JUNÇÃO ESPACIAL DE REGIÕES POLIGONAIS USANDO CAMPOS ESCALARES

Ana Paula Teixeira Tinoco Xavier

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Claudio Esperança

Rio de Janeiro
Setembro de 2013

JUNÇÃO ESPACIAL DE REGIÕES POLIGONAIS USANDO CAMPOS ESCALARES

Ana Paula Teixeira Tinoco Xavier

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Claudio Esperança, Ph.D.

Prof. Julia Celia Mercedes Strauch, D.Sc.

Prof. Paulo Roma Cavalcanti, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2013

Xavier, Ana Paula Teixeira Tinoco

Junção Espacial de Regiões Poligonais Usando Campos Escalares/Ana Paula Teixeira Tinoco Xavier. – Rio de Janeiro: UFRJ/COPPE, 2013.

XII, 55 p.: il.; 29, 7cm.

Orientador: Claudio Esperança

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2013.

Referências Bibliográficas: p. 54 – 55.

1. Junções espaciais. 2. Polígonos. 3. Campos escalares. I. Esperança, Claudio. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Aos meus filhos Pedro e Thiago,
para que alcancem voos mais
altos que os meus.

Agradecimentos

A Deus, pela certeza de sua presença que nunca me abandona.

A minha família: meu marido Jonatas por me apoiar e compreender minhas ausências, minha mãe Estela e meus sogros Carmem e Ivanildo pelo apoio nas tarefas de casa e cuidado com meus filhos nas vezes em que precisei me ausentar.

À Marinha do Brasil, minha segunda casa e local de trabalho, pela oportunidade de cursar este mestrado.

Ao meu orientador, Prof. Claudio Esperança, pelas oportunidades que me ofereceu durante o mestrado, que me fizeram amadurecer, pelo aprendizado proporcionado e pela sua dedicação.

Ao Prof. Jano Moreira de Souza, pelo apoio durante o curso.

Aos professores Paulo Roma e Julia Strauch, por aceitarem prontamente participar da banca.

A minha amiga e Sicleidi, pelas aulas particulares, apoio, torcida e amizade de valor inestimável.

Aos meus amigos, pelo incentivo e apoio, especialmente a: Alayde, Cássia, Rita, Márcia e Mussia.

Aos funcionários do PESC, Solange, Sônia e Gutierrez, por sua colaboração nos procedimentos administrativos.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

JUNÇÃO ESPACIAL DE REGIÕES POLIGONAIS USANDO CAMPOS ESCALARES

Ana Paula Teixeira Tinoco Xavier

Setembro/2013

Orientador: Claudio Esperança

Programa: Engenharia de Sistemas e Computação

Este trabalho apresenta uma estrutura de dados para representação de regiões em Bancos de Dados Geográficos. Diferentemente da concepção tradicionalmente empregada, onde uma região é vista como um conjunto de pontos delimitado por linhas poligonais, regiões são representadas como campos escalares. As diversas regiões que compõem uma partição poligonal do plano são distinguidas utilizando uma função que mapeia cada ponto no plano em um valor de 0 a n. São descritos os algoritmos que permitem desenhar e realizar consultas espaciais sobre regiões, incluindo diversos tipos de junção espacial. Uma implementação-protótipo foi construida como prova de conceito, tendo sido empregada para realizar consultas de junção espacial utilizando diversos mapas de regiões.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

POLYGONAL REGION SPATIAL JOIN USING SCALAR FIELDS

Ana Paula Teixeira Tinoco Xavier

September/2013

Advisor: Claudio Esperança

Department: Systems Engineering and Computer Science

This work presents a data structure to represent regions in Geographic Databases systems. Unlike traditional conception, where a region is seen as a set of points bounded by polygonal lines, regions are represented as scalar fields. The various regions that make up a polygonal partition of the plane are distinguished using a function that maps each point on the plane to an integer value from 0 to n. Algorithms to perform spatial queries on regions, including various types of spatial join operations are described. An implementation prototype was constructed, and used to perform spatial join queries on several region maps.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Um sistema de banco de dados espacial é um sistema de banco de dados capaz de armazenar objetos espaciais, provendo tipos de dados adequados para este fim, bem como uma linguagem de consulta que permita manipular estes objetos, oferecendo, no mínimo, indexação espacial e métodos de junção. Sistemas de banco de dados espaciais oferecem a tecnologia de banco de dados fundamental para sistemas de informação geográfica e outras aplicações. [9]

Um objeto espacial é composto ao menos de um atributo espacial que descreve a geometria do objeto. Este atributo contém dados em duas ou três dimensões de um tipo comum, tais como pontos, linhas, polígonos e superfícies, bem como tipos ainda mais complexos compostos de tipos simples. Em uma analogia com bancos de dados relacionais, uma coleção de objetos espaciais definidos sob os mesmos atributos é denominada *relação espacial*. Por exemplo, a *relação espacial* Cidade (CNome, Codigo Postal, População, CRegião) contém o atributo geométrico *CRegião* que descreve suas fronteiras utilizando um polígono [10].

A representação de objetos espaciais é uma importante questão nas aplicações de bancos de dados espaciais, onde ainda persiste uma busca por aperfeiçoamento na construção de estruturas de dados que os suportem e realizem seu processamento de forma eficiente. Particularmente em sistemas de informações geográficas (SIG), onde o foco é a representação e análise de dados geográficos, este problema é de especial interesse.

O termo sistemas de informação geográfica (SIG) é usado para designar sistemas que realizam o tratamento computacional de dados geográficos. A principal diferença entre um SIG e um sistema de informação convencional é sua capacidade de armazenar tanto os atributos descritivos como as geometrias dos diferentes tipos de dados geográficos, sendo seu principal objetivo a representação e análise de dados geográficos.

Todo o dado geográfico possui três componentes <A; X; T >. “A” é “o que” (atributos), “X” é o “onde” “A” ocorre (posição) e “T” é “quando” “X” e “A” foram

medidos. Mais do que armazenar essas informações, um Banco de Dados Geográficos precisa ser capaz de recuperar e processar essas componentes [11]. Desta forma, um SIG consiste de mecanismos para gerenciar e armazenar dados, bem como analisar e manipulá-los eficientemente por meio algoritmos que permitam a recuperação e visualização do conteúdo.

Diferentemente da concepção tradicionalmente empregada em SIGs, onde uma região é vista como um conjunto de pontos delimitado por linhas poligonais, este trabalho propõe representar regiões como campos escalares. Assim, por exemplo, enquanto uma partição poligonal do plano em n regiões é tradicionalmente representada como uma coleção de polígonos, propomos distinguir as diversas regiões da partição através de uma função que mapeia cada ponto do plano em um valor de 0 a n. Para tanto, propomos uma nova estrutura de dados para representação de regiões em Bancos de Dados Geográficos. Neste estudo, o atributo geométrico correspondente a uma região é definido por coleções de arestas ponderadas, sendo cada aresta definida por seu ponto inicial e final, expressos cada um através de pares ordenados (x, y), e por um peso definido por um número inteiro. As operações de desenho, consulta de janela e *sobreposição de mapas* são solucionadas pela execução de algoritmos de varredura aplicados sobre a estrutura.

O objetivo deste trabalho é descrever algoritmos que permitam realizar operações de *sobreposição de mapas* de região utilizando a representação por arestas ponderadas. A principal contribuição é demonstrar que através da abstração de campos escalares é possível exprimir e realizar consultas espaciais sofisticadas sobre regiões, incluindo diversos tipos de junção espacial. Uma implementação-protótipo foi construida como prova de conceito, tendo sido empregada para realizar consultas de junção espacial utilizando diversos mapas de regiões.

Este trabalho está organizado da seguinte forma: No capítulo 2 são apresentados os conceitos de mapas poligonais, polígonos simples, partições poligonais do plano e mapas temáticos, em seguida, discute-se algumas formas computacionais utilizadas para a representação de polígonos. O capítulo 3 discute o processamento de consultas espaciais, com ênfase à junção espacial e sobreposição de mapas. O Capítulo 4 introduz a idéia de representação por campos escalares, descreve a estrutura de representação por vértices e apresenta a idéia de representação por arestas ponderadas, estrutura de dados proposta para representar regiões em Bancos de Dados Geográficos. O capítulo 5 descreve as operações sobre coleções de arestas ponderadas, que possibilitam manipular regiões poligonais definidas por campos escalares, entre elas, desenho, transformação escalar e algoritmo de varredura do plano. No capítulo 6, são mostrados alguns resultados obtidos com a implementação do protótipo e no capítulo 7 são apresentadas as conclusões do trabalho.

Capítulo 2

Mapas poligonais

Mapas são representações do espaço geográfico feitas geralmente em uma superfície plana com a finalidade de apresentar informações da realidade. Um mapa poligonal é uma subdivisão planar do espaço e representa eventos que podem ser particionados em entidades distintas e identificáveis, onde cada entidade é definida por uma fronteira fechada. O mapa da Figura 2.1 é um exemplo.

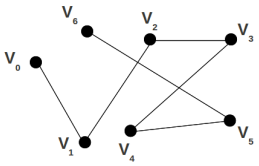
2.1 Polígonos simples

Algumas definições são necessárias: uma curva poligonal é uma sequência finita de segmentos de reta, denominados arestas, conectados ponto a ponto. Os pontos delimitantes das arestas denominam-se vértices (veja Figura 2.2(a)). Uma curva poligonal é fechada se seu último ponto delimitante corresponde ao primeiro, isto é, $v_0 = v_n$ (veja Figura 2.2(b)). Uma curva poligonal é simples se não há auto-interseção, ou seja, cada aresta não intersecta nenhuma outra exceto em seus pontos delimitantes que são compartilhados com a aresta adjacente.

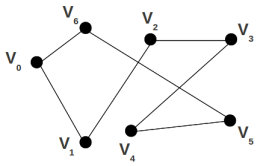
Figura 2.1: Mapa político do Brasil Fonte: Adaptado internet.

Um polígono simples é a região do plano delimitada por uma curva poligonal fechada simples (veja Figura 2.2(c)). A fronteira do polígono divide o plano em duas regiões: o interior e o exterior. Estes polígonos possuem comumente uma circulação predeterminada garantindo implícita ou explicitamente que ao percorrer seu bordo, o interior do polígono estará à esquerda ou à direita da ordem em que se atravessa a circulação. Normalmente a circulação dos vértices é anti-horária e desta forma, o interior do polígono estará à esquerda de sua borda.

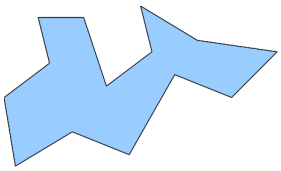
Além da representação por polígonos simples, regiões podem ser descritas por meio de polígonos simples com buracos, isto é, um polígono simples do qual buracos poligonais simples foram recortados. Esta representação é feita utilizando polígonos com circulações inversas, isto é, um polígono com circulação anti-horária representando a região e um ou mais polígonos com circulação horária representando os buracos. A representação poligonal do estado brasileiro de Goiás com a área do Distrito Federal recortada em seu interior, é um exemplo (veja Figura 2.2(d)). Podem ainda ocorrer casos em que um ou mais polígonos disjuntos referem-se a uma mesma feição de região. A representação poligonal do estado do Pará (veja Figura 2.2(e)) ilustra um exemplo.



(a) Curva Poligonal



(b) Curva poligonal fechada não simples



(c) Polígono simples

(d) Polígono simples com buraco representando o estado de Goiás

(e) Polígonos disjuntos representando o estado do Pará

Figura 2.2: Curvas poligonais

2.2 Partições poligonais do plano

Uma partição poligonal do plano divide o plano em polígonos onde cada ponto pertence somente a uma região, ou é compartilhado por dois ou mais polígonos em sua fronteira, isto é, qualquer par de polígonos da representação tem interseção nula, ou intersectam-se apenas em suas bordas. Assim, por exemplo, quando se deseja representar características de regiões, o plano é repartido em regiões poligonais, estando cada região associada a atributos como por exemplo: divisão política de estados, a distribuição de população, tipo de solo, etc. Desta maneira, o mapa poligonal que representa a divisão política do território brasileiro ilustrada pela Figura 2.1 é uma partição poligonal do plano, onde cada polígono representa um estado da federação.

Um mapa poligonal pode ser visto sob a forma de uma função em $\Re^2$ que associa pontos do plano a um identificador de região (número inteiro) ($f : \Re^2 \rightarrow \mathbb{N}$). Assim, cada polígono está associado a um escalar e representa a variação de regiões acerca de um tema. Observe que a associação de outros atributos a cada região pode ser feita através de funções que mapeiam o espaço de identificadores no espaço de atributos. Por exemplo, é possível atribuir nomes a regiões associando cada nome ao identificador correspondente.

2.3 Mapas temáticos

Praticamente todas as características do espaço geográfico podem ser representadas em um mapa. No entanto, tais características não podem ser agrupadas em uma única carta cartográfica, pois sua compreensão ficaria comprometida. Diante disso, os cartógrafos criaram mapas que abordam temas específicos, dando origem aos mapas temáticos. Como exemplos, temos: mapa político, que explicita as divisões territoriais; físico, que informa aspectos naturais; econômico, que relata as riquezas de uma determinada região; e histórico, que denota aspectos do passado.

Segundo [3], um mapa temático classifica as regiões em classes distintas sem ordem inerente, como rótulos que podem ser quaisquer símbolos. Um exemplo é o uso e cobertura da terra, com rótulos como "floresta", "área urbana" e "área agrícola".

Ao representar diferentes temas acerca de uma mesma parte do mundo, estes são organizados em estruturas de dados individuais chamadas de "camadas" (*map layers*) [5]. Desta forma, cada camada armazena somente um tipo de informação como, por exemplo, uma camada acerca de posições de cidades, uma camada a respeito do tipo de solo em cada região, uma camada com dados populacionais, etc [12]. A Figura 2.3 ilustra um mapa que descreve a variação climática no território brasileiro.

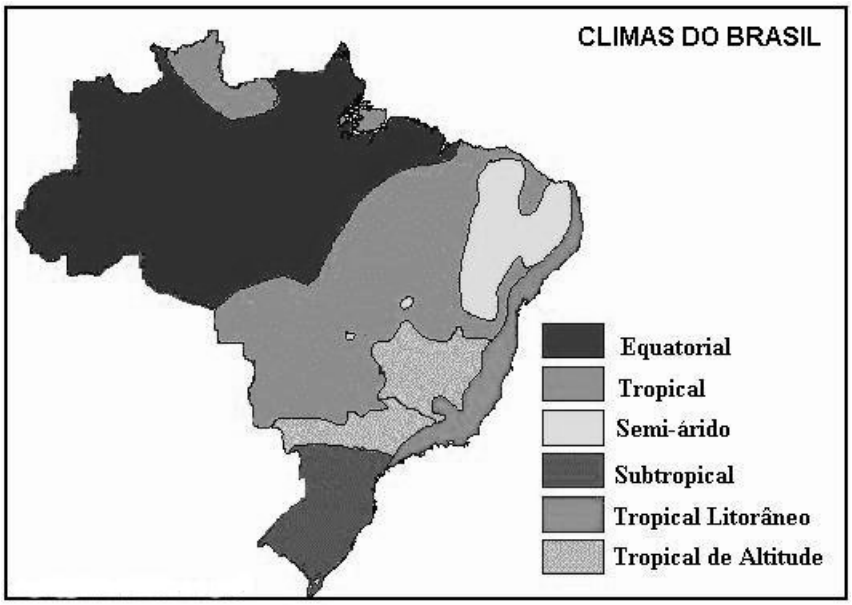


Figura 2.3: Mapa temático descrevendo a variação climática no território brasileiro. Fonte: Geografia para todos [1].

2.4 Representação de polígonos

Quando deseja-se representar geometricamente objetos espaciais, os tipos de representação utilizados em Bancos de Dados Geográficos podem ser divididos em duas grandes classes: estruturas vetoriais e estruturas matriciais (*raster*).

No modelo vetorial, a localização e a aparência gráfica de cada objeto são dados por coleções de pontos e vetores representados por coordenadas. É uma representação bastante intuitiva, que geralmente utiliza o sistema de coordenadas cartesianas. A representação de um elemento ou objeto é uma tentativa de reproduzí-lo o mais exatamente possível. Qualquer entidade de um mapa pode ser representado por três elementos gráficos: ponto, linha poligonal e área (polígono). Na maioria dos casos, um polígono é definido por uma ou mais circulações de vértices e um mapa poligonal é uma coleção contígua de polígonos que compartilham seus bordos [3, 13].

O modelo matricial supõe que o espaço pode ser tratado como uma superfície plana, onde cada célula está associada a uma porção do terreno. Consiste no uso de uma malha quadriculada regular de tamanho e formato uniformes (*raster*), sobre a qual se constrói, célula a célula, o elemento que está sendo representado. O espaço é particionado em um número finito de células de tamanho $N \times M$ comumente conhecidas como *picture element* ou sua contração *pixels*. A Figura 2.4 mostra um exemplo. A resolução do sistema é dada pela relação entre o tamanho da célula no mapa ou documento e a área por ela coberta no terreno.

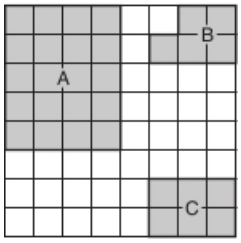


Figura 2.4: Raster Fonte: Samet [2]

2.4.1 Comparação entre representação vetorial e matricial

Mapas temáticos podem ser representados utilizando o modelo vetorial ou matricial. Segundo [3], para operações onde se requer maior precisão, a representação vetorial é mais adequada. As operações de álgebra de mapas são mais facilmente realizadas no formato matricial. No entanto, para um mesmo grau de precisão, o espaço de armazenamento requerido por uma representação matricial é substancialmente maior. Isto é ilustrado na Figura 2.5.

A representação vetorial representa bem qualquer tamanho de escala de trabalho e é mais adequada para representar relações espaciais entre os objetos. Seu armazenamento por coordenadas é mais eficiente e facilita na associação de atributos e elementos gráficos no banco de dados. Em contrapartida, a representação matricial representa melhor fenômenos com variação contínua no espaço e o processamento de algoritmos é feito, em geral, de forma mais rápida e eficiente. Entretanto, a representação matricial é mais adequada a pequenas escalas de trabalho. Por ser uma representação explícita, seu espaço de armazenamento é proporcional à área do polígono e à resolução da grade.

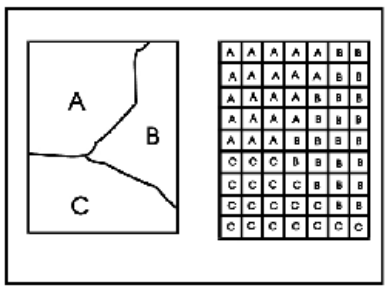


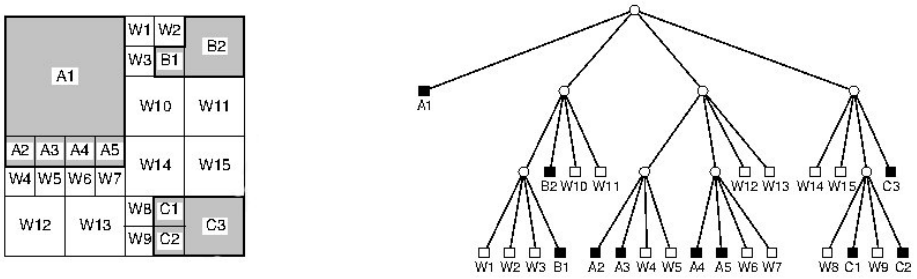
Figura 2.5: Comparação entre representações vetorial e matricial Fonte: Casanova [3]

2.4.2 Representações alternativas

É possível representar mapas poligonais utilizando representações alternativas tais como uma *quadtree* de região e suas variantes [2, 14, 15]. A *quadtree* de região é

um método hierárquico de divisão regular, onde o ambiente que contém o objeto é dividido recursivamente em 4 blocos até que este esteja completamente ocupado por um objeto ou esteja vazio. Os blocos são quadrados com lados cujo tamanho é correspondente a uma potência de 2. A Figura 2.6(a) representa a divisão da Figura 2.4, em uma *quadtree* de região.

A vantagem desta em relação à representação por raster é que esta representação implícita consome espaço de armazenamento proporcional ao perímetro das regiões e é implementada em uma estrutura de acesso de árvore como na Figura 2.6(b) correspondente à *quadtree* de região da Figura 2.6(a). É relativamente eficiente no suporte a consultas de classificação de pontos, já que pode-se determinar a região à qual pertence um ponto em $O(\log n)$, onde n é o número total de *pixels* de uma grade regular equivalente. Observe, entretanto, que grades podem responder a consulta de classificação de pontos em $O(1)$.



(a) *quadtree* de região

(b) Estrutura de acesso para a *quadtree*

Figura 2.6: Hierarquias de objeto Fonte: Samet [2]

Outra estrutura utilizada para mapas poligonais é a *PMR-quadtree* [15], uma variante da *PM-quadtree* [14] com representação baseada em bordos que visa representar mapas poligonais de forma exata. Não há distinção entre vértices e arestas e a subdivisão é realizada visando obter *buckets* que contenham um número de objetos próximo de um limite ótimo, por exemplo do tamanho de um bloco de disco. A regra de subdivisão visa obter uma subdivisão probabilisticamente ótima.

A *PMR-quadtree* é construída inserindo os segmentos um a um, a partir de uma estrutura vazia, formada por um só bloco. Cada segmento é inserido em todos os blocos que ele intersecta. Durante este processo, é verificado se a inserção ultrapassou o limite de objetos pré-estabelecidos para um bloco. Se um bloco excede sua capacidade, o bloco é dividido uma e somente uma vez em quatro blocos de igual tamanho. Se a subdivisão não for capaz de reduzir a população de um bloco para menos que o limite, os dados excedentes são colocados em blocos de *overflow*. Uma nova subdivisão será tentada se um novo dado for inserido naquele nó. A Figura 2.7 ilustra uma *PMR-quadtree*, onde o limite para divisão do bloco é igual a 2.

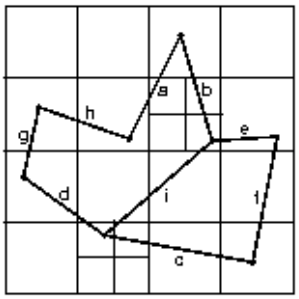


Figura 2.7: *PMR-quadtree* com limite de divisão igual a 2, onde os segmentos de reta foram inseridos em ordem alfabética. Fonte: Samet [2]

Capítulo 3

Processamento de consultas espaciais

Uma consulta espacial está relacionada com o atributo geométrico do objeto espacial. Por exemplo, uma consulta espacial comum é conhecida como consulta de "janela", e consiste em selecionar objetos espaciais que intersectam um retângulo dado. Já uma junção espacial é definida por duas ou mais relações. Uma junção espacial calcula um subconjunto do produto cartesiano combinando objetos espaciais destas relações de acordo com seus atributos geométricos, ou seja, estes atributos devem obedecer a um predicado espacial [10].

Devido aos grandes volumes de dados e também devido à alta complexidade dos objetos e consultas, os Bancos de Dados Geográficos impõem requisitos rigorosos ao processamento de consultas. Consultas espaciais tais como seleção por ponto, seleção por região, seleção por janela, seleção por vizinhança e junção espacial são operações básicas em um Banco de Dados Espacial. Estas operações servem de base para operações mais complexas baseadas em aplicação, como por exemplo a operação de *sobreposição de mapas* em SIGs. Portanto, uma implementação eficiente de consultas espaciais é um importante requisito para o bom desempenho de um banco de dados espacial e suas aplicações.

3.1 Junção espacial

Junção espacial é uma das operações mais frequentes em Banco de Dados Espaciais. O desempenho de um Banco de Dados Espacial é mais dependente de uma boa implementação de junções espaciais do que, por exemplo, por implementações pobres de consultas de janela. Isto porque uma junção espacial tende a acessar um mesmo objeto várias vezes, fazendo desta uma das mais caras operações em termos de consumo de recursos computacionais, custo este que depende em grande parte da

estratégia adotada para a execução de cada etapa no processamento da junção.

Existem muitas abordagens para o processamento da operação de junção espacial [16, 17]. Normalmente a execução é realizada em uma sequência de passos. A abordagem tradicional [18] prevê dois passos conhecidos como filtragem e refinamento. No primeiro deles é aplicado um método de acesso espacial que, na maioria dos casos, utiliza apenas uma aproximação das componentes espaciais a fim de reduzir o espaço de busca. Por exemplo, determinando o retângulo mínimo envolvente (*Minimum Bounding Rectangle (MBR)*). Este passo não produz o resultado da junção, mas um conjunto de pares candidatos que contém todas as respostas, embora possa conter pares de objetos que não cumprem o predicado de junção. O passo de refinamento determina se os objetos detectados pelo passo anterior verdadeiramente satisfazem o predicado, através da análise da exata geometria do objeto (3.1.a). Segundo vários autores, [4, 10, 17] este é o passo computacionalmente mais caro, exigindo tempo de I/O para procurar e ler os objetos espaciais a partir do disco e tempo de CPU para calcular a resposta exata. Esta abordagem em dois passos pode ser aumentada por um passo extra, incluído entre o primeiro e o segundo, que tem por objetivo promover uma etapa de filtragem mais refinada (3.1.b). Por exemplo, Brinkhoff *et al.* [10] utiliza um filtro geométrico para comparar os pares de candidatos resultantes do primeiro passo através de uma representação compacta e aproximada do objeto, tentando reter as suas características principais. Ou, ainda como exemplo, o trabalho de Azevedo *et al.* [4] que utiliza técnicas de rasterização na etapa de filtragem.

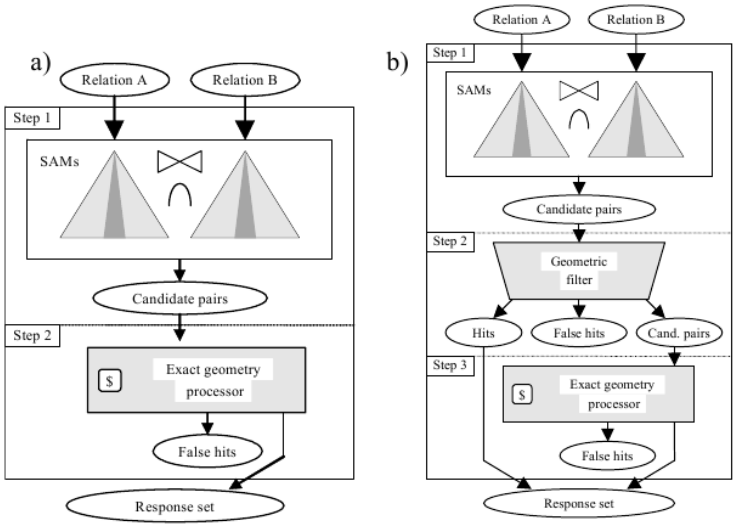


Figura 3.1: a)Arquitetura em dois passos para o processamento de junção espacial b) Arquitetura em três passos Fonte: Azevedo *et al.* [4].

Na etapa de refinamento, particularmente nos casos em que os objetos espaciais

correspondem a regiões, o processamento da consulta recai em problemas geométricos sobre polígonos [6, 9, 19]. Há ainda trabalhos que solucionam o problema da junção espacial sem utilizar-se de índices espaciais, suprimindo a etapa de filtragem [17, 20].

3.1.1 Sobreposição de mapas

Tipicamente, a inspeção de dados geográficos em um SIG é realizada através da visualização das camadas de um dado mapa. Por exemplo, para obter informações sobre o uso e cobertura da terra, o usuário solicitaria a exibição da camada que armazena esta informação. Entretanto, se o objetivo é visualizar as áreas do mapa onde, digamos, o grau de poluição é maior ou igual a 2 e o uso e cobertura da terra corresponde à florestas ou uso agrícola, seria necessário analisar ambas as camadas "poluição do solo" e "o uso e cobertura da terra" de forma a inferir as regiões onde as características desejadas se encontram. Para este fim, SIGs possibilitam a seus usuários operação conhecida como *sobreposição de mapas* (veja um exemplo na Figura 3.2).

A operação de sobreposição de mapas (*Map Overlay*) é um tipo especial de junção espacial onde dois ou mais mapas são combinados em relação a um ou mais predicados, gerando um novo mapa. A combinação de atributos temáticos ou de propriedades geológicas ou topológicas das áreas de entrada são controladas por uma função de sobreposição *overlay function* f, que é definida ou selecionada pelo usuário [5].

Quando duas ou mais camadas são mostradas juntas, interseções na sobreposição são áreas de especial interesse [12]. No exemplo apresentado da Figura 3.2 , o resultado representa a interseção das regiões com grau de poluição do solo maior ou igual a 2 na camada "poluição do solo", com as regiões que representam florestas e uso agrícola na camada "uso e cobertura da terra", e a respectiva função de sobreposição, correspondente aos atributos temáticos desejados.

Da mesma forma, se desejarmos visualizar uma relação entre o grau de poluição do solo e a finalidade de sua utilização, o resultado representará a união das camadas "poluição do solo" e "uso e cobertura da terra". Ou ainda, para visualizar áreas do mapa onde o uso e cobertura da terra corresponde à florestas e onde não existe grau de poluição do solo, o resultado representará a diferença entre as regiões que representam florestas no mapa temático "uso e cobertura da terra", com relação às regiões com algum grau de poluição no mapa "poluição do solo".

A avaliação da inteseção, união ou diferença ente polígonos são casos especiais do problema de sobreposição. [5]. Algoritmos eficientes para este fim têm sido extensivamente estudados [5, 6, 17, 19].

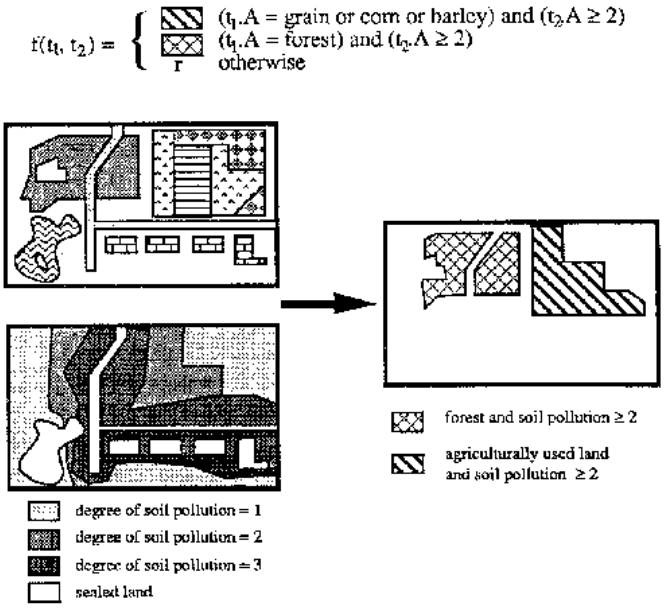


Figura 3.2: Exemplo de *Map Overlay* e uma *overlay function*. Fonte: Kriegel [5]

Margalit *et al.*[6] propôs um algoritmo para determinar a união, interseção e diferença entre polígonos. O algoritmo utiliza uma representação por bordos e é capaz de manipular polígonos irregulares na entrada ou na saída, utilizando para o processamento duas listas encadeadas e uma tabela *hash*. O domínio do algoritmo é a classe de polígonos denominada vértice-completos, cujas componentes mínimas irredutíveis são polígonos simples. Cada par destas componentes mínimas ou são disjuntas uma da outra e possuem a mesma orientação, ou uma é aninhada em relação a outra e possuem orientação oposta. Um polígono vértice-completo não possui arestas que se cruzam, mas pode possuir vértices coincidentes, arestas colineares e muitas outras sequências fechadas de arestas.

O algoritmo recebe como entrada um código de operação que determina a operação a ser processada (união, interseção ou diferença), um código indicador que especifica se o polígono de saída deve ser regular ou não, e duas listas encadeadas A e B contendo os polígonos a serem processados e seus tipos: *ilha* ou *buraco*. O algoritmo constrói como resultado uma série de polígonos irredutíveis e os armazena em uma lista encadeada C.

O processamento é realizado da seguinte forma: primeiro as orientações dos polígonos são normalizadas, em seguida os vértices originais de cada polígono são classificados como *interior*, *exterior* ou *bordo* em relação ao outro polígono e os vértices classificados dos dois polígonos da entrada são inseridos em duas listas circulares de vértices AV e BV, onde cada dois pontos adjacentes definem um fragmento de aresta. Depois, para cada aresta em um polígono, encontra-se todas as arestas do outro polígono que são intersectadas por ela, calculando todos os pontos de inter-

seção. Cada ponto de interseção é analisado segundo sua característica e inserido na lista circular correspondente. Então, cada fragmento de aresta pertencente a um polígono é classificado como *interior*, *exterior* ou *bordo* em relação ao outro polígono. Os fragmentos de aresta são selecionados e organizados de acordo com o tipo de operação e de polígono especificados na entrada e são então armazenados em uma tabela *hash EF*. Neste processo é utilizada uma tabela, ilustrada na figura 3.3 que sumariza o tipo de polígono e a operação a ser realizada com o tipo do fragmento de aresta que deve ser utilizado para produzir o resultado.

polygon types		operation							
		$A \cap B$		$A \cup B$		$A - B$		$B - A$	
A	B	A	B	A	B	A	B	A	B
island	island	inside	inside	outside	outside	outside	inside	inside	outside
island	hole	outside	inside	inside	outside	inside	inside	outside	outside
hole	island	inside	outside	outside	inside	outside	outside	inside	inside
hole	hole	outside	outside	inside	inside	inside	outside	outside	inside

Figura 3.3: tabela: Tipo de fragmento de aresta a ser selecionado de acordo com a operação e o tipo de polígono de entrada. Fonte: Margalit *et al.*[6]

Finalmente, o polígono resultante é construído varrendo a estrutura de dados *EF* de maneira que toda sequência fechada de aresta seja relatada como um polígono resultante e transferido para um *array* de saída, garantindo assim a formação do maior número de polígonos possíveis. A complexidade total de pior caso do algoritmo é $O((n_A.n_B)^2)$, onde n_A e n_B representam o número de vértices e arestas no polígono A e no polígono B respectivamente.

Kriegel et. al. em [5] propõe que a operação de sobreposição de mapas seja executada utilizando métodos de acesso espaciais juntamente com um algoritmo de varredura do plano (*Plane sweep*) [12]. Considerando-o adequado para solucionar o problema da sobreposição de mapas por definir e utilizar uma relação de ordem entre os objetos no plano, permitindo assim estabelecer uma partição espacial entre os mapas de entrada.

O funcionamento do algoritmo de varredura do plano segue, em linhas gerais, os seguintes passos: as coordenadas dos objetos (*event points*) são projetadas no eixo x e são processadas de acordo com a relação de ordem neste eixo. Os *event points* são armazenados em uma estrutura de dados (fila) denominada *event point scheduler*. Se novos *event points* forem computados durante o processo, a *event point scheduler* deve estar habilitada a inserí-los após sua inicialização. Uma linha vertical (*sweep line*) varre o plano da esquerda para a direita parando a cada *event point*. O estado do plano na posição em que se encontra a linha de varredura (*sweep line*) é guardado em uma tabela (*sweep line status*). Os *event points* que foram ultrapassados pela linha de varredura, são apagados da *event point scheduler*. A Figura 3.4 mostra um exemplo da *event point scheduler*, contendo os pontos iniciais

e finais das arestas que ainda não foram transpassadas pela linha de varredura, ordenadas por sua coordenada x e a *sweep line status*, que mostra o status da linha de varredura em um dado momento, contendo as arestas que a intersectam ordenadas por sua coordenada y.

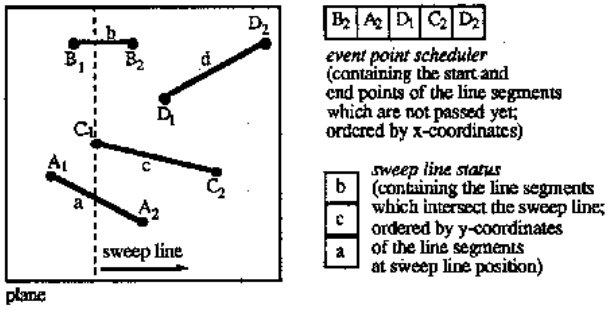


Figura 3.4: Exemplo de algoritmo de varredura do plano (*Plane sweep*). Fonte: Kriegel [5]

No algoritmo proposto por Kriegel et. al. [5], o ponto de partida é o trabalho de Nievergelt e Preparata [7], que utiliza um algoritmo de varredura do plano para determinar regiões em polígonos com auto-interseção. No algoritmo de Nievergelt e Preparata, os vértices e os pontos de interseção computados são armazenados na *event point scheduler*. As arestas que ora intersectam a linha de varredura são mantidas na *sweep line status* ordenadas pela sua coordenada y em relação a posição da linha de varredura. Os *event points* são classificados em umas das quatro categorias: *i) Start point* quando ambas as extremidades da aresta estão à direita do ponto, *ii) end point* quando ambas as extremidades da aresta estão à esquerda do ponto, *iii) bend* quando uma extremidade da aresta está à direita e a outra extremidade está à esquerda do *event point* e *iv) intersection point* quando duas arestas se intersectam no *event point*.

Para computar regiões que surgem entre as arestas, o status da linha de varredura guarda ainda dois ponteiros para a lista de pontos (*point list*) de cada aresta. A *point list* descreve os bordos da região. Uma lista correspondente é estendida por um *event point*, quando a linha de varredura encontra este ponto, dependendo da sua categoria. Para cada região entre duas arestas na *sweep line status* existe um número identificador de região (*region ID*). A Figura 3.5 apresenta um exemplo.

Kriegel et. al. [5] realiza uma generalização da classificação dos *Event points*, como sugerido por Nievergelt e Preparata [7], a fim de gerar um algoritmo capaz de manipular mapas arbitrários. Para efeito de representação, as regiões são descritas por polígonos simples com buracos em representação vetorial. As regiões calculadas pelo algoritmo formam as áreas do mapa de saída. Entretanto, o algoritmo deve ser capaz de identificar polígonos dos mapas de entrada cobrindo novas regiões. Estes

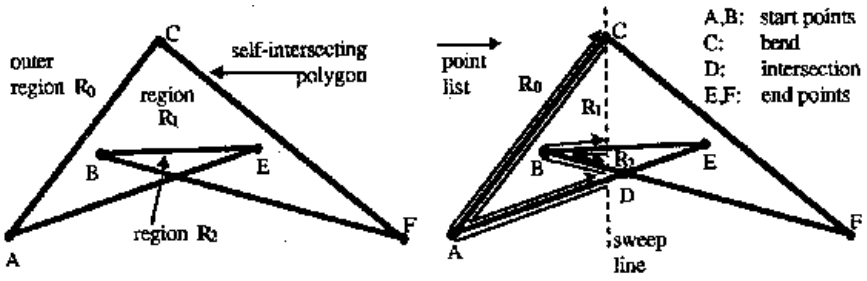


Figura 3.5: Exemplo do algoritmo de Nievergelt e Preparata. Fonte: Nievergelt e Preparata [7]

polígonos são denominados *parent polygons* e são necessários para calcular o valor do atributo temático de uma região pela função de sobreposição.

Para lidar corretamente com a lista de pontos, toda aresta que toca um *event point* é associada e ele. São consideradas as arestas que se iniciam, terminam ou cruzam o *event point*. Arestas que cruzam o *event point*, são consideradas como uma combinação de uma aresta que termina com outra que se inicia. Esta situação e o tratamento dos *event points* estão representados na Figura 3.6.

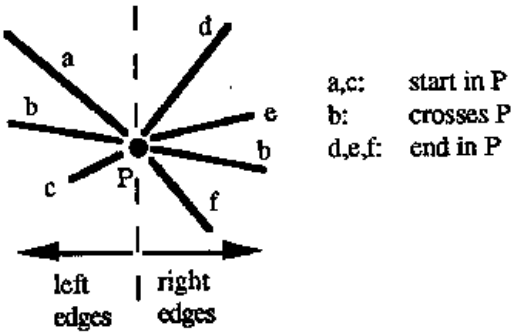


Figura 3.6: Tratamento dos *event points* realizado em duas etapas: 1.Trata as arestas à esquerda dos *event points* (n) 2.Trata as arestas à direita dos *event points* (m) . Fonte: Kriegel [5]

Quando mais de um polígono é tratado, os vértices ou pontos de interseção dos diferentes polígonos podem coincidir, e devem ser combinados em um *event point*. As arestas associadas a um *event point* são ordenadas por sua inclinação. Sendo P o *event point* atual, s_1 a s_n as arestas à esquerda e t_1 a t_n as arestas à direita associadas a P, ordenadas de cima para baixo. Para cada aresta k existem dois ponteiros $A(k)$ e $B(k)$. $A(k)$ aponta para extremidade da lista acima da aresta k e $B(k)$ aponta para o topo da lista abaixo da aresta k. O status da linha de varredura é denotado por y.

Dado um *event point*, o algoritmo analisa as arestas à sua esquerda (n) e, de acordo com o número de arestas encontradas, procede da seguinte maneira: veja Figura 3.7

Caso 1: $n = 0$ indica o início de uma nova região. Uma nova lista de ponto exterior é criada, consistindo somente do ponto P. $A(t_1)$ e $B(t_m)$ passam a apontar para esta lista.

Caso 2: $n = 1$ as duas listas de pontos referenciadas por $A(s_1)$ e $B(s_n)$ são extendidas pelo ponto P e anexadas aos ponteiros $A(t_1)$ e $B(t_m)$ respectivamente. s_1 é removido de y.

Caso 3: $n > 1$

Se $m = 0$**,** a lista de ponto exterior deve ser fechada. se $A(s_1)$ e $B(s_n)$ apontam para a mesma lista, esta lista, incluindo o ponto P descreve uma região ou um buraco poligonal. Senão, as listas e P são concatenadas.

Se $m > 0$ a lista de ponto exterior $A(s_1)$ e $B(s_n)$ são extendidas por P e anexadas a $A(s_1)$ e $B(s_n)$ tal como descrito no caso 2.

Para todas as regiões entre as arestas s_i e s_{j+1} existem ainda dois subcasos: Se $B(s_i) = A(s_{j+1})$ a lista interior $B(s_i)$, incluindo P descreve uma nova região que é armazenada no mapa de saída. Caso contrário, $A(s_{j+1})$,P e $B(s_i)$ são concatenados.

s_1,...s_n são removidos de y.

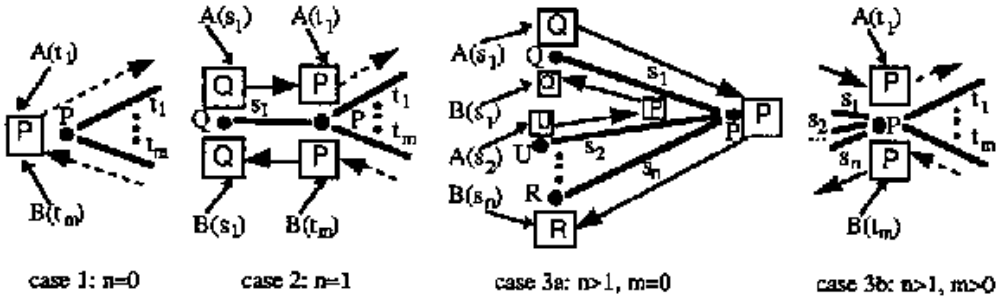


Figura 3.7: Tratamento das arestas à esquerda dos *event points*. Fonte: Kriegel *et al.* [5]

As arestas localizadas à direita dos *event points* são examinadas em uma segunda etapa do algoritmo. Adicionalmente, os pontos de interseção são computados e inseridos na *event point scheduler*. Três casos são analisados:

Caso 1: $m = 0$ as duas regiões, antes separadas pelas arestas s_1 e s_n, passam a ser idênticas (veja Figura 3.7 caso 3a) e os identificadores de região, se forem diferentes, devem ser ajustados. As arestas s_1 e s_n são retiradas da linha de varredura y e duas arestas de y, que antes não tinham relação de vizinhança, passam a ser vizinhas e necessitam ser testadas quanto a interseção.

Caso 2: $m = 1$ a aresta t_1 é iserida em y e testada quanto a interseção com suas arestas vizinhas.

Caso 3: $m > 1$ as arestas t_1 a t_m são inseridas em y. Então, t_1 é testada quanto a interseção com a aresta vizinha acima dela e t_m testada quanto a interseção com a aresta vizinha abaixo dela. Para todo $i = 1, ..., m-1$ uma nova região está começando entre as arestas t_1 e t_{i+1} recebendo um novo identificador de região. Além disso, uma nova lista de ponto interior consistindo de P é gerada onde os dois ponteiros $A(t_i)$ e $B(t_{i+1})$ referem-se a esta lista. Veja Figura 3.8.

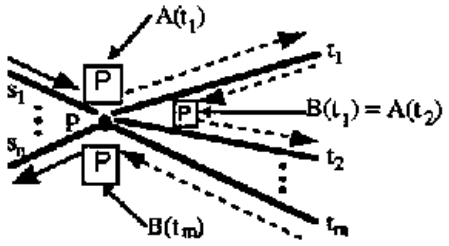


Figura 3.8: Tratamento das arestas à direita dos *event points*. Fonte: Kriegel [5]

A região que descreve um buraco poligonal pode ser associada a região de seu entorno através do identificador de região.

Na operação de sobreposição de mapas, é necessário identificar os *parent polygons* de uma região a fim de determinar o atributo temático de cada região no mapa gerado na saída. O algoritmo de varredura proposto permite uma simples e eficiente solução do problema pois a vizinhança das regiões que intersectam a linha de varredura são inerentemente armazemazenados pelo status da linha de varredura.

Em cada mapa de entrada M_i, existe exatamente um ou nenhum *parent polygon* para cada região avaliada. O algoritmo atribui a cada aresta de um polígono de entrada uma marca superior e uma inferior. A marca, que é definida, indica o lado onde o polígono está localizado e contém um ponteiro para o polígono. A outra marca é vazia. Enquanto os *event points* são combinados, arestas coincidentes são descobertas. O conjunto de *parent polygons* $P(R)$ de uma nova região R é identificado quando uma região está começando (Caso 3 da segunda etapa do algoritmo). A esta altura, os *parent polygons* da região Q situada acima de R são conhecidos, assim como as marcas para a aresta e entre R e Q.

Sendo n o número total de arestas de todos os polígonos e k o número total de interseções entre todas as arestas, se nenhum vértice coincide, o número máximo de *event points* encontrados é de $n + k$. Portanto, a operação na *event point scheduler* pode ser executada em $O(\log n)$ (porque $k < n^2$). Porque n é o número total de arestas, a operação no status da linha de varredura pode se executada em tempo $O(\log(n))$. Os vértices do polígono podem ser ordenados em tempo $O^*(\log(n))$.

Se assumido que o número de arestas anexadas a um *event point* é limitado por uma constante, a operação de sobreposição proposta por Kriegel *et al.*[5] pode ser executada em tempo: $t(n,k) = O((n+k)^* \log(n))$. A mesma complexidade do algoritmo de Nievergelt e Preparata [7].

O uso dos métodos de acesso espacial nesta abordagem parte do pressuposto que o Sistema de Banco de Dados do SIG utiliza um método de acesso espacial para gerenciar o acesso ao banco de dados, e que o acesso aos objetos do banco de dados é implementado em uma estrutura de acesso em árvore de diretório onde os nós internos são denominados *directory pages* e as folhas das árvores *data pages*. Juntos, eles definem uma partição do espaço. Um registro em uma *data page* consiste em ao menos no retângulo mínimo envolvente (MBR), no valor do atributo temático e em uma descrição do polígono ou em um ponteiro para esta descrição.

A partição do mapa é realizada utilizando métodos de acesso espacial e o algoritmo de sobreposição proposto. Somente os polígonos que intersectam a linha de varredura, ou próximos a ela são mantidos em memória. Isto implica em ler as *data pages* do meio de armazenamento secudário logo que a linha de varredura os intersecte. Esta abordagem é chamada de *sweep-line partition*. Contrariamente a uma partição orientada pela linha de varredura, uma partição ortogonal é igualmente possível de ser utilizada.

Sob o ponto de vista de banco de dados, uma junção espacial entre duas camadas de mapa de região poderia ser realizada obtendo-se o produto cartesiano das duas camadas de mapas a serem pesquisadas, seguido de uma operação de seleção sobre os itens que se deseja obter. Esta operação poderia ser facilitada pela adoção de uma estrutura de armazenamento mais eficiente para representar regiões, em substituição a representação por listas circulares de vértices, comum em SIGs.

Capítulo 4

Representação por meio de campos escalares

Um campo escalar é uma função $\Re^n \rightarrow \Re$, isto é, mapeia cada ponto de um espaço n-dimensional em um valor escalar, sendo que, em Bancos de Dados Geográficos, o domínio da função normalmente é algum espaço bidimensional associado à superfície da terra. Como exemplo, podemos citar mapas que indicam variação de temperatura ou pressão atmosférica (veja Figura 4.1). Entretanto, esta representação não limita-se ao espaço bidimensional, podendo ser considerada ainda a variação em função da altitude.

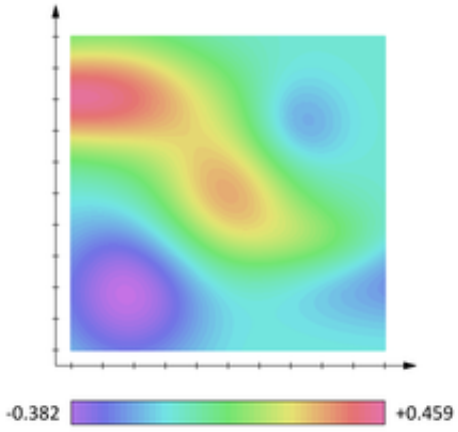


Figura 4.1: Campo escalar representando temperatura ou pressão, onde a intensidade do campo é graficamente representada por diferentes tonalidades de cor.

4.1 Representação por vértices

Esperança e Samet [8] descreveram uma estrutura de dados denominada *Representação por Vértices* (*Vertex Representation*) com a finalidade de representar e

manipular campos escalares ortogonais em aplicações de computação gráfica. Nessa estrutura, o interior e os bordos de regiões são representados por coleções de vértices que marcam os ápices de cones ortogonais infinitos. Cada cone é equivalente a um objeto ilimitado formado pela interseção de d semi-espaços ortogonais perpendiculares aos d eixos coordenados e que contêm o vértice.

Um *Vértice* é um ponto no espaço ao qual um valor escalar, denominado peso, foi associado. Uma *Representação por Vértices*, é um conjunto de vértices aos quais as seguintes restrições são impostas: (1) dois vértices não podem estar posicionados em um mesmo ponto no espaço, e (2) vértices não podem ter peso igual a zero.

A estrutura tem como objetivo modelar campos escalares discretos, cujas bordas são perpendiculares aos eixos coordenados. Assim, por exemplo, uma imagem digital, que mapeia células de uma grade retangular em tons de cinza, pode ser pensada como um campo escalar e portanto ser descrita por uma representação por vértices. De maneira semelhante, uma partição ortogonal do plano, onde regiões distintas recebem rótulos inteiros pode ser também descrita por uma tal representação onde o valor do campo corresponde ao rótulo de cada ponto do plano.

Uma *Representação por Vértices* nula representa um campo escalar nulo por definição, ou seja, um campo escalar onde todos os pontos são mapeados em zero. Quando considerado um conjunto contendo somente um vértice v colocado em um ponto $p(v)$ com peso igual a $w(v)$, então o efeito de v sobre o campo é adicionar $w(v)$ a todos os pontos q sujeitos a $p(v) \leq q$. A relação que denota "$\leq$" é definida da seguinte maneira: Seja $p1(v), p2(v), ..., pd(v)$, os d valores das coordenadas de um ponto $p(v)$, e seja $q_1, q_2, ..., q_d$ os d valores das coordenadas de um ponto q. Então, $p(v) \leq q$ se e somente se para todo $i = 1...d, p_i(v) \leq q_i$. É possível visualizar a região do espaço em que um vértice v é colocado sob a influência de $p(v)$ imaginando um cone de faces planas que tem seu ápice em $p(v)$, como exemplificado na Figura 4.2.

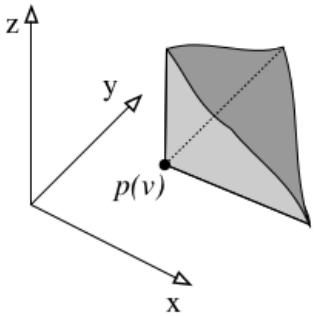


Figura 4.2: Cone de um vértice v colocado no ponto $p(v)$ em $\Re^3$. Fonte: Esperança *et al.* [8].

O campo escalar se modifica à medida que mais vértices são adicionados à representação. Se considerada uma representação V constituída de n vértices $v_1, v_2, ..., v_n$,

onde a posição do vértice v_i é denotada por $p(v_i)$ e seu peso por $w(v_i)$, então o valor do campo escalar Qv correspondente em um ponto q é denotado por:

$$Qv(q) = \sum_{p(v_i) \leq q} w(v_i)$$

A construção de uma representação capaz de mapear para 1 (um) todos os pontos do interior de um hiper-retângulo e para 0 (zero) os demais pontos inicia-se no espaço unidimensional, onde um retângulo é um intervalo definido por dois pontos a e b. Formalmente, dado um intervalo $R1 = (a, b)$, a construção do campo escalar Q_{R1} é definida por:

$$Q_{R1}(q) = \begin{cases} 1 & \text{se } a \leq q < b, \\ 0 & \text{caso contrário} \end{cases}$$

A representação é construída com dois vértices: Um na posição a com peso $+1$ e outro na posição b com peso -1 (veja a Figura 4.3).

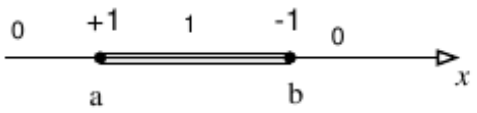


Figura 4.3: Representação por Vértices para um retângulo unidimensional. Fonte: Esperança *et al.* [8].

Em uma representação bidimensional do problema, um retângulo de eixos alinhados Q_{R2} pode ser definido por dois pontos extremos a e b. Construindo uma representação por campo escalar definida por:

$$Q_{R2}(q) = \begin{cases} 1 & \text{se } a_1 \leq q_1 < b_1 \wedge a_2 \leq q_2 < b_2, \\ 0 & \text{caso contrário} \end{cases}$$

A representação por vértices obtida no caso unidimensional provê a metade da resposta para o problema em duas dimensões. Adicionando um segundo valor de coordenada igual a a_2 para cada um dos dois vértices, obtém-se um retângulo que se estende de a_2 para o infinito na direção positiva do eixo y (Veja Figura 4.4.a). Resta então adicionar dois vértices em (a_1, b_2) e (b_1, b_2) com pesos -1 e $+1$ respectivamente (veja Figura 4.4.b).

A representação é facilmente extensível a dimensões maiores através da utilização de um processo recursivo.

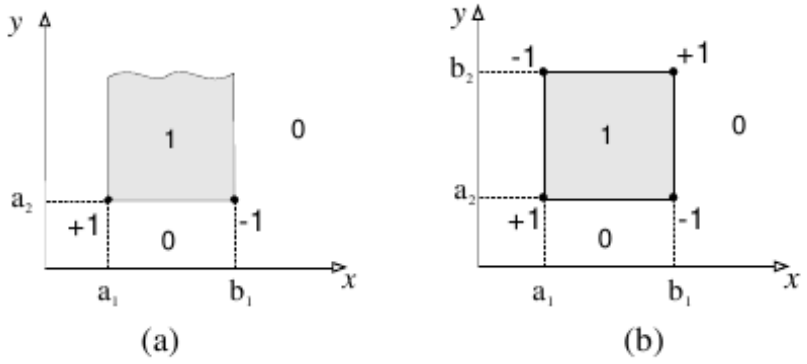


Figura 4.4: a)Retângulo unidimensional colocado em uma representação bidimensional b) e "fechado" com um segundo retângulo unidimensional. Fonte: Esperança *et al.* [8].

4.2 Representação por coleção de arestas ponderadas

Este trabalho propõe a adoção de uma nova estrutura de dados para representação de regiões em Bancos de Dados Geográficos, com o objetivo de simplificar a operação de sobreposição de mapas. A idéia tem como ponto de partida a *Representação por Vértices* proposta por Esperança e Samet [8], justamente por sua capacidade de representar objetos em uma partição ortogonal do plano, atribuindo um escalar para cada região distinta. Entretanto, para representar partições poligonais do plano, a estrutura deve ser capaz de representar e manipular objetos não ortogonais, como por exemplo, polígonos simples conforme descrito anteriormente e ilustrado pela Figura 2.2(c).

Uma representação por coleção de arestas ponderadas é uma extensão da idéia de *Representação por Vértices* [8], possuindo como diferencial a capacidade de representar polígonos não ortogonais por meio de campos escalares definidos por suas arestas não verticais. A seguir, são apresentadas algumas definições:

Uma *Aresta* é um segmento de reta associado a dois pontos no espaço, cada qual um par (x, y): um ponto inicial p_1 e um ponto final p_2. O ponto inicial p_1 é ocupado pelo ponto de menor coordenada y, ou pelo ponto de menor coordenada x no caso de ambos possuírem a mesma coordenada y, ou seja, quando se tratar de uma aresta horizontal. A cada *Aresta* é associado um valor escalar w (denominado peso) que indica uma variação do campo escalar w acima da aresta. A estrutura de dados *Coleção de Arestas* é uma lista capaz de representar partições poligonais do plano.

De forma similar a uma *Representação por Vértices* [8], uma *Coleção de Arestas* é uma lista de *Arestas* que obedecem as seguintes restrições: (1) duas arestas não

podem se sobrepor total ou parcialmente, e (2) arestas não podem ter peso igual a zero.

Uma *Coleção de Arestas* nula representa um campo escalar nulo por definição. Por exemplo, um campo escalar onde todos os pontos são mapeados em zero. Considerando uma coleção contendo uma única aresta a definida pelos pontos $p_1(a)$ e $p_2(a)$ com peso $w(a)$, o efeito de a é adicionar w a todos os pontos q, se e somente se q é contido na aresta a ou está acima desta.(Veja Figura 4.5)

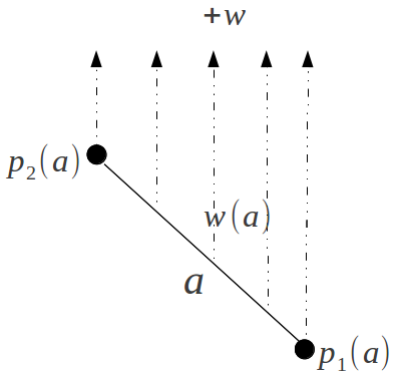


Figura 4.5: Coleção de Arestas composta por uma única aresta a e seu campo de influência $w(a)$

Em outras palavras, o campo correspondente à a mapeia um ponto q do plano segundo:

$$Q_a(q) = \begin{cases} w(a) & \text{se } cobre(a, q), \\ 0 & \text{caso contrário} \end{cases}$$

onde a função $cobre(a, q)$ é verdadeira se e somente se q está acima ou sobre a reta que contém a aresta a, e $xmin(a) \leq q_x < xmax(a)$, considerando que $xmin(a)$ e $xmax(a)$ são definidos, respectivamente como a menor e maior coordenada x de a.

O campo escalar se modifica à medida que novas arestas são acrescentadas à representação, de forma análoga ao que ocorre com os vértices em uma uma *Representação por Vértices* [8]. Se considerada uma representação V constituída de n arestas $a_1, a_2, ..., a_n$, então o valor do campo escalar Q_V correspondente em um ponto q qualquer é denotado por:

$$Q_V(q) = \sum_{a_i \in V} Q_{a_i}(q)$$

Por exemplo, ao acrescentarmos uma nova aresta b definida pelos pontos $p_1(b)$ e $p_2(b)$ com peso $w(b)$ ao exemplo da Figura 4.5, e considerarmos a existência de um ponto q posicionado acima do campo de influência das arestas a e b o valor do campo escalar em q será igual a $w(a) + w(b)$. Veja a Figura 4.6.

A tarefa de construir de uma representação que mapeie para 1 (um) todos os

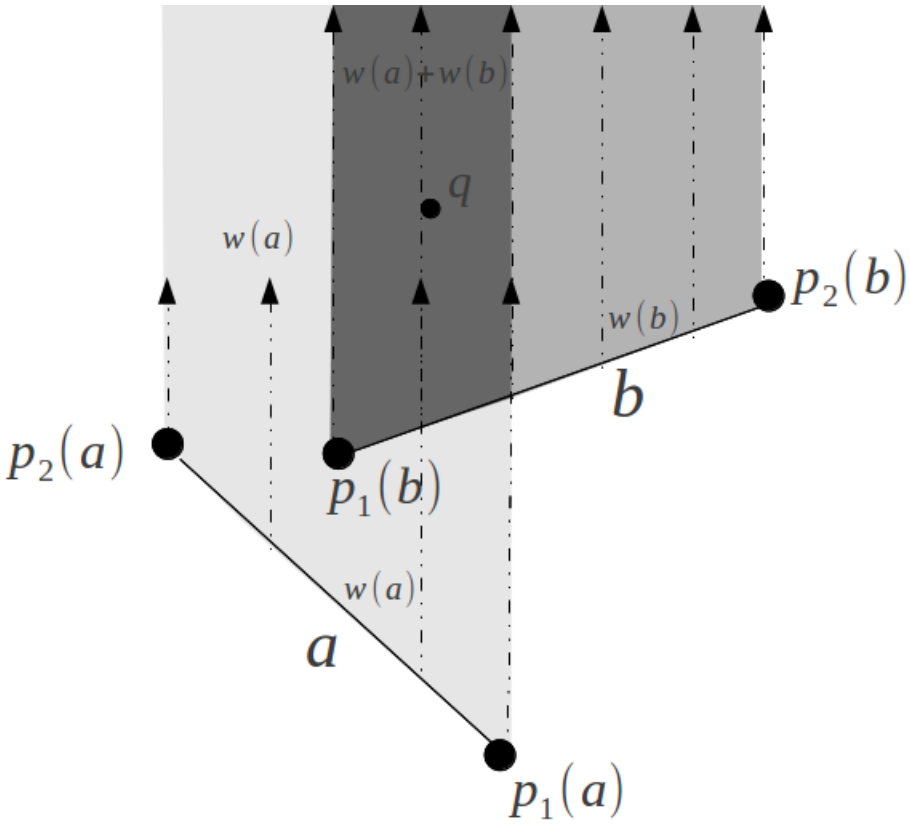


Figura 4.6: Coleção de Arestas composta por duas arestas a e b e a variação do campo escalar provocado por elas.

pontos do interior de um hiper-retângulo e para 0 (zero) os demais pontos, tal como ilustrado na Figura 4.4 com a utilização de uma *representação por vértices*, é realizada de forma mais simples com a estrutura de coleção de arestas, utilizando somente duas arestas horizontais, a primeira aresta a_1 com peso $w(a_1)$ mapeado para 1 e a segunda aresta a_2 com peso $w(a_2)$ mapeado para -1, anulando assim o campo de influência imposto pela aresta a_1. Veja a Figura 4.7. A representação armazena somente arestas não verticais do polígono, uma vez que arestas verticais não influenciam na avaliação do campo, que é feita de baixo para cima.

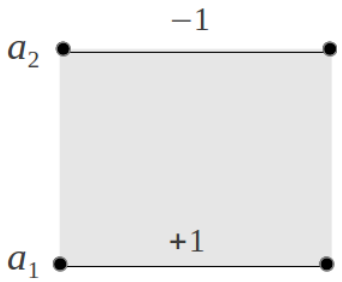
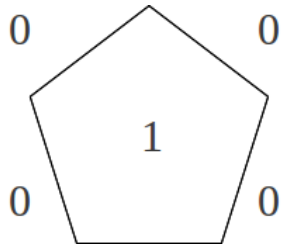


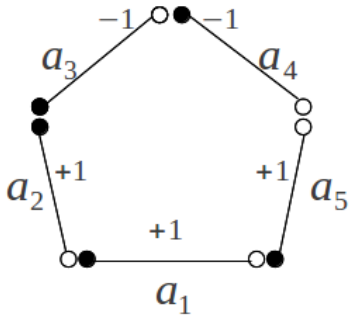
Figura 4.7: Representação por Coleção de Arestas formando um hiper-retângulo com interior mapeado para 1. Arestas verticais não são representadas.

Um outro exemplo é a construção de um polígono simples de 5 lados onde de-

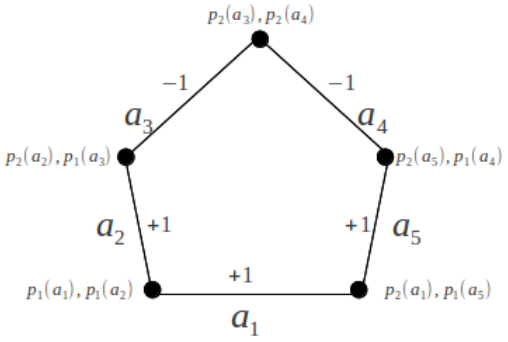
sejamos mapear todos os pontos em seu interior para 1 (veja Figura 4.8(a)), sua representação na estrutura *Coleção de Arestas* corresponde à Figura 4.8(b).



(a) Polígono simples de 5 lados onde todos os pontos no seu interior foram mapeados para 1.



(b) Representação do polígono em (a) usando arestas ponderadas, onde é possível visualizar o campo de influência exercido por cada aresta.



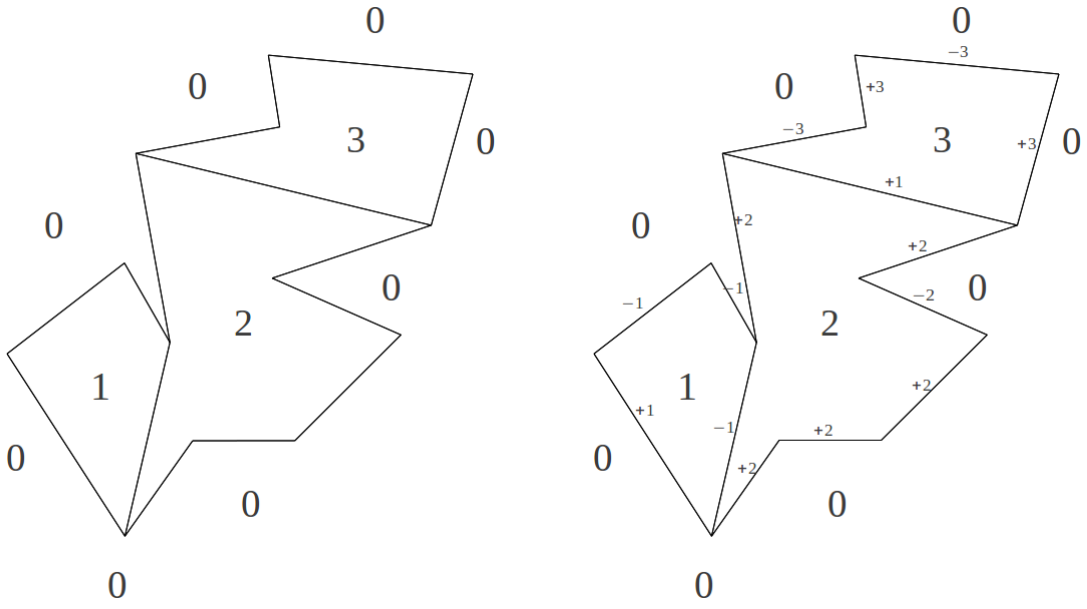
(c) Coleção de Arestas composta pelas arestas a_1, a_2, a_3, a_4 e a_5 representando um polígono simples de 5 lados onde seu interior foi mapeado para 1.

Figura 4.8: Construção de um polígono simples de 5 lados com interior mapeado para 1.

Note que na *Coleção de Arestas* representada na Figura 4.8(b), as arestas estão separadas somente para proporcionar uma melhor visualização, o limite estabelecido pelo campo de influência de cada aresta é ilustrado com terminador cheio em $xmin(a_n)$, que está sobre o campo de influência da aresta, e vazio em $xmax(a_n)$ que está fora do campo de influência da aresta. Em sua representação computacional, arestas podem ter pontos coincidentes: por exemplo, o ponto p_2 da aresta a_1 é coincidente com o ponto p_1 da aresta a_5 como mostra a Figura 4.8(c). A condição $xmin(a) \leq q_x < xmax(a)$ da função $cobre(a_n, q)$, garante o correto mapeamento de um ponto q qualquer no interior da figura e nos pontos coincidentes. Note ainda que o peso -1, associado às arestas a_3 e a_4 tem como efeito anular o campo de influência projetado pelas arestas a_1, a_2 e a_5 cujo peso é 1. Vale ressaltar que em qualquer ponto de a_3 e a_4 o campo escalar vale zero (0).

As arestas contidas em uma *Coleção de Arestas* são organizadas sob a forma de lista, obedecendo a uma ordenação de acordo com a ordem em que as arestas seriam encontradas se uma varredura do plano fosse realizada. Em se tratando de uma estrutura bidimensional, as arestas de uma coleção são ordenadas primeiramente pelo menor valor da coordenada y e em seguida pelo menor valor da coordenada x do ponto inicial da aresta (p_1). Quando duas arestas compartilharem o mesmo ponto inicial, a ordenação será realizada avaliando o seu ponto final (p_2). Segundo este critério, a ordenação das arestas no exemplo da Figura 4.8(c) será exatamente a_1, a_2, a_5, a_3 e a_4.

Uma partição poligonal do plano pode ser facilmente descrita por uma *Coleção de Arestas*. Tomemos como exemplo a partição poligonal do plano representada na Figura 4.9(a) onde cada polígono está associado a um escalar e representa a variação de regiões acerca de um tema. Observe que a associação de atributos a cada região pode ser feita através de funções que mapeiam o espaço de identificadores no espaço de atributos. Consideremos hipoteticamente que a figura representa o grau de desmatamento em uma determinada região: 1 para área de preservação, 2 para área em recuperação ambiental e 3 para área degradada. É possível atribuir nomes, ou mesmo cores a regiões associando cada nome ao identificador correspondente. A coleção de arestas correspondente a Figura 4.9(a) é apresentada na Figura 4.9(b). Observe que se existe uma aresta q que separa 2 regiões a e b com pesos $w(a)$ e $w(b)$ respectivamente, o peso atribuído à aresta q será exatamente $w(b) - w(a)$ se a região b está acima da região a e, $w(a) - w(b)$ se a região a está acima da região b



(a) Partição poligonal do plano onde cada região está associada a um escalar.

(b) Arestas ponderadas representando a partição poligonal do plano em (a).

Figura 4.9: Partição poligonal do plano utilizando coleção de arestas.

Capítulo 5

Operações espaciais usando coleções de arestas ponderadas

Neste capítulo são descritas as operações que possibilitam manipular regiões poligonais definidas por campos escalares. As operações de desenho e tranformação escalar, são solucionadas com o auxílio de um algoritmo de varredura do plano [12] aplicado sobre a estrutura. São definidas também algumas operações elementares essencialmente importantes para a execução de consultas sobre coleções de arestas ponderadas. Entre elas a função *cobertura*, utilizada pelo método que responde sobre o valor do campo em um determinado ponto (x, y).

Por simplicidade assumimos que as arestas de uma coleção estão em *posição geral* [1]. Isto significa por exemplo, que não existem arestas horizontais, três ou mais arestas não se intersectam em um mesmo ponto e, uma aresta não inicia ou termina sobre outra aresta.

5.1 Operações elementares

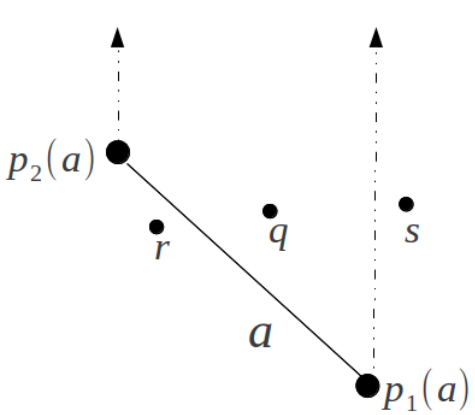
Uma aresta a, tal como definida no capítulo anterior, é capaz de responder algumas perguntas sobre si mesma:

Cobertura: Dado um ponto (x, y) no espaço, informa se este está sobre o campo de influência de a, conforme ilustrado na Figura 5.1(a).

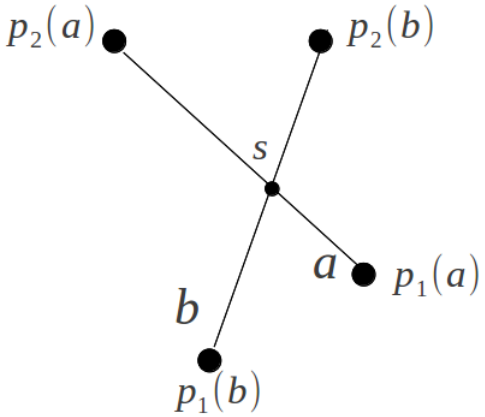
Interseção: Dada uma aresta a', informa se esta intersecta a e, caso positivo, informa o ponto (x, y) onde a interseção ocorre, conforme a Figura 5.1(b). Ou ainda, dada uma coordenada y, informa a coordenada x correspondente à interseção de a com a linha horizontal que passa por uma dada coordenada y, conforme ilustrado pela Figura 5.1(c).

[1] Termo comum na literatura de geometria computacional, que significa pressupor valores de entrada que ignoram tudo aquilo que possa confundir os conceitos geométricos com os quais se está lidando [12].

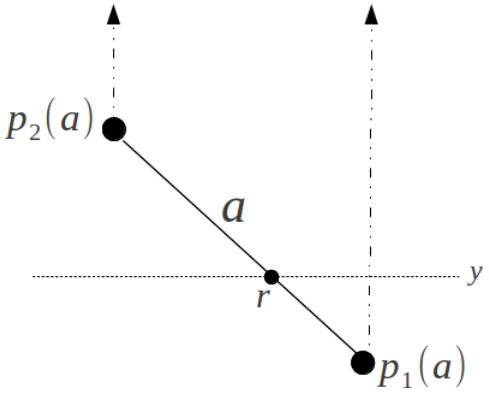
Inclinação: Dada uma aresta a, retorna sua inclinação em relação a uma cota y. A inclinação, também conhecida como declividade, é negativa se a reta apresenta um declive da esquerda para a direita e positiva se o declive é da direita para a esquerda. Esta classificação é útil em várias etapas do processamento de uma coleção de arestas, como por exemplo, para estabelecer a ordem entre duas arestas que possuam as mesmas coordenadas (x, y) em seus pontos iniciais p_1 e finais p_2. Veja a Figura 5.1(d).



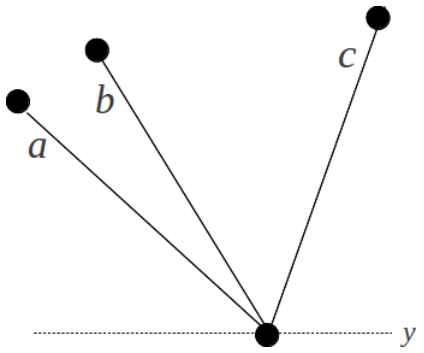
(a) Aresta a e os pontos q,r,s. Quanto à cobertura, somente o ponto q está sobre o campo de influência da aresta.



(b) Interseção das arestas a e b que ocorre no ponto s.



(c) Interseção dada pelo ponto r, da aresta a com a linha horizontal que passa por y.



(d) Inclinação de uma aresta em relação a uma cota y. As arestas a e b possuem inclinação negativa e a aresta c possui inclinação positiva.

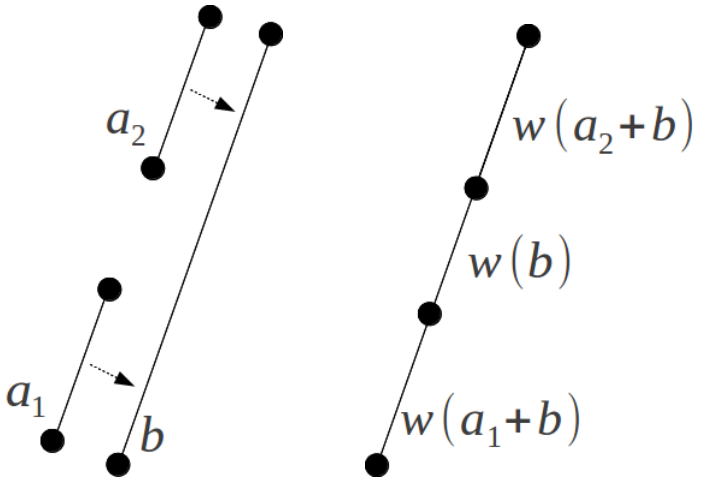
Figura 5.1: Operações elementares sobre arestas ponderadas

O processamento de uma *Coleção de Arestas* utiliza algumas operações elementares que são empregadas na maioria de seus algoritmos.

Multiplicação por um escalar: Multiplicar uma coleção de arestas C por um escalar α, cujo resultado é denotado por $\alpha.C$, significa multiplicar os pesos de todas as arestas contidas na coleção C por α, onde α é um inteiro diferente de zero.

Soma: A soma de duas coleções de arestas C e T, é uma coleção de arestas que

representa o campo escalar $Q_C + Q_T$. Esta coleção é composta de todas as arestas de C e T reordenadas conforme descrito no capítulo anterior. Um algoritmo de *merge* é usado para percorrer as duas listas simultaneamente e escolher qual aresta será a próxima a ser acrescida ao resultado. Se as duas arestas forem coincidentes e a soma de seus pesos for diferente de zero, então será acrescido ao resultado uma aresta na posição, com peso igual à soma dos pesos das arestas. É evidente que podem haver casos degenerados onde, por exemplo, somente parte de uma aresta é coincidente com outra. Neste caso, o algoritmo deve proceder a "quebra" da aresta em segmentos menores, somando os pesos correspondentes. Veja o exemplo da Figura 5.2(a). Imagine que ao somar duas coleções de arestas, encontramos as arestas a_1 de peso $w(a_1)$ e a_2 de peso $w(a_2)$, que são coincidentes com a aresta b de peso $w(b)$, a aresta b é particionada em três novas arestas que terão seus pesos calculados conforme mostra a Figura 5.2(b).



(a) Quando somadas, as arestas a_1, a_2 incidem sobre b.

(b) A Aresta b é particionada e a cada nova aresta á atribuído o peso correspondente.

Figura 5.2: Solucionando casos degenerados durante a soma de duas coleções de arestas.

5.2 Varredura

A operação de varredura é engrenagem principal no processamento de uma coleção de arestas. Implementada como um gerador [2], informa a cada alteração no campo escalar, o valor do campo abaixo de uma determinada coordenada y e o próximo ponto (x, y) onde o campo escalar sofrerá alteração.

No processamento de coleções de arestas, realiza-se uma varredura vertical do

[2]Em ciência da computação, um gerador é uma rotina especial que pode ser utilizada para controlar o comportamento de um *loop* de iteração.

plano, onde uma linha de varredura perpendicular ao eixo y percorre a coleção de arestas, partindo de $ymin$ até $ymax$. Veja um exemplo na Figura 5.3.

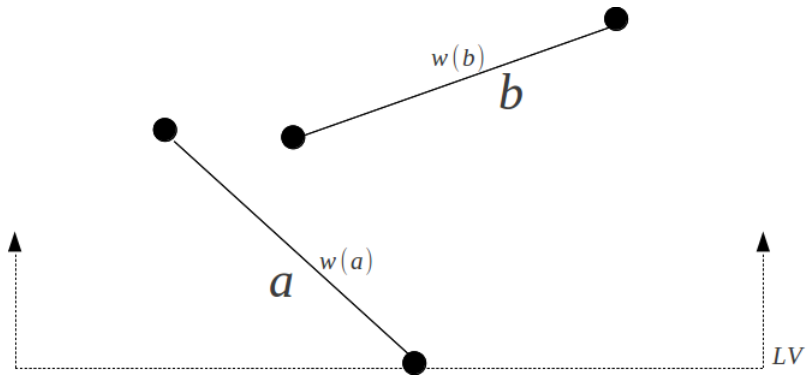


Figura 5.3: Linha de Varredura (LV) em uma Coleção de Arestas.

A linha de varredura é uma estrutura que armazena as arestas que a intersectam em um dado momento. Isto é denominado *status* da linha de varredura. O *status* da linha de varredura muda à medida que ela se movimenta em direção a $ymax$, porém isto não ocorre continuamente mas somente em alguns pontos onde uma atualização do *status* é necessária. Estes pontos recebem o nome de *event points* da linha de varredura [12]. *Event points* que ocorrem em uma mesma cota y, são processados em ordem crescente de x. Em uma coleção de arestas, os *event points* são os que marcam o início (BOTTOM), o fim das arestas (TOP) ou as interseções (INTERSECTION) entre arestas. Veja um exemplo na Figura 5.4.

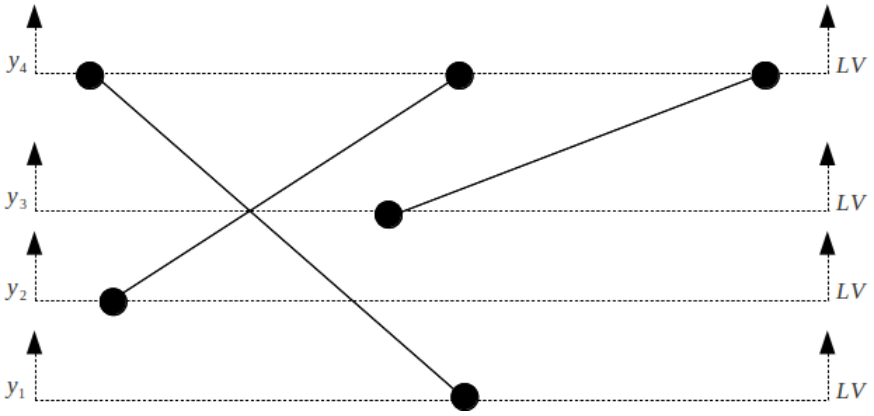


Figura 5.4: *Event points* de uma linha de Varredura (LV) em uma Coleção de Arestas. Enquanto se movimenta em direção a $ymax$, a linha de varredura para em y_1 em um evento do tipo BOTTOM, em y_2 em outro evento do tipo BOTTOM, em y_3 em um evento INTERSECTION e um BOTTOM e finalmente, em y_4 em três eventos do tipo TOP.

Um *evento* é um objeto que descreve um *event point* para o processamento da linha de varredura. Um evento de um determinado tipo ocorre em um ponto com

coordenada x, y e está associado a uma aresta. Os eventos de interseção possuem duas arestas associadas.

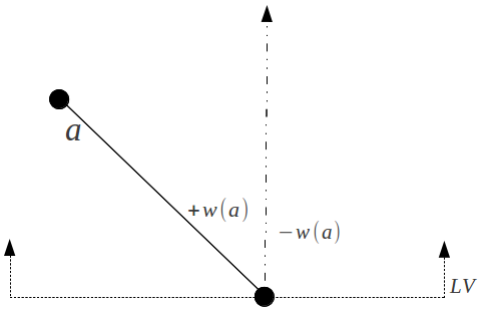
Os eventos estão organizados em uma *lista de eventos* implementada como uma fila de prioridades que obedece a seguinte ordem: primeiramente são processados os eventos de menor coordenada y; havendo dois ou mais eventos sobre a mesma coordenada y recebe maior prioridade aquele com menor coordenada x; se os eventos recaem sobre as mesmas coordenadas (x, y), é processado primeiro o evento cuja aresta possui a menor inclinação.

5.2.1 Arestas verticais

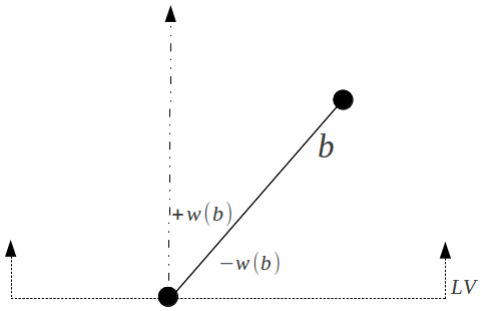
Apesar de não serem representadas em uma coleção de arestas, arestas verticais são necessárias durante o processamento da linha de varredura. Isto porque uma aresta, tal como definida para uma representação por coleção de arestas ponderadas, possui a propriedade de alterar o valor do campo escalar acima dela (vide Figura 4.5). Isto significa que cada aresta deixa uma espécie de "rastro" que se prolonga verticalmente a partir de suas extremidades estendendo-se até o infinito ou até que uma outra aresta anule ou incremente o campo escalar projetado por ela. Para representar este "rastro", utilizamos uma aresta vertical em cada extremidade da aresta a fim de delimitar o espaço onde acontece a alteração do campo escalar.

Durante o processamento da lista de eventos, quando um evento do tipo BOTTOM é encontrado, uma aresta vertical é adicionada à linha de varredura para delimitar o campo de influência da aresta. Desta forma, cada aresta que entra na linha de varredura é na realidade registrada como duas: a própria aresta e uma aresta vertical que funciona como um delimitador para o campo de influência. A Figura 5.5(a), mostra a entrada de uma aresta a na linha de varredura. Como a aresta a possui inclinação negativa, ela é inserida na linha de varredura com o peso $w(a)$ positivo e em seguida uma aresta vertical com o peso negativo é inserida em $x = p_1(a)$. Já o exemplo da Figura 5.5(b) mostra a inserção da aresta b, que possui inclinação positiva. Neste caso, uma aresta vertical com peso $w(b)$ positivo é inserida na linha de varredura na posição $x = p_1(b)$, e, logo em seguida, é inserida a aresta b com peso negativo.

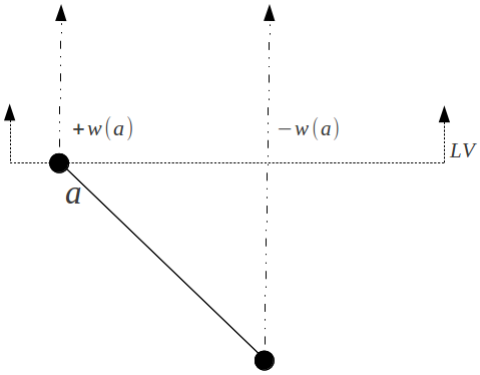
Quando a linha de varredura alcança o evento TOP, as arestas a e b são retiradas da linha de varredura e substituídas por uma aresta vertical em $x = p_2$. A inclinação da aresta determina o peso da aresta vertical (positivo ou negativo). Veja as figuras 5.5(c) e 5.5(d), referentes ao evento TOP das arestas a e b respectivamente.



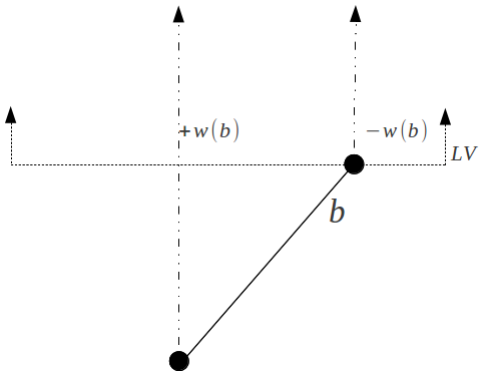
(a) Entrada de uma aresta a com inclinação negativa em uma linha de varredura.



(b) Entrada de uma aresta b com inclinação positiva em uma linha de varredura.



(c) Linha de varredura encontra o fim da aresta a.



(d) Linha de varredura encontra o fim da aresta b.

Figura 5.5: Utilização de arestas verticais no processamento da linha de varredura.

5.2.2 Processamento da linha de varredura

A *linha de varredura* é implementada como uma lista e registra as mudanças no campo escalar que ocorrem nos valores crescentes de x, em uma determinada cota y. Assume-se que antes do primeiro ponto, o campo escalar vale zero (0). Cada *elemento* sucessivo da linha de varredura registra uma mudança no campo escalar e consiste no ponto onde a aresta intersecta a linha de varredura em y constante e o correspondente incremento no campo escalar, que altera o valor do campo à direita da interseção, além da menor coordenada y afetada pelo elemento. Considerando o exemplo apresentado na Figura 5.6, vamos analisar as mudanças de status da linha de varredura enquanto ela percorre uma coleção de arestas realizando o processamento dos eventos. A representação é composta das arestas a, b e c cada qual com peso igual a 1, as arestas verticais adicionadas a linha de varredura serão referenciadas pela aresta a ela associada seguida de $'$ para a primeira aresta vertical adicionada e $''$ para a segunda.

Passo 1: Em y_0, antes de qualquer evento, o campo escalar vale zero (0). Ao avançar para y_1, a linha de varredura encontra o evento BOTTOM da aresta b que possui inclinação negativa e peso +1. São inseridas as arestas b e b' que delimitam o campo de influência de b. Note que em y_1, o campo vale 1 apenas

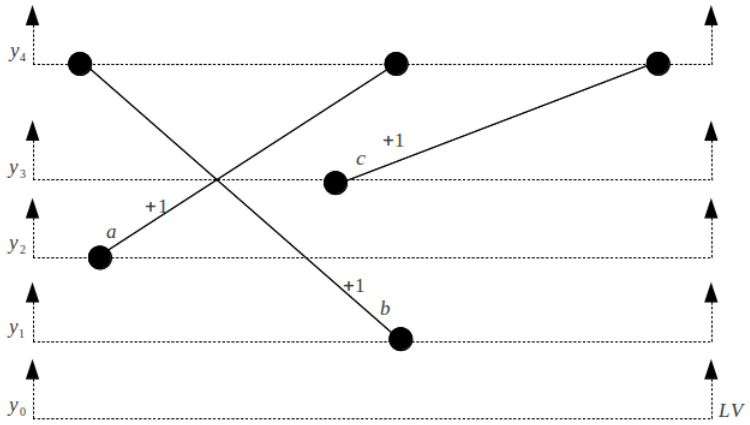


Figura 5.6: Varredura sobre uma coleção de arestas.

em um ponto: onde as arestas b e b' se iniciam. [3] A Figura 5.7 mostra o *status* da linha de varredura após processar o evento em y_1. Valores à direita das arestas indicam a variação do campo escalar.

Passo 2: A linha de varredura avança e para em y_2 no evento BOTTOM da aresta a. São inseridas as arestas a' com peso $+1$ e a aresta a com peso -1. O status da linha de varredura em y_2 pode ser visto na Figura 5.8.

Passo 3: Em y_3 a linha de varredura encontra um evento INTERSECTION, que por sua vez, provoca alteração no campo escalar e na ordem dos elementos (arestas) na linha de varredura: antes a', a, b e b' agora a', b, a e b'. Logo após, ocorre o evento BOTTOM da aresta c que insere as arestas c' com peso $+1$ e c com peso -1, alterando para a', b, a, c', c e b' a ordem dos elementos na linha de varredura. A Figura 5.9 mostra o status da linha de varredura após o processamento de todos os eventos em y_3.

Passo 4: A Linha de varredura encontra em y_4, três eventos TOP das arestas b, a e c nesta ordem. O evento é processado retirando as arestas originais da linha de varredura e inserindo as arestas b'' com peso $+1$, a'' e c'' com peso -1 cada. O *status* da linha de varredura após y_4 pode ser visto na Figura 5.10.

A cada inserção ou retirada de um elemento na linha de varredura, o algoritmo testa as vizinhanças à procura de interseção entre arestas que, quando detectada, gera um novo evento INTERSECTION a ser processado em um ponto mais adiante. Observe que entre y_3 e y_4 há três pontos onde o campo escalar se altera, conforme mostra a Figura 5.11. Estas interseções foram geradas durante a execução do algoritmo e processadas da mesma maneira que o evento de INTERSECTION em y_3

[3] Na verdade, o campo só passa a valer 1 um pouco acima deste ponto. Isto devido ao limite aberto da aresta dado pela equação $xmin(b) \leq q_x < xmax(b)$, que denota a variação do campo escalar em um ponto q acima ou sobre a reta que contém a aresta.

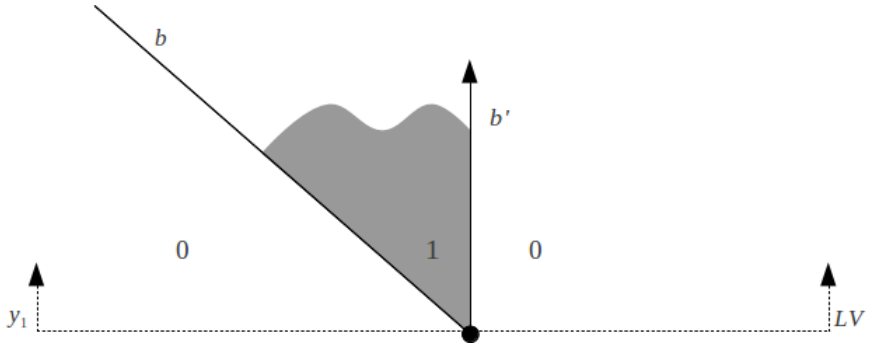


Figura 5.7: *Status* da linha de varredura em y_1.

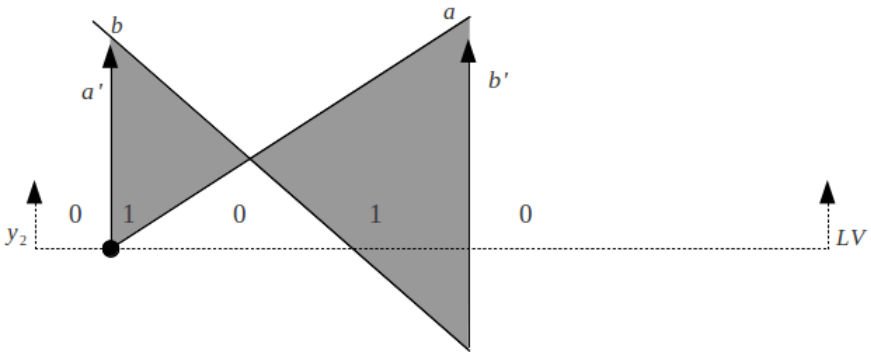


Figura 5.8: *Status* da linha de varredura em y_2.

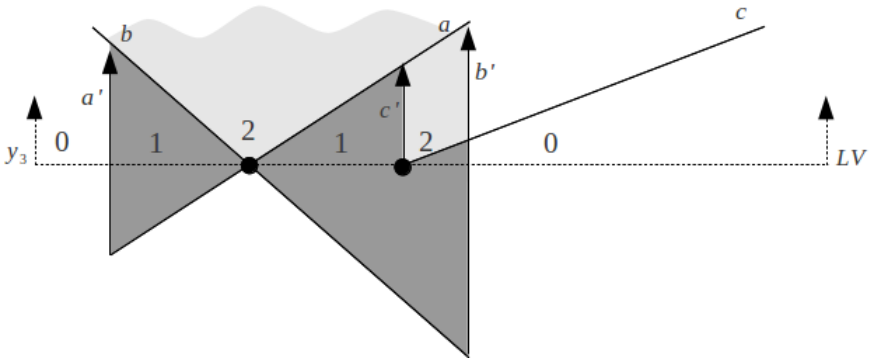


Figura 5.9: *Status* da linha de varredura em y_3.

descrito no passo 3. O tratamento destes eventos foram aqui omitidos com intuito de simplificar a explicação.

Durante o processamento da linha de varredura, registra-se todos os acontecimentos em uma determinada cota y, percorrendo a linha de varredura a partir de $xmin$ em direção a $xmax$. Desta forma, é possível avaliar o valor do campo escalar em qualquer ponto da linha de varredura e também em qualquer y acima dela desde que não haja inversão de ordem entre as arestas, ou seja, antes do próximo evento que causará alteração no campo.

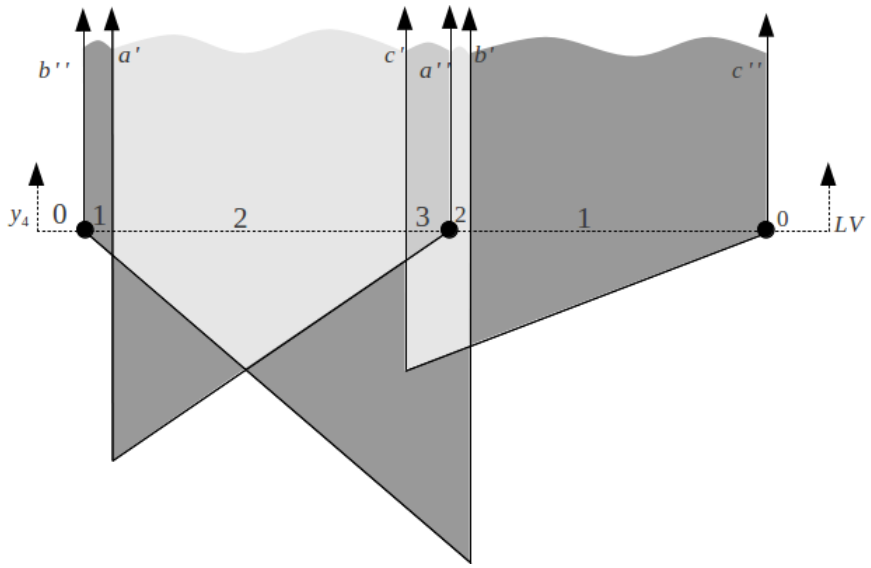


Figura 5.10: *Status* da linha de varredura em y_4.

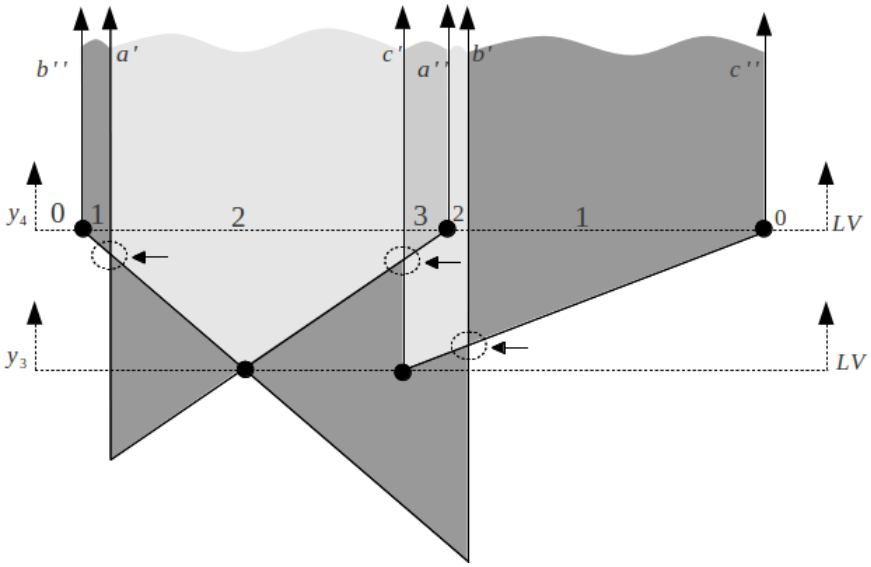


Figura 5.11: Eventos de interseção gerados durante o processamento da linha de varredura entre y_3 e y_4.

5.3 Desenho

A tarefa de desenhar polígonos representados por coleções de arestas necessita de uma abordagem diferente dos programas gráficos tradicionais que normalmente só desenham polígonos simples expressos por listas conectadas de vértices. Considerando que os polígonos que compõem uma coleção podem ser um pouco mais complexos, conter buracos ou ser delimitados por contornos desconectados, durante a operação de desenho a forma original da figura é recuperada utilizando um modelo de decomposição trapezoidal [12] orientada pelo algoritmo de varredura.

Ao percorrer a coleção de arestas, verifica-se o *status* da linha de varredura a

cada novo evento, avaliando as arestas que estão na linha de varredura a fim de decidir quando emitir os trapézios. A decomposição trapezoidal para o exemplo da Figura 5.6 é mostrado na Figura 5.12

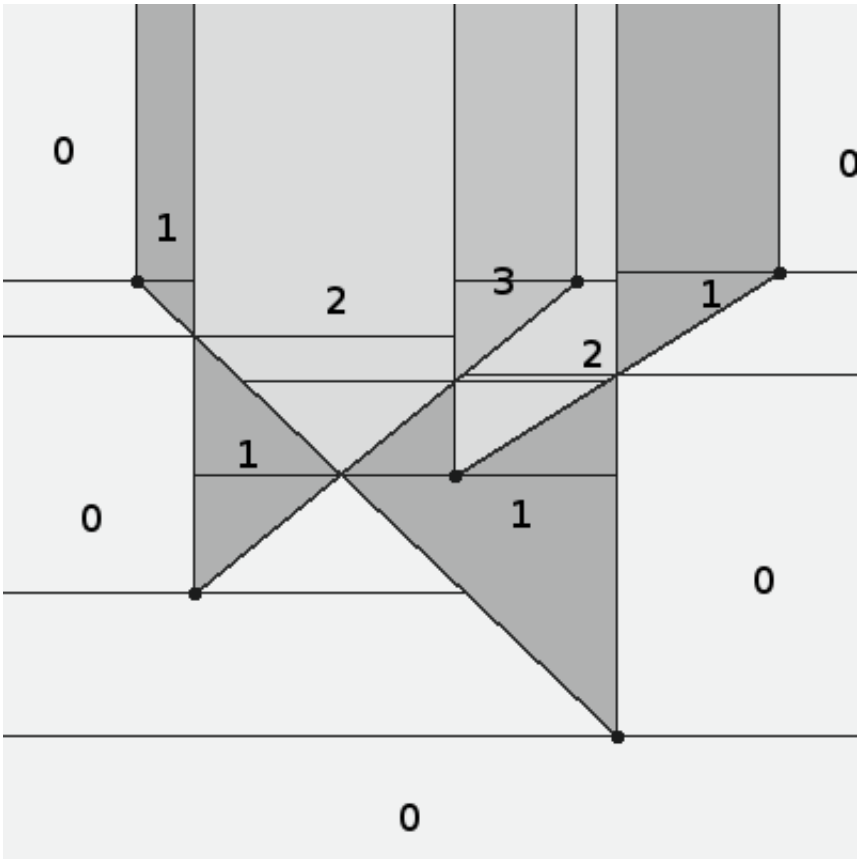


Figura 5.12: Trapézios desenhados para o exemplo da Figura 5.6. Observar que neste exemplo a variação do campo escalar estende-se ao infinito.

5.4 Transformação escalar

A transformação escalar é o operador mais importante no processamento de uma coleção de arestas, permitindo realizar diversos tipos de consultas espaciais sobre regiões. A forma como as coleções de arestas são representadas aliada à técnica de varredura aplicada à estrutura lhe conferem esta vantagem.

Tomemos como exemplo a Figura 4.9(b) aqui reapresentada na Figura 5.13(a). A divisão poligonal representa um mapa temático de uma região hipotética descrita por uma coleção de arestas. Observe que o peso atribuído a uma aresta q, que define a fronteira entre duas regiões a e b quaisquer, é exatamente $w(a) - w(b)$ quando uma região a está acima de uma região b ou vice-versa. Em outras palavras, o peso atribuído a cada aresta da coleção registra exatamente a diferença no valor do campo escalar entre uma região e outra, o que é vantajoso quando se deseja fazer junções de regiões, que podem ser obtidas somando (intercalando) duas representações.

Considere a Figura 5.13(b) que mostra um outro mapa temático acerca da mesma região. Imagine que deseja-se sobrepor estes mapas a fim de extrair informações sobre a relação entre crescimento populacional e degradação do meio ambiente. A sobreposição dos mapas pode ser facilmente executada realizando a soma das duas coleções de arestas. O resultado é uma nova representação poligonal combinando os dois temas conforme mostra a Figura 5.13(c).

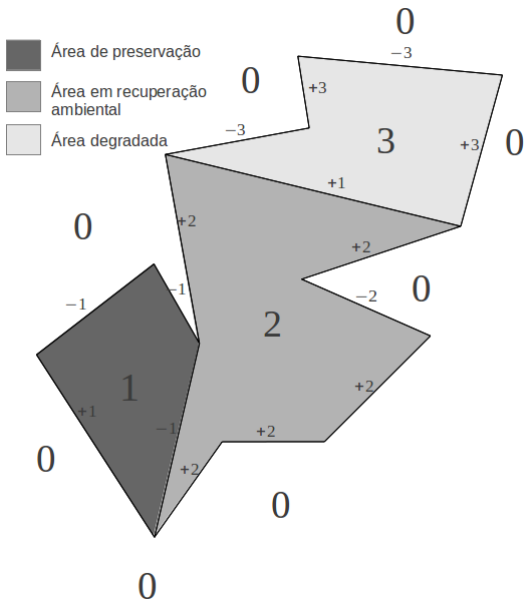
A partir do resultado da soma das regiões aplicamos uma transformação, isto é, uma função de $\Re \rightarrow \Re$ ao campo escalar. Combinando a operação de transformação com a soma de campos escalares, pode-se obter a operação de junção espacial desejada. Por exemplo, seja I a coleção de arestas que representa o mapa temático da Figura 5.13(a) e J a coleção de arestas que representa o mapa temático da Figura 5.13(b), deseja-se obter um mapa K com regiões cujos rótulos são dados por: 10 para região de área degradada com população superior a 200 mil habitantes e 20 para região de preservação com população menor que 100 mil habitantes, sendo as demais regiões mapeadas para 0. Para obter o resultado, somamos as coleções I e J obtendo a coleção K' representada pela Figura 5.13(c) aplica-se a esta uma função de transformação escalar para se obter a coleção final k mostrada na Figura 5.13(d). A função para esta consulta é expressa a seguir:

$$f(x) = \begin{cases} 10 & \text{se } x = 303, \\ 20 & \text{se } x = 101, \\ 0 & \text{caso contrário.} \end{cases}$$

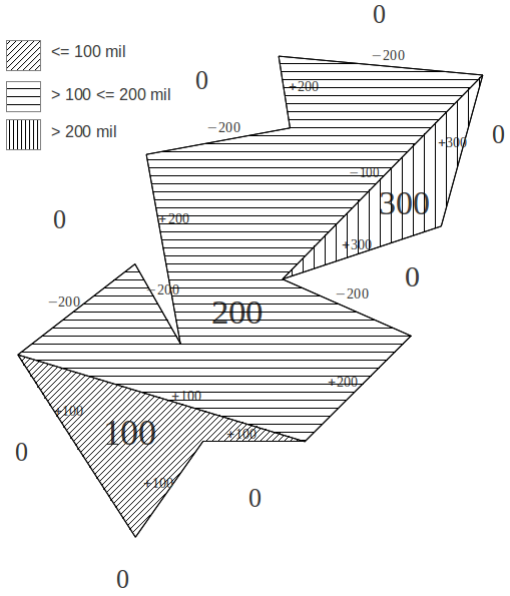
De forma similar, para realizar operações de união, interseção e diferença entre regiões expressas por coleções de arestas basta somar as coleções e aplicar uma transformação escalar. Consideremos duas coleções de arestas C e T, cada qual representando um polígono onde todos os pontos de seu interior foram mapeados para 1, veja a Figura 5.14(a). Consideremos ainda que quando somadas $(Q_C + Q_T)$, produzem um campo escalar onde todos os pontos no espaço estão mapeados para 0, 1 ou 2 (observe a Figura 5.14(b)). Pontos mapeados para 0 estão fora de C e fora de T; pontos mapeados para 1 estão no interior de C ou no interior de T e pontos mapeados para 2 estão no interior de ambas as coleções C e T. Então, o problema de computar $C \cup T$ fica reduzido ao problema de computar uma coleção de arestas que represente a função $f \cup (Q_{C+T})$, onde:

$$f_{\cup}(x) = \begin{cases} 1 & \text{se } x > 0 \\ 0 & \text{caso contrário} \end{cases}$$

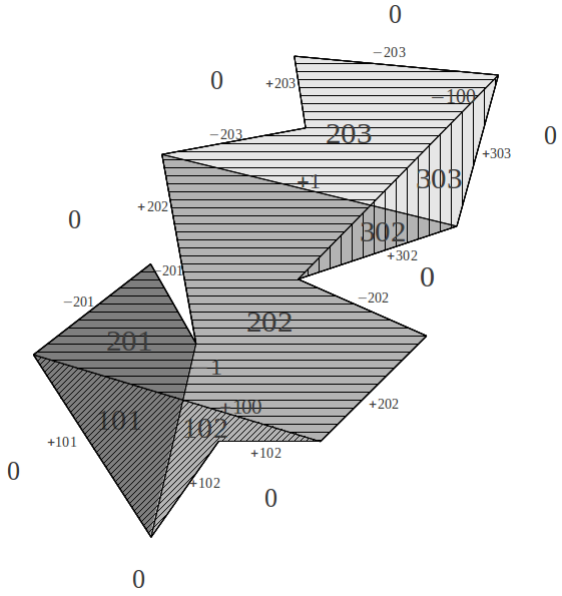
Veja o resultado na Figura 5.14(c). Similarmente, a interseção $(C \cap T)$ entre duas coleções de arestas é a coleção de arestas que representa a função $f \cap (Q_{C+T})$,



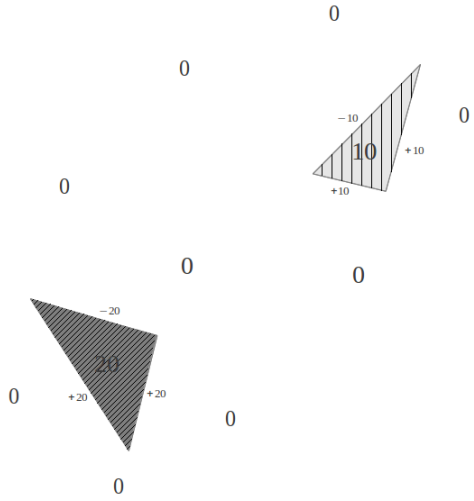
(a) Grau de desmatamento em uma região.



(b) Região por número de habitantes.



(c) Campos escalares obtidos pela soma das coleções de arestas em 5.13(a) e 5.13(b).



(d) Trasformação escalarar aplicada às coleções de arestas 5.13(a) e 5.13(b).

Figura 5.13: Transformação escalar.

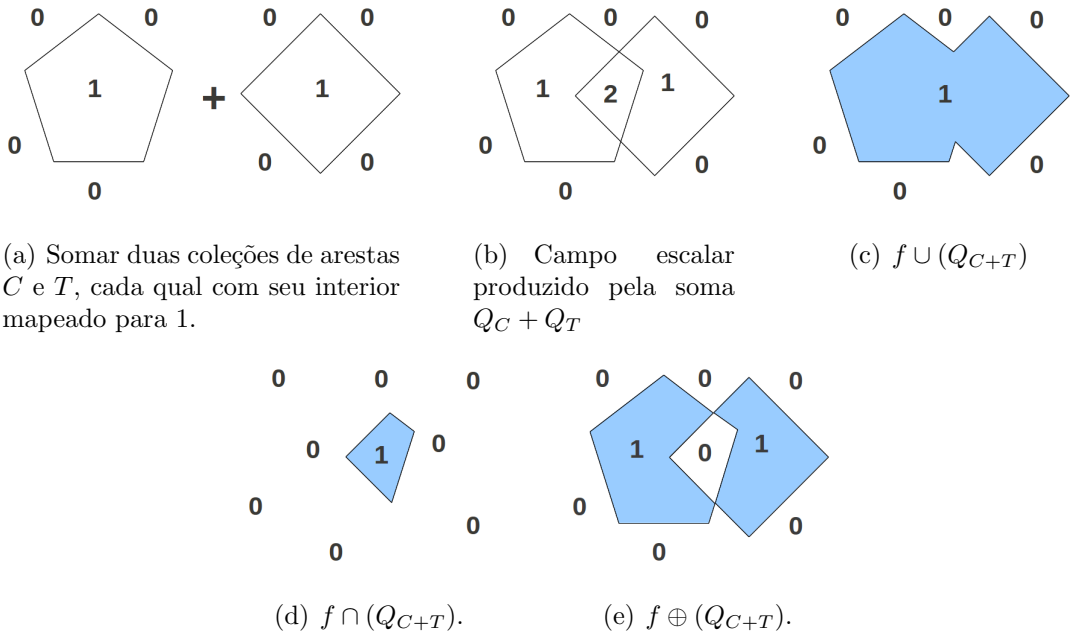
onde:

$$f_{\cap}(x) = \begin{cases} 1 & \text{se } x > 1 \\ 0 & \text{caso contrário} \end{cases}$$

O resultado pode ser visto na Figura 5.14(d). A operação de diferença, também conhecida por *disjunção exclusiva* ou *XOR* $(C \oplus T)$ é a coleção de arestas que representa a função $f \oplus (Q_{C+T})$, onde:

$$f_{\oplus}(x) = \begin{cases} 1 & \text{se } x = 1 \\ 0 & \text{caso contrário} \end{cases}$$

Veja a Figura 5.14(e)



(a) Somar duas coleções de arestas C e T, cada qual com seu interior mapeado para 1.

(b) Campo escalar produzido pela soma $Q_C + Q_T$

(c) $f \cup (Q_{C+T})$

(d) $f \cap (Q_{C+T})$.

(e) $f \oplus (Q_{C+T})$.

Figura 5.14: Operações de união, interseção e diferença utilizando transformação escalar.

5.4.1 Algoritmo de transformação escalar

A execução de uma transformação escalar sobre uma coleção de arestas retorna uma nova coleção de arestas que representa o campo escalar dado por $f(x)$. Esta representação é construída analisando o *status* da linha de varredura (LV) após o processamento de todos os eventos em uma cota y a fim de fazer as modificações necessárias no valor do campo escalar. A cada vez que a linha de varredura retrata uma mudança no campo escalar, ou seja, há uma mudança de *status*, o algoritmo de varredura retorna o controle para o algoritmo que processa a transformação escalar e este, por sua vez, analisa cada elemento na linha de varredura retornada, executa as alterações necessárias para que o valor do campo corresponda à função de transformação escalar recebida como argumento e as registra em uma linha de varredura transformada, aqui referenciada por fLV, que dará origem à coleção de arestas da saída. Suponha uma transformação escalar aplicada a uma coleção de arestas cujos

rótulos são dados pela função:

$$f(x) = \begin{cases} 5 & \text{se } x = 1, \\ 0 & \text{caso contrário.} \end{cases}$$

Vejamos passo a passo o comportamento do algoritmo de transformação escalar se submetermos a ele uma coleção de arestas composta por uma só aresta de peso 1 (veja a Figura 5.15) e passado um argumento contendo a função definida acima.

Passo 1: O algoritmo de varredura retorna o *status* da linha de varredura em y_1. É realizada a análise da linha de varredura original da esquerda para a direita, inferindo o valor do campo para cada aresta encontrada. Na Figura 5.16(a) observa-se a mudança do campo de 0 para 1 ao encontrar uma aresta com inclinação negativa com peso $+1$ e, subsequentemente, uma mudança de 1 para 0 ao encontrar uma aresta vertical com peso -1.

Passo 2: Aplicar $f(x)$ ao campo original (Figura 5.16(b)). Desta forma, observa-se que para efetuar a mudança desejada no campo seriam necessárias uma aresta com inclinação negativa e peso $+5$ e uma aresta vertical com peso -5.

Passo 3: Neste momento a linha de varredura transformada ainda não sofreu alterações e registra o campo escalar igual a 0 (Figura 5.16(c)). Para que esta corresponda à $f(x)$, registra-se na linha de varredura transformada uma aresta cujo peso é a diferença entre o que deve ser registrado (Figura 5.16(b)) e o que já está registrado (Figura 5.16(c)) e sua respectiva aresta vertical. O valor da aresta a ser registrada, apesar de bastante intuitivo neste exemplo, só pode ser calculado realizando esta operação.

É importante ressaltar que há uma distinção entre a linha de varredura transformada, que reflete as alterações no campo escalar segundo $f(x)$ e a coleção de arestas retornada pela função. Apesar de registrada na linha de varredura transformada, as arestas verticais apenas delimitam o valor do campo e não compõem o registro de saída. Neste momento, o algoritmo de transformação escalar apenas registra que uma aresta de peso 5 iniciou-se em um ponto (x, y), onde y corresponde à altura da linha de varredura e x corresponde à interseção da aresta com a linha horizontal determinada pela coordenada y. Esta aresta estende-se ao infinito e aguarda que um evento posterior determine seu encerramento (Figura 5.16(d)).

Passo 4: O algoritmo de varredura alcança o ponto final da aresta e retorna o *status* da linha de varredura em y_2 (Figura 5.17(a)). Observa-se a mudança

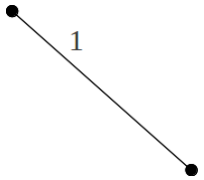
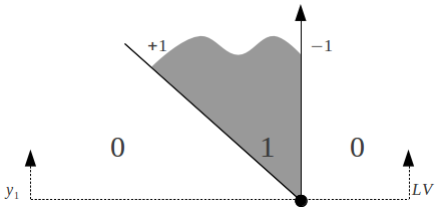
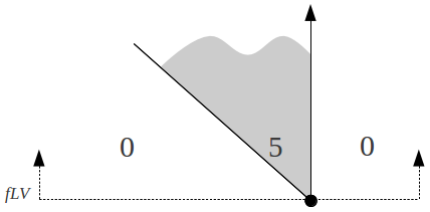


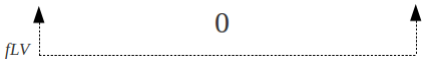
Figura 5.15: Coleção contendo uma aresta de peso 1.



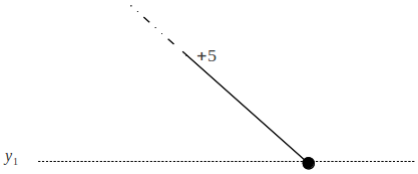
(a) Linha de varredura em y_1 recebida pela função de transformação escalar.



(b) Transformação escalar desejada.



(c) Valor já registrado na linha de varredura transformada.



(d) Aresta emitida pelo algoritmo de transformação escalar.

Figura 5.16: Execução do algoritmo de transformação escalar no ponto inicial de uma aresta.

do campo de 0 para 1 ao encontrar uma aresta vertical com peso +1 e, subsequentemente, uma mudança de 1 para 0 ao encontrar uma outra aresta vertical com peso −1.

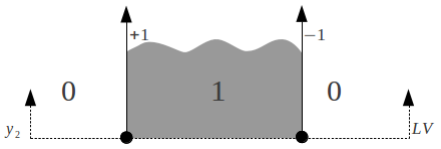
Passo 5: Aplicar $f(x)$ ao campo original, a fim de efetuar a mudança desejada no campo. Observa-se que seriam necessárias uma aresta vertical com peso +5 e uma segunda aresta vertical com peso −5 (Figura 5.17(b)).

Passo 6: A linha de varredura transformada já registra uma variação no campo escalar, (Figura 5.17(c)) que retrata como está o campo um pouco acima do y corrente da fLV. Para que corresponda a $f(x)$, a linha de varredura transformada deve ser igual a diferença entre o se quer registrar (Figura 5.17(b)) e o que já está registrado (Figura 5.17(c)). Observe que para isto, seria necessário inserir uma aresta com inclinação negativa e peso −5 na fLV, e, no mesmo ponto, insere-se uma aresta vertical de peso +5 (Figura 5.17(d))

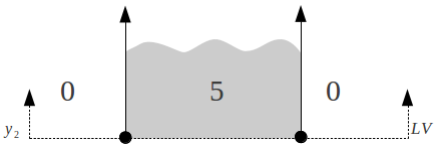
Passo 7: Ao identificar que duas arestas se sobrepõem parcialmente, o algoritmo promove a "quebra" da maior aresta, somando os pesos correspondentes. Efetuando a soma dos pesos das arestas (+5−5), obtém-se zero (0) como resultado (Figura 5.18(a)). O algoritmo de transformação escalar faz o corte da maior

aresta encerrando-a neste ponto e emite para a coleção de saída uma aresta de inclinação negativa e peso +5 (Figura 5.18(b)). Deste modo, a linha de varredura transformada passa a corresponder à figura 5.17(b).

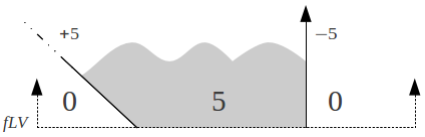
Passo 8: Termina o processamento da linha de varredura e a função de transformação emite a coleção de arestas transformada, que neste exemplo é constituída de apenas uma aresta de peso 5.



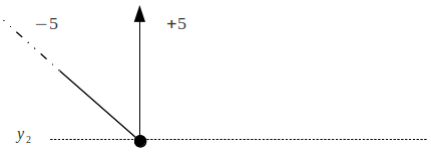
(a) Linha de varredura em y_2 recebida pela função de transformação escalar.



(b) Transformação escalar desejada em y_2 .

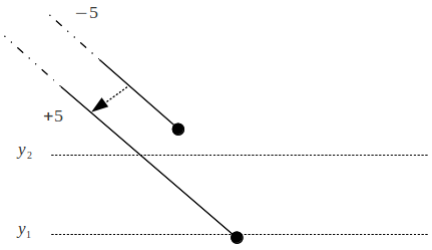


(c) Valor já registrado na linha de varredura transformada.

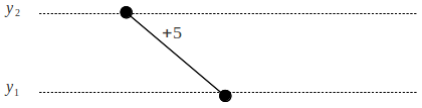


(d) Arestas emitidas pelo algoritmo de transformação escalar.

Figura 5.17: Execução do algoritmo de transformação escalar no ponto final de uma aresta.



(a) Duas arestas, que estendem-se ao infinito, sobrepõem-se parcialmente e a soma de seus campos escalares resulta 0.



(b) Aresta resultante da operação em (a).

Figura 5.18: Resultado da transformação escalar.

Considere agora aplicar uma transformação escalar de interseção à coleção de arestas mostrada na Figura 5.12. A transformação consiste em mapear para 1 as regiões onde o campo é maior que 1 e para 0 as demais regiões. Faremos uma análise do comportamento do algoritmo de transformação escalar em uma determinada cota y para mostrar o modo como são tratados os acontecimentos registrados na linha de varredura.

Passo 1: Nada é registrado na linha de varredura transformada (fLV) antes de a linha de varredura alcançar y_3, pois o campo escalar abaixo de y_3 varia entre 0 ou 1 e foi mapeado para 0 de acordo com a função passada como argumento.

Passo 2: Quando a linha de varredura alcança y_3 (veja Figura 5.19(a)) o controle do processamento retorna para o algoritmo de transformação escalar. Ao iniciar a análise da linha de varredura original, a função de transformação escalar encontra a aresta a' que altera para 1 o campo escalar após ela. Deste modo, nada é registrado em fLV pois $f(1) = 0$ e assim o campo escalar continua valendo 0.

Passo 3: Em seguida, a aresta b, de inclinação negativa e peso $+1$ é encontrada e verifica-se que ela altera o valor do campo escalar para 2. Ao executar $f(2) = 1$, a função de transformação escalar insere uma aresta em fLV com a mesma inclinação de b, porém iniciando no ponto de interseção de b com a coordenada y da linha de varredura e peso igual a $+1$. No mesmo ponto insere-se uma aresta vertical de peso -1 (Veja Figura 5.19(b)).

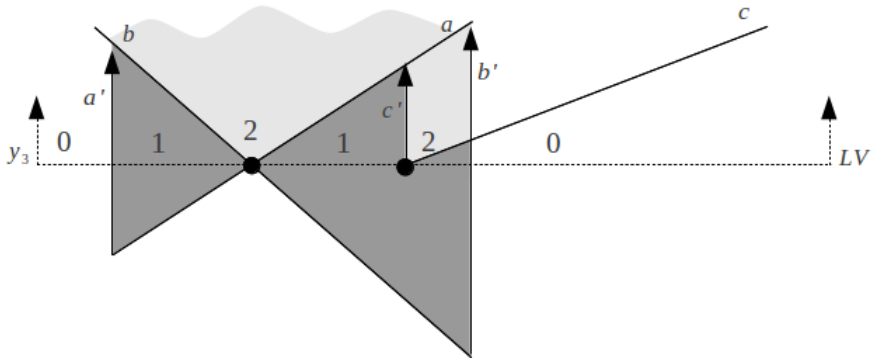
Passo 4: A seguir é encontrada a aresta a, que altera para 1 o valor do campo escalar após ela. Se olharmos a linha de varredura, antes de a o campo vale 2, e, depois de a vale 1. Ao aplicar a transformação escalar $f(2) = 1$ e $f(1) = 0$, a função de transformação escalar insere em fLV uma aresta com a mesma inclinação de a e peso -1, que inicia-se no ponto de interseção de a com a coordenada y da linha de varredura. No mesmo ponto insere-se uma aresta vertical de peso $+1$. Ao tentar inserir a aresta vertical com valor $+1$, o algoritmo verifica que já há uma aresta vertical na mesma posição com valor -1, isto faz com que as arestas se anulem e ambas sejam eliminadas de fLV, permanecendo apenas a aresta com inclinação positiva equivalente a a de peso -1 (Veja Figura 5.19(c)).

Passo 5: As próximas arestas a serem avaliadas na Linha de Varredura são a aresta c' que altera para 2 o campo escalar após ela e a aresta c que faz o campo voltar a valer 1. Aplicando a transformação $f(2) = 1$ e $f(1) = 0$, a função de transformação escalar insere em fLV duas novas arestas: uma aresta vertical iniciando no ponto de interseção de c' com a coordenada y da linha de varredura e peso igual a $+1$ e uma segunda aresta iniciando-se no mesmo ponto, com a mesma inclinação da aresta c e peso -1 fazendo com que o campo volte a valer 0.

Passo 6: Prosseguindo na linha de varredura original, encontramos a aresta b' e observamos que o campo escalar vale 1 antes e 0 após esta aresta. Sendo

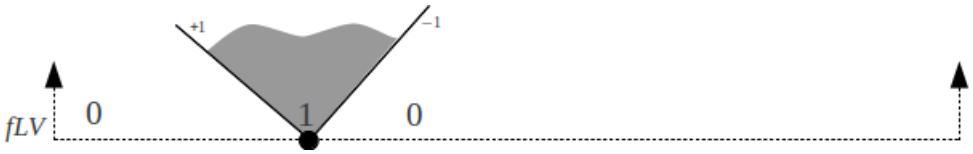
assim, nada é registrado na fLV pois $f(1) = 0$ e $f(0) = 0$ e o campo escalar permanece igual a 0. O algoritmo alcança o fim das alterações na linha de varredura e nada mais é alterado (Veja Figura 5.19(d)).

Note que algumas vezes a aresta original da linha de varredura pode necessitar ser cortada, da maneira como ocorreu neste exemplo, onde a aresta inicia-se em um ponto mais acima que sua correspondente no campo original, ou ainda, quando após uma interseção com outra aresta há uma alteração no campo escalar e este deixa de atender à função expressa por $f(x)$. Somente arestas não verticais compõem a coleção de arestas retornada ao final da função de trasformação. A Figura 5.19(e) mostra o resultado da execução da transformação escalar de interseção sobre a coleção de arestas da Figura 5.12.

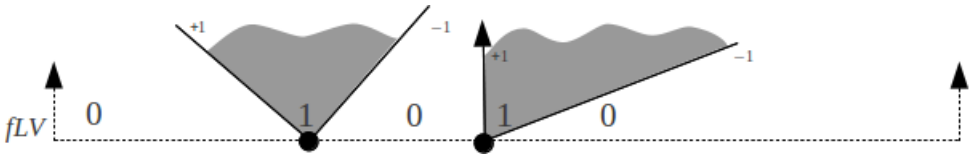


(a) Linha de varredura em y_3.

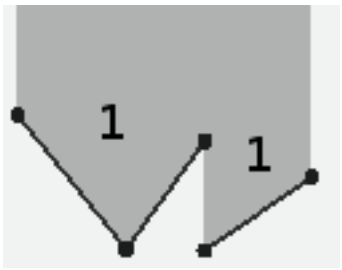
(b) Linha de varredura transformada após processar aresta b.



(c) Linha de varredura transformada após processar aresta a.



(d) Linha de varredura transformada após análise de y_3.



(e) Resultado da transformação escalar de interseção.

Figura 5.19: Execução da transformação escalar de interseção nas arestas da Figura 5.12.

Capítulo 6

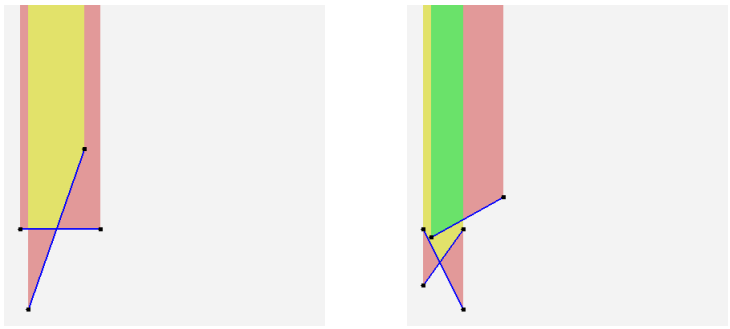
Implementação

Uma implementação-protótipo foi desenvolvida em linguagem Python com o intuito de provar a viabilidade de uso da estrutura, bem como, a correção dos algoritmos descritos neste trabalho. Inicialmente, foi criada uma codificação para verificar a projeção do campo de influência de uma aresta e a correta variação do campo escalar quando estas se sobrepõem. A Figura 6.1 mostra alguns resultados.

A decomposição trapezoidal utilizada na funcionalidade de desenho, pode ser vista na Figura 6.2 onde os trapézios emitidos foram desenhados.

Operações de união, interseção e diferença entre regiões expressas por coleções de arestas, foram realizadas aplicando a operação de transformação escalar utilizando dois polígonos simples, cada qual com seu interior mapeado para 1, conforme descritos no Capítulo 5. Veja os resultados na Figura 6.3

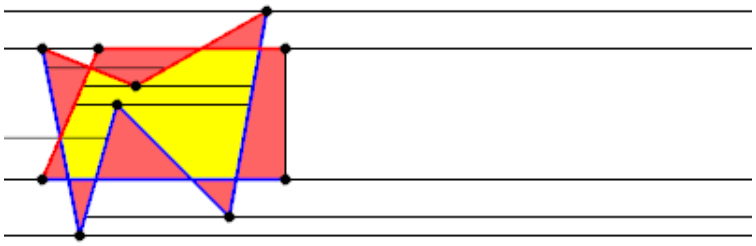
Mapas vetoriais de região no formato *shapefile* [1] foram convertidos para o formato de coleções de arestas ponderadas e submetidos à operações de transformação escalar. As figuras 6.4, 6.5 e 6.6 mostram alguns resultados.



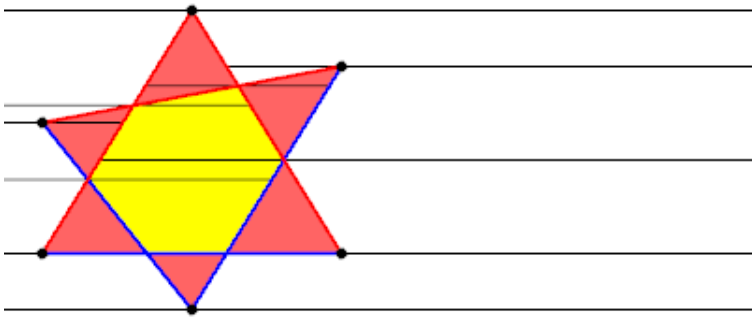
(a) (b)

Figura 6.1: Arestas emitidas pela implementação-protótipo.

[1] Formato de arquivo contendo dados geospaciais desenvolvido como uma especificação aberta para interoperabilidade de dados.

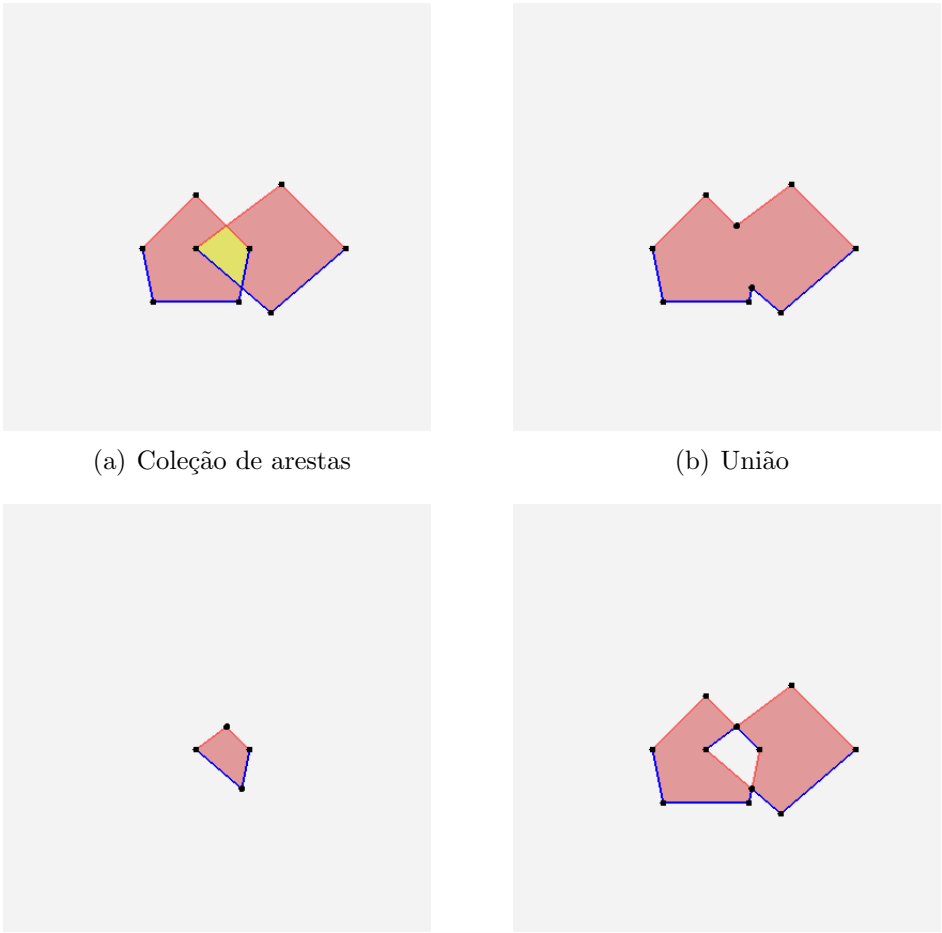


(a)



(b)

Figura 6.2: Trapézios desenhados pela implementação-protótipo.



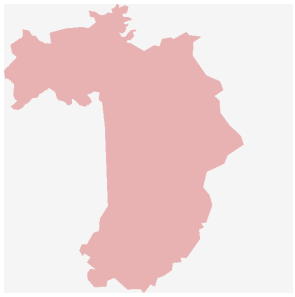
(a) Coleção de arestas

(b) União

(c) Interseção

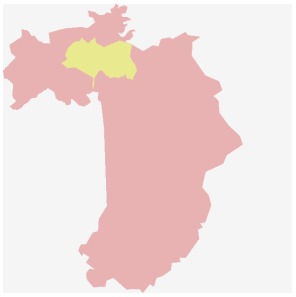
(d) Diferença

Figura 6.3: União, interseção e diferença entre regiões expressas por coleções de arestas.

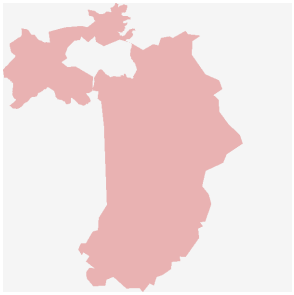


(a) Mapa de região A.

(b) Mapa de região B.

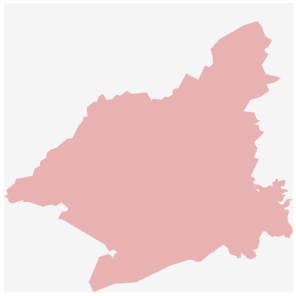


(c) Junção espacial $A \cup B$



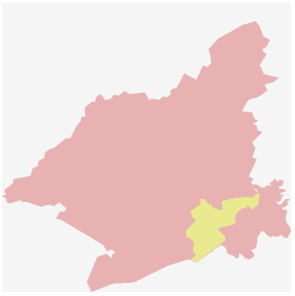
(d) Junção espacial $A \oplus B$

Figura 6.4: Mapas de região A e B e operações de junção espacial emitidos pela implementação-protótipo.

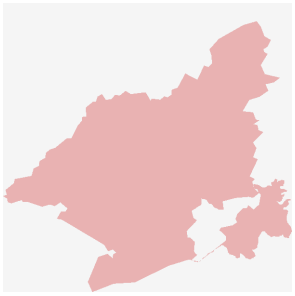


(a) Mapa de região C.

(b) Mapa de região D.

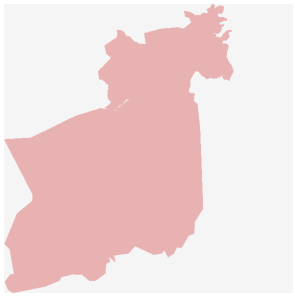


(c) Junção espacial $C \cup D$



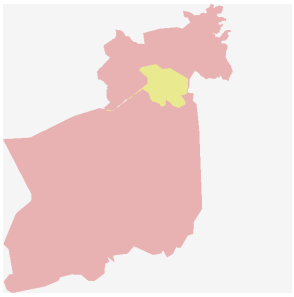
(d) Junção espacial $C \oplus D$

Figura 6.5: Mapas de região C e D e operações de junção espacial emitidos pela implementação-protótipo.

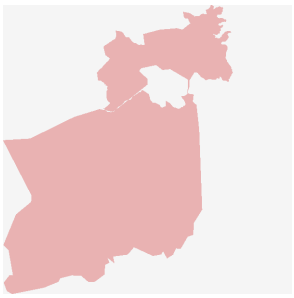


(a) Mapa de região E.

(b) Mapa de região F.



(c) Junção espacial $E \cup F$



(d) Junção espacial $E \oplus F$

Figura 6.6: Mapas de região E e F e operações de junção espacial emitidos pela implementação-protótipo.

Capítulo 7

Conclusões

Este trabalho propõe uma forma alternativa para a representação e o processamento de partições poligonais do plano baseada em campos escalares e algoritmos de varredura. A representação por coleção de arestas ponderadas, descrita em detalhes no Capítulo 4, nos permite representar regiões por meio de campos escalares definidos por suas arestas não verticais. Deste modo, mapas poligonais são armazenados sob a forma de coleções de arestas ponderadas e processados utilizando os algoritmos descritos no capítulo 5, que tem como engrenagem principal um algoritmo de varredura aplicado sobre a estrutura. Dentre os algoritmos desenvolvidos, destaca-se o algoritmo que executa a operção de transformação escalar, que permite a realização de diversos tipos de junção espacial apenas somando duas ou mais coleções de arestas e aplicando sobre o resultado a operação de seleção desejada, que é definida por uma função de $\Re \rightarrow \Re$.

Uma implementação-protótipo foi construída em linguagem Python para verificar a aplicabilidade dos algoritmos, tendo sido utilizada para executar diversos testes.

Não houve neste trabalho um enfoque rigoroso nas cotas de complexidade, quer de espaço, quer de tempo. Apesar disto, nos parece claro que a representação por arestas ponderadas é linear com relação ao número de vértices das regiões poligonais. De forma semelhante a outros algoritmos de varredura de polígonos [7], os algoritmos descritos no Capítulo 5, claramente possuem complexidade de pior caso quadrática. Na realidade, esta complexidade poderia tornar-se cúbica se para os n vértices de uma representação fossem processados $O(n)^2$ eventos, o que só ocorreria se para a cada aresta da coleção existisse um evento de interseção, o que a rigor não ocorre em dados que descrevem mapas de regiões. Entretanto, é possível adaptar os algoritmos para utilizar estruturas de dados que permitam realizar de forma mais eficiente o processamento de operações de busca, inserção e remoção de eventos, tais como árvores balanceadas, o que deve levar a uma cota de complexidade sub-quadrática.

7.1 Trabalhos futuros

Durante este trabalho houve a preocupação em demonstrar que através da abstração de campos escalares é possível realizar consultas espaciais sobre regiões de forma conceitualmente mais simples que os métodos atualmente empregados. Após a definição dos conceitos, desenvolvimento dos algoritmos e construção do protótipo, foram realizados testes exaustivos a fim de retirar os erros concceituais e realizar todas as consultas previstas utilizando o aplicativo.

Futuramente, seria interessante aprimorar o protótipo reimplementando o código base em uma linguagem que proporcione um maior desempenho na realização das consultas, desenvolver algoritmos e estruturas de dados para indexação da estrutura de campos escalares representados por arestas ponderadas, implementar estruturas mais eficientes para o processamento de linha de varredura e realizar testes com massas de dados mais representativas de bancos de dados georeferenciados.

Referências Bibliográficas

[1] ANSELMO LÁZARO BRANCO, CLAUDIO MENDONÇA, E. A. L. "Geografia para todos". "`http://www.geografiaparatodos.com.br/index.php?pag=mapastematicos`".

[2] SAMET, H. *Foundations of Multidimensional and Metric Data Structures*. San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 2005.

[3] CASANOVA, M., CÂMARA, G., DAVIS, C., et al. *Bancos de Dados Geográficos*. São José dos Campos, MundoGEO, 2005.

[4] AZEVEDO, L. G., GÜTING, R. H., RODRIGUES, R. B., et al. "Filtering with raster signatures". In: *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, GIS '06, New York, NY, USA, 2006. ACM.

[5] KRIEGEL, H.-P., BRINKHOFF, T., SCHNEIDER, R. "An Efficient Map Overlay Algorithm Based on Spatial Access Methods and Computational Geometry". In: *Proceedings of the International Workshop on DBMS's for Geographic Applications*. Springer Verlag, 1991.

[6] MARGALIT, A., KNOTT, G. D. "An algorithm for computing the union, intersection or difference of two polygons." *Computers & Graphics*, 1989.

[7] NIEVERGELT, J., PREPARATA, F. P. "Plane-sweep algorithms for intersecting geometric figures", *Commun. ACM*, 1982.

[8] ESPERANÇA, C., SAMET, H. "Vertex representations and their applications in computer graphics." *The Visual Computer*, 1998.

[9] GÜTING, R. H. "An Introduction to Spatial Database Systems", *VLDB J.*, v. 3, n. 4, pp. 357–399, 1994.

[10] BRINKHOFF, T., KRIEGEL, H.-P., SCHNEIDER, R., et al. "Multi-Step Processing of Spatial Joins". In: *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1994.

[11] LAURINI, R., THOMPSON, D. *Fundamentals of spatial information systems*. A.P.I.C. studies in data processing. Academic Press, 1992.

[12] DE BERG, M., VAN KREVELD, M., OVERMARS, M., et al. *Computational geometry: algorithms and applications*. Springer, 2008.

[13] OGC. "Open Geospatial Consortium (OGC)". "`http://www.opengeospatial.org/`".

[14] SAMET, H., WEBBER, R. E. "Storing a collection of polygons using quadtrees", *ACM Trans. Graph.*, 1985.

[15] NELSON, R. C., SAMET, H. "A consistent hierarchical representation for vector data". In: *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pp. 197–206, New York, NY, USA, 1986. ACM.

[16] JACOX, E. H., SAMET, H. "Spatial join techniques", *ACM Trans. Database Syst.*, 2007.

[17] ZHU, H., SU, J., IBARRA, O. H. "Toward Spatial Joins for Polygons". In: *Proceedings of the 12th International Conference on Scientific and Statistical Database Management*, SSDBM '00, Washington, DC, USA, 2000. IEEE Computer Society.

[18] ORENSTEIN, J. A. "Spatial query processing in an object-oriented database system". In: *Proceedings of the 1986 ACM SIGMOD international conference on Management of data*, SIGMOD '86, New York, NY, USA, 1986. ACM.

[19] BECKER, L., GIESEN, A., HINRICHS, K. H., et al. "Algorithms for Performing Polygonal Map Overlay and Spatial Join on Massive Data Sets". In: *Advances in Spatial Databases-6th International Symposium, SSD '99, Hong Kong*. Springer-Verlag, 1999.

[20] JACOX, E. H., SAMET, H. "Iterative spatial join", *ACM Trans. Database Syst.*, 2003.