizar a construção do diagrama de Voronoi foi o algoritmo de Fortune, o principal motivo para esta decisão é o fato de este algoritmo poder ser modificado para o caso em que os *sites* são formados por pontos com peso ou segmentos de reta.

O algoritmo de Fortune possui duas interpretações. Em [3] o algoritmo foi apresentado pela primeira vez, apesar disso a maioria das implementações deste algoritmo para o caso em que os *sites* são representados por pontos, condizem com a interpretação do algoritmo de Fortune proposta em [2], acredita-se que isso se deve ao fato de que esta nova interpretação possui maior simplicidade, no entanto [2] não aponta uma modificação possível para quando os *sites* são formados por pontos com peso, além disso sua modificação para quando os *sites* são formados por segmentos de reta não é tão bem elaborada quanto a modificação apresentada por Fortune.

Pelos motivos já comentados a interpretação apresentada e implementada neste trabalho será do algoritmo de Fortune proposto por Fortune, consequentemente os lemas e teoremas apresentados e suas respectivas demonstrações foram retiradas de [3].

3.1 Introdução ao algoritmo de Fortune

O algoritmo de Fortune se utiliza da técnica de Varredura do Plano (*Line Sweep*[5]). A Varredura do Plano é um paradigma para a construção de algoritmos de natureza geométrica, a técnica consiste em:

- Varrer o plano com uma linha L.
- Manter um *status* de L que representa os objetos envolvidos no problema cujo termino do processamento depende dos futuros eventos que vão ser encontrados por L. O *status* de L pode ser visto como o conjunto de objetos sendo interceptados por L.
- Os eventos ocorrem quando L está localizada sobre algum local específico do plano. Quando um evento ocorre deve-se processar este evento de acordo com a solução do problema e se utilizando do *status* de L disponível no momento. Durante o processamento

[5] A técnica de *Line Sweep* é utilizada em vários algoritmos de natureza geométrica como o algoritmo do Capítulo 2 do livro [2].

de um evento deve-se também atualizar o *status* de L para ser utilizado nos próximos eventos.

A direção em que a varredura do plano é feita não é imposta pelo paradigma, assim como a inclinação de L. Neste algoritmo por escolha arbitrária a varredura será feita de baixo para cima e L será uma reta horizontal.

Sejam p e q dois pontos no plano, diz-se que $p < q$, se $p_y < q_y$ ou $p_y = q_y$ e $p_x < q_x$. Uma reta, semirreta ou segmento de reta l está abaixo de um ponto p se existe um ponto $q \in l$ tal que $p_x = q_x$ e $q_y < p_y$.

Um primeiro ponto do algoritmo a ser notado é que não é necessário uma varredura completa do plano, ou seja, em nenhum momento L precisa estar sobre algum ponto z do plano se em z nada de relevante acontece, considera-se a movimentação de L apenas por pontos que possuem alguma importância para a construção do diagrama, estes pontos são as posições do plano onde acontecem os eventos.

Tendo em vista que inicialmente se tem apenas um conjunto de N *sites* no plano, alguns dos locais obrigatórios para o acontecimento de um evento são as posições do plano onde se encontra um *site*, a constatação da existência de um *site* é de suma importância para a construção da estrutura do diagrama, pois para a construção de um bissetor que realiza a divisão do plano dois *sites* são necessários. Portanto quando a linha varredora L está sobre um *site* um evento ocorrerá, este tipo de evento é chamado de Evento de *Site*(ES).

O processamento de um ES para um *site* p deve consistir na construção dos bissetores entre p e os outros *sites* já encontrados, ou seja, é levado em consideração as informações contidas no *status* de L.

Considerando a Figura 3.1, quando L está sobre a o *status* ainda não contém nenhuma informação, agora o diagrama possui somente o *site* a e a região $R_a = \mathbb{R}^2$, a única ação a ser tomada é a inserção de a e R_a no *status* de L.

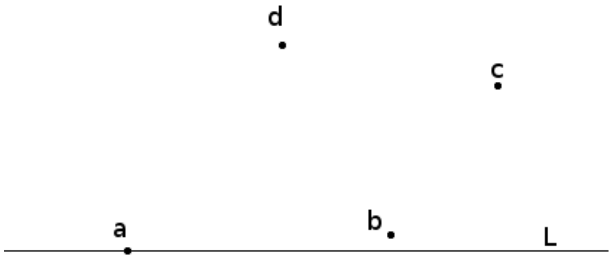

Figura 3.1 – ES para o *site* a

Quando L avança para o próximo ES, o *site* b é encontrado, analisando o *status* disponível nota-se que b está contido em R_a, consequentemente R_a deve ser igualmente particionada entre a e b, portanto retiramos R_a do *status*, e o particionamento de R_a vai ser realizado pelo bissetor B_{ab}, como mostra a Figura 3.2 agora existem duas regiões no diagrama R_a e R_b, as duas regiões e a aresta e_{ab} devem ser inseridas no *status*.

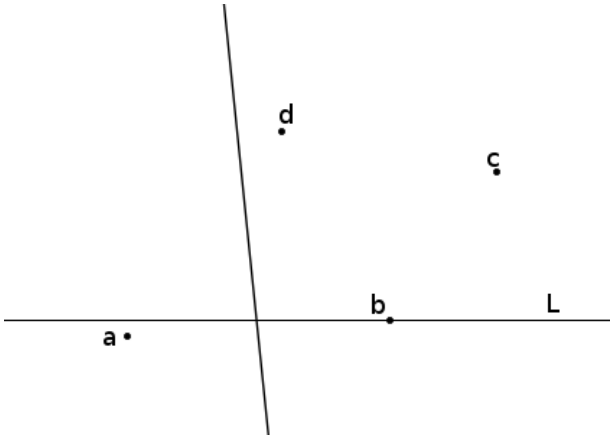

Figura 3.2 – L sobre o *site* b

Durante o processamento do ES para c (Figura 3.3) percebe-se que c está contido em R_b, retiramos R_b do *status* e particionamos a região igualmente utilizando o bissetor B_{bc}, agora constata-se que B_{bc} intercepta e_{ab} no ponto u, portanto o bissetor B_{ab} não será mais utilizado por completo, logo e_{ab} que é formada por B_{ab} deve ser retirada do *status*, as arestas e_{ab} e e_{bc} são formadas por semirretas contidas em seus respectivos bissetores com extremo em u e localizadas abaixo de u, agora c vai estar contido em R_a, portanto o bissetor B_{ac} será dado origem a aresta e_{ac} que interceptará os outros dois bissetores no ponto u, agora as arestas e_{ab}, e_{ac} e e_{bc} estão se interceptando no ponto u, logo u é um vértice do diagrama.

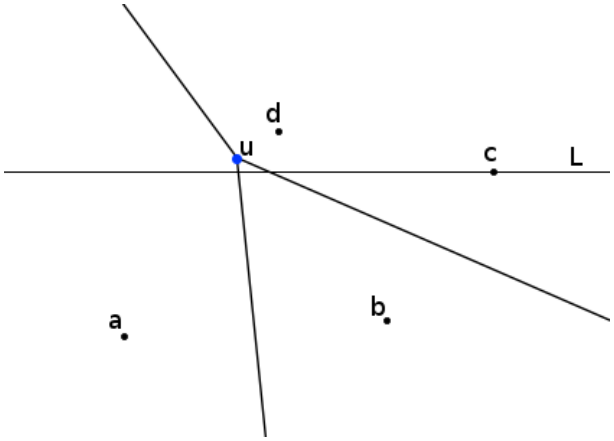

Figura 3.3 – Diagrama após o processamento do ES para o *site* c

Durante o processamento do ES para c foi realizado uma re-estruturação quase que por completo do diagrama. Agora o processamento do ES para d deverá ser realizado, na Figura 3.3 em que o vértice u e parte das três arestas estão muito próximos de d, é notável o fato de que u e parte das três arestas estão contidos no interior da futura região de d, consequentemente serão retiradas as três arestas e o vértice u do diagrama e do *status* e será refeita a construção do diagrama do zero, pois nenhuma aresta pode continuar sendo utilizada, este caso pode acabar gerando complexidade no processamento de um ES, pois poderiam existir mais arestas no diagrama contidas no interior da futura região de d e todas deveriam ser retiradas.

A situação descrita no processamento do ES para d aconteceu porque os bissetores B_{bc} e B_{ac} foram criados desconsiderando a existência de d, caso d fosse levado em conta poderia ser notado que partes dos bissetores estavam contidas no interior da futura região de d, porém d não pode ser levado em consideração pois L ainda não esteve sobre d, logo ainda não se sabe da existência de d.

O trabalho de retirada de u e das arestas poderia ser evitado, uma vez que u e as arestas fossem criados somente depois que se tivesse certeza que eles não estão contidos em uma região, é necessário encontrar d durante a varredura antes de u e as arestas serem criados, isso é impossível, pois no diagrama o vértice u é o menor ponto de R_d, por este motivo uma transformação geométrica do diagrama de Voronoi será feita, a transformação deve garantir que o primeiro ponto de uma região a ser encontrado será o *site* dono da região.

Na Figura 3.5 é apresentado o diagrama para os *sites* a, b, c e d, nota-se que o vértice u após a transformação se encontra acima de d, e d é o primeiro ponto da região R_d a ser encontrado por L. Levando em consideração o diagrama após a transformação geométrica da Figura 3.5, caso a criação de u e das arestas ligadas em u seja adiada até que L esteja sobre u,

pode-se verificar a existência de d e cancelar a criação de u e das arestas, pois já é sabido sobre a existência do *site* d.

Para evitar a complexidade de casos como o processamento do ES de d, primeiro é computado a transformação geométrica do diagrama $V^*(S)$, e depois então o diagrama Voronoi $V(S)$.

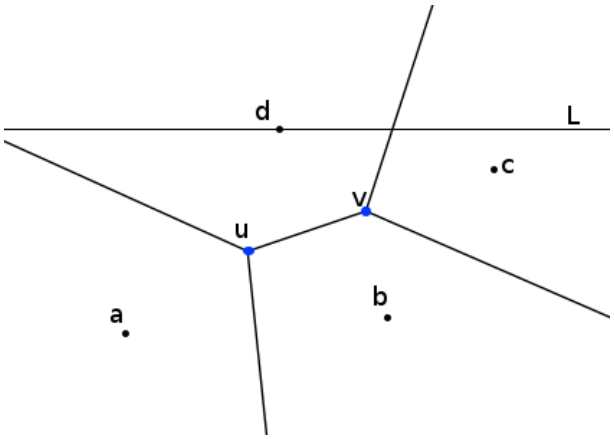

Figura 3.4 – Diagrama após o processamento do ES para o *site* d

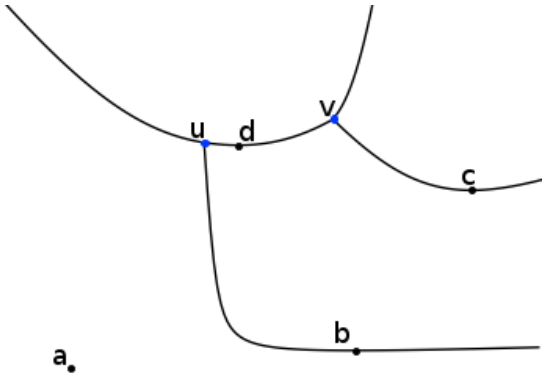

Figura 3.5 – Transformação geométrica do diagrama da Figura 3.4

3.2 Transformação geométrica

A transformação geométrica é feita através do mapeamento $* : \mathbb{R}^2 \to \mathbb{R}^2$, onde $*(z) = (z_x, z_y + d(z))$, seja S um conjunto com N *sites*, $*$ possui as seguintes propriedades para os *sites* e $V(S)$: Seja z um ponto no plano, $*(z) = z'$ onde z' é o maior ponto na circunferência de $C_S(z)$. O mapeamento é contínuo. Seja p um *site* de S é verdade que $*(p) = p$, visto que $d(p) = 0$. Seja l uma linha vertical, para todo ponto $l_i \in l$ nota-se que $*(l_i) \in l$. Seja B um

subconjunto do plano, $*(B)$ é denotado por B^*, ou seja, B^* é igual ao mapeamento aplicado a todo ponto em B, por exemplo: $*(V(S)) = V^*(S)$. Em algumas situações é utilizado também o mapeamento auxiliar $*_p : \mathbb{R}^2 \to \mathbb{R}^2$, onde $*_p(z) = (z_x, z_y + dist(z, p))$, para todo ponto z em R_p é valido que $*(z) = *_p(z)$.

Primeiro é verificado como a transformação se comporta no diagrama, para isso é necessário analisar a estrutura da forma geométrica das regiões, dos bissetores e dos vértices após aplicarmos $*$ em cada um deles. Após isso é possível verificar algumas propriedades da transformação.

Como $d(z) \geq 0$ para qualquer $z \in \mathbb{R}^2$, consequentemente $*(z) = z'$ e $z' \geq z$, esta observação é válida também para $*_p$ e é utilizada para analisar o comportamento da transformação de um bissetor.

Para um conjunto inicial $S = \{p, q\}$, seja B_{pq} o bissetor dos *sites* p e q, primeiro é tratado o caso em que B_{pq} não é vertical, para isso é necessário que $p_y \neq q_y$, neste caso considera-se que $p_y > q_y$, com isso todos os pontos em B_{pq} são mapeados para locais diferentes e resultam em uma hipérbole, além disso como cada ponto sofrerá alteração apenas na coordenada y o mapeamento $*_p$ é injetivo neste caso, como mostra o próximo Lema.

Lema 3.1. *Suponha que l é uma linha não vertical. Então $*_p$ é injetivo quando restrito a l e $*_p(l)$ é uma hipérbole.*

Demonstração. Para provar que $*_p$ é injetivo temos que provar que:

$$*_p(u) = *_p(v) \to u = v, u \text{ e } v \in l \tag{3.1}$$

Comparando as coordenadas:

$$\begin{gathered} *_p(u) = *_p(v) \\ (u_x, u_y + dist(p, u)) = (v_x, v_y + dist(p, v)) \end{gathered} \tag{3.2}$$

Notamos que $u_x = v_x$, logo $u_y = v_y$, pois l não é vertical, portanto $*_p$ é injetivo.

Como l não é vertical então podemos considerar que l é representada por $y = mx + b$, aplicando $*_p$ em l temos:

$$*_p(l) = \{(x, z) : z = mx + b + \sqrt{(mx + b - p_y)^2 + (x - p_x)^2}\} \tag{3.3}$$

Colocando $mx + b$ no lado esquerdo da igualdade e elevando os dois lados ao quadrado chegamos em:

$$(z - (mx + b))^2 = (mx + b - p_y)^2 + (x - p_x)^2 \tag{3.4}$$

Da equação 3.4 temos uma seção cônica, pois 3.4 representa uma curva definida por uma equação quadrática de duas variáveis. Agora consideramos o conjunto de pontos:

$$H = \{(x, y + dist((x, y), r)) : (x, y) \in B_{pq}\} \tag{3.5}$$

onde $r \in l$ é o ponto mais próximo de p. H vai ser a união de duas semirretas com extremo em r. As duas semirretas que formam H serão assíntotas a $*_p(l)$, visto que à medida em que $(x, y) \in B_{pq}$ está se afastando de p, $dist((x, y), r)$ se aproxima de $dist((x, y), p)$, consequentemente $*_p(l)$ é uma cônica e possui duas assíntotas, ou seja, uma hipérbole. □

Quando B_{pq} é uma reta vertical, identifica-se que $p_y = q_y$, e $*_p(B_{pq})$ não resulta em uma hipérbole visto que todo ponto em B_{pq} possui o mesmo valor de coordenada x, o primeiro caso do Lema seguinte apresenta a forma geométrica de B_{pq}, além disso o primeiro e o segundo caso são úteis para a verificação do resultado do mapeamento dos pontos em uma região.

Lema 3.2. *Seja l uma linha vertical em l_x.*

1. *Se $p \notin l$ então $*_p$ é injetivo em l, e $*_p(l)$ é uma semirreta acima do ponto (l_x, p_y).*
2. *Se $p \in l$ então $*_p(l)$ vai resultar em uma semirreta com o extremo no site p, todos os pontos abaixo de p serão mapeados para p, e $*_p$ é injetivo para todos os pontos acima de p.*

Demonstração.

1. Obviamente para todo ponto $z \geq (l_x, p_y)$ temos $*_p(z) > (l_x, p_y)$. Consideramos um ponto $z < (l_x, p_y)$, como $dist(p, z)$ é a hipotenusa do triângulo retângulo formado pelos pontos z, p e (l_x, p_y), temos $dist^2(p, z) = (z_y - p_y)^2 + (l_x - p_x)^2$, logo $dist^2(p, z) > (z_y - p_y)^2$, consequentemente $*_p(z) > (l_x, p_y)$.

   Agora suponha que dois pontos distintos r e s em l sejam mapeados por $*_p$ para o mesmo ponto q. Logo:

$$q_y = dist(p, s) + s_y \tag{3.6}$$

$$q_y = dist(p, r) + r_y \tag{3.7}$$

   De 3.6 temos:

$$\begin{aligned} q_y &= \sqrt{(l_x - p_x)^2 + (p_y - s_y)^2} + s_y \\ (q_y - s_y)^2 - (p_y - s_y)^2 &= (l_x - p_x)^2 \\ q_y^2 - p_y^2 + 2s_y(p_y - q_y) &= (l_x - p_x)^2 \end{aligned} \tag{3.8}$$

Fazendo os mesmos passos para 3.7 chegamos em:

$$\begin{aligned} q_y &= \sqrt{(l_x - p_x)^2 + (p_y - r_y)^2} + r_y \\ (q_y - r_y)^2 - (p_y - r_y)^2 &= (l_x - p_x)^2 \\ q_y^2 - p_y^2 + 2r_y(p_y - q_y) &= (l_x - p_x)^2 \end{aligned} \tag{3.9}$$

Igualando 3.8 e 3.9 temos:

$$q_y^2 - p_y^2 + 2r_y(p_y - q_y) = q_y^2 - p_y^2 + 2s_y(p_y - q_y)$$

Retirando $q_y^2 - p_y^2$ e dividindo os dois lados da equação por $2(p_y - q_y)$ concluímos:

$$r_y = s_y$$

Portanto para que $*_p(r) = *_p(s)$ é necessário que $r = s$, então $*_p$ é injetiva nesse caso.

2. Quando fazemos $*_p(z)$ para um ponto z tal que $p_x = z_x$ e $p_y \geq z_y$, temos o seguinte ponto resultante:

$$\begin{aligned} *(z) &= (z_x, z_y + d(z)) \\ &= (z_x, z_y + dist(z, p)) \\ &= (z_x, z_y + \sqrt{(p_x - z_x)^2 + (p_y - z_y)^2}) \\ &= (z_x, z_y + (p_y - z_y)) \\ &= (z_x, p_y) \end{aligned}$$

Ou seja, todo ponto abaixo de p em l vai ser mapeado para p.

Seja r e s dois pontos de l acima de p. $*p$ é injetivo acima de p, ou seja, se $*p(r) = *p(s)$ então $r = s$, uma vez que l é vertical é verdade que $r_x = s_x$, assumindo que seja verdade que $*p(r) = *p(s)$, tem-se:

$$\begin{aligned} r_y + dist(r, p) &= s_y + dist(s, p) \\ r_y + r_y - p_y &= s_y + s_y - p_y \\ 2r_y &= 2s_y \end{aligned}$$

Portanto neste caso para dois pontos serem mapeados para o mesmo local os pontos devem ser iguais. Consequentemente $*_p$ é injetivo nesse caso.

□

As formas geométricas das arestas devem ser as mesmas formas geométricas dos bissetores, como mostra o próximo Lema.

Lema 3.3. *Seja e_{pq} uma aresta de $V(S)$, então e^*_{pq} é um trecho de hipérbole ou uma semirreta.*

Demonstração. Uma vez que e_{pq} está contida em B_{pq}, e e_{pq} possui as mesmas propriedades de B_{pq}, ou seja, para todo ponto z em e_{pq} temos $d(z) = d_q(z) = d_p(z)$, pelos Lemas 3.1 e 3.2 e_{pq} é um trecho de hipérbole ou uma semirreta. □

O mapeamento deve garantir que o primeiro ponto a ser encontrado em uma região R^*_p, seja o próprio *site* p, isto é necessário para ajudar a evitar a situação comentada anteriormente na Subseção 3.1. O próximo Lema esclarece como os pontos dentro de uma região são mapeados.

Lema 3.4. *O site p é o único menor ponto de R^*_p.*

Demonstração. Pelo primeiro caso do Lema 3.2 todo ponto que está em uma reta vertical diferente de $x = p_x$, será mapeado por $*_p$ para um ponto acima de p. Pelo segundo caso do Lema 3.2 todo ponto que está na reta $x = p_x$ será mapeado para um ponto maior ou igual a p, logo o *site* p é o único menor ponto de R^*_p. □

As Figuras 3.6 e 3.7 mostram o resultado do mapeamento. Todo vértice é mapeado para o maior ponto na circunferência de C_S. Cada *site* é o ponto mais abaixo de uma hipérbole formada pela aresta mais abaixo em sua região, isso é verdade para todos os *sites*, exceto para o *site* mais abaixo em S.

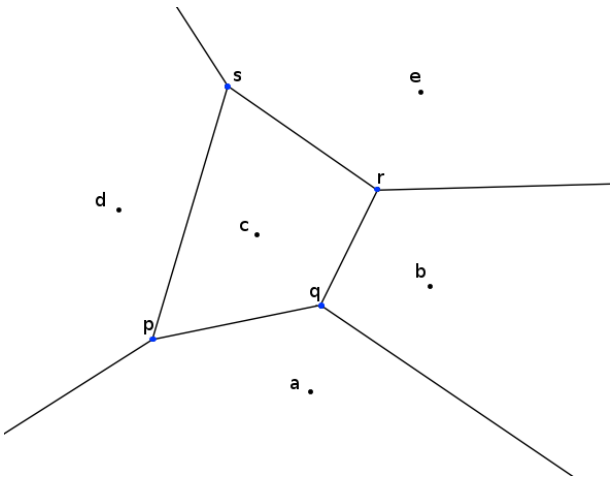

Figura 3.6 – Diagrama de Voronoi

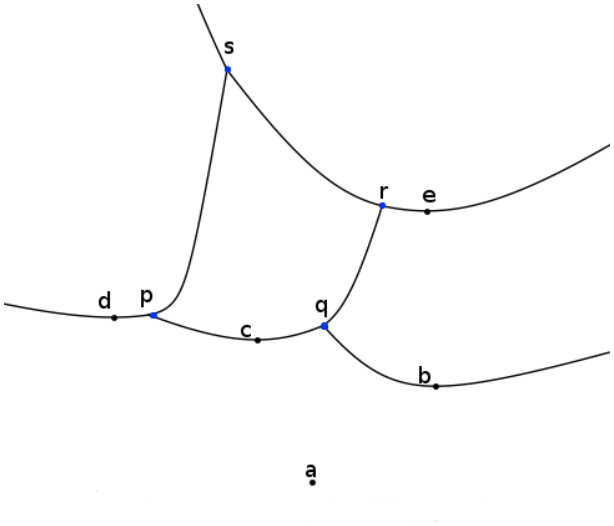

Figura 3.7 – Transformação geométrica do diagrama de Voronoi da Figura 3.6

Uma vez que será construído primeiro $V^*(S)$, é necessário garantir que é possível construir $V(S)$ a partir de $V^*(S)$. Quando um ponto z é mapeado, deve-se garantir que é possível colocar z em sua posição original, para isso o fato de que $*$ é injetivo em $V(S)$ é útil, logo os pontos de regiões diferentes nunca são mapeados para o mesmo lugar. Porém em apenas um caso $*$ não é injetivo, quando B_{pq} for uma reta vertical $*$ não é injetivo nos pontos abaixo de p_y, mas este caso pode ser tratado separadamente.

Lema 3.5. *O mapeamento $*$ é injetivo em $V(S)$.*

Demonstração. Seja l uma linha vertical, sabemos que $*$ é contínuo portanto preserva a ordem dos pontos em l.

Notemos que $R_p \cap l$ representa um segmento ou uma semirreta contida em R_p, portanto todos os pontos de R_p podem ser representados por $R_p \cap l$, como $* = *_p$ em R_p, pelo Lema 3.2 temos que $*$ não é injetivo em $R_p \cap l$ na semirreta abaixo de p somente se $p \in (R_p \cap l)$. Como p não pode ser o maior ponto de $R_p \cap l$, todos os pontos de $R_p \cap l$ nunca serão mapeados para o mesmo local.

Uma vez que $V(S) \cap R_p$ resulta nos pontos pertencentes as arestas e vértices na fronteira de R_p, $V(S) \cap R_p \cap l$ resulta no conjunto de pontos na fronteira de R_p pertencentes a linha vertical l.

Caso não exista nenhuma aresta de R_p abaixo de p então $(V(S) \cap R_p \cap l)$ resulta em pontos pertencentes a fronteira de R_p acima de p, logo $*$ é injetivo neste caso.

Caso exista uma aresta de R_p abaixo de p, essa aresta não pode ser vertical, consequentemente $V(S) \cap R_p \cap l$ resulta em apenas um ponto abaixo de p, logo $*$ também é injetivo neste

caso. □

Quando existe um vértice no diagrama é necessário saber qual vai ser o resultado do mapeamento das arestas ligadas em cada vértice, o Lema a seguir descreve o mapeamento de arestas, além disso, ajuda a identificar os possíveis locais de um ponto em $V^*(S)$.

Lema 3.6. *Suponha que v é um vértice de $V(S)$. Seja r e s os sites na circunferência de $C_S(v)$ no sentido anti-horário e horário de v^*, respectivamente.*

1. *Se v^* não é um site, então a aresta e^*_{rs} se extende para cima a partir de v^*, e as outras arestas se extendem abaixo.*

2. *Se v^* é um site, então as arestas $e^*_{rv^*}$ e $e^*_{v^*s}$ se extendem para cima a partir de v^*, e as outras arestas se extendem abaixo.*

Demonstração. Toda aresta e_{pq} incidente ao vértice v é um segmento do bissetor B_{pq} de dois sites p e q que estão presentes na circunferência de $C_S(v)$. O bissetor B_{pq} é dividido em duas semirretas por v, e_{pq} está contido na semirreta que está interceptando A_{pq}, onde A_{pq} é o arco de $C_S(v)$ que contém somente p e q e nenhum outro site. Nós estabelecemos que e^*_{pq} se extende para cima a partir de v^* se e somente se $v^* \in A_{pq}$.

Primeiramente suponha que $p \neq v^*$ e $q \neq v^*$. Como sabemos v divide B_{pq} em duas semirretas, sendo elas h e h'. Seja h a semirreta que intercepta A_{pq} e h' a outra semirreta. Se $p_y \neq q_y$, consideramos que $p_y > q_y$, então $*_p(B_{pq})$ é uma hipérbole que passa por v^* com o ponto minímo em p, onde $*(h)$ se extende para cima a partir de v^*, e $*(h)$ se extende para baixo a partir de v^*.

Se $p_y = q_y$, então $*_p(B_{pq})$ é uma seção da semirreta vertical acima de v, $*_p(h)$ é a semirreta acima de v^*, e $*_p(h')$ é um segmento a partir do ponto localizado no meio do segmento $\overline{pq}$ até o ponto v^*

Nos dois casos, se A_{pq} contém v^*, então $e_{pq} \subseteq h$, $*p = *$ em e_{pq}, e e^*_{pq} se extende para cima a partir de v^*, se A_{pq} não contém v^*, então $e_{pq} \subseteq h'$, temos que $*p = *$ em e_{pq}, e e^*_{pq} se extende para abaixo a partir de v^*. Sem perda de generalidade assuma que $p = v^*$, então e^*_{pq} é uma seção da hipérbole com ponto minímo p, e como $p = v^*$ sabemos que A_{pq} contém v^*, consequentemente e^*_{pq} se extende para cima a partir de v^*. □

A partir do Lema 3.6 o mapeamento de um ponto z do plano pode ser classificado em um dos seguintes casos:

1. $*(z)$ pode estar contido no interior de uma região R_p^*, caso z não esteja contido em nenhuma aresta ou vértice de $V(S)$.

2. $*(z)$ pode estar contido em uma aresta e_{pq}^* e não ser um vértice nem um site, caso z esteja presente na aresta e_{pq}.

3. $*(z)$ pode estar presente no ponto minímo de uma região R_z^* e de uma aresta e_{zp}^*, e $*(z)$ é um *site*, e não é um vértice.

4. $*(z)$ é um vértice mas não um *site*, z possui no minímo duas arestas incidentes pelo lado de baixo, e exatamente uma aresta incidente pelo lado de cima, isso pode ser constatado pelo caso 1 do Lema 3.6.

5. $*(z)$ é um vértice e um *site*, com exatamente duas arestas incidentes pelo lado de cima, e uma aresta incidente pelo lado de baixo, isso pode ser constatado pelo caso 2 do Lema 3.6.

O inverso do mapeamento $*$ é denotado por $*^{-1}$, quando é aplicado $*^{-1}$ em um ponto p do plano, existem dois casos possíveis, se p é um *site* $*^{-1}(p)$ resulta em uma semirreta com extremo em p e que se extende para baixo, caso contrário é verdade que $*^{-1}(*(p)) = p$

3.3 Algoritmo de Fortune para construir $V^*(S)$ e $V(S)$.

Primeiramente é apresentado o algoritmo proposto em [3] para a construção de $V^*(S)$, logo após é mostrado sua corretude, e então o Teorema 3.3 mostrará como construir $V(S)$.

Deve-se notar que em $V^*(S)$ um *site* é o menor ponto de uma região, os vértices de cada região são encontrados após ser conhecido a existência do *site* dono da região, porém é possível saber da existência de um vértice de uma região quando encontramos um *site*.

A varredura feita por L é realizada de baixo para cima, como L deve se mover apenas por pontos que possuem alguma relevância, é necessário utilizar uma fila de eventos Q, tal que Q vai possuir as posições onde L deve estar posicionada.

Dado que um ES deve ser um ponto relevante para a construção do diagrama, assim L deve estar sobre cada *site*, portanto inicialmente Q deve conter todos os *sites*.

Seja p um *site* de S, considerando que será processado o ES para p, considerando ainda que dois *sites* q e s já foram encontrados e seus respectivos ESs já foram processados, e que consequentemente existe um bissetor B_{qs} para realizar a partição do plano, e sem perda de

generalidade assume-se que p esteja contido em R_q, logo vamos criar o bissetor B_{pq} para realizar a partição da atual região R_q, caso p,q e s não sejam colineares os bissetores B_{pq} e B_{qs} vão se interceptar em algum ponto v do plano, v pode ser um vértice de $V(S)$, como está sendo computando $V^*(S)$ ao invés de $V(S)$, não vamos realizar a criação do vértice v^* durante o processamento do ES para p, pois L só estará sobre v^* quando estiver sobre o último ponto de $C_S(v)$, assim sendo a criação de v^* deve entrar na fila de eventos, e quando L estiver sobre v^* ocorrerá um Evento de Vértice(EV), enquanto não ocorre o EV para v^*, o ponto v^* é apenas uma intersecção.

Um EV pode ser cancelado, quando o ES de um *site* p ocorre os EVs de vértices contidos em R_p são cancelados.

Para a simplificação da apresentação do pseudocódigo e da prova do Teorema 3.1, um bissetor B^*_{pq} dos *sites* p e q será particionado em dois arcos hiperbólicos C^-_{pq} e C^+_{pq}, que podem ser chamados de fronteira de uma região, levando em conta que $p > q$, C^-_{pq} representa os pontos de B^*_{pq} à esquerda e acima de p, e C^+_{pq} representa os pontos de B^*_{pq} à direita e acima de p. Quando a notação C_{pq} for utilizada não é importante de qual arco estamos falando, ou é possível determinar o arco em questão através do contexto.

O Algoritmo 1 mostra como os EVs e ESs são processados. Após a apresentação do algoritmo os casos de degeneração são discutidos.

Algoritmo 1: Construção de $V^*(S)$

Entrada: Um conjunto S com $n \geq 1$ *sites*, com somente um menor *site*.

Saída: Os bissetores e vértices de $V^*(S)$.

Estrutura de Dados:

- Q : Uma fila de prioridade onde os eventos serão mantidos em ordem crescente. Além de sua localidade no plano cada evento possui uma flag para apontar se o mesmo é um ES ou EV. Q pode conter eventos no mesmo local, neste caso a ordem dos eventos é irrelevante.
- T : Uma lista que representa uma sequência de arcos hiperbólicos e de regiões. Uma região pode aparecer mais de uma vez em T ao mesmo tempo.

```
1  Inserir todos os sites de S em Q.
2  p ← extract_min(Q)
3  T ← R_p.
4  enquanto Existe um evento em Q faça
5      p ← extract_min(Q)
6      se p é um site então
7          Encontre uma região R_q em T que contém p.
8          Crie o bissetor B_pq.
9          Atualize T substituindo R*_q por R*_q, C⁻_pq, R*_p, C⁺_pq, R*_q.
10         Exclua de Q as intersecções dos arcos de R_q com
           algum outro arco, se existir alguma.
11         Insira em Q a intersecção entre C⁻_pq e seu vizinho à
           esquerda em T, se existir alguma, insira em Q a
           intersecção entre C⁺_pq e seu vizinho à direita em
           T, se existir alguma.
12     senão //L está sobre uma intersecção.
13         Seja p a intersecção dos arcos C_qr e C_rs.
14         Crie o bissetor B*_qs.
15         Atualize T tal que C_qr, R*_r, C_rs sejam substituidos por
           C_qs = C⁻_qs ou C⁺_qs como apropriado.
16         Exclua de Q a intersecção entre C_qr e seu vizinho à
           esquerda e entre C_rs e seu vizinho à direita.
17         Insira em Q as intersecções entre C_qs e seus
           vizinhos à esquerda e à direita.
18         Marque p como um vértice e como extremo dos
           bissetores B*_qr, B*_rs e B*_qs.
19     fim
20 fim
```

O algoritmo possui alguns casos degenerados, estes casos devem ocorrer quando existir um *site* sobre um bissetor, quando existirem quatro ou mais *sites* cocirculares, ou quando existirem três ou mais *sites* colineares. Caso existir um *site* p sobre um bissetor, pode-se escolher

qual das regiões vai conter p. Se S possuir quatro ou mais *sites* cocirculares, e este caso não for tratado de forma explicita, o algoritmo vai criar um bissetor de tamanho 0 localizado no centro do círculo que contém os pontos cocirculares.

Teorema 3.1. *Seja S um conjunto de pontos, com somente um menor site. O Algoritmo 1 computa $V^*(S)$.*

Demonstração. Dizemos que uma região ou fronteira E está ativa em p, se quando L está sobre p, L intercepta E, e o ponto minímo de E não está à direita de p, e o ponto máximo de E não está à esquerda de p. Utilizaremos três invariantes, a inviariante três(I3) será a invariante do laço da linha 4, as invariantes um(I1) e dois(I2) são intermitentes, (I1) e (I2) são proposições verdadeiras após a retirada de um evento p de Q.

(I1) A lista T contém todas as regiões e fronteiras de regiões de $V^*(S)$ ativas em p, em ordem de intersecção com L.

(I2) Se e_{rs} é uma aresta de $V(S)$ e e^*_{rs} contém um ponto menor ou igual a p, então o bissetor B^*_{rs} já foi criado. Se v é um vértice de $V(S)$ e $v^* \leq p$, então o vértice v^* já foi criado e marcado como extremo de todas as arestas de $V^*(S)$ incidentes em v^*.

(I3) Se duas fronteiras de região são adjacentes em T, e se interceptam acima de L quando L está sobre P, então a intersecção entre as duas fronteiras está em Q.

A invariante I1 é verdade inicialmente, pois seja b o menor *site* L intercepta R^*_b em b. A invariante I2 é verdade inicialmente, pois não existe nenhuma aresta ou vértice contém um ponto menor que o menor *site*. A invariante I3 é verdade inicialmente, pois ainda não existem fronteiras de regiões.

A invariante I3 é mantida pelas linhas $11, 12, 17$ e 18. A invariante I2 é mantida pelo fato de que processamos os eventos em ordem crescente, obviamente se processamos um evento p todos os eventos abaixo de p já foram processados, logo para os vértices isso é verdadeiro pois qualquer vértice já processado é menor do que p, já uma aresta e^*_{rs} tem seu ponto minímo no *site* r ou no *site* s, como $r \leq p$ e $s \leq p$, a invariante está correta.

Para verificarmos a corretude de I1, primeiramente notemos que o conjunto de regiões e fronteiras ativas muda somente quando L está sobre um *site* ou quando L está sobre um vértice, pois cada *site* é o menor ponto de sua região, e alguns vértices são o maior ponto de uma região. Pelo Lema 3.6 duas arestas incidentes a um vértice v^* que não é um *site*, devem possuir um

trecho abaixo de v^*, logo as fronteiras devem estar ativas quando interceptadas por L, quando L está realizando a transição da posição de um dos *sites* para a posição do vértice v^*.

Agora provaremos que T é atualizada corretamente, quando L está sobre um *site* ou vértice. Suponha que p é um *site*, pelo Lema 3.4, p é o ponto minímo de R_p^*. Se p está contido em R_q^* quando ele foi encontrado pela primeira vez, então p está acima de e_{pq}, e p está contido em e_{pq}^*, consequentemente as linhas 8-10 atualizam T corretamente.

Agora suponha que p é um vértice, seja $C_{q_1,q_2}, C_{q_2,q_3}, ..., C_{q_{m-1},q_m}$ onde $m \geq 3$, uma sequência de fronteiras se interceptando em p, nesta ordem da esquerda para a direita e abaixo de p, e suponha que p está para ser retirado de Q pela primeira vez. Pela hipótese indutiva de I1, T contém a sequência $R_{q_1}^*, C_{q_1,q_2}, ..., C_{q_{m-1},q_m}, R_{q_m}^*$, está sequência será chamada de T_0. Pelo Lema 3.6, T_0 vai ser substituída por $R_{q_1}^*, C_{q_1,q_m}, R_{q_m}^*$, depois que p for retirado de Q pela última vez. A invariante I4 é verdade até p ser retirado de Q pela última vez.

(I4) $R_{q_1}^*$ e $R_{q_m}^*$ nunca são deletadas de T_0, e se $R_{q_i}, C_{q_i,q_j}, R_{q_j}, C_{q_j,q_k}, R_k$ são adjacentes em T_0, então C_{q_i,q_j} e C_{q_j,q_k} se interceptam no ponto p.

Pela invariante I3, Q vai conter uma cópia de p para cada par consecutivo de fronteiras em T_0. A invariante I4 é verdade após p ser retirado de Q pela primeira vez. Enquanto I4 é verdade, a cada iteração do laço um par de fronteiras que se interceptam e são adjacentes em T_0 é substituído por uma única fronteira. Uma vez que todos os *sites* $q_1, ..., q_m$ são equidistantes a $*^{-1}(p)$, a nova fronteira intercepta os vizinhos à esquerda e à direita de p, e a invariante I4 é mantida. Quando p é retirado de Q pela última vez, restará somente $R_{q_1}, C_{q_1,q_m}, R_{q_m}$, e a invariante I1 se mantém.

Agora suponha que p é um *site* em uma fronteira, ou p é um *site* e duas ou mais fronteiras se interceptam em p. Seja $C_{q_1,q_2}, ..., C_{q_{m-1},q_m}$ onde $m \geq 2$, fronteiras incidentes em p, Pelo Lema 3.6 essa sequência de fronteiras e regiões deve ser substituída por $C_{q_1,p}, R_p, C_{p,q_m}$. Mostraremos novamente que I4 é uma invariante do laço para as iterações em que p é retirado de Q. Como p é um *site* e um vértice, temos que tratar apenas o caso em que p é um *site*, pois já tratamos o caso em que p é um vértice. Quando p é um *site*, p deve estar contido em uma região R_{q_i}, e então as fronteiras $C_{q_i,p}^-$ e $C_{q_i,p}^+$ são criadas. De qualquer forma, p e $q_1, ..., q_m$ são todos equidistantes a $*^{-1}(p)$. Consequentemente se $i \neq 1$, então $C_{q_i,p}^-$ intersepta seu vizinho à esquerda em p, e se $i \neq m$ então $C_{q_i,p}^+$ intercepta seu vizinho à direita em p. □

Teorema 3.2. *O Algoritmo 1 tem complexidade de tempo $O(N\ log\ N)$ e complexidade de espaço $O(N)$.*

Demonstração. Por uma análise similar ao Teorema 3.1, o número de iterações do loop para um ponto p que não é um *site*, é o número de fronteiras abaixo de p e incidentes em p menos um (podemos notar isso através da invariante I4), quando p é um *site* o número de interações é igual ao número de fronteiras abaixo de p e incidentes em p mais um. Como a soma do grau de todos os vértices é $O(N)$ o número de iterações do loop também é $O(N)$.

Q será utilizada no máximo duas vezes por cada fronteira e uma vez por *site*, logo será utilizada $O(N)$ vezes. A fila de prioridade Q deve suportar as operações de inserção, remoção, e retirada do próximo evento, uma vez que Q pode ser implementada como uma heap binária todas as operações podem ser feita com um custo de tempo $O(logN)$ e um custo total de espaço de $O(N)$. T é utilizada no máximo $O(N)$ vezes, uma vez que L intercepta cada bissetor no máximo duas vezes. A lista T deve suportar as operações de inserção, remoção e busca, T pode ser implementada como uma árvore binária balanceada, respondendendo assim cada operação em tempo $O(logN)$ e utilizando $O(N)$ de espaço. Logo cada operação do loop possui custo de tempo $O(logN)$, portanto o Algoritmo 1 tem custo de tempo $O(NlogN)$ e $O(N)$ de espaço. □

Teorema 3.3. *O Algoritmo 1 pode ser modificado para computar $V(S)$ em tempo $O(N\ log\ N)$ e utilizando $O(N)$ de espaço.*

Demonstração. Seja v a intersecção entre duas fronteiras, o mapeamento de v pode ser feito somando à coordenada y de v a distância de v para qualquer um dos *sites* que determinam as fronteiras. A lista T pode conter regiões não transformadas e bissetores não transformados, o mapeamento pode ser computado explicitamente durante a busca da linha 8. □

4 IMPLEMENTAÇÃO DO ALGORITMO DE FORTUNE PARA PONTOS

Como parte do resultado deste trabalho, este capítulo mostra uma possibilidade de implementação do diagrama de Voronoi para o caso em que os *sites* não possuem peso. Uma vez que a entrada é formada por *sites* com coordenadas representadas por números inteiros, esta variante do diagrama pode ser implementada sem a utilização de ponto flutuante, porém as operações envolvidas para evitar o uso de raiz quadrada e divisão acabam por utilizar valores muito grandes. Para evitar o uso da divisão as operações atiméticas podem ser realizadas como fração, mas para simplificar a apresentação dos cálculos as operações em fração são omitidas.

Sabendo que os valores envolvidos nas operações podem ser muito altos, é mostrado um exemplo no final deste capítulo apontando este problema, para contorná-lo é necessário a utilização de uma biblioteca para manipulação de inteiros grandes, ou pode-se utilizar ponto flutuante evitando assim *overflow*, mas trazendo erros de cálculo para as operações.

4.1 Operações com frações

Quando existir uma atribuição $m = \frac{a}{b}$ considere que m ainda é uma fração. Seja $m = \frac{a}{b}, n = \frac{c}{d}$ onde $a, c \in \mathbb{Z}$ e $b, d \in \mathbb{N}_{>0}$, quando é feito $m * n$, m/n, $n - m$ ou $n + m$ as seguintes operações estão sendo realizadas:

$$n * m = \frac{a * c}{b * d}$$
$$n/m = \frac{a * d}{b * c}$$
$$n - m = \frac{a * d - b * c}{b * d}$$
$$n + m = \frac{a * d + b * c}{b * d}$$

Sempre que o denominador de uma fração resultante de qualquer operação aritmética for menor do que 0, multiplica-se o denominador e o númerador por -1, isso facilita a comparação entre frações, quando é feito:

$$\frac{a}{b} < \frac{c}{d}$$

Faz-se:

$$a * d < c * b$$

Caso os denominadores não fossem mantidos no conjunto $\mathbb{N}_{>0}$, casos como:

$$\frac{1}{5} < \frac{1}{-5}$$

Deveriam ser tratados separadamente, pois:

$$1 * (-5) < 1 * 5$$

Seria verdadeiro, o que é incorreto, uma vez que $\frac{1}{5} < \frac{1}{-5}$ é falso.

4.2 Transformação geométrica

A partir dos Lemas 3.1 e 3.2, conclui-se que existem apenas dois casos para o resultado do mapeamento de uma reta l, caso l seja vertical $l' = *(l)$ é uma semirreta vertical, caso contrário $h = *(l)$ é uma hipérbole. Seja p e q dois *sites* de um diagrama V, o bissetor B_{pq} é uma reta cuja a equação pode ser deduzida facilmente, pois para todo ponto $z \in B_{pq}$ é verdade que:

$$\sqrt{(z_x - p_x)^2 + (z_y - p_y)^2} = \sqrt{(z_x - q_x)^2 + (z_y - q_y)^2} \tag{4.1}$$

Elevanto os dois lados da equação 4.1 ao quadrado e agrupando os termos equivalentes chega-se a seguinte equação de reta:

$$2(q_x - p_x)z_x + 2(q_y - p_y)z_y + p_x^2 + p_y^2 - (q_x^2 + q_y^2) = 0 \tag{4.2}$$

Quando $p_y = q_y$ faz-se:

$$\begin{aligned} z_x &= \frac{q_x^2 + q_y^2 - (p_x^2 + p_y^2)}{2(q_x - p_x)} \\ &= \frac{(p_x + q_x)(q_x - p_x)}{2(q_x - p_x)} \\ &= \frac{p_x + q_x}{2} \end{aligned} \tag{4.3}$$

Pela equação 4.3 e pelo primeiro caso do Lema 3.2, tem-se que uma reta vertical mapeada pode ser representada pelos pontos $\{z \in B_{pq} : z_x = \frac{p_x + q_x}{2}\ e\ z_y > p_y\}$.

Caso $p_y \neq q_y$ é possível representar o bissetor pela equação de reta:

$$z_y = \frac{p_x - q_x}{q_y - p_y} z_x + \frac{q_x^2 + q_y^2 - (p_x^2 + p_y^2)}{2(q_x - p_x)} \tag{4.4}$$

Fazendo:

$$\begin{aligned} m &= \frac{p_x - q_x}{q_y - p_y} \\ b &= \frac{q_x^2 + q_y^2 - (p_x^2 + p_y^2)}{2(q_x - p_x)} \end{aligned}$$

Obtém-se:

$$z_y = mz_x + b \tag{4.5}$$

O mapeamento deve ser feito em cada ponto na forma (z_x, mz_x+b), portanto deve existir um $*(z) \in B^*_{pq}$ tal que:

$$*\,(z)_x = z_x \tag{4.6}$$

$$*\,(z)_y = m * z_x + b + \sqrt{(m * z_x + b - p_y)^2 + (z_x - p_x)^2} \tag{4.7}$$

Ainda na equação 4.7 colocando os termos que estão fora da raiz para o lado esquerdo da igualdade e elevando ambos os lados ao quadrado:

$$(*(z)_y - (m * z_x + b))^2 = (m * z_x + b - p_y)^2 + (z_x - p_x)^2 \tag{4.8}$$

Apenas para evitar ambiguidade na simbologia faz-se: $y = *(z)_y$. Desenvolvendo os quadrados e agrupando os termos equivalentes têm-se:

$$-z_x^2 - 2mz_xy + y^2 + 2(p_ym + p_x)x - 2by + 2bp_y - (p_y^2 + p_x^2) = 0 \tag{4.9}$$

Nota-se que a hipérbole está em sua forma geral e $-2m \neq 0$, ou seja, não necessariamente os dois focos da hipérbole estão ambos contidos em uma reta paralela ao eixo X, ou ambos contidos em uma reta paralela ao eixo Y.

Como está relatado em [3] primeiramente é calculado a intersecção entre dois bissetores ainda não transformados e só então computa-se a transformação geométrica da intersecção. Seja p, q e s três *sites* de um diagrama V, considerando que B_{pq} e B_{qs} se interceptam em algum ponto z do plano, a transformação geométrica do ponto z é representada por:

$$*\,(z) = (z_x, z_y + \sqrt{z_d}) \tag{4.10}$$

Onde z_d é a distância ao quadrado entre o ponto z e qualquer um dos três *sites* envolvidos, é importante lembrar que a operação de raiz quadrada não será feita, apenas o valor z_d deve ser armazenado de alguma forma junto ao ponto z.

4.3 Organização dos Eventos

Os eventos devem aguardar em uma fila de prioridade para que sejam atendidos em ordem, existem duas possibilidades de organização dos eventos, pode-se fazer uma fila para os eventos de vértice e uma fila para os eventos de *site* e então compara-se o primeiro evento de

cada fila para verificar qual será o próximo a ser atendido, também é possível manter somente uma fila para ambos os tipos de evento. Independente da escolha de organização da espera dos eventos, três tipos de comparação devem ser utilizadas, sendo elas:

1. $p < q$
2. $v < p$
3. $v < w$

Onde p e q são *sites* e v e w são intersecções aguardando por um evento de vértice.

A comparação de número 1 feita entre os *sites* p e q pode ser facilmente realizada apenas com a comparação entre as coordenadas:

Algoritmo 2: Comparação entre *sites*.

```
se p_y ≠ q_y então
    p_y < q_y
senão
    p_x < q_x
fim
```

Na comparação 2 o cálculo da raiz quadrada contida na coordenada y do vértice v pode ser evitado, as comparações $v_y + \sqrt{v_d} < p_y$ ou $v_x < p_x$ podem ser feita com os passos:

Algoritmo 3: Comparação entre vértice e *site*.

```
delta_y = p_y − v_y
se delta_y < 0 então
    não é verdade que v < p.
senão se v_d < delta_y^2 então
    é verdade que v < p.
senão se delta_y^2 = v_d e v_x < p_x então
    é verdade que v < p.
senão
    não é verdade que v < p.
```

A terceira comparação exige uma análise de alguns casos para verificar se $v < w$, primeiramente nota-se que:

$$v_y + \sqrt{v_d} < w_y + \sqrt{w_d} \tag{4.11}$$

É equivalente a comparação:

$$v_y - w_y < \sqrt{w_d} - \sqrt{v_d} \tag{4.12}$$

Agora é apresentada uma análise dos sinais das expressões de ambos os lados da comparação 4.12, a tabela a seguir ilustra as possíveis combinações dos sinais:

Tabela 4.1 – Casos possíveis para o sinal de cada lado da comparação 4.12.

$k = v_y - w_y$	$\sqrt{w_d} - \sqrt{v_d}$	
+	-	caso 1
-	+	caso 2
+	+	caso 3
-	-	caso 4

É importante notar que o sinal do número resultante de $\sqrt{w_d} - \sqrt{v_d}$, é o mesmo sinal do número resultante de $w_d - v_d$.

A tabela 4.1 não leva em conta os casos em que um dos lados da comparação resulte em 0, pois uma vez que isso aconteça a comparação se torna muito mais fácil, o algoritmo 4 cobre estes casos.

4.3.1 Casos 1 e 2 da tabela 4.1

No caso 1 como $v_y - w_y > 0$, é verdade que:

$$v_y > w_y \tag{4.13}$$

E uma vez $\sqrt{w_d} - \sqrt{v_d} < 0$, também é verdade que:

$$\sqrt{v_d} > \sqrt{w_d} \tag{4.14}$$

Somando as inequações 4.13 e 4.14, tem-se:

$$v_y + \sqrt{v_d} > w_y + \sqrt{w_d} \tag{4.15}$$

Portanto o resultado da comparação 4.11 é falso no caso 1.

A análise para o caso 2 é similar a análise feita para o caso 1, apenas é invertido o sinal das comparações e conclui-se que o resultado da comparação 4.11 é verdadeiro.

4.3.2 Casos 3 e 4 da tabela 4.1.

Para os casos 3 e 4 a inequação 4.12 é desenvolvida como segue.

A partir de:

$$v_y - w_y < \sqrt{w_d} - \sqrt{v_d}$$

Faz-se:

$$k + \sqrt{v_d} < \sqrt{w_d} \tag{4.16}$$

Elevando os dois lados da inequação 4.16 ao quadrado:

$$k^2 + v_d + 2k\sqrt{v_d} < w_d$$

Deixando somente $2k\sqrt{v_d}$ do lado direito da inequação:

$$k^2 + v_d - w_d < -2k\sqrt{v_d}$$

$$k' = k^2 + v_d - w_d$$

Obtendo:

$$k' < -2k\sqrt{v_d} \tag{4.17}$$

Não se pode simplesmente elevar os dois lados da inequação 4.17 ao quadrado, se isso fosse feito, o fato de que k' ou k podem ser menores que 0 estaria sendo ignorado, é necessário uma análise dos sinais das expressões dos dois lados da comparação feita pela inequação 4.17, a tabela a seguir mostra as combinações dos sinais:

Tabela 4.2 – Casos possíveis para o sinal de cada lado da comparação 4.17.

k'	$-2k\sqrt{v_d}$	
+	-	caso 5
-	+	caso 6
+	+	caso 7
-	-	caso 8

Agora é feita a análise dos casos ilustrados na tabela 4.2 para o caso 3 separadamente do caso 4.

4.3.2.1 Análise dos casos da tabela 4.2 para o caso 3 da tabela 4.1

Primeiramente nota-se que no caso 3 é verdade que $k > 0$, logo $-2k\sqrt{v_d} < 0$, portanto os casos 6 e 7 da tabela 4.2 não são possíveis para o caso 3.

Uma vez que $k' > 0$ no caso 5, a comparação 4.17 resultará sempre em falso no caso 3.

No caso 8 eleva-se os dois lados da inequação ao quadrado para evitar a raiz quadrada, e como ambos os lados são negativos o sinal da inequação é invertido, portanto faz-se $k'^2 > (-2k\sqrt{v_d})^2$.

4.3.2.2 Análise dos casos da tabela 4.2 para o caso 4 da tabela 4.1

No caso 4 tem-se que $k < 0$, consequentemente $-2k\sqrt{v_d} > 0$, o que mostra que os casos 5 e 8 da tabela 4.17 não são possíveis para o caso 4.

Uma vez verificado que $k' < 0$, pode-se concluir que para o caso 6 da tabela 4.2 o resultado da inequação 4.17 será sempre verdadeiro.

Já no caso 7 pode-se elevar os dois lados da inequação ao quadrado, obtendo $k'^2 < (-2k\sqrt{v_d})^2$.

Segue o algoritmo para comparação entre dois vértices.

Algoritmo 4: Comparação entre os vértices v e w.

```
1  k = v_y − w_y
2  delta_d = w_d − v_d
3  se k = 0 então
4  |  se v_d < w_d então
5  |  |  é verdade que v < p.
6  |  senão
7  |  |  não é verdade que v < p.
8  senão se delta_y = 0 então
9  |  se v_y < w_y então
10 |  |  é verdade que v < p.
11 |  senão
12 |  |  não é verdade que v < p.
13 senão se k > 0 e delta_d < 0 então
14 |  não é verdade que v < p.
15 senão se k < 0 e delta_d > 0 então
16 |  é verdade que v < p.
17 senão se k > 0 e delta_d > 0 então
18 |  k' = k^2 + v_d − w_d
19 |  se k' > 0 então
20 |  |  não é verdade que v < p.
21 |  senão se k'^2 > (−2k√v_d)^2 então
22 |  |  é verdade que v < p.
23 |  senão
24 |  |  não é verdade que v < p.
25 senão se k < 0 e delta_d < 0 então
26 |  k' = k^2 + v_d − w_d
27 |  se k' < 0 então
28 |  |  é verdade que v < p.
29 |  senão se k'^2 < (−2k√v_d)^2 então
30 |  |  é verdade que v < p.
31 |  senão
32 |  |  não é verdade que v < p.
```

4.4 Criação de fronteiras

Na Seção 3.3 foi apresentado uma definição de fronteira para a apresentação do algoritmo 1 e das provas dos Teoremas 3.1 e 3.2. Durante a implementação descrita neste trabalho a definição de fronteira também foi utilizada pra dividir um bissetor em dois arcos hiperbólicos. Dado um bissetor B^*_{pq} sendo que $p > q$, tem-se que $C^-_{pq} = \{z \in B^*_{pq} : z_x \leq p_x\}$, e $C^+_{pq} = \{z \in B^*_{pq} : p_x \geq z_x\}$, por convenção quando B^*_{pq} for vertical só será criado a fronteira C^-_{pq}.

Em alguns momentos o sinal $+$ ou $-$ é omitido, uma vez que não seja necessário a presença do mesmo, ou que o sinal em questão possa ser determinado pelo contexto. Na implementação alguma informação que aponte que uma fronteira C_{pq} é na realidade C^+_{pq} ou C^-_{pq} deve ser armazenada junto à fronteira C_{pq}.

O local do plano onde ocorreu um evento que originou a criação de uma fronteira C^-_{pq} ou C^+_{pq} é chamado de base da fronteira, por exemplo, caso uma fronteira C^+_{pq} seja criada durante o evento de um vértice v, logo a posição no plano do vértice v é a base de C^+_{pq}, o mesmo acontece quando ocorre um ES. Nota-se que a base de uma fronteira sempre é um ponto da fronteira, no caso de um ES a base da nova fronteira é um *site* que pertence a fronteira, no caso de um evento de vértice a base da nova fronteira é um vértice que está na intersecção da nova fronteira com outras duas fronteiras existentes. Também é importante que a base de uma fronteira seja armazenada junto à fronteira de alguma forma.

Tanto a nova definição de fronteira quanto a definição de base são úteis na próxima Seção.

4.5 Organização do *status* da linha varredora

Ao contrário do que o pseudocódigo de [3] mostra, não será inserido nenhuma estrutura no *status* para representar uma região.

Neste trabalho o *status* da linha varredora l tem as seguintes responsabilidades:

1. Manter as fronteiras interceptadas pela linha varredora.
2. Encontrar a primeira fronteira à direita, e a primeira fronteira à esquerda de uma fronteira qualquer.
3. Encontrar a primeira fronteira à direita de um *site*.

As três operações que devem ser feitas pelo *status* da linha devem possuir complexidade de tempo $O(logN)$, para isso ser possível o *status* é representado por uma árvore binária balanceada(BST), a BST deve conter todas as fronteiras interceptadas pela linha varredora ordenadas pela coordenada x do ponto de interceptação das fronteiras com l. Mais formalmente seja T a BST que contém as m fronteiras do *status* de l, é verdade que $C_k < C_j$, para $k \neq j$ e $1 \leq k, j \leq m$, quando para os pontos $A = C_k \cap l$ e $D = C_j \cap l$, é verdade que $A_x < D_x$.

Independente do evento que está sendo processado, uma nova fronteira C_{pq} com base b deve ser criada e inserida no *status* de l, e como C_{pq} é criada em um evento que está sendo processado, logo l está sobre o ponto que originou este evento, portanto $b = C_{pq} \cap l$ e $b_y = l_y$. Quando C_{pq} está sendo adicionada na BST devem existir somente comparações entre C_{pq} e as fronteiras já existentes na BST, ou seja, considerando que C_{rs} já está contida na BST, a seguinte comparação deve ser feita $(C_{pq} \cap l) < (C_{rs} \cap l)$, que é equivalente a $b < C_{rs} \cap l$, portanto a única comparação necessária para a BST do *status* de l, é entre a base da nova fronteira que está sendo adicionada, e o ponto de intersecção entre as fronteiras já contidas na BST e a reta l.

4.5.1 Operações no *status* durante um ES

Como mostra o algoritmo 3.3, encontrar a região R_q^* que contém o *site* p é o primeiro passo no processamento de um ES para p, a região R_q^* é inferida a partir da primeira fronteira à direita de p.

Seja T a BST que contém todas as fronteiras sendo interceptadas por l, quando ocorre um ES para um *site* s é feito uma busca em T pela primeira fronteira C_{rq} à direita de s, se não existir tal fronteira considera-se a fronteira C_{rq} mais à direita em T. Para realizar a busca pela fronteira C_{rq} é necessário realizar comparações entre o *site* s e as fronteiras contidas em T, ou seja, $s_x < I_x$, onde $I = C_{pq} \cap l$ e C_{pq} é uma fronteira qualquer em T.

4.5.1.1 Comparação entre *site* e fronteira

Primeiramente deve-se computar o ponto $I = C_{pq} \cap l$, na prática este cálculo envolve algumas condições, a fórmula para um bissetor não vertical B_{pq}^* é da forma $Ax^2 + Bxy + Cy^2 + Dx + Ey + F$, calculando $B_{pq}^* \cap l$, tem-se:

$$Ax^2 + Bl_yx + Cl_y^2 + Dx + El_y + F = 0$$

Fazendo:

$$c_0 = A$$

$$c_1 = Bl_y + D$$

$$c_2 = Cl_y^2 + El_y + F$$

Obtém-se a equação de segundo grau:

$$c_0x^2 + c_1x + c2 = 0 \tag{4.18}$$

Pela fórmula de Bhaskara tem-se as duas raízes:

$$x' = \frac{\sqrt{\Delta} - c_1}{2c_0} \tag{4.19}$$

$$x'' = \frac{-(\sqrt{\Delta} + c_1)}{2c_0} \tag{4.20}$$

A maior raiz representa a coordenada x de $I_{pq}^{+} \in C_{pq}^{+}$, e a menor raiz representa a coordenada x de $I_{pq}^{-} \in C_{pq}^{-}$, a Figura 4.1 ilustra esta afirmação, o *subscript* $_{pq}$ é omitido dos pontos I para melhor apresentação da simbologia.

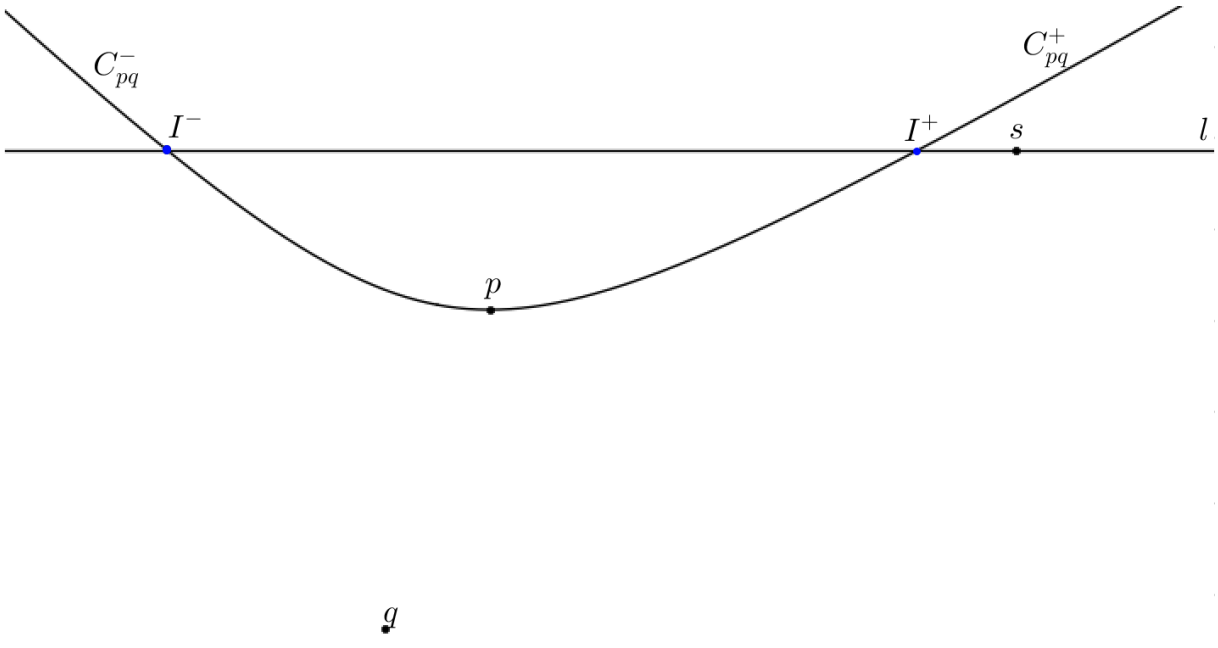

Figura 4.1 – ES para o *site* s

Para decidir qual raiz será tomada como coordenada I_x^- ou I_x^+, é necessário efetuar uma comparação entre as raízes 4.19 e 4.20, esta comparação pode ser feita apenas verificando o

sinal do denominador em comum $2c_0$, pois a partir de:

$$x'' < x'$$
$$\frac{-(\sqrt{\Delta} + c_1)}{2c_0} < \frac{\sqrt{\Delta} - c_1}{2c_0}$$

Retirando os termos equivalentes em lados contrários da inequação:

$$\frac{-\sqrt{\Delta}}{2c_0} < \frac{\sqrt{\Delta}}{2c_0} \tag{4.21}$$

Obviamente, quando $2c_0 < 0$ consequentemente $x'' > x'$, já quando $2c_0 > 0$ conclui-se que $x'' < x'$. Uma vez que $c_0 = A$ e a equação 4.9 apresentada na Seção 4.2 mostra que $A = -1$ então sempre é verdade que $x'' > x'$. Portanto, como é conhecido que $s_y = l_y$, consegue-se dizer que $I^- = (x', s_y)$ e $I^+ = (x'', s_y)$. O algoritmo 5 apresenta os passos para efetuar a comparação $s_x < x'$ sem utilizar raiz quadrada, já a comparação $s_x < x''$ é feita no algoritmo 6 também sem utilizar raiz quadrada.

Algoritmo 5: Algoritmo para comparação $s_x < x'$.

```
1 k = −2s_x + c_1
2 se k < 0 então
3 |  não é verdade que s_x < x'.
4 senão se k² > Δ então
5 |  é verdade que s_x < x'.
6 senão
7 |  não é verdade que s_x < x'
```

Algoritmo 6: Algoritmo para comparação $s_x < x''$.

```
1 k = 2s_x − c_1
2 se k < 0 então
3 |  é verdade que s_x < x''.
4 senão se k² < Δ então
5 |  é verdade que s_x < x''.
6 senão
7 |  não é verdade que s_x < x''
```

A comparação entre o *site* s e uma fronteira C_{pq} pode ser feita através do algoritmo 7.

Algoritmo 7: Algoritmo para comparação $s_x < C_{pq}$.

```
1 se C_pq = C⁻_pq então
2 |  utilizar o algoritmo 5 para verificar se s_x < x'.
3 senão
4 |  utilizar o algoritmo 6 para verificar se s_x < x''.
```

4.5.1.2 Como decidir qual região contém um *site*

Uma vez que se descobriu a primeira fronteira à direita de um *site* s, é fácil descobrir em qual região s está contido. Supondo que C_{pq} é a primeira fronteira à direita de s e que $p > q$, existem somente duas possibilidades, s está contido em R_p^* ou em R_q^*, o algoritmo 8 mostra como proceder nestas duas possibilidades.

Algoritmo 8: Algoritmo para verificar em qual região s está contido.

```
se C_pq = C_pq^- então
    Utilizar o algoritmo 7 para verificar se s_x < C_pq.
    se s_x ≤ C_pq^- então
        s está contido em R_q^*
    senão
        s está contido em R_p^*
senão
    se s_x ≤ C_pq^+ então
        s está contido em R_p^*
    senão
        s está contido em R_q^*
```

Observação: Quando um *site* está contido exatamente sobre uma fronteira o algoritmo 8 considera que o *site* está contido na região do *site* mais a baixo.

4.5.1.3 Inserção de fronteiras no *status* durante ES

Durante o processamento de um ES para um *site* s e após ter encontrado em qual região s está contido, deve-se criar novas fronteiras para repartir a região em que s está contido.

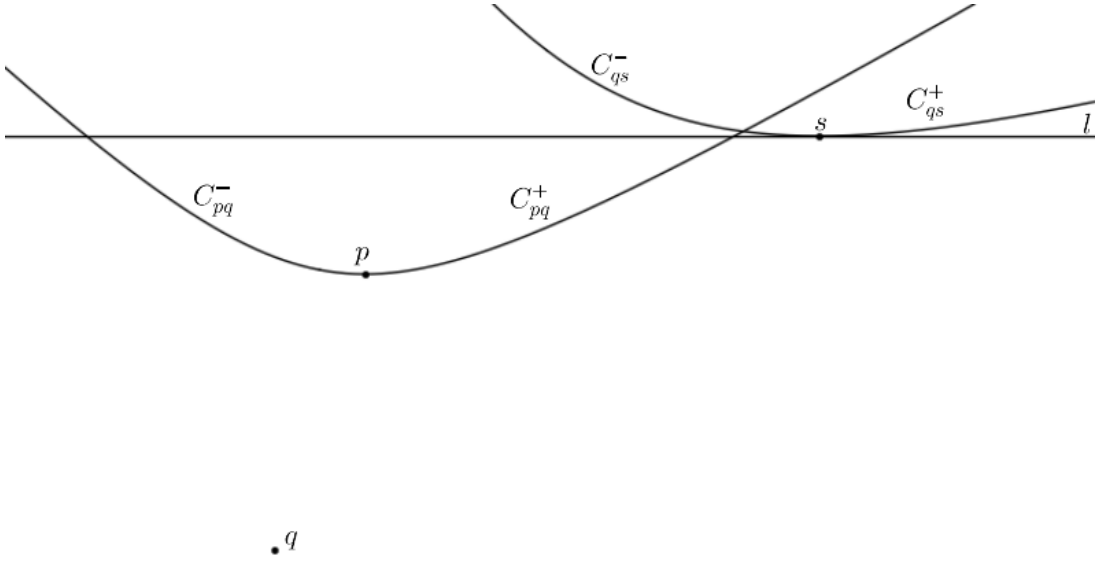

Figura 4.2 – ES para o *site* s

Levando em consideração o cenário da Figura 4.2, s está contido em R_q^*, portanto são criadas as fronteiras C_{qs}^- e C_{qs}^+, agora deve-se inserir ambas as fronteiras em T, durante a inserção de C_{qs} são feitas comparações somente entre C_{qs} e outras fronteiras já contidas em T, sendo assim, considerando que C_{pr} é uma fronteira qualquer em T a comparação $C_{qs} < C_{pr}$ será feita. Como já se sabe a comparação:

$$C_{qs} < C_{pr}$$

Deve ser feita pela comparação dos pontos:

$$(C_{qs} \cap l) < (C_{pr} \cap l)$$

Pelo fato de que $s = C_{qs} \cap l$, percebe-se que:

$$s < (C_{pr} \cap l) \tag{4.22}$$

A operação 4.22 pode ser feita pelo algoritmo 7.

Ainda é necessário tratar a comparação $C_{pq}^- < C_{pq}^+$ separadamente, neste caso considera-se que C_{pq}^- é lexicograficamente menor que C_{pq}^+.

4.5.1.4 Validação de intersecções

Após a inserção de uma fronteira C_{pq} em T é necessário verificar as intersecções entre C_{pq} e seus vizinhos em T, como já foi mostrado na Seção 4.2 a detecção de intersecções pri-

meiramente é feita nos bissetores ainda não transformados, somente após a detecção de uma intersecção v entre dois bissetores B_{pq} e B_{pr} é feito o mapeamento de v, por exemplo, suponha que C^-_{pq} é vizinha de C^+_{pr} em T, portanto será testado a intersecção entre os bissetores B_{pq} e B_{pr}, levando em conta que $v = B_{pq} \cap B_{pr}$, é possível que $v \in C^+_{pq}$ ao invés de $v \in C^-_{pq}$, para validar a intersecção v é necessário verificar se de fato é verdade que $v \in C^-_{pq}$ e $v \in C^+_{pr}$, isso pode ser feito utilizando a definição de fronteira mostrada na Seção 4.4.

4.5.2 Operações no *status* durante um EV

No início do processamento de um EV para um vértice $v = C_{rp} \cap C_{pq}$, é preciso criar a fronteira C_{rq}, primeiramente será criado uma fronteira C_{rq} em que a base é o *site* mais acima entre r e q, assumindo que $r > q$ e que tanto C_{rp} quanto C_{pq} não são verticais, caso $v_x > r_x$ então $C_{rq} = C^+_{rq}$, caso contrário $C_{rq} = C^-_{rq}$. Se C_{rp} ou C_{pq} for vertical é necessário verificar outros dois casos, as imagens 4.3 e 4.4 ilustram os dois casos possíveis quando C_{pq} é vertical.

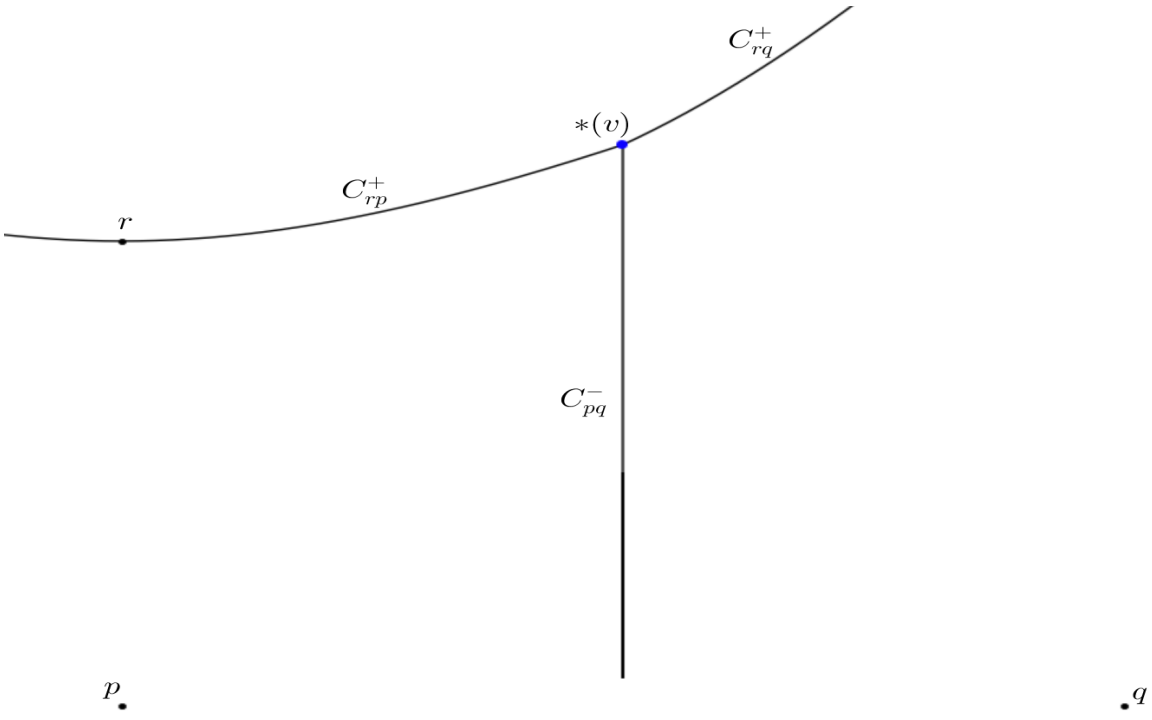

Figura 4.3 – Caso 1

Quando o *site* r está à esquerda da fronteira C^-_{pq} a fronteira criada durante o EV de v deve ser C^+_{rq}.

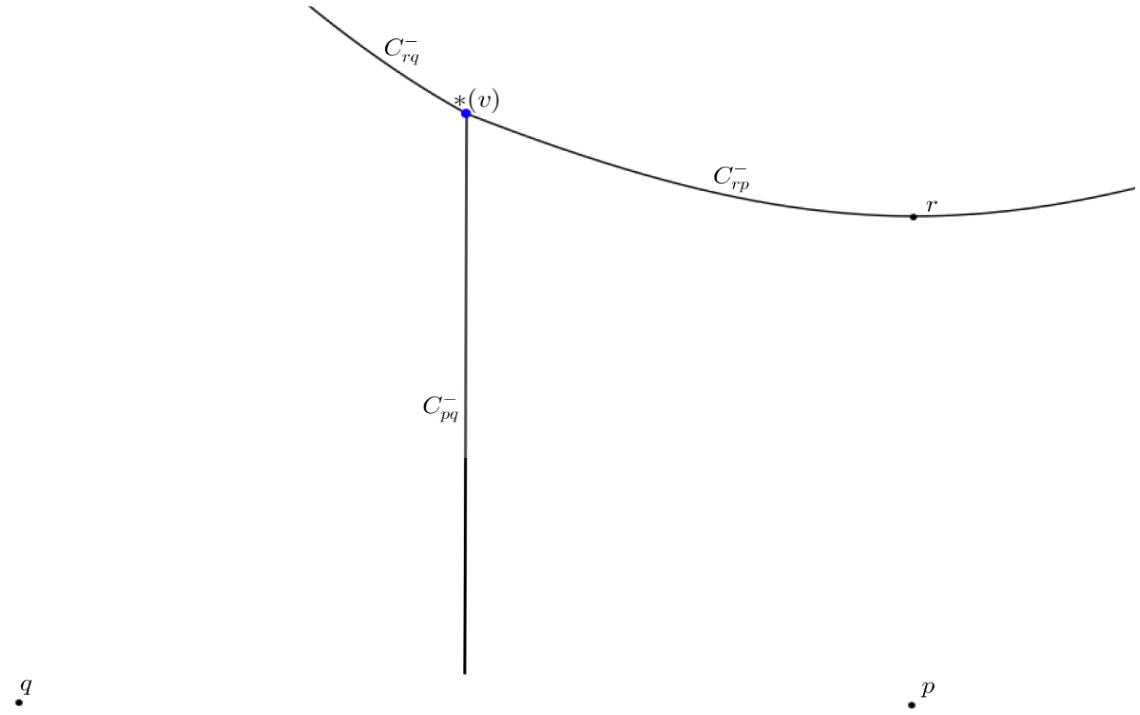

Figura 4.4 – Caso 2

Quando r está à direita de C_{pq}^{-} é necessário criar a fronteira C_{rq}^{-}.

4.5.2.1 Inserção de fronteira durante um EV

A inserção da fronteira C_{rq} durante o EV de um vétice $*(v) = (x, y + \sqrt{v_d})$ exige mais operações do que a inserção de uma fronteira durante um ES, pelo fato de que $l_y = y + \sqrt{v_d}$, e não somente $l_y = s_y$ como no caso de um ES, isso acaba por dificultar a comparação entre a nova fronteira que é inserida durante um EV. No entanto, assim foi explicado na Subseção 4.5.1.1 só existiram comparações entre a nova fronteira C_{rq} e as fronteiras já contidas em T, e também só é necessário comparar o vértice $*(v)$ e as fronteiras já contidas em T.

4.5.2.2 Comparação entre um vértice e uma fronteira

A comparação entre um vértice v e uma fronteira C_{rs} já contida em T, é feita através da comparação de v com o ponto $C_{rs} \cap l$, a intersecção $C_{rs} \cap l$ é calculada através da intersecção $B_{rs}^{*} \cap l$, como $B_{rs}^{*} \cap l$ resulta em dois pontos é necessário decidir qual ponto pertence à fronteira C_{rs}. Calculando $B_{rs}^{*} \cap l$:

$$Ax^2 + Bl_y x + Cl_y^2 + Dx + El_y + F = 0$$

Como $l_y = *(v)_y = v_y + \sqrt{v_d}$:

$$Ax^2 + B(v_y + \sqrt{v_d})x + C(v_y + \sqrt{v_d})^2 + Dx + E(v_y + \sqrt{v_d}) + F = 0$$

Desenvolvendo os quadrados e agrupando os termos equivalentes, obtém-se a equação de segundo grau:

$$Ax^2 + (B\sqrt{v_d} + Bv_y + D)x + (E + 2Cv_y)\sqrt{v_d} + Cv_y^2 + Ev_y + Cv_d + F = 0$$

Fazendo:

$$c_0 = A$$
$$a_0 = B$$
$$a_1 = Bv_y + D$$
$$c_1 = a_0\sqrt{v_d} + a_1$$
$$a_2 = E + 2Cv_y$$
$$a_3 = Cv_y^2 + Ev_y + Cv_d + F$$
$$c_2 = a_2\sqrt{v_d} + a_3$$

Chega-se em:

$$c_0x^2 + c_1x + c_2 = 0 \tag{4.23}$$

A equação 4.23 tem as raízes:

$$x' = \frac{\sqrt{\Delta} - c_1}{2c_0} \tag{4.24}$$

$$x'' = \frac{-(\sqrt{\Delta} + c_1)}{2c_0} \tag{4.25}$$

Assim como na Subseção 4.5.1.1 é preciso verificar qual das raízes x' e x'' é a maior e qual é a menor, visto que a menor raiz representa a coordenada x do ponto $C_{rs}^- \cap l$, e a maior representa a coordenada x do ponto $C_{rs}^+ \cap l$, como $2c_0 < 0$ é sabido que $x'' > x'$.

A comparação $v_x < x''$ pode ser feita como segue:

$$v_x < \frac{-(a_1 + a_0\sqrt{v_d} + \sqrt{\Delta})}{-2}$$
$$2v_x - a_1 - a_0\sqrt{v_d} < \sqrt{\Delta}$$
$$a_4 = 2v_x - a_1$$
$$a_4 - a_0\sqrt{v_d} < \sqrt{\Delta}$$

Caso $a_4 - a_0\sqrt{v_d} < 0$ então é verdade que $v_x < x''$, já quando $a_4 - a_0\sqrt{v_d} \geq 0$ eleva-se os dois lados de $a_4 - a_0\sqrt{v_d} < \sqrt{\Delta}$ ao quadrado e desenvolve-se as expressões contidas em $\Delta = c_1^2 - 4c_0c_2$, após agrupar os termos equivalentes chega-se em:

$$a_4^2 - (a_1^2 + 4a_3) < (2a_0a_4 + 2a_0a_1 + 4a_2)\sqrt{v_d} \tag{4.26}$$

Para simplificação faz-se $a_5 = a_4^2 - (a_1^2 + 4a_3)$ e $a_6 = 2a_0a_4 + 2a_0a_1 + 4a_2$, obtendo $a_5 < a_6\sqrt{v_d}$, tanto a comparação $a_4 - a_0\sqrt{v_d} < 0$ quanto $a_5 < a_6\sqrt{v_d}$ podem ser feitas pelo algoritmo apresentado a seguir:

Algoritmo 9: Algoritmo para efetuar comparações do tipo $k_1 > k_2\sqrt{k}$.

```
se k1 < 0 e k2 < 0 então
    se k1² < k2²k então
        é verdade que k1 > k2√k
    senão
        não é verdade que k1 > k2√k
se k1 < 0 e k2 > 0 então
    não é verdade que k1 > k2√k
se k1 > 0 e k2 < 0 então
    é verdade que k1 > k2√k
senão
    se k1² > k2²k então
        é verdade que k1 > k2√k
    senão
        não é verdade que k1 > k2√k
```

Para realizar a comparação $v_x < x'$ o processo é análogo:

$$v_x < x'$$
$$-2v_x + a_1 + a_0\sqrt{v_d} > \sqrt{\Delta}$$
$$a_4 = -2v_x + a_1$$
$$a_4 + a_0\sqrt{v_d} > \sqrt{\Delta}$$

Caso $a_4 + a_0\sqrt{v_d} < 0$ então não é verdade que $v_x < x'$, caso contrário eleva-se os dois lados de $a_4 + a_0\sqrt{v_d} > \sqrt{\Delta}$ ao quadrado e desenvolve-se as expressões contidas em $\Delta = c_1^2 - 4c_0c_2$, chegando em:

$$a_4^2 - (a_1^2 + 4a_3) > (2a_0a_4 + 2a_0a_1 + 4a_2)\sqrt{v_d} \tag{4.27}$$

Tanto a comparação $a_4 + a_0\sqrt{v_d} < 0$ quanto 4.27 podem ser feitas com o algoritmo 9, para isto basta multiplicar ambos os lados das duas equações por -1.

O algoritmo 10 mostra os passos para a comparação $v < C_{rs}$.

Algoritmo 10: Algoritmo para comparação $v < C_{pq}$.

```
1  Seja Ax^2 + Bxy + Cy^2 + Dx + Ey + F a equação do bissetor B*_rs,
     portanto:
2  a_0 = B
3  a_1 = Bv_y + D
4  a_2 = E + 2Cv_y
5  a_3 = Cv_y^2 + Ev_y + Cv_d + F
6  se C_pq = C^-_pq então
7  |  a_4 = -2v_x + a_1
8  senão
9  |  a_4 = 2v_x - a_1
10 a_5 = a_4^2 - (a_1^2 + 4a_3)
11 a_6 = 2a_0a_4 + 2a_0a_1 + 4a_2
12 se C_pq = C^-_pq então
13 |  //Fazer comparação v_x < x'.
14 |  utilizar o algoritmo 9 para verificar se a_4 + a_0√v_d < 0
15 |  utilizar o algoritmo 9 para verificar se a_5 > a_6√v_d
16 |  se a_4 + a_0√v_d ≥ 0 e a_5 > a_6√v_d então
17 |  |  é verdade que v < C_rs
18 |  senão
19 |  |  não é verdade que v < C_rs
20 senão
21 |  //Fazer comparação v_x < x''.
22 |  utilizar o algoritmo 9 para verificar se a_4 - a_0√v_d < 0
23 |  utilizar o algoritmo 9 para verificar se a_5 < a_6√v_d
24 |  se a_4 - a_0√v_d < 0 ou a_5 < a_6√v_d então
25 |  |  é verdade que v < C_rs
26 |  senão
27 |  |  não é verdade que v < C_rs
```

4.5.3 Capacidade númerica

Apesar do algoritmo 10 não utilizar ponto flutuante e portanto garantir precisão, o algoritmo utiliza números muito grandes, mesmo que a fronteira C_{pq} não seja criada a partir de *sites* que não possuem coordenadas com um valor alto, e que o prórprio vértice v não possua coordenadas com um valor alto. Por exemplo, considerando o seguinte cenário onde existem os *sites* $p = (1000, 900)$ e $q = (0, 898)$, e um vértice $v = (-600, 1000)$ cuja distância para p ao quadrado é $v_d = 2570000$, os bissetores não mapeados e mapeados são:

$$B_{pq} : y = -500x + 250899$$

$$B^*_{pq} : -1x^2 + 1000xy + 1y^2 - 898000x - 501798y + 449808200 = 0$$

Através do algoritmo 10 calcula-se:

$$a0 = 1000$$

$$a1 = 102000$$

$$a2 = -499798$$

$$a3 = -48419800$$

$$a4 = 103200$$

$$a5 = 439919200$$

$$a6 = 408400808$$

Para realizar a comparação $v < C^{-}_{pq}$, é preciso verificar se $a_4 + a_0\sqrt{v_d} \geq 0$ e $a_5 > a_6\sqrt{v_d}$, é facil notar que $a_4 + a_0\sqrt{v_d} \geq 0$, quando o algoritmo 9 vai realizar a comparação $a_5 > a_6\sqrt{v_d}$, será elevado os dois lados da equação ao quadrado, obtendo: $193528902528640000 > 428653435335885860480000$, uma vez que $428653435335885860480000 > 2^{64} - 1$ ocorrerá *overflow* em qualquer tipo de dados primitivo da linguagem $C++$ que estiver sendo utilizado. O caso de entrada apresentado nesta subseção resulta nos mesmos valores para as variáveis mesmo que as frações sejam mantidas em sua forma irredutível, pois para este caso de entrada não existe nenhuma divisão.

O problema de *overflow* pode ser contornado, o algoritmo 10 pode ser substituído por um algoritmo mais simples que se utiliza de ponto flutuante e raiz quadrada como o algoritmo 11, no entanto este algoritmo pode possuir erros de cálculo.

Algoritmo 11: Algoritmo para comparação $v < C_{pq}$.

```
1  Seja c_0x^2 + c_1x + c_2 = 0 a equação de intersecção B*_rs ∩ l.
2  se C_pq = C^-_pq então
3  |  //Fazer comparação v_x < x'.
4  |  se v_x < (√Δ − c_1)/(2c_0) então
5  |  |  é verdade que v < C_rs
6  |  senão
7  |  |  não é verdade que v < C_rs
8  senão
9  |  //Fazer comparação v_x < x''.
10 |  se v_x < (−(√Δ + c_1))/(2c_0) então
11 |  |  é verdade que v < C_rs
12 |  senão
13 |  |  não é verdade que v < C_rs
```

A implementação desenvolvida neste trabalho utiliza ponto flutuante e raiz quadrada, a

implementação funciona corretamente para casos gerais e também para os casos degenerados apontados em [3]. A seguir é mostrado o resultado de alguns casos em que o diagrama resultado foi gerado corretamente.

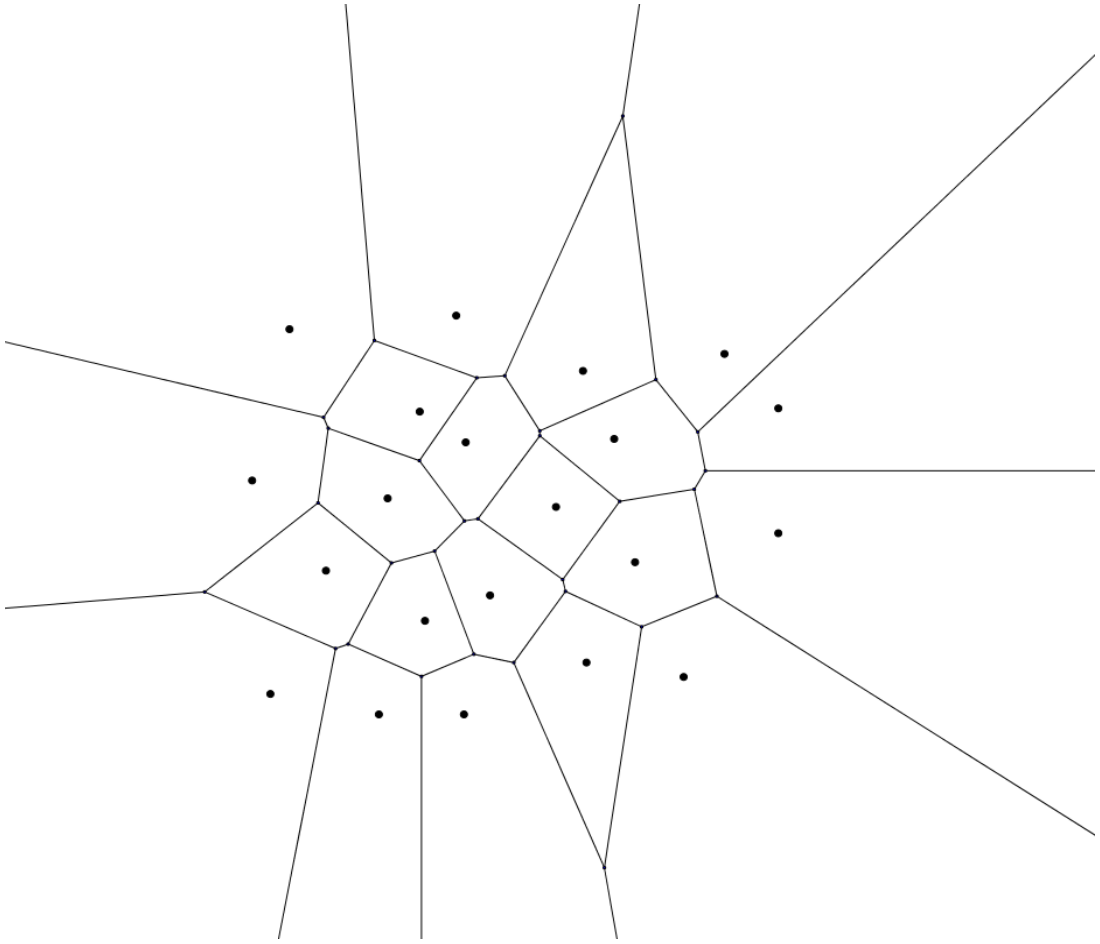

Figura 4.5 – Caso geral

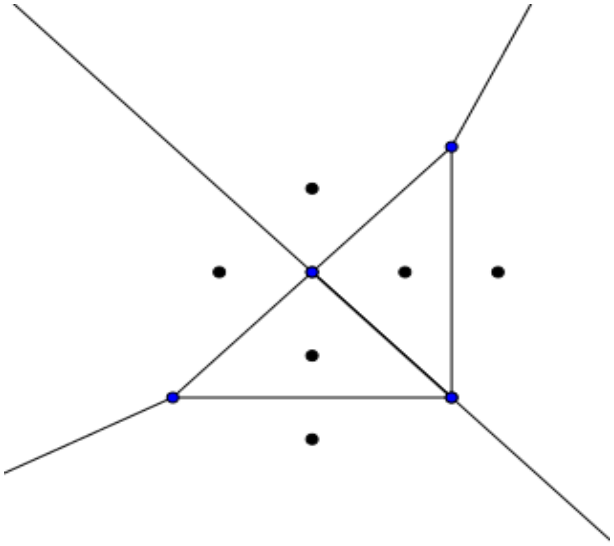

Figura 4.6 – Caso degenerado 1

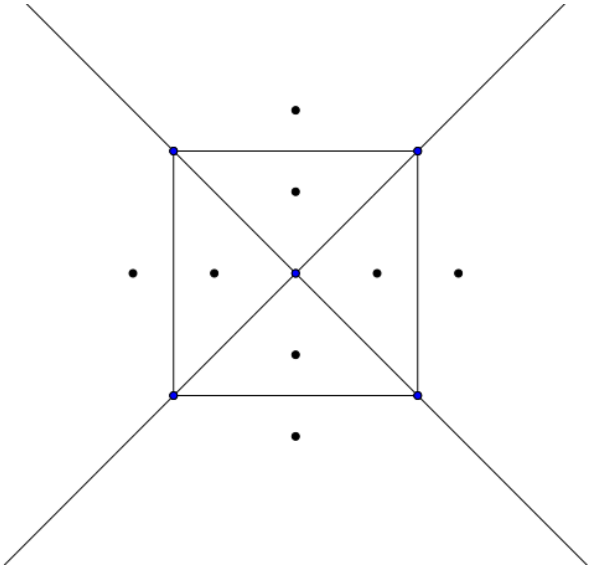

Figura 4.7 – Caso degenerado 2

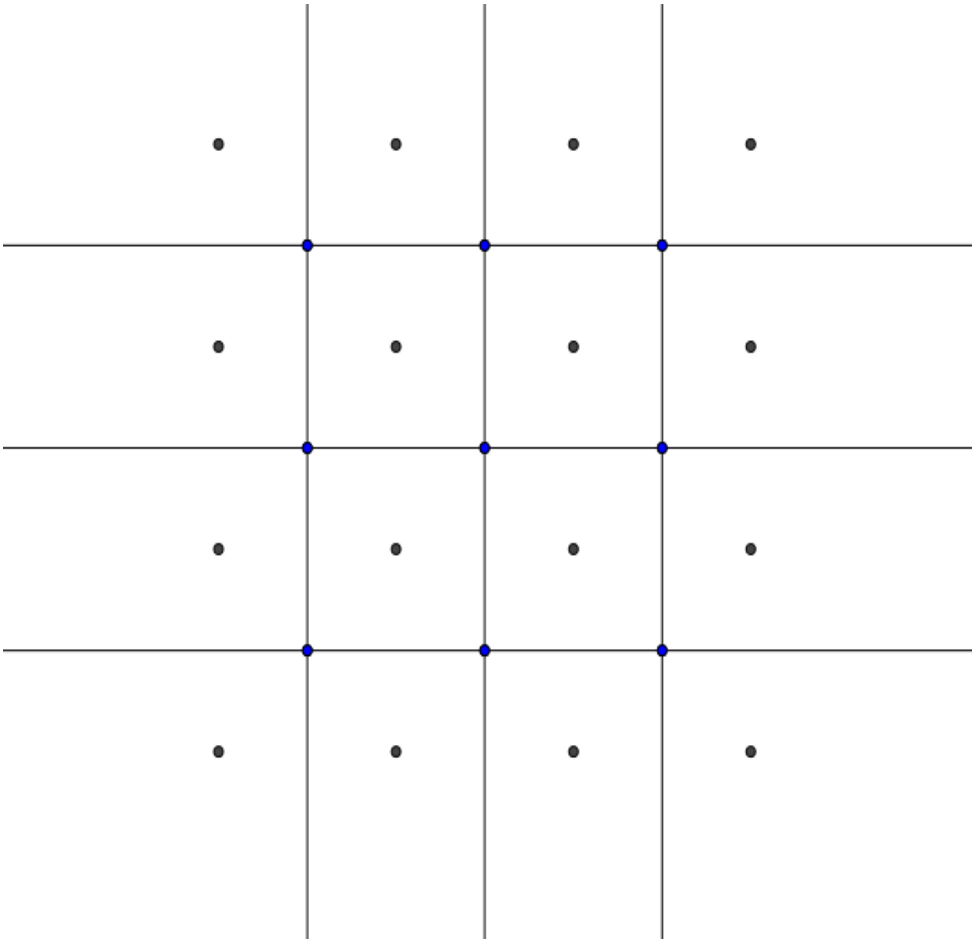

Figura 4.8 – Caso degenerado 3.

5 O DIAGRAMA DE VORONOI PARA PONTOS COM PESO

A variante do diagrama de Voronoi para pontos com peso possui uma métrica diferente de distância entre os *sites* e os pontos do plano, nesta variante cada *site* p possui um peso não-negativo w_p associado, a distância entre p e um ponto z do plano é calculada por $dist(p, z) + w_p$. O uso da nova métrica acaba gerando novas formas geométricas como bissetores e como regiões, além disso, a tarefa de computar os vértices do diagrama acaba se tornando mais complicada, no entanto a nova métrica permite a equivalência entre o diagrama de Voronoi para pontos com peso e o diagrama de Voronoi para círculos.

A função de distância $d : \mathbb{R}^2 \rightarrow \mathbb{R}$ definida no Capítulo 2 na Seção 2.1 sofre uma pequena alteração, a partir deste ponto considera-se que $d(z) = \min_{p \in S}(dist(p, z) + w_p)$, a função é útil em vários momentos neste e no próximo Capítulo.

Dado um conjunto S de *sites* com peso, o bissetor entre dois *sites* $p, q \in S$ é definido por:

$$B_{pq} = \{z \in \mathbb{R}^2 : dist(p, z) + w_p = dist(q, z) + w_q\} \tag{5.1}$$

Já a região de um *site* p é definida por

$$R_p = \{z \in \mathbb{R}^2 : dist(p, z) + w_p = d(z)\} \tag{5.2}$$

O conjunto S de pontos com peso pode ser visto como um conjunto de círculos, basta converter o peso de cada *site* por um raio correspondente, para isso faz-se $W = \max_{p \in S} w_p$ e define-se o raio de cada *site* p por $r_p = W - w_p$, o centro do círculo é o próprio *site* p. No caso do diagrama para círculos a distância entre um ponto z no plano e um *site* p é computada por $dist(p, z) - r_p$, esta métrica de distância resulta no fato de que quando z está contido no interior do círculo a distância é negativa, caso contrário a distância é não negativa.

A definição de bissetor continua sendo válida após converter o peso de um *site* em um raio, pois utilizando a métrica de distância entre ponto e círculo e sabendo que o bissetor entre dois *sites* são os pontos equidistantes aos dois *sites*, tem-se:

$$dist(p, z) - r_p = dist(q, z) - r_q$$

$$dist(p, z) - W + w_p = dist(q, z) - W + w_q$$

$$dist(p, z) + w_p = dist(q, z) + w_q$$

Analogamente a definição de região continua válida.

Da definição de região surgem alguns casos passíveis de análise, segue uma descrição de cada um deles.

1. Caso $R_p = \emptyset$ sabe-se que nenhum ponto do plano pertence a região de p, ou seja, nem mesmo o próprio *site* p pertence a sua região, para isso ser possível deve existir um *site* $q \in S$ tal que $dist(q,p) + w_q < w_p$, neste caso a região R_p é vazia e diz-se que q domina p.

2. Se para todo ponto $z \in R_p$ existe um outro *site* $q \in S - \{p\}$ que satisfaça a igualdade

$$dist(p,z) + w_p = dist(q,z) + w_q$$

   Então não existe nenhum ponto z contido no interior de R_p, neste caso R_p possui o interior vazio, pois todos os seus pontos estão contidos no bissetor entre o *site* p e outros *sites* em S.

3. Caso existir pelo menos um ponto $z \in R_p$, tal que $\forall q \in S - \{p\}$, e seja verdade que:

$$dist(p,z) + w_p < dist(q,z) + w_q$$

   Então R_p não é vazia.

Para os bissetores também existem três possíveis formas geométricas. Seja p e q dois *sites* com os pesos w_p e w_q respectivamente, o bissetor entre os *sites* surge da igualdade em 5.1, segue uma descrição das formas geométricas dos bissetores.

1. Caso $w_p = w_q$, colocando w_p do lado direito da igualdade em 5.1, percebe-se que o bissetor dos *sites* p e q é uma reta assim como no diagrama para pontos sem peso.

2. Caso $dist(q,p) + w_q = w_p$, o bissetor é uma semirreta com extremo em p e contida na reta que passa pelos pontos p e q.

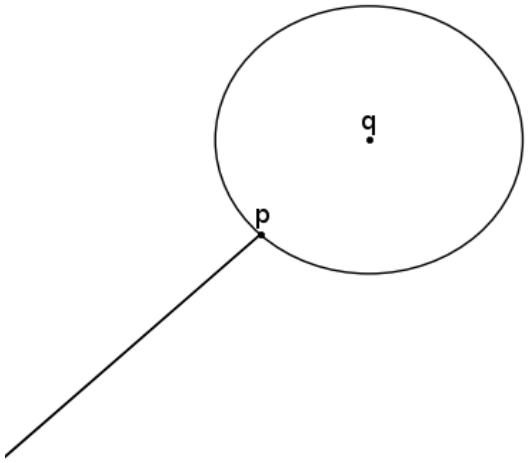

Figura 5.1 – Bissetor para diagrama de pontos com peso / círculos, caso $w_p = w_q + dist(q, p)$

3. A partir da equação em 5.1 colocando w_p do lado direito da igualdade e $dist(q, z)$ do lado esquerdo tem-se:

$$dist(p, z) - dist(q, z) = w_q - w_p \tag{5.3}$$

A equação 5.3 representa uma hipérbole que possui os *sites* p e q como foco, somente um ramo da hipérbole será utilizado como bissetor, caso $w_p > w_q$ o ramo que possui p como foco será utilizado, caso contrário, o ramo que possui q como foco será utilizado como bissetor.

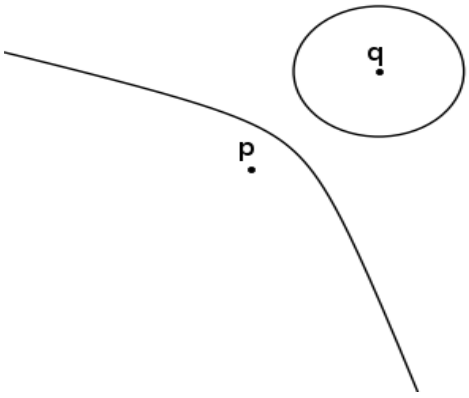

Figura 5.2 – Bissetor para diagrama de pontos com peso / círculos, caso $w_p > w_q$

Visto que os bissetores entre dois *sites* com peso podem ser retas, semirretas ou hipér-

boles, as arestas utilizadas no diagrama também podem assumir somente essas formas geométricas, o trabalho [11] contém a prova desta propriedade.

Os vértices do diagrama para pontos com peso são pontos no plano equidistantes a três ou mais *sites*, mais formalmente, um ponto v no plano é um vértice se existir um conjunto $S' \subseteq S$ com três ou mais *sites*, tal que para toda tripla de *sites* $p, q, s \in S'$ seja verdade que:

$$dist(p, v) + w_p = dist(q, v) + w_q = dist(s, v) + w_s = d(v) \tag{5.4}$$

Utilizando a métrica de distância para círculos tem-se a relação:

$$dist(p, v) - r_p = dist(q, v) - r_q = dist(s, v) - r_s = d(v) \tag{5.5}$$

A partir da equação 5.5 nota-se que o vértice v possui a mesma distância para três *sites*, dado que os *sites* são círculos, então v possui a mesma distância para três pontos p', q' e s' respectivamente pertencentes as circunferências dos *sites* p, q e s, portanto pode-se dizer que v é o centro de um círculo C_v de raio $dist(p, v) - r_p$ que possui p', q' e s' em sua circunferência, ou seja, C_v é tangente aos círculos dos *sites* p, q e s (Ver Figura 5.3).

Das definições de bissetor, região e vértice é possível dizer que o diagrama de Voronoi de um conjunto de *sites* com peso é definido por $V(S) = \{z \in \mathbb{R}^2, p\ e\ q \in S, p \neq q : d(z) = dist(p, z) + w_p = dist(q, z) + w_q\}$.

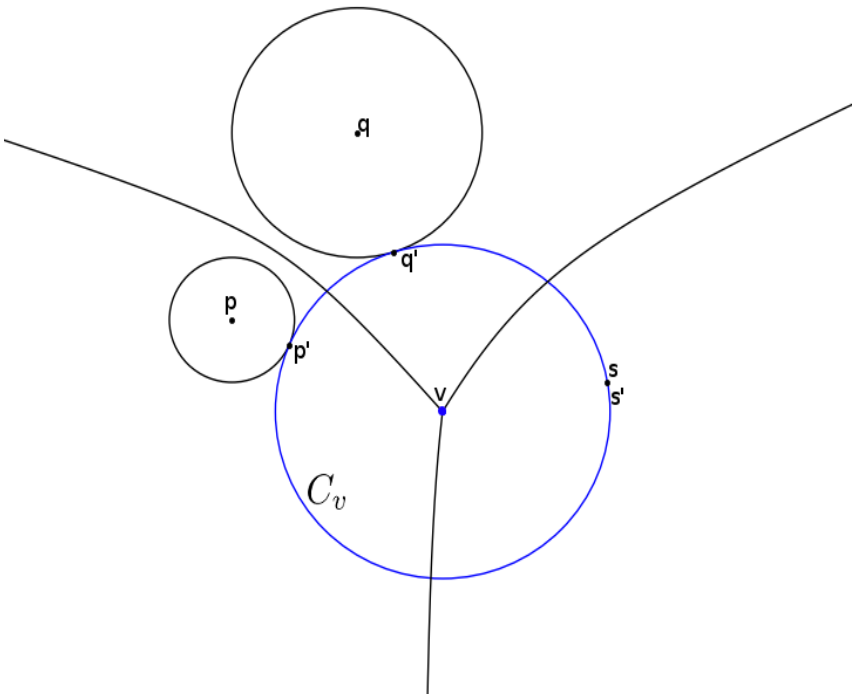

Figura 5.3 – Vértice do diagrama de Voronoi para pontos com peso / círculos

O problema de computar o círculo C_v é o décimo problema de Apolônio de Perga, o problema consiste em computar um círculo tangente a três outros círculos, em [9] é apresentado uma solução para este problema, já em [6] é apresentado uma solução que tem como objetivo garantir a robustez dos cálculos necessários para encontrar o círculo. No próximo Capítulo as duas soluções são discutidas e também é relatado como a solução [6] foi utilizada para computar os vértices do diagrama.

Os bissetores na Figura 5.3 se interceptam somente no ponto v, porém, como os bissetores podem assumir a forma de uma hipérbole, três bissetores também podem se interceptar em dois locais, como mostra a próxima Figura.

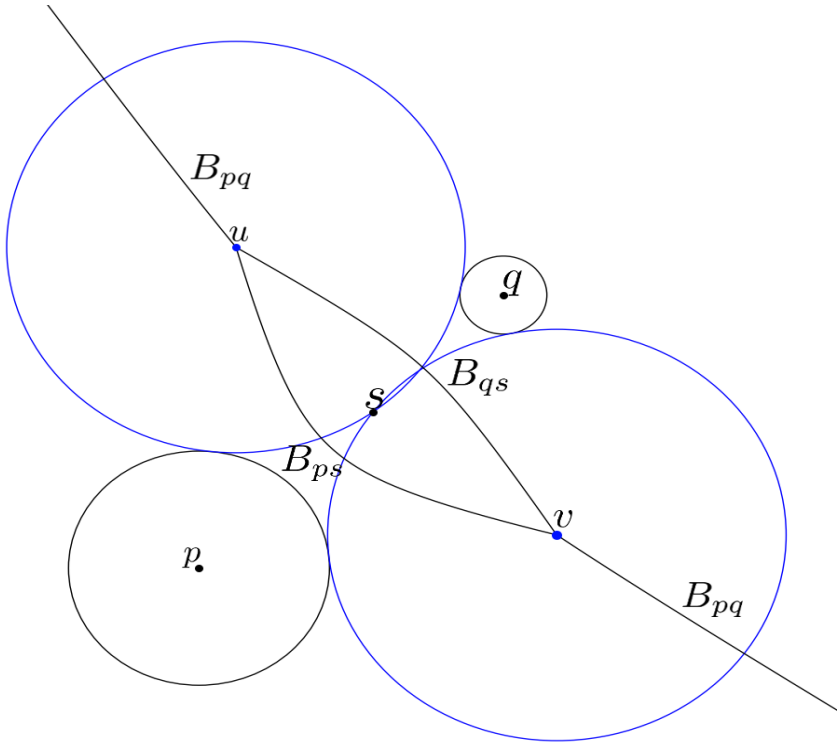

Figura 5.4 – Dois vértices gerados a partir de três bissetores

Na realidade a intersecção entre três bissetores $I = B_{pq} \cap B_{qs} \cap B_{ps}$ pode possuir no máximo dois pontos, uma prova para esta propriedade pode ser encontrada em [11]. No caso do diagrama na Figura 5.4 foram necessários três bissetores para repartir o plano, o motivo se da pelo fato de que $w_s > w_p$ e $w_s > w_q$. Considerando a mesma posição para os *sites* p, q e s mas levando em conta que $w_s < w_p$ e $w_s < w_q$, não são mais necessários três bissetores para realizar a partição do plano, como ilustra a próxima Figura apenas são necessários dois bissetores.

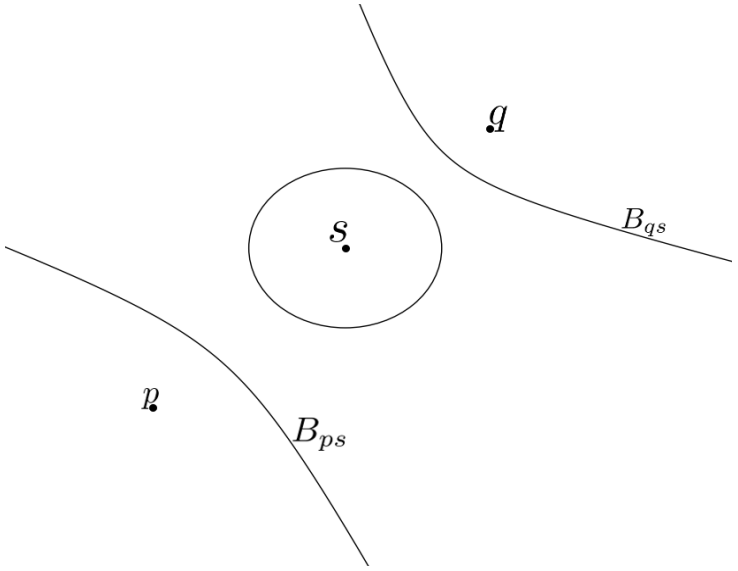

Figura 5.5 – Diagrama que possui somente bissetores

O diagrama da Figura 5.5 não possui vértice, pois as arestas contidas nos bissetores não se tocam em nenhum ponto, por isso cada aresta está contida em um componente diferente, por este fato diz-se que o diagrama é desconexo.

Um componente consiste de vértices e arestas do diagrama, duas arestas e_{pq} e e_{ps} estão contidas em um mesmo componente V' se existir uma sequência de arestas $W = e_1, e_2, ..., e_{m-1}, e_m$ tal que toda aresta em W está contida em V' e $e_1 = e_{pq}$, $e_m = e_{ps}$, além disso, cada aresta e_i deve estar conectada com a aresta e_{i+1} através de um vértice $u \in V'$ caso $i < m$.

Um diagrama V é dito desconexo se possui mais de um componente, um diagrama desconexo pode possuir $O(N)$ componentes onde N é o número de *sites*, a próxima Figura 5.6 mostra um exemplo deste fato. Uma vez que cada componente pode possuir um conjunto de vértices e arestas que fazem parte do diagrama, é possível notar que podem existir $O(N)$ arestas e $O(N)$ vértices em um componente, mas a soma do número de arestas e de vértices de todos os componentes é $O(N)$. As provas das afirmações feitas referentes a complexidade da estrutura do diagrama podem ser encontradas em [11].

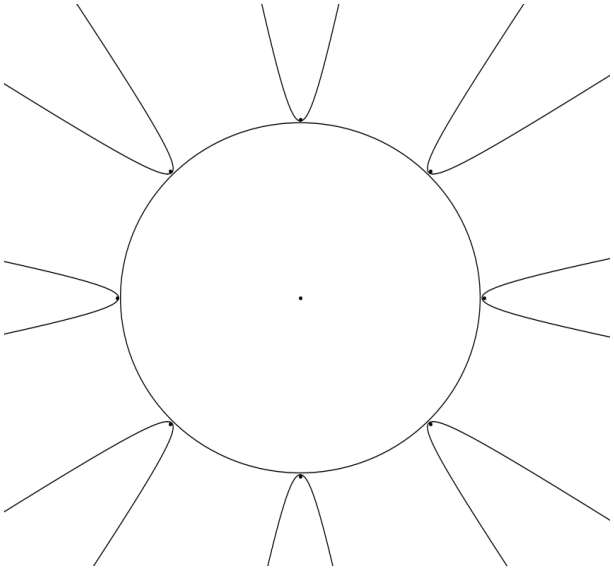

Figura 5.6 – Diagrama de Voronoi com N-1 componentes

6 IMPLEMENTAÇÃO DO ALGORITMO DE FORTUNE PARA PONTOS COM PESO

Neste capítulo são apresentadas algumas formas de se realizar alguns passos do algoritmo de Fortune para pontos com peso, os passos mostrados neste capítulo fazem parte dos detalhes não mostrados em [3], como por exemplo na Seção 6.1 é mostrada uma forma de se computar a transformação geométrica dos elemento de um diagrama de pontos com peso utilizando o método de construção de cônicas descrito em [8], na Seção 6.3 é apresentado um meio de se computar os vértices do diagrama a partir do resultado publicado em [6].

6.1 Transformação Geométrica

Seja V o diagrama para pontos com peso e $V^* = *(V)$, esta seção vai apresentar o comportamento da transformação e uma maneira de computar a transformação para os elementos de V. A transformação geométrica de um ponto z no plano para o caso em que os *sites* possuem peso é dada por:

$$*(z) = (z_x, z_y + d(z))$$

Já o mapeamento auxiliar $*p(z)$ é:

$$*p(z) = (z_x, z_y + dist(p, z) + w_p)$$

Com a definição do mapeamento percebe-se que um *site* p será mapeado para a posição $*(p) = (p_x, p_y + w_p)$.

Algumas propriedades são semelhantes as propriedades da transforação para pontos sem peso, como por exemplo, a propriedade mostrada no próximo Lema é similar a propriedade mostrada no Lema 3.4.

Lema 6.1. $*(p) = (p_x, p_y + w_p)$ *é o menor ponto da região* R^*_p.

Demonstração. Similar à prova do Lema 3.4. □

As Figuras 6.1 e 6.2 ilustram a propriedade do Lema 6.1.

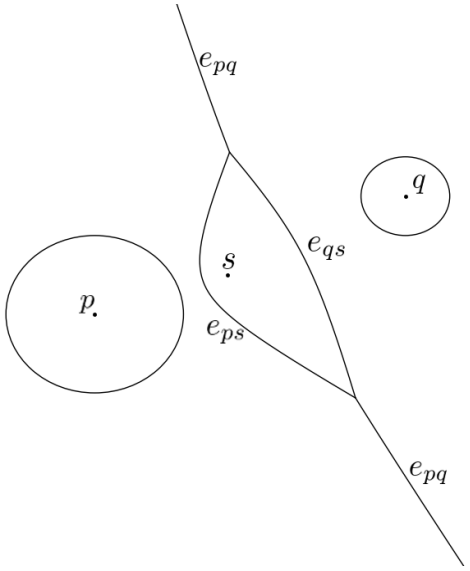

Figura 6.1 – Região não mapeada

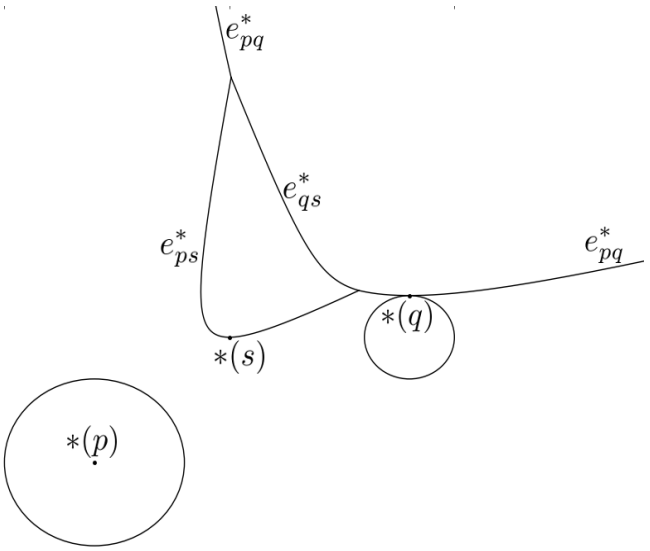

Figura 6.2 – Região mapeada

6.1.1 Como computar a transformação geométrica

Computar a trasformação geométrica dos vértices do diagrama para pontos com peso, não é uma tarefa muito diferente de computar o mapeamento $*$ para um vértice de um diagrama para pontos como foi feito na Seção 4.2. Seja S um conjunto de *sites* com peso, V o diagrama de S e v um vértice equidistante aos *sites* $p, q, s \in S$, o mapeamento $*(v)$ pode ser realizado através de:

$$*(v)_x = v_x$$

$$*(v)_y = v_y + \sqrt{v_d} + w_p$$

Onde v_d é a distância de v para p ao quadrado, fazendo $v_a = v_y + w_p$, $*(v)$ tem a seguinte forma:

$$\begin{aligned} *(v)_x &= v_x \\ *(v)_y &= v_a + \sqrt{v_d} \end{aligned} \tag{6.1}$$

Pode-se notar que a forma do vértice $*(v)$ mostrada em 6.1, é a mesma forma de um vértice para um diagrama sem pontos com peso declarada em 4.10.

Verificar o comportamento do mapeamento $*$ nos bissetores de V exige a análise de três casos, visto que existem três possíveis formas geométricas de bissetores como foi mostrado no Capítulo 5. Considerando que p e q sejam dois *sites* com peso, existem os seguintes casos para o mapeamento de B_{pq}:

1. Quando $p_y \neq q_y$ e $w_p = w_q$ o bissetor B_{pq} é uma reta não vertical, o mapeamento de uma reta não vertical segundo o Lema 3.1 B^*_{pq} é uma hipérbole, e de acordo com o Lema 4.3 do trabalho [3] existe somente uma reta horizontal tangente ao bissetor B^*_{pq}, e esta reta passa sobre o *site* mais acima entre $*(p)$ e $*(q)$.

2. Se $p_y + w_p = q_y + w_q$ existem dois casos, uma vez que $p_y = q_y$ e $w_p = w_q$, logo B^*_{pq} é uma semirreta vertical com extremo no primeiro ponto acima de $((p_x + q_x)/2, p_y + w_p)$. Quando $w_p \neq w_q$, B^*_{pq} é uma semirreta, visto $C = 0$ na equação de B^*_{pq} (ainda nesta seção será mostrado como computar B^*_{pq}). Na implementação realizada neste trabalho este caso não foi levado em conta, para a simplificação dos casos de borda.

3. Para o caso em que $p_y + w_p > q_y + w_q$ e $w_p \neq w_q$ o mapeamento foi exaustivamente observado, e como mostram as quatro Figuras a seguir quando mapeamos um bissetor B_{pq} que é uma hipérbole o resultado do mapeamento também será uma hipérbole.

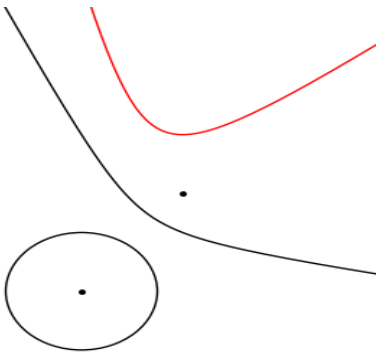

Figura 6.3 – *Site* com menor peso abaixo e à esquerda do *site* com maior peso

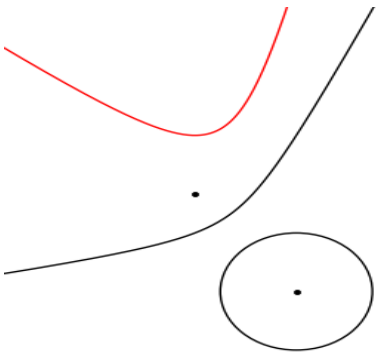

Figura 6.4 – *Site* com menor peso abaixo e à direita do *site* com maior peso

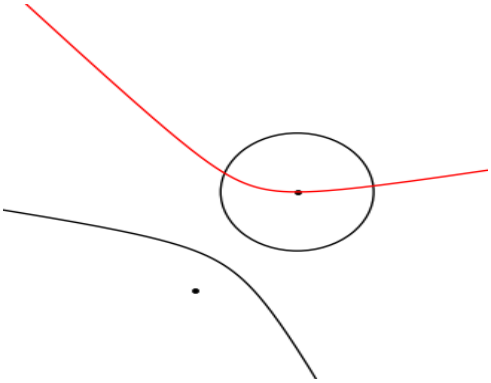

Figura 6.5 – *Site* com menor peso acima e à direita do *site* com maior peso

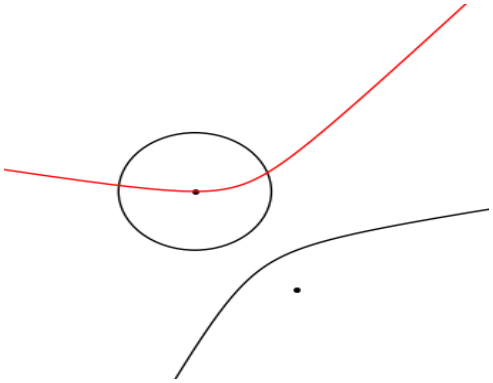

Figura 6.6 – *Site* com menor peso acima e à esquerda do *site* com maior peso

O caso em que $dist(p, q) + w_p = w_q$ também não foi levado em consideração, para simplificação da implementação.

Seja B_{pq} o bissetor não vertical entre dois *sites* com peso p e q onde $p > q$, e B_{rs} o bissetor não vertical entre dois *sites* sem peso r e s onde $r > s$, vale a pena notar que a hipérbole resultante de $*(B_{pq})$ pode não ter as mesmas propriedades da hipérbole resultante de $*(B_{rs})$. A hipérbole B^*_{rs} como foi mostrado na Seção 3.3 é monótona e decrescente até o *site* r, e monótona e crescente após o *site* r, o mesmo não é verdade para a hipérbole B^*_{pq} (Ver Figuras 6.7 e 6.8).

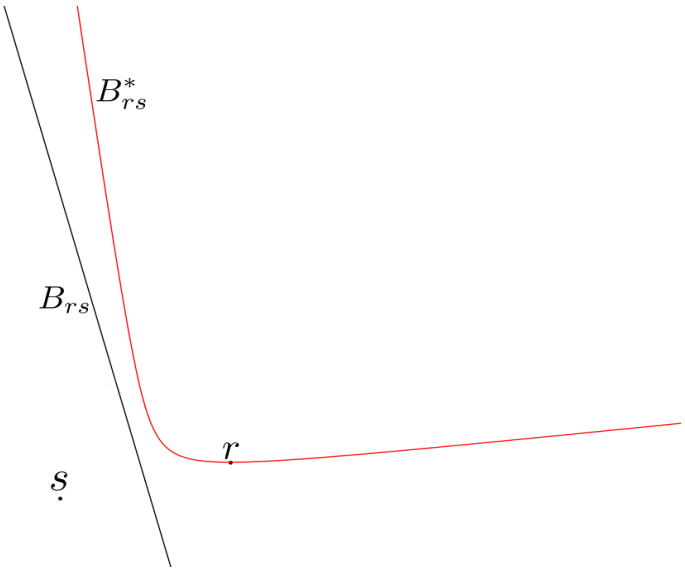

Figura 6.7 – Hipérbole (em vermelho) que representa o mapeamento de uma reta

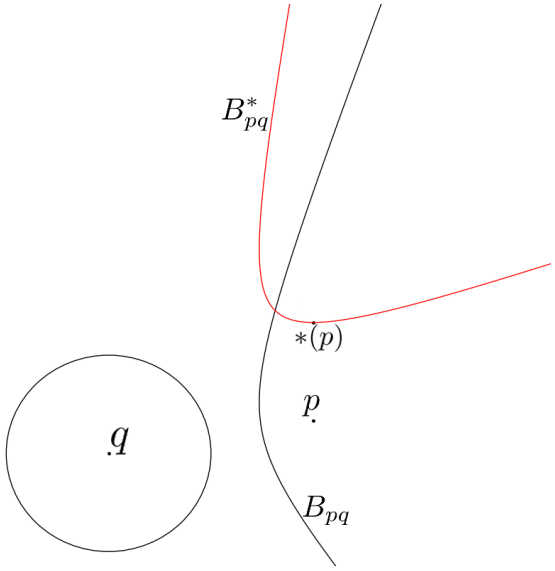

Figura 6.8 – Hipérbole (em vermelho) representando o mapeamento de uma hipérbole

A equação da hipérbole B_{pq} é da forma:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \tag{6.2}$$

Devido aos termos envolvidos na equação 6.2, a equação para B^*_{pq} não pode ser calculada tão facilmente quanto no caso de B^*_{rs} como foi mostrado na Seção 4.2. Dividindo todos os coeficientes de 6.2 por F, tem-se:

$$A'x^2 + B'xy + C'y^2 + D'x + E'y + 1 = 0 \tag{6.3}$$

Escolhendo um ponto arbitrário $z_1 = (x_1, y_1)$ em B_{pq} e utilizando suas coordenadas em 6.3, conclui-se que as variáveis que devem ser calculadas são os coeficientes A', B', C', D', E' da equação $A'x_1^2 + B'x_1y_1 + C'y_1^2 + D'x_1 + E'y_1 + 1 = 0$, uma vez que existem cinco variáveis são necessárias cinco equações para um sistema de equações lineares com uma possível solução, portanto escolhe-se um conjunto de pontos arbitrários $B' = \{z_1, z_2, z_3, z_4, z_5\}$ onde $z_i = (x_i, y_i)$ e $B' \subset B_{pq}$ tal que não existam dois pontos iguais em B', efetua-se o mapeamento de cada ponto em B' e então coloca-se as coordenadas de cada ponto em 6.3, obtendo o seguinte sistema de equações lineares:

$$
\begin{aligned}
x_1^2A' + x_1y_1B' + y_1^2C' + x_1D' + y_1E' + 1 = 0 \\
x_2^2A' + x_2y_2B' + y_2^2C' + x_2D' + y_2E' + 1 = 0 \\
x_3^2A' + x_3y_3B' + y_3^2C' + x_3D' + y_3E' + 1 = 0 \\
x_4^2A' + x_4y_4B' + y_4^2C' + x_4D' + y_4E' + 1 = 0 \\
x_5^2A' + x_5y_5B' + y_5^2C' + x_5D' + y_5E' + 1 = 0
\end{aligned}
\tag{6.4}
$$

Os valores encontrados para os coeficientes A', B', C', D', E' são os coeficientes da equação para o bissetor mapeado B^*_{pq}. Quatro dos cinco pontos em B' podem ser escolhidos em B_{pq} a partir de intersecções de retas com B_{pq}, como mostra a Figura 6.9 as intersecções de duas retas distintas com o bissetor B_{pq}(pontos a, b, c, d) podem ser quatro pontos distintos, no entanto, os cálculos das intersecções de duas retas com B_{pq} exigem duas vezes o uso de raiz quadrada.

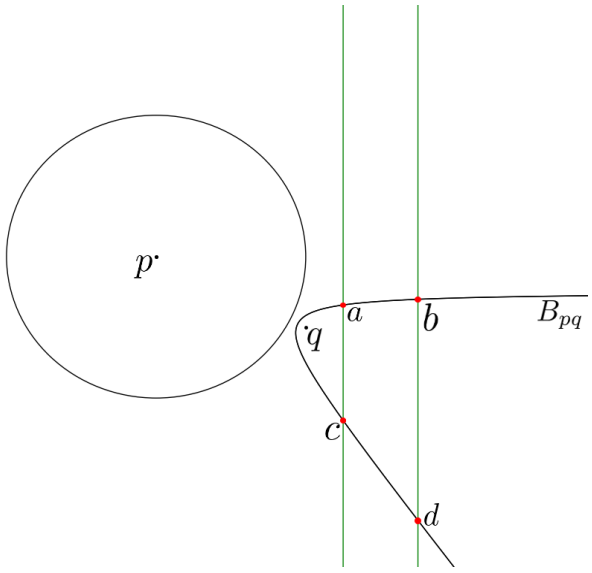

Figura 6.9 – Escolha de pontos em B_{pq}

Levando em conta que $w_q > w_p$ como o exemplo da Figura 6.9, apesar de q não pertencer ao bissetor B_{pq} pode-se utilizar q como um possível ponto para B', pois segundo o Lema 4.3 de [3] é verdade que $*(q) \in B^*_{pq}$, e após mapearmos os outros quatro pontos em B', obtém-se cinco pontos em B^*_{pq} o que torna possível a construção do sistema 6.4 (ver Figura 6.10).

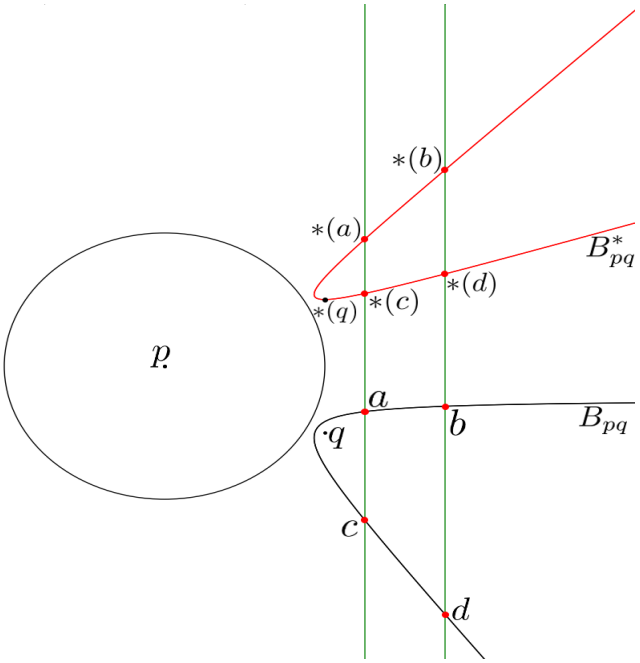

Figura 6.10 – Escolha de pontos em B_{pq}

Computar a solução do sistema 6.4 utilizando o método de eliminação de Gauss envolve o uso de 130 operações aritméticas, além disso, no capítulo três do livro [4] é mostrado um exemplo de que mesmo para sistemas com duas equações o método de Gauss possui instabilidade numérica, portanto ao invés de computar os coeficientes de B^*_{pq} apenas solucionando o sistema 6.4, é utilizada a equação 6.5 desenvolvida em [8] para construir cônicas a partir de cinco pontos no plano através de alguns determinantes envolvendo os cinco pontos.

$$[r, z_1, z_4][r, z_2, z_3][z, z_1, z_3][z, z_2, z_4] - [r, z_1, z_3][r, z_2, z_4][z, z_1, p_4][z, z_2, z_3] = 0 \qquad (6.5)$$

Onde os pontos z_1, z_2, z_3 e z_4 são os pontos retirados de B_{pq} e mapeados, r é o *site* mapeado com o maior peso entre p e q, z é um ponto qualquer do plano, e $[a, b, c]$ denota o determinante de uma matriz:

$$\begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix}$$

Para computar o bissetor B^*_{pq} no exemplo da Figura 6.10 basta fazer $z_1 = *(a), z_2 = *(b), z_3 = *(c), z_4 = *(d)$ e $r = *(q)$ e calcular os determinantes em 6.5.

Desenvolvendo os determinantes envolvidos na equação 6.5 e que não possuem incog-

nita (coordenadas do ponto z) tem-se:

$$\begin{aligned} det_1 &= [r, z_1, z_4] = r_y(z_{4x} - z_{1x}) + r_x(z_{1y} - z_{4y}) + z_{1x}z_{4y} - z_{4x}z_{1y} \\ det_2 &= [r, z_2, z_3] = r_y(z_{3x} - z_{2x}) + r_x(z_{2y} - z_{3y}) + z_{2x}z_{3y} - z_{3x}z_{2y} \\ det_{12} &= det_1 * det_2 \\ det_3 &= [r, z_1, z_3] = r_y(z_{3x} - z_{1x}) + r_x(z_{1y} - z_{3y}) + z_{1x}z_{3y} - z_{3x}z_{1y} \\ det_4 &= [r, z_2, z_4] = r_y(z_{4x} - z_{2x}) + r_x(z_{2y} - z_{4y}) + z_{2x}z_{4y} - z_{4x}z_{2y} \\ det_{34} &= det_3 * det_4 \end{aligned} \tag{6.6}$$

Obtendo a equação:

$$det_{12} * [z, z_1, z_3][z, z_2, z_4] - det_{34} * [z, z_1, p_4][z, z_2, z_3] = 0 \tag{6.7}$$

Cada um dos determinantes $[z, z_1, z_3], [z, z_2, z_4], [z, z_1, p_4]$ e $[z, z_2, z_3]$ representa um uma reta, isto é:

$$\begin{aligned} a_1 &= z_{3y} - z_{1y},\ b_1 = z_{1x} - z_{3x},\ c_1 = z_{3x}z_{1y} - z_{1x}z_{3y} \\ [z, z_1, z_3] &= a_1z_x + b_1z_y + c_1 \\ a_2 &= z_{4y} - z_{2y},\ b_2 = z_{2x} - z_{4x},\ c_2 = z_{4x}z_{2y} - z_{2x}z_{4y} \\ [z, z_2, z_4] &= a_2z_x + b_2z_y + c_2 \\ a_3 &= z_{4y} - z_{1y},\ b_3 = z_{1x} - z_{4x},\ c_3 = z_{4x}z_{1y} - z_{1x}z_{4y} \\ [z, z_1, p_4] &= a_3z_x + b_3z_y + c_3 \\ a_4 &= z_{3y} - z_{2y},\ b_4 = z_{2x} - z_{3x},\ c_4 = z_{3x}z_{2y} - z_{2x}z_{3y} \\ [z, z_2, z_3] &= a_4z_x + b_4z_y + c_4 \end{aligned} \tag{6.8}$$

Efetuando os produtos da equação 6.7 e agrupando os termos equivalentes pode-se concluir as seguintes equações para cada coeficiente:

$$\begin{aligned} A &= det_{12}a_1a_2 - det_{34}a_3a_4 \\ B &= det_{12}(a_1b_2 + b_1a_2) - det_{34}(a_3b_4 + b_3a_4) \\ C &= det_{12}b_1b_2 - det_{34}b_3b_4 \\ D &= det_{12}(a_1c_2 + c_1a_2) - det_{34}(a_3c_4 + c_3a_4) \\ E &= det_{12}(b_1c_2 + c_1b_2) - det_{34}(b_3c_4 + c_3b_4) \\ F &= det_{12}c_1c_2 - det_{34}c_3c_4 \end{aligned} \tag{6.9}$$

A soma total de operações aritméticas feitas em 6.6, 6.8 e 6.9 é igual a 99, ou seja, um número menor de operações do que a eliminação Gaussiana realiza.

6.2 Criação de fronteiras

A definição de fronteira para esta variante sofreu uma pequena alteração, visto que as hipérboles que assumem a forma de alguns bissetores após o mapeamento não possuem as mesma propriedades das hipérboles após o mapeamento para a variante de pontos sem peso. Seja B^*_{pq} um bissetor de dois *sites* com peso p e q tal que $p > q$, l' uma reta horizontal qualquer sobre ou acima de p, k_1 e k_2 dois pontos que resultam da intersecção entre l' e B^*_{pq} onde $k_{1x} \leq k_{2x}$, a fronteira C^-_{pq} é o trecho de B^*_{pq} que contém k_1 e C^+_{pq} é o trecho de B^*_{pq} que contém k_2, caso B^*_{pq} seja uma reta vertical somente C^-_{pq} será utilizada. A base de cada fronteira continua possuindo as mesmas características descritas na Seção 4.4.

6.3 Algoritmo para computar vértices do diagrama

Computar um vértice do diagrama para pontos com peso é o mesmo problema de computar um círculo tangente a três outros círculos(círculo de Apolônio), sabendo que um vértice v do diagrama é equidistante a três *sites* p, q e s e cada *site* pode ser visto como um círculo(como foi colocado no Capítulo 5), o vértice v deve ser equidistante a três pontos p', q' e s' pertencentes às respectivas circunferências dos *sites* p, q e s

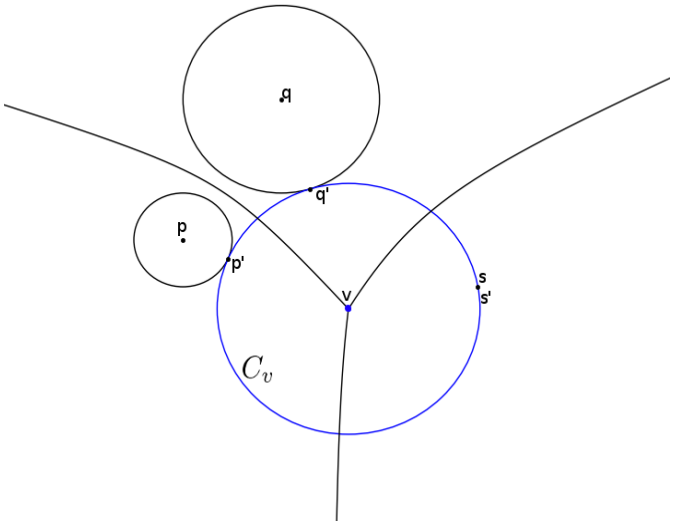

Figura 6.11 – Vértice do diagrama de Voronoi para pontos com peso / círculos

É importante deixar claro que o centro t de um círculo T tangente a três *sites* pode ser um vértice do diagrama, se T não contém nenhum *site*, isto é verdade devido à definição de vértice mostrada em 5.4 e 5.5.

A Figura 6.11 ilustra um exemplo de um círculo tangente a três *sites*, o raio do círculo representado pelo *site* s é 0, obviamente isso nem sempre será verdade. Sobre a distância de v com os *sites* é possível notar que:

$$\begin{aligned} dist(v, p') &= dist(v, p) - r_p \\ dist(v, q') &= dist(v, q) - r_q \\ dist(v, s') &= dist(v, s) - r_s \end{aligned} \tag{6.10}$$

Sabendo que $dist(v, p') = dist(v, q') = dist(v, s') = r_v$, onde r_v é o raio do círculo C_v que possui v como centro, pode-se fazer:

$$\begin{aligned} r_v &= dist(v, p) - r_p \\ r_v &= dist(v, q) - r_q \\ r_v &= dist(v, s) - r_s \end{aligned} \tag{6.11}$$

Colocando os raios r_p, r_q e r_s do lado esquerdo de suas respectivas igualdades e elevando ambos os lados de cada equação ao quadrado, tem-se:

$$\begin{aligned} (r_v + r_p)^2 &= (v_x - p_x)^2 + (v_y - p_y)^2 \\ (r_v + r_q)^2 &= (v_x - q_x)^2 + (v_y - q_y)^2 \\ (r_v + r_s)^2 &= (v_x - s_x)^2 + (v_y - s_y)^2 \end{aligned} \tag{6.12}$$

As incógnitas v_x, v_y e r_v envolvidas no sistema de equações 6.12, podem ser encontradas como foi mostrado em [9], ou seja, subtraindo a segunda equação da primeira equação, subtraindo a terceira equação da segunda equação, e então agrupando os termos equivalentes, chegando em:

$$\begin{aligned} Av_x + Bv_y &= F - Dr_v \\ Hv_x + Jv_y &= M - Kr_v \end{aligned} \tag{6.13}$$

Onde:

$$\begin{aligned} A &= 2(q_x - p_x) \\ B &= 2(q_y - p_y) \\ D &= 2(r_q - r_p) \\ F &= q_x^2 + q_y^2 - r_q^2 - p_x^2 - p_y^2 + r_p^2 \\ H &= 2(s_x - q_x) \\ J &= 2(s_y - q_y) \\ K &= 2(r_s - r_q) \\ M &= s_x^2 + s_y^2 - r_s^2 - q_x^2 - q_y^2 + r_q^2 \end{aligned} \tag{6.14}$$

Resolvendo o sistema 6.13 para v_x e v_y em função de r_v, obtem-se as seguintes equações lineares para os valores de v_x e v_y:

$$\begin{aligned} v_x &= \frac{(F - Dr_v)J - (M - Kr_v)B}{AJ - HB} \\ v_y &= \frac{A(M - Kr_v) - H(F - Dr_v)}{AJ - HB} \end{aligned} \tag{6.15}$$

Substituindo os valores de v_x e v_y em qualquer uma das equações do sistema 6.12 encontra-se uma equação quadrática em v_r, portanto é necessário utilizar raiz quadrada para descobrir o valor de v_r, esta solução apresentada em [9] não possui nenhuma análise de eficiência e robustez computacional. Já em [6] é apresentado um algoritmo para computar um círculo tangente a três outros círculos, e também é afirmado que o algoritmo é computacionalmente eficiente e robusto, porém não foi encontrado em [6] como realizar um dos passos do algoritmo, mas visto que é garantido a robustez de todos os outros passos este foi o método utilizado para computar os vértices do diagrama na implementação feita neste trabalho. A seguir é mostrado de forma breve como utilizar o método apresentado em [6], para computar o círculo de Apolônio T_0 mostrado na Figura 6.12.

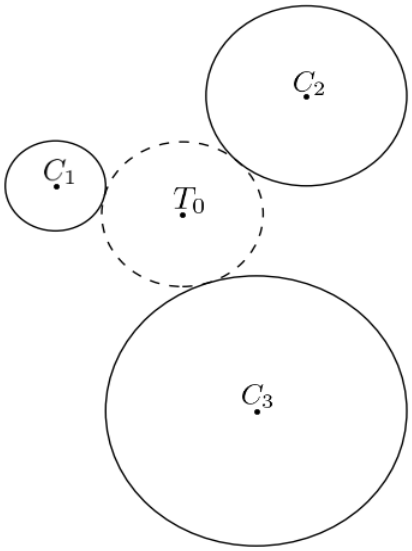

Figura 6.12 – Círculo tangente a três outros círculos

Considera-se que os círculos na Figura 6.12 estão contidos no plano euclidiano, e que o objetivo é computar o círculo tracejado T_0, primeiramente é transformado o problema de computar o círculo T_0, no problema de computar um círculo T_1 tangente a dois círculos e passando por um ponto. Seja $G_i = (C_i, r_i)$ para $i \in \{1, 2, 3\}$, onde C_i é o centro do círculo i e

para os raios r_i é verdade que $r_1 \leq r_2 \leq r_3$, é feito:

$$\begin{aligned} R_2 &= r_2 - r_1 \\ R_3 &= r_3 - r_1 \end{aligned} \tag{6.16}$$

É considerado que após as transformações feitas nos raios dos círculos em 6.16, os círculos $G_i' = (C_i, R_i)$ com $i \in \{2, 3\}$ e o ponto C_1 vão estar contidos em um plano complexo Z que contém pontos com a forma $z = x + iy$, resultando no cenário da Figura 6.13. Nesta mesma Figura é possível notar que o centro do círculo T_1 e o centro do círculo T_0 são o mesmo ponto.

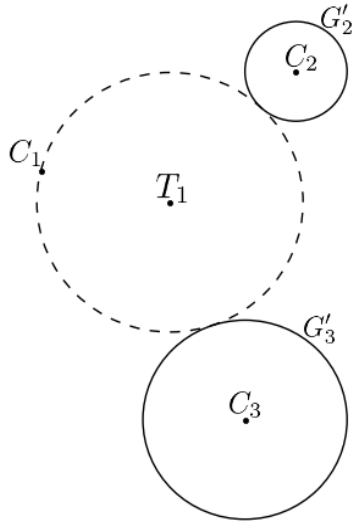

Figura 6.13 – Círculo tangente a dois outros círculos e passando por um ponto

O próximo passo é utilizar uma transformação de Möbius definida por $W(z) = \frac{1}{z - C_1}$, a transformação é útil para transformar o problema de computar o círculo T_1, no problema de computar uma reta tangente a dois círculos. A tranformação $W(z)$ realiza o mapeamento de um ponto z no plano Z, para um ponto w em um outro plano complexo W, esta transformação é útil pois possui algumas propriedades que facilitam o cálculo de T_1.

As propriedades afirmadas no Lema 3 de [6] transformam o problema de computar o círculo T_1 no plano Z, no problema de computar uma reta tangente a dois círculos no plano W, isso é possível pelo fato de que $W(z)$ mapeia círculos que passam pelo ponto C_1 no plano Z, para retas no plano W, além disso, $W(z)$ mapeia círculos que não passam pelo ponto C_1 no plano Z, para círculos no plano W, já o próprio ponto C_1 é mapeado para o ponto no infinito no plano W. Consequentemente, o círculo T_1 vai ser mapeado para uma reta L_1, e os círculos G_2' e G_3' que são tangentes a T_1 são mapeados para outros círculos, levando ao fato de que L_1 vai ser

tangente a G'_2 e G'_3 após o mapeamento dos mesmos, esta afirmação está provada no Teorema 4 de [6].

Cada círculo G'_i em Z é mapeado por $W(Z)$ para um círculo $W_i = (w_i, \frac{R_i}{k_i})$ no plano W, onde o centro $w_i = (\frac{C_{ix}-C_{1x}}{k_i}, \frac{C_{1y}-C_{iy}}{k_i})$, $k_i = (C_{ix} - C_{1x})^2 + (C_{iy} - C_{1y})^2 - R_i^2$ e $i \in \{2, 3\}$. O próximo passo é computar uma reta tangente aos círculos W_1 e W_2 que represente o círculo T_1 após o mapeamento $W(Z)$. A Figura 6.14 mostra as quatro possíveis retas tangentes a dois círculos.

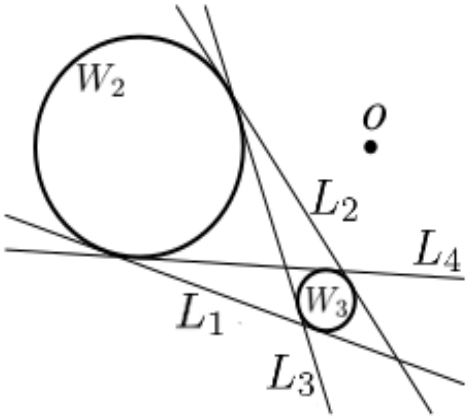

Figura 6.14 – Quatro possíveis retas tangentes a dois círculos, imagem retirada de [6] e editada. O ponto O representa a origem do plano

Das quatro retas possíveis mostradas na Figura 6.14 somente uma reta é resultado do mapeamento do círculo T_1, cada reta mostrada na Figura pode separar o plano em dois semi-planos, o trabalho [6] garante que a reta que separa o plano em dois semi-planos de tal forma com que os dois círculos W_2 e W_3 e a origem do plano fiquem no mesmo semi-plano, é a reta que representa o mapeamento do círculo T_1. Pode-se notar na Figura 6.14 a única linha que satisfaz esta condição é a linha L_1.

Como computar a linha L_1 é exatamente o passo que não foi encontrado em [6], para computar esta linha na implementação deste trabalho foi utilizado a construção explicada na solução do primeiro problema de [7], computar a linha L_1 desta forma exige uma vez o uso de raiz quadrada e de algumas operações aritméticas, por isto seria necessário verificar se este passo pode ser feito de forma robusta e eficiente, para então garantir com que todo o algoritmo seja eficiente. Seja $au + bv + 1 = 0$ a equação obtida para a linha L_1, é garantido por [6] que o centro de T_1 e T_0 é o ponto $z = (\frac{-a}{2} + C_{1x}, \frac{b}{2} + C_{1y})$.

6.4 Organização do *status* da linha varredora

O *status* da linha varredora para a construção do diagrama para pontos com peso vai conter somente as fronteiras já construídas e que estão sendo interceptadas pela linha varredora.

As operações necessárias no *status* continuam sendo as mesmas apresentadas na Seção 4.5. Portanto é possível evitar o uso de raiz quadrada durante as comparações feitas nas inserções das fronteiras. No entanto sempre é necessário utilizar ponto flutuante, devido ao modo com que as fronteiras foram construídas (Seção 6.1.1), dado este fato e a quantidade de operações aritméticas necessárias nos algoritmos para comparações que evitam o uso de ponto flutuante mostrados no Capítulo 4, pode-se considerar mais a possibilidade de se utilizar os algoritmos mais simples que utilizam raiz quadrada para efetuar as comparações.

Quando ocorre um ES para um *site* p deve-se verificar em qual região do diagrama o *site* p está contido, para que seja feita a repartição desta região, esses são os passos das linhas 7 e 8 do algoritmo 1. Para verificar em qual região p está contido, primeiramente é feito uma busca pela primeira fronteira C_{rs} à direita de p, uma vez que C_{rs} é encontrada é possível verificar em qual região p está contido, esses passos podem ser realizados como foi descrito na subseção 4.5.1.2. Porém, para o caso em que os *sites* possuem peso deve-se levar em conta que p pode ser dominado por r ou s, na próxima Seção é mostrado que p não pode dominar r ou s.

6.5 *Sites* dominantes

Um *site* q domina um *site* p quando:

$$w_p > dist(p, q) + w_q \tag{6.17}$$

Neste caso o *site* p está totalmente contido no interior da região do *site* q. Considerando que q domine p, a partir de 6.17 tem-se:

$$(w_p - w_q)^2 > (q_x - p_x)^2 + (q_y - p_y)^2 \tag{6.18}$$

Ou seja, $w_p - w_q > q_y - p_y$, logo $w_p + p_y > w_q + q_y$, portanto o *site* q vai ser encontrado antes do *site* p ser encontrado.

No entanto durante o desenvolvimento deste trabalho não foi possível tratar o caso em que existam *sites* dominantes na entrada, pois durante o processamento do ES para o *site* p, é possível verificar a primeira fronteira C_{rs} à direita de p, no entanto não foi possível concluir que se existe algum *site* t, tal que t domine p, t vai estar acessível a partir de C_{rs}.

7 CONCLUSÕES

Uma vez que as coordenadas dos *sites* são representadas apenas por números inteiros, é garantido que uma maneira de garantir a exatidão de um diagrama computado com o algoritmo de Fortune é utilizar apenas números inteiros, garantindo assim a precisão nas operações aritméticas. No Capítulo 4 foi realizado uma tentativa de evitar o uso de ponto flutuante na implementação do algoritmo de Fortune, no entanto a abordagem utilizada acaba por elevar muito rapidamente os valores das variáveis inteiras, trazendo a necessidade do uso de uma biblioteca para manipulação de inteiros grandes. Porém, o uso de uma biblioteca pode afetar o desempenho de tempo do algoritmo. Contudo, é possível realizar uma implementação robusta do algoritmo de Fortune para computar diagramas de Voronoi onde os *sites* são formados por pontos.

Para o caso em que os *sites* possuem peso foi necessário o uso de algumas restrições para os bissetores, mas dos casos de bissetores que foram levados em consideração, foi possível verificar a forma geométrica de cada um após a transformação geométrica, esta verificação é de muita importância, pois todos os passos do algoritmo exigem algum tipo de operação com bissetores mapeados. As operações de comparação entre os elementos do diagrama após a transformação geométrica podem ser feitas da mesma forma com que foram feitas para pontos sem peso, entretanto é necessário o uso de ponto flutuante, devido a maneira com que foram construídos os bissetores para pontos com peso.

As duas implementações do algoritmo de Fortune realizadas neste trabalho podem ser encontradas em [6].

[6] https://github.com/matheusdallrosa/voronoi-diagram-construction

REFERÊNCIAS

[1] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[2] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd edition, 2008.

[3] S. Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2(1-4):153–174, 1987.

[4] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[5] L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of delaunay and voronoi diagrams. *Algorithmica*, 1992.

[6] D. Kim, D.-S. Kim, and K. Sugihara. Apollonius tenth problem via radius adjustment and möbius transformations. *Computer-Aided Design*, 38(1):14–21, 2006.

[7] J. LAZERGES. Mathematics, advanced geometry, junior 8, 2009-2010. http://jml.ecole-alsacienne.org/Jingshan_Maths_English/@_JUNIOR_8/Junior8_oct28_ANSW.pdf, acesso em 2 de julho de 2017.

[8] J. Richter-Gebert. *Perspectives on projective geometry: A guided tour through real and complex geometry*. Springer Science & Business Media, 2011.

[9] B. Saelman. Determination of a circle tangent to three given circles. *Mechanism and Machine Theory*, 13(5):519–522, 1978.

[10] M. I. Shamos and D. Hoey. Closest-point problems. In *Foundations of Computer Science, 1975., 16th Annual Symposium on*, pages 151–162. IEEE, 1975.

[11] M. Sharir. Intersection and closest-pair problems for a set of planar discs. *SIAM Journal on Computing*, 14(2):448–468, 1985.