Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Engenharia de Computação e Automação Industrial

Adler Cardoso Gomes da Silva

Simplificação Consistente de Linhas em Mapas Cartográficos

Orientadora: Profa. Dr.-Ing. Wu, Shin-Ting

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: **Engenharia de Computação**.

Campinas, SP
Junho de 2007

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

Adler Cardoso Gomes da Silva

Simplificação Consistente de Linhas em Mapas Cartográficos

Orientadora: Profa. Dr.-Ing. Wu, Shin-Ting

Banca Examinadora

Prof. Dr. Clodoveu Augusto Davis Jr. .... INF/PUC-MG
Prof. Dr. Clésio Luis Tozzi .... DCA/FEEC/UNICAMP
Prof. Dr. José Mario De Martino .... DCA/FEEC/UNICAMP

Campinas, SP

Junho de 2007

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA
DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

da Silva, Adler Cardoso Gomes

Si38s Simplificação Consistente de Linhas em Mapas Cartográficos / Adler Cardoso Gomes da Silva. – Campinas, SP: [s.n.], 2007.

Orientadora: Wu, Shin-Ting.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Sistema de informação geográfica. 2. Generalização. 3. Geometria computacional. 4. Computação gráfica. 5. Topologia. I. Wu, Shin-Ting. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês:	Consistent Line Simplification in Cartographic Maps
Palavras-chave em Inglês:	Geographic Information System, Generalization, Computational Geometry, Computer Graphics, Topology
Área de Concentração:	Engenharia de Computação
Titulação:	Mestre em Engenharia Elétrica
Banca examinadora:	Clodoveu Augusto Davis Jr., Clésio Luis Tozzi e José Mario De Martino
Data da defesa:	15/06/2007
Programa de Pós-Graduação:	Engenharia Elétrica

Resumo

A simplificação de linha é a operação da generalização cartográfica que remove os detalhes desnecessários de uma linha, preservando os principais aspectos da sua forma. A simplificação consistente de linha preocupa-se tanto em remover estes detalhes, quanto em gerar um mapa que seja consistente ao original. Um mapa é dito consistente ao original se possuir a mesma topologia que este e conservar espaçamentos entre os seus objetos. Os trabalhos em simplificação consistente presentes na literatura ainda apresentam diversas limitações, das quais se destacam a aplicabilidade restrita a determinados tipos de objetos, a ineficácia na preservação da topologia em certos casos, a ausência da conservação de espaçamentos e o alto custo computacional. Este trabalho propõe uma nova solução para o problema da simplificação consistente, procurando contornar estas limitações. Do ponto de vista teórico, ele determina um conjunto de condições que se aplicam a todos os tipos de objetos e garantem a consistência do mapa resultante. Do ponto de vista prático, ele apresenta um algoritmo que, com base nestas condições, é capaz de simplificar consistente e eficientemente as linhas de um mapa. O algoritmo proposto também apresenta outras características importantes para a simplificação consistente, tais como a capacidade de produzir de mapas independentes de escala e a invariância do mapa resultante em relação à ordem de processamento dos dados de entrada.

Palavras-chave: Cartografia Automatizada, Generalização Cartográfica, Simplificação de Linha, Consistência Topológica, Geometria Computacional, Computação Gráfica.

Abstract

Line simplification is the cartographic generalization operation that reduces the complexity of a line, while preserving its main shape features. Consistent line simplification involves not only the reduction of the line complexity, but also the consistency between the original and the simplified maps. A map is said to be consistent to the original one, if it preserves the topology and keeps the same proximity relation of the objects. Current works in consistent simplification still present drawbacks, such as the applicability to just certain types of objects, the failure while preserving topology in particular cases, the lack of proximity handling and the high computational cost. To overcome these drawbacks, this work proposes a new way for handling the consistent simplification problem. From the theoretical point of view, it presents a set of conditions applicable to all types of objects, which guarantees the consistency between the original and simplified maps. From the practical point of view, it presents an algorithm that, based on these conditions, can consistently and efficiently simplify the lines of the original map. The algorithm also presents other important properties to the consistent simplification, such as the capacity of producing scale-independent maps and the ability of yielding the same simplified map for different data arrangements in the original map.

Keywords: Automated Cartography, Cartographic Generalization, Line Simplification, Topological Consistency, Computational Geometry, Computer Graphics.

Aos meus pais,
Almeri e Edi.

Agradecimentos

Aos meus pais, Almeri e Edi, à minha irmã, Emily, ao meu cunhado, Otto, e aos meus sobrinhos, Anderson e Otávio, pelo amor, carinho e apoio enviados de tão longe.

À minha orientadora, Ting, pelo apoio intelectual na árdua concepção deste trabalho e pela amizade cultivada durante estes anos de pesquisa.

Aos professores Clodoveu Augusto Davis Jr., Clésio Luis Tozzi e José Mario De Martino por participarem da banca examinadora desta dissertação e pelas valiosas sugestões agregadas à sua versão final.

À minha namorada, Luciana, pelo amor, carinho e companheirismo cruciais à cansativa etapa de escrita da dissertação.

Aos meus amigos da Pocilga, do DCA, da FEEC, do Nordestão e das peladas, pela grande amizade, companheirismo e incentivo durante esta jornada.

Às pessoas que de algum modo contribuíram, por existirem.

À CAPES, pelo suporte financeiro.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$\mathfrak{L}$	Conjunto de linhas de um mapa
$\mathfrak{L}'$	Conjunto de linhas de um mapa simplificado
$\mathfrak{L}''$	Modificação no conjunto $\mathfrak{L}'$
$\mathcal{L}$	Linha poligonal
$\mathcal{L}'$	Linha simplificada de $\mathcal{L}$
$\mathcal{L}''$	Modificação na linha $\mathcal{L}'$
v_i, u_i, t_i	i-ésimo vértice de uma linha poligonal
$v_{\alpha(i)}$, $u_{\beta(i)}$	i-ésimo vértice de uma linha simplificada
P	Conjunto de pontos de um mapa
p	Ponto de um mapa
$\mathfrak{U}$	Conjunto de URCs do algoritmo proposto
$\mathcal{L}_{[i,j]}$	Sublinha da linha $\mathcal{L}$ que vai do vértice v_i ao vértice v_j
$\mathcal{P}_{\mathcal{L}_{[i,j]}}$	Polígono associado à sublinha $\mathcal{L}_{[i,j]}$
$\mathcal{U}_{\mathcal{L}_{[i,j]}}$	URC que a monitora consistência entre a sublinha $\mathcal{L}_{[i,j]}$ e o segmento $\overline{v_i v_j}$

Lista de Siglas

ACI	Associação Cartográfica Internacional
ASC	Algoritmo de Simplificação em Contexto
ASI	Algoritmo de Simplificação Isolada
ASG	Algoritmo de Simplificação Global
FC	Fecho Convexo
RDP	Ramer-Douglas-Peucker
REAM	Retângulo Envolvente Alinhado Mínimo
REIM	Retângulo Envolvente Inclinado Mínimo
REOM	Retângulo Envolvente Orientado Mínimo
SIG	Sistema de Informação Geográfica
TD	Triangulação de Delaunay
TDR	Triangulação de Delaunay Restrita
URC	Unidade de Recuperação de Consistência

Trabalhos Publicados pelo Autor

1. A. C. G. da Silva, S.-T. Wu. "A Robust Strategy for Handling Linear Features in Topologically Consistent Polyline Simplification". *Anais do VIII Simpósio Brasileiro em Geoinformática (GEOINFO)*, pp. 19–33, Novembro 2006.

   Este trabalho considera um problema no tratamento da consistência topológica de linhas, quando reduzidas a uma seqüência de pontos, e propõe uma estratégia robusta para preservar a topologia deste tipo de objeto. Ele compõe a base do estudo teórico em simplificação consistente apresentado no Capítulo 3 desta dissertação e um trecho do algoritmo proposto presente no Capítulo 4.

2. A. C. G. da Silva, S.-T. Wu. "Preserving Coincidence and Incidence Topologies in Saalfeld's Polyline Simplification Algorithm". *Anais do VII Simpósio Brasileiro em Geoinformática (GEOINFO)*, pp. 107–121, Novembro 2005.

   Este trabalho propõe duas melhorias ao algoritmo de Saalfeld [37]: a preservação da sobreposição de linhas coincidentes em mapas redundantes e a conservação das relações de proximidade entre os seus objetos. Esta segunda melhoria, em particular, é a base da estratégia de conservação de espaçamentos entre objetos apresentada no Capítulo 4 desta dissertação.

3. S.-T. Wu, A. C. G. da Silva, M. R. G. Márquez. "The Douglas-Peucker Algorithm: Sufficient Conditions for Non-Self-Intersecting". *Journal of the Brazilian Computer Society*, Volume 9, Número 3, pp. 67–79, Setembro 2004.

   Este trabalho adiciona ao tradicional algoritmo Douglas-Peucker [35, 10], referenciado atualmente na literatura como algoritmo Ramer-Douglas-Peucker (ou RDP), a capacidade de gerar linhas simplificadas sem auto-interseções. Este trabalho não está presente diretamente nesta dissertação, mas corresponde ao início do estudo em simplificação consistente aqui apresentado.

Capítulo 1

Introdução

A generalização cartográfica é o processo de diminuição dos detalhes de um mapa com o intuito de torná-lo legível ao ser representado em uma escala reduzida. A generalização é composta por um conjunto de operadores, do qual faz parte a simplificação de linha. A simplificação consiste basicamente na redução do número de pontos de uma linha, preservando-se os principais aspectos da sua forma. Em se reduzindo o número de pontos, espera-se eliminar detalhes desnecessários da linha que, em uma escala menor, apenas prejudicariam a sua legibilidade. Uma questão importante da generalização é a consistência do mapa resultante. Recentemente, alguns trabalhos vêm considerando a questão da consistência dentro do processo simplificador. Estes trabalhos, no entanto, ainda apresentam certas limitações, dentre elas o alto custo computacional e, em alguns casos, a baixa legibilidade dos mapas resultantes. Esta dissertação descreve uma nova proposta de simplificação consistente que busca contornar estas limitações.

Este capítulo está organizado da seguinte maneira. A Seção 1.1 contextualiza o problema da simplificação de linha no domínio da cartografia. Ela serve de embasamento àqueles que não têm grande conhecimento neste domínio. Os leitores que se sentem à vontade sobre o assunto podem passar à seção seguinte. A Seção 1.2 apresenta uma breve introdução à simplificação e expõe as principais questões relacionadas a este problema, com atenção especial à questão da consistência. Mais adiante, a Seção 1.3 discute resumidamente o conteúdo deste trabalho, a sua hipótese e principais contribuições. Por fim, a Seção 1.4 descreve brevemente como está organizado o restante da dissertação.

1.1 Cartografia

Há cerca de oito mil anos, a cartografia passou a integrar a história da humanidade. Começando pelas pinturas nas cavernas, passando pelos mapas antigos da Babilônia e chegando aos mapas digitais dos dias de hoje, o homem vem fabricando e utilizando mapas como ferramentas essenciais para exploração, navegação e conhecimento do universo em que vive e dos que imagina. A fabricação de um mapa é um processo que envolve habilidades avançadas, assim como uma grande capacidade de visualização do universo de maneira abstrata e em uma escala reduzida. De acordo com alguns especialistas, os mapas representam um passo significante do desenvolvimento intelectual do homem e servem como registro do avanço do conhecimento da raça humana.

Bicanic e Solaric [1] afirmam que projetar um mapa é uma tarefa complicada que envolve decisões subjetivas. Segundo eles, o cartógrafo não pode simplesmente apresentar os dados em um pedaço de papel sem se preocupar com a sua legibilidade. É de interesse do leitor do mapa uma apresentação que forneça uma visão clara e consistente dos dados. A Associação Cartográfica Internacional (ACI) estabeleceu, em 1966, um conceito de cartografia bem aceito nos dias atuais: "a cartografia apresenta-se como o conjunto de estudos e operações científicas, técnicas e artísticas que, tendo por base os resultados de observações diretas ou da análise de documentação, se voltam para a elaboração de mapas, cartas e outras formas de expressão ou representação de objetos, elementos, fenômenos e ambientes físicos e socioeconômicos, bem como a sua utilização."

A partir da década de 1960, com o advento e a expansão dos computadores, a cartografia atravessou uma enorme revolução. A maioria dos mapas de boa qualidade, tradicionalmente feitos à mão, passaram a ser confeccionados com o auxílio de programas de computador. Criou-se o campo de pesquisa conhecido hoje como cartografia automatizada, formalmente definida como a ciência do processo de fabricação de um mapa com o suporte de um computador. Esta área consistia, inicialmente, na automatização de algumas tarefas básicas realizadas pelos cartógrafos e evoluiu, posteriormente, ao auxílio do processo de concepção de mapas por meio de Sistemas de Informação Geográfica (SIGs). Atualmente, o grande desafio da cartografia automatizada ainda permanece sendo, entretanto, automatizar completamente o processo de fabricação de um mapa.

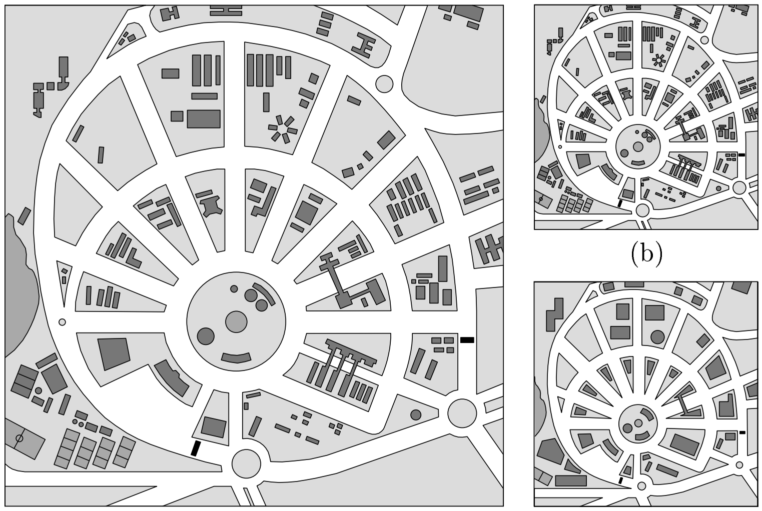
1.1.1 Cartografia Automatizada e SIGs

Desde meados da década de 1990, os SIGs vêm se tornando surpreendentemente populares. Tal popularização deve-se principalmente à recente expansão tecnológica, em particular, dos meios de comunicação móvel e da internet. Por este motivo, a cartografia automatizada preocupa-se não somente com a criação de mapas estáticos, geralmente voltados à impressão, mas também com a concepção de mapas dinâmicos e interativos que possam ser manipulados virtualmente via SIGs. Na realidade, as áreas da cartografia automatizada e dos SIGs sempre foram intimamente relacionadas e evoluíram concomitante e simbioticamente. Por um lado, os SIGs fornecem meios de interação entre o cartógrafo e os algoritmos da cartografia automatizada e, pelo outro, a cartografia automatizada fornece mapas digitais estruturados para a utilização nos SIGs.

O dinamismo e a interatividade dos mapas utilizados nos SIGs requerem que o processo cartográfico seja capaz de elaborar representações confiáveis da realidade em diferentes graus de abstração, permitindo que o usuário navegue entre múltiplas escalas com níveis adequados de detalhamento. Muitas vezes, estas escalas são obtidas a partir de uma única representação do mapa com um nível enorme de detalhamento. Se nenhum processamento fosse realizado, o mapa seria apresentado com um nível exagerado de detalhamento e este excesso de detalhes poderia prejudicar consideravelmente a sua legibilidade. A solução para este problema recai sobre o campo da generalização cartográfica, por meio da qual, de acordo com Ware *et al.* [49], o mapa é adaptado para atender as restrições impostas pela escala à qual se destina e pelo seu propósito.

1.1.2 Generalização Cartográfica

Segundo Spiess [39], a tarefa da generalização cartográfica é prevenir uma representação de dados geográficos de uma total queda no nível de informação, quando o espaço reservado se torna escasso ou a complexidade dos dados causa confusão para os usuários aos quais se destina. Tome como exemplo o mapa do campus da Unicamp ilustrado pela Figura 1.1(a). Se a generalização não fosse aplicada, a redução da sua escala implicaria na diminuição dos seus símbolos e do espaço entre eles, comprometendo a legibilidade da informação nele contida, como ilustra a Figura 1.1(b). A generalização, de acordo com



(a) (b) (c)

Figura 1.1: Mapas da Unicamp: (a) original na escala de 1:10.000, (b) reduzido à escala de 1:25.000 sem generalização e (c) reduzido à escala de 1:25.000 com generalização.

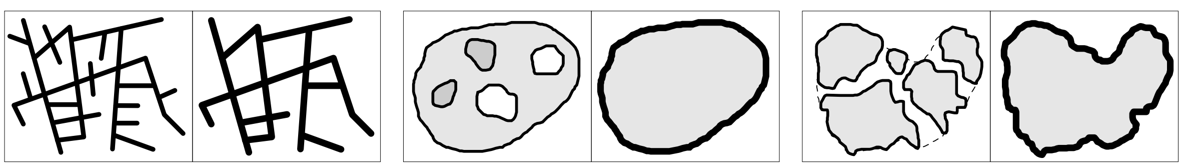
Justo [21], deve remover seletivamente a informação do mapa com o intuito de simplificar padrões sem distorcer o seu conteúdo global, sob a restrição de que as precisões espacial e estética, além da hierarquia lógica da informação, sejam preservadas. A Figura 1.1(c) ilustra o resultado da generalização cartográfica sobre o mapa da Figura 1.1(b).

O cartógrafo precisa, portanto, decidir quais são os aspectos relevantes e quais podem ser omitidos. A causa dos conflitos é a combinação dos efeitos da redução da escala geométrica com o fato de que os objetos são, em geral, bem maiores do que seriam se estivessem na escala correta para aquele nível. É tarefa do cartógrafo perceber qual a realização do mapa que mais se adéqua a critérios de visualização, como a escala e o propósito do mapa. A escala indica o maior nível de detalhes que o cartógrafo pode representar sem comprometer a legibilidade do mapa. O propósito indica que tipo de objeto deve ser realçado. Em mapas rodoviários, por exemplo, as estradas e os objetos relacionados a elas devem ser realçados em detrimento de outros tipos de objetos.

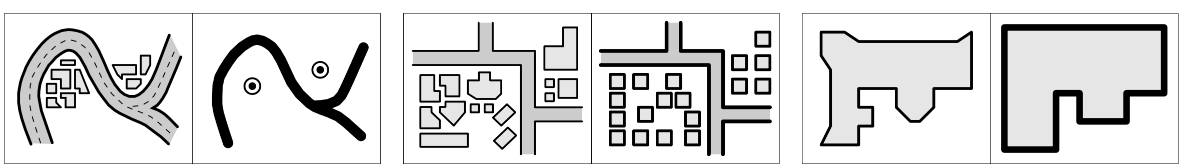
As ferramentas de generalização cartográfica disponíveis atualmente remetem, em geral, às da generalização manual. Os esforços no desenvolvimento de ferramentas automatizadas são voltados às operações manuais, porque, muitas vezes, a qualidade dos mapas generalizados por computador é medida comparando-se estes a mapas equivalentes produzidos manualmente. Embora Müller *et al.* [33], assim como outros autores, tenham questionado a validade da comparação entre resultados automatizados e manuais, ela continua a ser utilizada, na literatura, como um indicador de qualidade da generalização automatizada. As ferramentas típicas de generalização, mais conhecidas como operadores da generalização, são descritas em detalhes na seção seguinte.

1.1.3 Operadores da Generalização

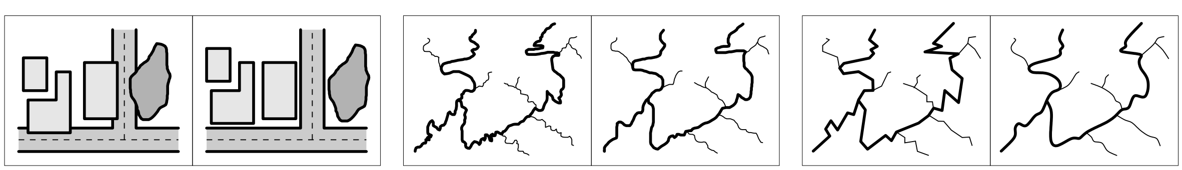
Um operador da generalização consiste em um tipo específico de transformação aplicado sobre objetos do mapa durante o processo de generalização cartográfica. O processo de generalização pode ser dito composto por subprocessos, cada qual envolvendo a aplicação de um determinado operador [27, 50, 51]. Não existe, até hoje, uma nomenclatura comum entre os cartógrafos para designar os operadores. Os mesmos operadores, muitas vezes, são denominados e descritos de maneiras distintas [36]. Os parágrafos seguintes apresentam uma possível classificação dos operadores típicos da generalização cartográfica. A Figura 1.2 ilustra exemplos comuns da aplicação destes operadores.



(a) Seleção (b) Reclassificação (c) Agregação



(d) Simbolização (e) Tipificação (f) Exagero



(g) Deslocamento (h) Simplificação (i) Suavização

Figura 1.2: Operadores típicos da generalização.

Seleção é o processo de remoção de objetos de baixa importância em relação a outros objetos do mapa. O nível de importância é computado a partir dos mais diversos fatores, tais como o propósito do mapa, o tamanho do objeto e a freqüência de ocorrência de um dado tipo de objeto. A Figura 1.2(a) ilustra um exemplo da seleção de ruas em função da intensidade do tráfego.

Reclassificação é o processo de mudança da categoria de objetos. Quando objetos conexos passam à mesma categoria, a reclassificação é acompanhada da aglutinação destes. A Figura 1.2(b) ilustra um exemplo da reclassificação das regiões de uma área florestal com diferentes vegetações, que passaram a uma única categoria.

Agregação é o processo de fusão de dois ou mais objetos próximos e de mesma categoria. O contorno do espaço entre os objetos agregados é geralmente computado por interpolação. A Figura 1.2(c) ilustra um exemplo da agregação de cinco territórios próximos em um único e maior território que engloba os cinco.

Simbolização é o processo de substituição de um ou mais objetos por um símbolo (uma abstração do objeto), que normalmente é associado a uma legenda explicativa. Quando a simbolização diminui a dimensionalidade de um dado objeto, ela é denominada colapso. A Figura 1.2(d) ilustra um exemplo da simbolização de regiões urbanas por círculos e de uma rodovia por uma linha espessa.

Tipificação é o processo de redução da complexidade de representação de um grupo de objetos de mesma categoria e com formas parecidas, através da substituição destes por um conjunto de objetos idênticos e tipificados. O posicionamento relativo dos objetos é geralmente preservado. A Figura 1.2(e) ilustra o exemplo da tipificação de um conjunto de prédios de um mapa urbano.

Exagero é o processo de realce gráfico de objetos de relativa importância. O exagero pode ser local, no qual um detalhe específico da forma do objeto é modificado em uma dada direção, ou global, na qual o objeto é modificado como um todo. O exagero global também é conhecido como dilatação. A Figura 1.2(f) ilustra um exemplo do exagero global ou dilatação de um prédio.

Deslocamento é o processo de reacomodação de objetos muito próximos ou sobrepostos com o intuito de eliminar conflitos gráficos. A forma dos objetos permanece inalterada. A Figura 1.2(g) ilustra um exemplo do deslocamento de prédios e de um lago, que remove a sobreposição entre estes e a rodovia.

Simplificação é o processo de redução da granularidade de linhas e contornos de áreas, eliminando-se os pontos de menor importância, preservando-se, porém, a essência da sua forma. A importância dos pontos é geralmente determinada por um medida geométrica calculada a partir dos pontos que formam a linha. A Figura 1.2(h) ilustra um exemplo da simplificação de uma rede de rios.

Suavização é o processo de atenuação visual de linhas e contornos de áreas. A suavização pode ser tanto realizada por curvas parametrizadas, quanto por processos de extrapolação, nos quais se aumenta a granularidade da linha. A Figura 1.2(i) ilustra um exemplo da suavização de uma rede de rios.

A maioria dos trabalhos em generalização desenvolveu soluções individuais para operadores específicos. Estas soluções são, em geral, publicadas sob a forma de algoritmos concebidos para executar uma tarefa particular sobre um determinado tipo de objeto do mapa. Tais soluções são, muitas vezes, agregadas em ferramentas interativas, nas quais o usuário escolhe como e em que momento um determinado operador deve ser utilizado. Devido à grande subjetividade e complexidade do fator interativo humano, poucos trabalhos preocuparam-se em substituir a intervenção humana pela autonomia, combinando estas soluções em uma solução única e sofisticada para a generalização do mapa completo. Na prática, a maneira como cada um destes operadores é implementado depende da estrutura de dados na qual o mapa é representado.

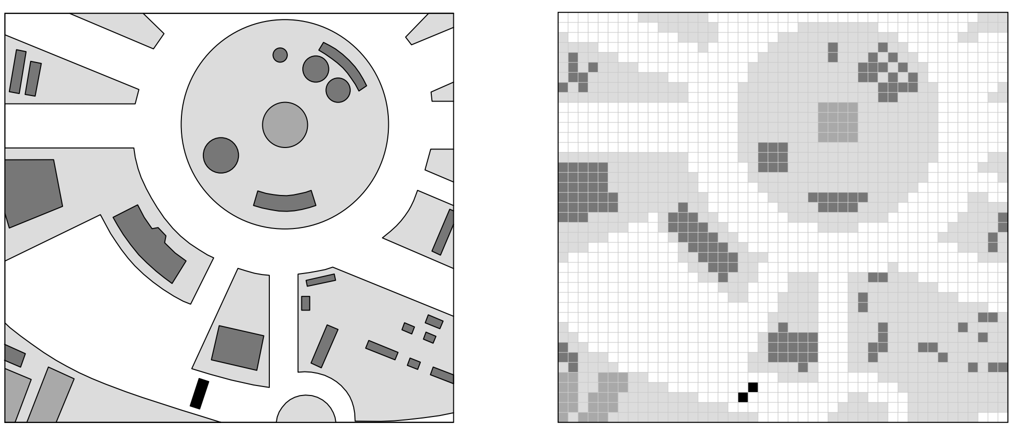
1.1.4 Estruturas de Dados Espaciais

O mapa cartográfico de uma região é uma representação gráfica e georreferenciada do conjunto de fenômenos[1] daquela região. Em se tratando de mapas bidimensionais,

[1]O termo fenômeno será utilizado no restante da dissertação para designar todo e qualquer tipo de dado geográfico real, seja ele um objeto, um elemento, um ambiente ou um fenômeno propriamente dito.

estes fenômenos são representados por três gêneros de objetos: pontos, linhas e áreas. Os pontos[2] representam os fenômenos dos quais se deseja informar apenas a localização, como, por exemplo, cabines telefônicas em mapas urbanos ou cidades em mapas políticos (fenômenos pontuais). As linhas representam fenômenos dos quais se deseja informar a forma e a extensão unidimensionais, como, por exemplo, rios em mapas hidrográficos ou estradas em mapas rodoviários (fenômenos lineares). As áreas representam fenômenos dos quais se deseja informar a forma e a extensão bidimensionais, como, por exemplo, países em mapas políticos ou regiões de mapas temáticos (fenômenos de área).

No contexto de mapas digitais, estes objetos podem ser codificados em duas estruturas de dados espaciais: a vetorial e a matricial. Na estrutura vetorial, os objetos são descritos em um espaço contínuo (Figura 1.3(a)). Os pontos são codificados como pares ordenados de coordenadas, as linhas, como seqüências de pontos interligados por segmentos de reta e as áreas, como encadeamentos cíclicos de linhas. Na estrutura matricial, os objetos são descritos em um espaço discreto (Figura 1.3(b)). O mapa é composto por uma grade regular na qual cada célula armazena um rótulo que identifica o fenômeno que ela representa. Os pontos são codificados como células individuais, as linhas, como células unidas em forma de arestas e as áreas, como grupos contíguos de células. As coordenadas de um objeto podem ser obtidas a partir da localização das suas células.



(a) Estrutura vetorial (b) Estrutura matricial

Figura 1.3: Mapa da Unicamp codificado nas duas estruturas de dados espaciais.

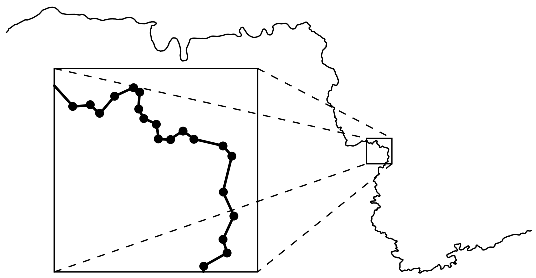
[2]Em muitos mapas, os pontos são substituídos por símbolos que geralmente representam tipos distintos de fenômenos reais e são diferenciados pela sua forma e cor.

As estruturas vetorial e matricial apresentam vantagens e desvantagens uma em relação à outra. Em virtude disto, muitos SIGs adotam codificações mistas para os seus dados, tomando proveito das vantagens de ambas as estruturas. Na realidade, é possível, desde que a resolução dos dados seja adequada, converter da estrutura vetorial para a matricial, e vice-versa, através dos processos de matricialização e vetorização, respectivamente. Assim sendo, os operadores da generalização são comumente concebidos para a estrutura à qual melhor se adéquam, porém podem ser utilizados para a outra, após efetuadas as devidas conversões de dados. Um exemplo desta abordagem é o operador de simplificação, que é geralmente idealizado para lidar com dados vetoriais, pois, nesta estrutura, as linhas são armazenadas explicitamente como seqüências de pontos.

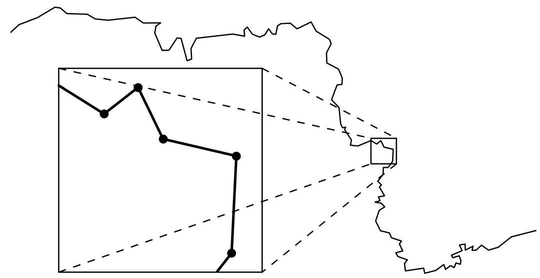
1.2 Simplificação de Linha

A simplificação de linha é provavelmente o tópico mais investigado em pesquisas na área da cartografia automatizada. Este não é um fato inesperado, visto que, quando são contados fenômenos lineares e contornos de fenômenos de área, as linhas compõem cerca de 80% dos objetos cartográficos [33]. Durante a generalização de um mapa, o processo simplificador é, de fato, aplicado sobre todas as suas linhas, sejam elas contornos de áreas ou linhas propriamente ditas. O resultado deste processo é um conjunto de linhas simplificadas que representam os mesmos fenômenos em um mapa apropriado para uma escala menor. Naturalmente, outros operadores também podem remover, modificar ou deslocar as linhas do mapa ao longo do processo de generalização.

A aquisição de uma linha é feita pela amostragem de uma seqüência de pontos do fenômeno a ser representado. Quanto maior a taxa de amostragem, maior o nível de detalhamento da representação. Ao visualizar a linha, os pontos adquiridos (daqui em diante denominados vértices) são ligados por segmentos de reta, constituindo uma representação denominada linha poligonal (Figura 1.4(a)). Em uma escala reduzida, alguns detalhes da linha não são percebidos pela visão humana. Quando aglomerados, estes detalhes podem prejudicar a legibilidade da linha e são, por este motivo, considerados nocivos. O objetivo da simplificação é eliminar tais detalhes, reduzindo o número de vértices da linha e, ao mesmo tempo, preservando a sua forma (Figura 1.4(b)).



(a) Linha original (1.270 vértices)



(b) Linha simplificada (110 vértices)

Figura 1.4: Remoção dos detalhes nocivos da linha através da simplificação.

No que diz respeito a mapas digitais para uso em SIGs, a simplificação traz outros benefícios além da eliminação dos detalhes nocivos da linha. A redução de vértices, cerca de 91% para o exemplo da Figura 1.4, além de tornar bem menor o espaço requerido para o armazenamento dos dados, diminui drasticamente o seu tempo de processamento. Operações sobre os dados como a visualização, a edição, a transmissão via internet, a matricialização, transformações espaciais ou qualquer outra análise espacial tornam-se bem mais eficientes. Além disso, em SIGs, pequeno espaço de armazenamento e rápido poder de processamento são sinônimos de alta qualidade, o que faz da redução de vértices de uma linha uma questão muito importante à generalização.

1.2.1 Algoritmos de Simplificação

Desde que o problema da simplificação de linha foi introduzido, um enorme número de algoritmos foi desenvolvido. Alguns deles projetam-se na direção heurística e executam em tempo linear ou próximo a isso, outros projetam-se na direção da simplificação ótima sob diferentes critérios e executam em tempo quadrático ou cúbico. A maioria dos algoritmos adota uma medida geométrica para determinar a importância de um vértice ou grupo de vértices à forma da linha. Esta medida pode ser, por exemplo, a distância do vértice à linha simplificada, a área do triângulo formado pelo vértice e pelos seus vizinhos ou a variação angular de uma seqüência de vértices. O critério que determina os vértices da linha simplificada depende geralmente do tipo de fenômeno que ela representa. Rios e curvas de nível, por exemplo, requerem critérios distintos de simplificação.

Quanto à determinação dos vértices da linha simplificada, os algoritmos de simplificação são agrupados em duas grandes classes. Na primeira classe, os algoritmos mantêm, na linha simplificada, apenas vértices da linha original. Em geral, eles calculam a medida geométrica para todos os vértices da linha original e preservam aqueles que têm maior medida, ou seja, os que mais contribuem com a forma da linha. Estes algoritmos são denominados extratores ou de filtragem. Na segunda classe, além de manter vértices da linha original, os algoritmos chegam a deslocá-los ou, até mesmo, a criar novos vértices. Algoritmos desta classe são denominados irrestritos. Uma revisão bibliográfica contendo os principais algoritmos de ambas as classes pode ser encontrada em [34].

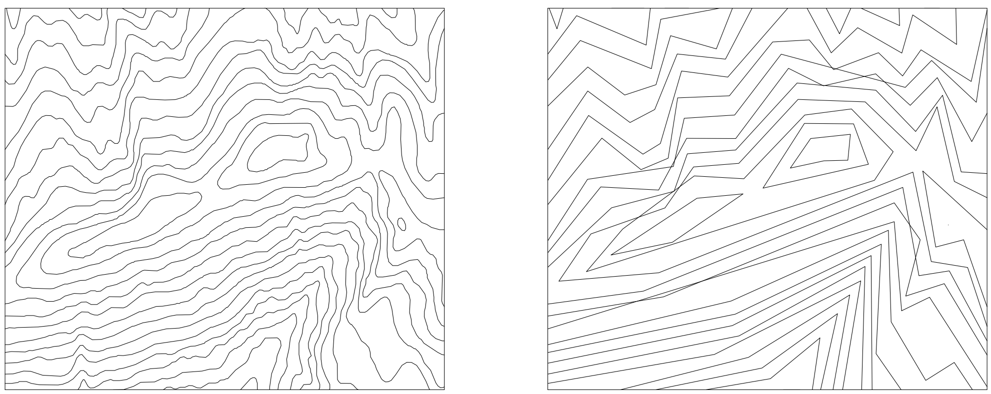
Muitos algoritmos extratores estabelecem uma ordem implícita de remoção dos vértices da linha com base na medida geométrica adotada.[3] A atribuição de uma ordem aos vértices permite que o algoritmo seja aplicado uma única vez sobre a linha em uma etapa de pré-processamento. Nesta etapa, ele associa a cada um dos vértices a sua medida geométrica. Quando uma escala é requerida, é suficiente percorrer os vértices da linha, buscando-se aqueles com medida maior ou igual à medida adequada à dada escala. Mapas com estas características são denominados independentes de escala. Algoritmos extratores que estabelecem uma ordem entre os vértices da linha são, portanto, capazes de produzir mapas independentes de escala. Veja o Apêndice A para mais detalhes.

Outra questão importante é a remoção hierárquica de subfenômenos. Um subfenômeno é uma parte distinguível de um fenômeno, como, por exemplo, uma curva em uma rodovia ou uma península em uma costa marítima. Remover hierarquicamente subfenômenos significa removê-los por ordem de importância. A remoção pode ser tanto explícita, na qual os subfenômenos são identificados antes de serem removidos, quanto implícita, na qual os subfenômenos são removidos indiretamente em função da ordem de remoção dos vértices. A remoção hierárquica de subfenômenos começou a ser discutida no início da década de 1990 com o trabalho de Visvalingam e Whyatt [45] e, a partir de então, passou a ser um requisito importante aos algoritmos de simplificação.

Até meados da década de 1990, os algoritmos de simplificação, com ou sem remoção hierárquica de subfenômenos, se preocupavam apenas com a aparência das linhas. Nesta

[3]É importante salientar que a ordem de remoção dos vértices é estabelecida pelo algoritmo extrator e não está, portanto, relacionada com a ordem em que estes vértices estão dispostos na linha.

época, os melhores algoritmos eram aqueles que reduziam consideravelmente o número de vértices da linha, ao mesmo tempo que a preservavam ao máximo parecida à original. Em algoritmos deste tipo, cada linha do mapa é simplificada isoladamente dos outros objetos e, por este motivo, eles são denominados Algoritmos de Simplificação Isolada (ASIs). Segundo Galanda [16], os ASIs mais utilizados atualmente são o Ramer-Douglas-Peucker (RDP) [35, 10], o de Visvalingam [46] e o de Wang e Müller [47]. A simplificação isolada, no entanto, é incapaz de preservar as relações topológicas dos objetos do mapa e acaba gerando resultados inconsistentes, como o ilustrado pela Figura 1.5.



(a) Mapa original (b) Mapa simplificado

Figura 1.5: Resultado inconsistente de uma simplificação isolada em curvas de nível.

1.2.2 Simplificação Consistente

O exemplo da Figura 1.5 demonstra que, quando as relações topológicas entre os objetos não são preservadas, a legibilidade do mapa é prejudicada. Na simplificação, estas relações dizem respeito ao lado de um objeto em relação a outro, às interseções entre eles e às suas auto-interseções. Algoritmos que preservam estas três relações efetuam uma simplificação de linha dita topologicamente consistente. Alguns algoritmos, no entanto, também conservam espaçamentos entre os objetos, para evitar que estes apareçam muito próximos no mapa simplificado e não consigam ser distinguidos pela visão humana. Como estes algoritmos, além de tratar a questão da topologia, também tratam a questão da proximidade, que não é uma relação topológica, considera-se que eles efetuam, de uma maneira mais abrangente, uma simplificação de linha dita consistente.

Algoritmos de simplificação consistente adicionam mais restrições à remoção dos vértices da linha, com o intuito de alcançar a consistência ao final do processo, como ilustra a Figura 1.6. A simplificação consistente de linha pode ser em contexto ou global. Na simplificação em contexto, o algoritmo processa cada linha do mapa, uma por vez, levando em consideração os objetos presentes na vizinhança da linha. Após simplificar todas as linhas e reuni-las, o mapa resultante como um todo torna-se consistente ao original. Na simplificação global, o algoritmo processa todas as linhas do mapa em conjunto, levando em consideração todos os outros objetos do mapa. Algoritmos desta abordagem geram de maneira direta um mapa simplificado consistente ao original.

Figura 1.6: Resultado de uma simplificação consistente das linhas da Figura 1.5(a).

Na simplificação em contexto, destaca-se o algoritmo de Saalfeld [37]. Ele propõe uma maneira simples de detectar as inconsistências introduzidas pelo RDP e tratá-las localmente. Basicamente, após aplicar o RDP sobre uma linha, ele verifica a consistência entre cada trecho da linha simplificada e o seu correspondente na linha original e preserva, nos trechos inconsistentes, vértices adicionais que recuperam a sua consistência. O algoritmo de Saalfeld, no entanto, ainda gera resultados inconsistentes em certos casos e não realiza nenhum tratamento de proximidade. Além disso, ao corrigir uma linha, ele leva em consideração os vértices das versões originais das outras linhas do mapa, o que diminui o seu desempenho e o leva a preservar mais vértices do que o necessário.

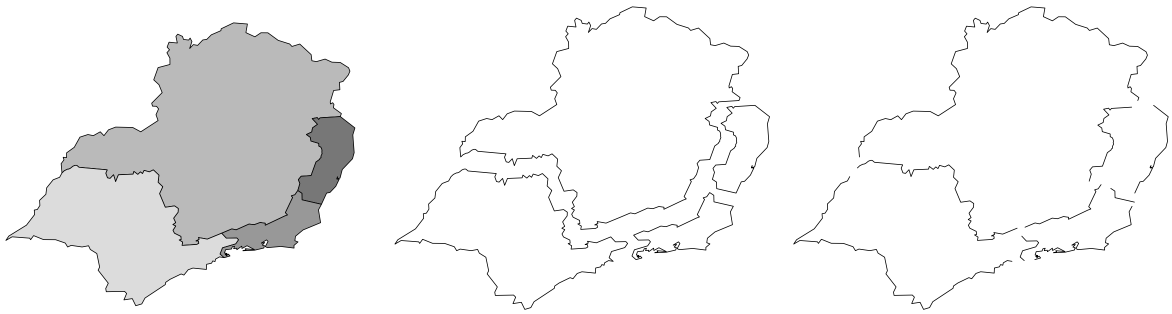
Na simplificação global, destaca-se o algoritmo de van der Poorten e Jones [41, 42]. Ele aplica a Triangulação de Delaunay Restrita (TDR) sobre o mapa original e, com base nela, preserva de maneira implícita a topologia do mapa. Ele remove os vértices das linhas em função de diversas métricas calculadas a partir da geometria dos triângulos resultantes da TDR. A cada remoção, a região do vértice ou vértices excluídos é retriangulada e as suas métricas são recalculadas, levando-se em conta os novos triângulos. O algoritmo de van der Poorten e Jones, no entanto, aplica-se apenas a certas configurações de linhas, não leva pontos em consideração e, tal qual o de Saalfeld, não trata a proximidade. Além disso, em razão da aplicação e reaplicação da TDR, o seu desempenho é muito baixo.

1.3 Conteúdo do Trabalho

Este trabalho propõe uma nova solução para o problema da simplificação consistente de linha, procurando contornar as limitações das soluções anteriores. Do ponto de vista teórico, esta solução consiste em um estudo do problema, que culmina em um conjunto de condições que garantem tanto a preservação da topologia, quanto a conservação de espaçamentos no mapa resultante. Já do ponto de vista prático, ela consiste em um algoritmo que, com base nestas condições de consistência, visa melhorar o aspecto visual do mapa resultante e aumentar o desempenho do processo. Além da questão da consistência, esta solução abrange outras questões importantes para o problema da simplificação, tais como a produção de mapas independentes de escala e a invariância do mapa resultante em relação à ordem de processamento dos dados de entrada.

1.3.1 Escopo

No escopo deste trabalho, assume-se que a entrada do algoritmo proposto é um mapa vetorial sem redundâncias, isto é, um mapa que não contém auto-interseções e cujas únicas interseções são entre extremos de linhas, como ilustra a Figura 1.7. Assim sendo, ele produz como saída um mapa simplificado sem auto-interseções, sem novas interseções e cujos lados dos objetos e interseções entre os extremos das linhas são preservados. É importante salientar que um mapa com redundâncias pode ser facilmente convertido em um equivalente sem redundâncias, e vice-versa. Está fora do escopo deste trabalho a implementação de outros operadores da generalização além da simplificação, muito embora isto seja discutido nesta dissertação e possa ser agregado em trabalhos futuros.

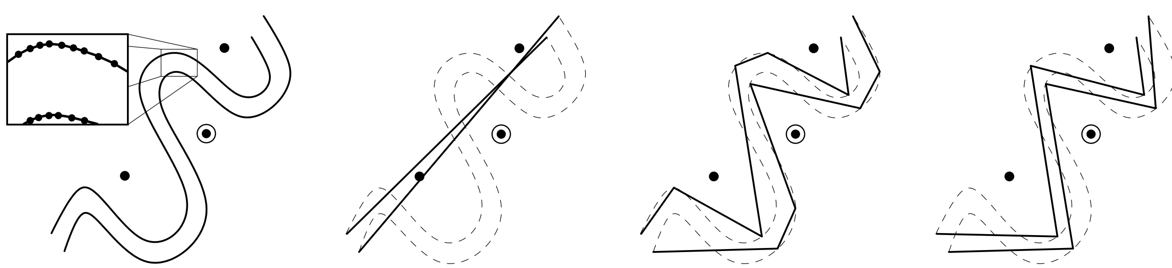


(a) (b) (c)

Figura 1.7: Duas representações possíveis para o (a) mapa da região Sudeste do Brasil: (b) com redundâncias e (c) sem redundâncias.

1.3.2 Hipótese

A Figura 1.8(a) ilustra um mapa composto por três cidades (pontos) e uma estrada (linhas). A Figura 1.8(b) exibe uma simplificação inconsistente, na qual as linhas foram substituídas pelos segmentos que ligam os seus extremos. A Figura 1.8(c), por sua vez, mostra um resultado consistente em que a simplificação de uma linha levou em conta os pontos e os vértices da versão original da outra linha (destacados na Figura 1.8(a)), e vice-versa. A hipótese básica deste trabalho é que, para simplificar consistentemente uma linha, é suficiente considerar os pontos e os vértices das versões simplificadas das outras linhas. Como será demonstrado ao longo desta dissertação, esta estratégia preserva menos vértices nas linhas para corrigir as inconsistências, conforme ilustra a Figura 1.8(d), o que implica em um ganho considerável no desempenho do processo.



(a) (b) (c) (d)

Figura 1.8: (a) O mapa original, (b) a sua simplificação inconsistente e as duas estratégias de correção das inconsistências: (c) levando-se em conta os pontos e os vértices das linhas originais; e (d) levando-se em conta os pontos e os vértices das linhas simplificadas.

1.3.3 Contribuições

A hipótese apresentada é comprovada em um estudo teórico do problema da simplificação consistente de linhas em cartográficos mapas não redundantes. A simplicidade da solução proposta reside principalmente na facilidade de agregar a hipótese à concepção do algoritmo. As outras contribuições deste trabalho residem na aplicabilidade deste algoritmo a múltiplos ASIs, além do RDP, e na sua já mencionada capacidade de produzir mapas independentes de escala, mantendo-o consistente para qualquer nível de detalhamento. Em resumo, este trabalho contribui em vários pontos para a simplificação consistente, que são descritos um a um nos parágrafos seguintes.

- O estudo teórico da consistência no processo simplificador resulta em um conjunto de condições suficientes, que são utilizadas pelo algoritmo proposto para garantir a consistência do mapa resultante. Estas condições abrangem tanto a questão da topologia do mapa, quanto a questão da proximidade entre os seus objetos.

- As linhas do mapa são processadas em conjunto, em uma estratégia de simplificação global. Isto permite que o algoritmo leve em consideração apenas os vértices das linhas simplificadas. Esta abordagem resulta em um ganho em eficiência e na redução do número de vértices adicionais preservados durante a correção.

- O algoritmo proposto pode ser utilizado, ao mesmo tempo, sobre uma diversidade de ASIs, adequados aos mais diferentes tipos de fenômenos. Isto permite especificar que ASI deve ser aplicado sobre que tipo de fenômeno, como também conceber novos ASIs sem se preocupar com a tratamento da consistência.

- O mapa resultante independe da ordem de processamento dos dados de entrada. Desde que os ASIs preencham certos pré-requisitos, o algoritmo proposto sempre gera o mesmo conjunto de linhas simplificadas, não sofrendo influência da ordem de correção das inconsistências, nem da ordem dos dados de entrada.

- O algoritmo proposto pode ser utilizado para produzir mapas independentes de escala. Como para os ASIs, a cada vértice de cada linha do mapa é atribuído um rótulo que indica a partir de que escala ele deve aparecer no mapa resultante. Só que, neste caso, para todas as escalas, o mapa é consistente ao original.

1.4 Organização da Dissertação

O restante da dissertação está organizada em três partes, que estão distribuídas em cinco capítulos. A primeira parte, composta apenas pelo Capítulo 2, descreve o estado da arte da simplificação consistente de linha. A segunda parte, composta pelos Capítulos 3 e 4, introduz o estudo teórico da simplificação consistente de linha e, em seguida, descreve o algoritmo proposto. A terceira e última parte, composta pelos Capítulos 5 e 6, apresenta os resultados do algoritmo proposto, as conclusões e os trabalhos futuros. Os parágrafos seguintes apresentam um breve descrição do conteúdo de cada capítulo.

Capítulo 2 introduz conceitos e notações relacionados à simplificação de linha e, em particular, ao tratamento da consistência durante o processo simplificador. Em seguida, ele detalha as diferentes abordagens de simplificação consistente, apresentando, para cada uma, os principais trabalhos presentes na literatura.

Capítulo 3 valida a hipótese deste trabalho, apresentando um conjunto de condições suficientes para a simplificação consistente de linha, que consideram as questões da preservação da topologia e da conservação de espaçamentos. Como já mencionado, estas condições são utilizadas para garantir a consistência dentro do algoritmo.

Capítulo 4 descreve o algoritmo de simplificação consistente proposto. Cada etapa do algoritmo é explicada em detalhes e ilustrada sob a forma de fluxo de dados. Em seguida, ele demonstra a invariância do mapa resultante em relação à ordem dos dados de entrada e explica como produzir mapas independentes de escala.

Capítulo 5 apresenta os resultados obtidos com o algoritmo proposto e compara-os aos de outros algoritmos de simplificação consistente. Ele destaca a melhoria em diferentes aspectos do mapa resultante e comprova o ganho em desempenho do processo, comparando os tempos de processamento dos algoritmos.

Capítulo 6 resume as principais conclusões deste trabalho, destacando as suas contribuições e limitações. Em seguida, apresenta algumas melhorias a serem agregadas em trabalhos futuros, como a utilização do algoritmo proposto com outros operadores da generalização, como a seleção e o deslocamento.

Capítulo 2

Trabalhos Correlatos

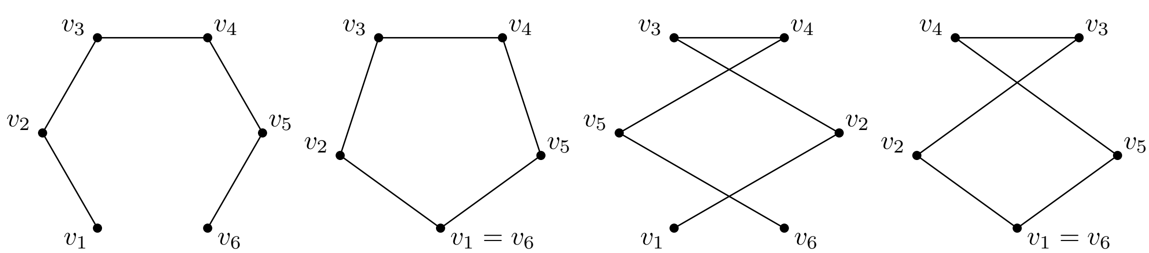
Embora a simplificação de linha seja um dos tópicos mais estudados na área da cartografia automatizada, a questão da consistência foi, até então, muito pouco explorada dentro deste processo. Na realidade, quase todos os trabalhos nesta área atacam o problema sob a abordagem da simplificação isolada. Nestes casos, as inconsistências introduzidas pelos algoritmos só podem ser corrigidas em uma etapa de pós-processamento, geralmente lenta e nem sempre bem sucedida. A simplificação consistente propriamente dita, ao contrário da simplificação isolada, preserva o mapa consistente já dentro do processo simplificador. Algoritmos de simplificação consistente generalizam as linhas de um mapa, ao mesmo tempo que preservam as relações topológicas entre os seus objetos. Como já mencionado, alguns destes algoritmos também são capazes de conservar espaçamentos entre os objetos. Este capítulo descreve os principais trabalhos e questões que representam o estado da arte da simplificação consistente de linha.

Este capítulo está organizado da seguinte maneira. A Seção 2.1 introduz conceitos e notações fundamentais que serão utilizados no restante da dissertação. Em seguida, a Seção 2.2 descreve dois dos algoritmos de simplificação isolada mais utilizados atualmente e os métodos pós-processados de recuperação da consistência. A Seção 2.3, por sua vez, apresenta os dois principais algoritmos de simplificação em contexto. Mais adiante, a Seção 2.4 descreve o principal algoritmo de simplificação global presente na literatura. Por fim, a Seção 2.5 apresenta as considerações finais deste capítulo, discutindo a relação do algoritmo proposto com os algoritmos apresentados.

2.1 Conceitos e Notações

Com o intuito de formalizar posteriores explicações e deduções matemáticas, esta seção introduz alguns conceitos e notações que serão utilizados no restante da dissertação. No contexto da simplificação em mapas vetoriais, o conceito mais fundamental é o de linha poligonal. Assume-se que uma linha poligonal é definida pela seqüência de segmentos de reta formada pela conexão dos pontos amostrados do fenômeno que ela representa. A linha é classificada em aberta ou fechada, em função da coincidência dos seus extremos, e em simples ou complexa[1], em função da presença de auto-interseções[2]. A Definição 1 formaliza este conceito. A Figura 2.1 ilustra exemplos de linha poligonal.

Definição 1. Sejam $v_1, v_2, \ldots, v_n$ pontos no plano, tal que $n \geq 2$. A seqüência de segmentos de reta $\overline{v_1v_2}, \overline{v_2v_3}, \ldots, \overline{v_{n-1}v_n}$ define a **linha poligonal** $\mathcal{L} = v_1v_2 \cdots v_n$, cujos vértices são os pontos $v_1, v_2, \ldots, v_n$ e cujas arestas são os segmentos $\overline{v_1v_2}, \overline{v_2v_3}, \ldots, \overline{v_{n-1}v_n}$. A linha poligonal $\mathcal{L}$ é dita **fechada**, se $v_1 = v_n$, e **aberta**, caso contrário. A linha poligonal $\mathcal{L}$ é dita **simples**, se não contiver auto-interseções, e **complexa**, caso contrário.



(a) Simples/aberta (b) Simples/fechada (c) Complexa/aberta (d) Complexa/fechada

Figura 2.1: Exemplos de linhas poligonais.

Em um mapa vetorial, um fenômeno de área pode ser codificado tanto como uma única linha poligonal fechada, quanto pelo encadeamento cíclico de várias linhas poligonais abertas. De maneira similar, um fenômeno linear pode ser codificado por uma única

[1]Na cartografia automatizada, apenas as linhas poligonais simples (Figuras 2.1(a) e 2.1(b)), são tratadas pela simplificação. As linhas complexas (Figuras 2.1(c) e 2.1(d)) devem ser convertidas em linhas simples antes do processo simplificador.

[2]Considera-se auto-interseção a interseção entre duas arestas de uma mesma linha poligonal que não seja entre os vértices comuns de duas arestas subseqüentes da linha, nem entre os vértices extremos da linha, isto é, entre o vértice v_1 da primeira aresta e o vértice v_n da última.

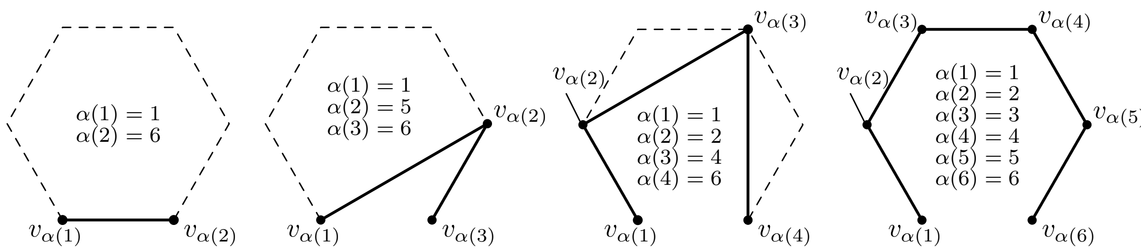
linha poligonal aberta ou pelo encadeamento acíclico de várias delas. A simplificação, no entanto, não precisa distinguir fenômenos lineares de contornos de fenômenos de área. Na realidade, ela é aplicada de maneira transparente em todas as linhas do mapa. Assim sendo, do ponto de vista da simplificação, o mapa é composto por um conjunto de N linhas poligonais abertas e fechadas, denotado por $\mathfrak{L} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_N\}$.

Quando o problema é observado sob o foco da consistência, as relações entre os diferentes objetos – pontos, linhas e áreas – devem ser levadas em consideração. Isto não significa, entretanto, que estes objetos precisam ser tratados explicitamente pelos algoritmos. No caso da área, como será mostrado mais adiante, é suficiente preservar as relações das linhas que a compõem, para tornar a sua simplificação consistente. Assim sendo, o tratamento de linhas e contornos de áreas na simplificação consistente ainda pode ser transparente. Além disso, os pontos poderiam ser vistos como linhas degeneradas com vértices idênticos. No entanto, decidiu-se, neste trabalho, tratá-los de maneira explícita. Portanto, para a simplificação consistente, o mapa é composto pelo conjunto de linhas $\mathfrak{L}$ e por um conjunto de M pontos, denotado por $P = \{p_1, p_2, \ldots, p_M\}$.

A simplificação de uma linha poligonal $\mathcal{L}$ é denotada por $\mathcal{L}'$ e o conjunto de linhas do mapa simplificado é denotado por $\mathfrak{L}' = \{\mathcal{L}'_1, \mathcal{L}'_2, \ldots, \mathcal{L}'_N\}$. As notações $\mathcal{L}''$ e $\mathfrak{L}''$ indicam modificações na simplificação $\mathcal{L}'$ e no conjunto $\mathfrak{L}'$, respectivamente. Já as notações $\mathcal{L}'''$ e $\mathfrak{L}'''$ indicam modificações em $\mathcal{L}''$ e $\mathfrak{L}''$, respectivamente. A simplificação de uma linha $\mathcal{L} = v_1 v_2 \cdots v_n$ sempre deve manter os seus extremos v_1 e v_n, para preservar conexões com outras linhas. No caso dos algoritmos extratores, tratados neste trabalho, além de manter os extremos, a linha simplificada $\mathcal{L}'$ deve conter apenas vértices da linha original $\mathcal{L}$ e preservar a ordem relativa entre eles. A Definição 2 formaliza este conceito. A Figura 2.2 ilustra exemplos de linhas simplificadas da linha da Figura 2.1(a).

Definição 2. Sejam $\mathcal{L} = v_1 v_2 \cdots v_n$ e $\mathcal{L}' = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m)}$ linhas poligonais, onde $n \geq m$. A linha $\mathcal{L}'$ é dita uma **linha simplificada** (ou **simplificação**) de $\mathcal{L}$, se $\alpha(1) = 1$, $\alpha(m) = n$ e $\alpha(i) < \alpha(j)$ para $1 \leq i < j \leq m$. A linha $\mathcal{L}' = v_1 v_n$, que contém apenas os extremos de $\mathcal{L}$, é denominada **linha simplificada** (ou **simplificação**) **trivial** de $\mathcal{L}$.

Outro conceito muito importante é o de sublinha. Uma sublinha de uma dada linha $\mathcal{L} = v_1 v_2 \cdots v_n$ é uma seqüência contígua de arestas de $\mathcal{L}$ ou simplesmente uma parte

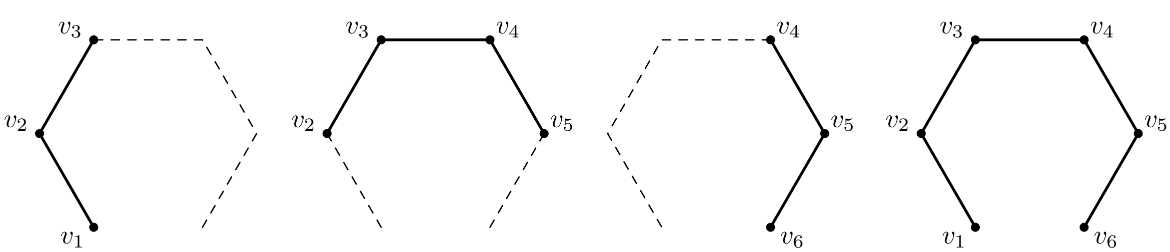


(a) Trivial (b) Três vértices (c) Quatro vértices (d) Todos os vértices

Figura 2.2: Exemplos de linhas simplificadas da linha da Figura 2.1(a).

de $\mathcal{L}$. A notação $\mathcal{L}_{[i,j]}$ designa a sublinha de $\mathcal{L}$ que inclui as arestas entre os vértices v_i e v_j. O texto subscrito $[i, j]$ indica o intervalo fechado de índices dos vértices de $\mathcal{L}$ que fazem parte da sublinha. Este conceito será bastante utilizado para indicar os trechos da linha original que foram substituídos por segmentos na sua simplificação. Em uma linha simplificada $\mathcal{L}' = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m)}$, por exemplo, cada segmento $\overline{v_{\alpha(i)} v_{\alpha(i+1)}}$ $(1 \leq i < m)$ está substituindo a sua sublinha correspondente $\mathcal{L}_{[\alpha(i), \alpha(i+1)]}$. A Definição 3 formaliza este conceito. A Figura 2.2 ilustra exemplos de sublinhas da linha da Figura 2.1(a).

Definição 3. Seja $\mathcal{L} = v_1 v_2 \cdots v_n$ uma linha poligonal. A linha $\mathcal{L}_{[i,j]} = v_i v_{i+i} \cdots v_{j-1} v_j$ define a **sublinha** de $\mathcal{L}$ que vai do vértice v_i ao vértice v_j, isto é, o trecho contíguo de arestas que inclui os vértices cujos índices estão contidos no intervalo $[i, j]$.



(a) $\mathcal{L}_{[1,3]}$ (b) $\mathcal{L}_{[2,5]}$ (c) $\mathcal{L}_{[4,6]}$ (d) $\mathcal{L}_{[1,6]}$

Figura 2.3: Exemplos de sublinhas da linha da Figura 2.1(a).

O último conceito a ser introduzido é o de vizinhança. A vizinhança de uma dada linha $\mathcal{L}$ é a região na qual uma simplificação $\mathcal{L}'$ pode influenciar na consistência do mapa. Vários trabalhos [7, 56, 37] constataram que, quando a linha $\mathcal{L}$ é simplificada por um algoritmo extrator, os únicos objetos que podem causar inconsistências topológicas são aqueles com pelo menos um vértice dentro do fecho convexo de $\mathcal{L}$. Isto porque, nestes casos, a simplificação $\mathcal{L}'$ sempre permanece no fecho convexo de $\mathcal{L}$ e, conseqüentemente, só pode intersectar ou mudar os lados dos objetos dentro do fecho. Portanto, a vizinhança, em algoritmos extratores, resume-se ao fecho convexo da linha, que pode ser usado para reduzir o espaço de busca por inconsistências. A Definição 4 formaliza este conceito. A Figura 2.4 destaca os objetos dentro da vizinhança de uma linha.

Definição 4. Seja $\mathcal{L} = v_1 v_2 \cdots v_n$ uma linha poligonal. O fecho convexo de $\mathcal{L}$, isto é, o menor conjunto convexo contendo os vértices $v_1, v_2, \ldots, v_n$, define a **vizinhança** de $\mathcal{L}$ no contexto da simplificação por algoritmos extratores.

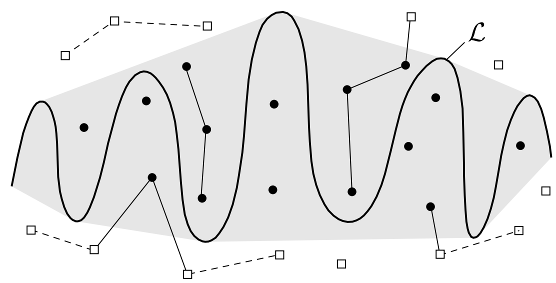


Figura 2.4: A região hachurada representa a vizinhança da linha $\mathcal{L}$ e as linhas cheias, os objetos que estão dentro desta região e que podem causar inconsistências.

2.2 Simplificação Isolada

A simplificação isolada foi a primeira abordagem dada ao problema da simplificação de linha. Por muito tempo, todos os algoritmos foram desenvolvidos levando-se em consideração somente a linha a ser simplificada. Um ASI ou Algoritmo de Simplificação Isolada recebe como entrada uma única linha poligonal e retorna como saída a sua simplificação. Para obter o mapa simplificado completo, o ASI deve ser aplicado em todas as suas linhas, conforme ilustra a Figura 2.5. Ao tratar uma linha isoladamente dos outros objetos do mapa, estes algoritmos têm apenas a possibilidade de evitar auto-interseções. Naturalmente, mesmo aqueles que as evitam, como o algoritmo de Wu, da Silva e Márquez [55, 54], ainda podem inserir outros tipos de inconsistências. As inconsistências introduzidas devem ser removidas em uma etapa de pós-processamento.

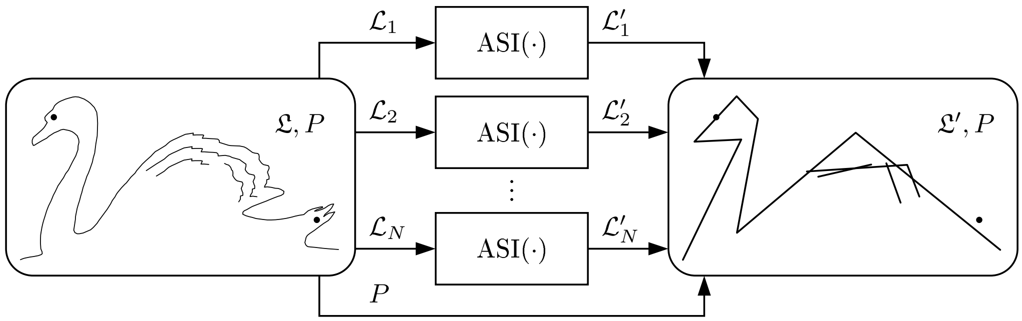


Figura 2.5: Fluxo de dados na aplicação de um ASI sobre um mapa.

Como mencionado na Seção 1.2.1, o algoritmo RDP [35, 10] e o de Visvalingam [46] são dois dos ASIs mais utilizados atualmente. O RDP foi, por muito tempo, apontado como o ASI que gerava os melhores resultados, devido à semelhança da linha que simplificava com a original [23, 52, 53] e com a generalizada manualmente [26, 25, 53]. Alguns trabalhos, no entanto, questionaram as medidas de qualidade utilizadas nestas avaliações [29, 31, 40, 45] e outros expressaram descontentamento quanto às pontas geradas pelo RDP [30, 9, 44, 32]. O algoritmo de Visvalingam, concebido duas décadas após o RDP, consagrou-se por ser capaz de remover hierarquicamente os subfenômenos da linha, uma característica ausente no RDP. Esta seção descreve estes dois algoritmos e apresenta os métodos pós-processados de recuperação da consistência do mapa.

2.2.1 Algoritmo RDP

Em 1973, Douglas e Peucker [10] descreveram, no contexto da cartografia automatizada, um método de simplificação de linha baseado em um critério de tolerância relacionado à distância da linha original à linha simplificada. Este método foi, por muito tempo, referenciado como algoritmo Douglas-Peucker. Em 1972, um algoritmo equivalente havia sido desenvolvido por Ramer [35] no contexto do processamento de imagens, com o intuito de aproximar curvas planas por um pequeno número de pontos presentes na curva. Muitos autores referem-se a estes algoritmos em trabalhos recentes [11, 2, 58, 57] como um só, denominando-o algoritmo Ramer-Douglas-Peucker (RDP).

Considere a linha $\mathcal{L} = v_1 v_2 \cdots v_n$ da Figura 2.6(a). A simplificação do RDP sobre $\mathcal{L}$ a uma tolerância ϵ pode ser resumida da seguinte maneira. Ele inicia o processo com a simplificação trivial, isto é, com o segmento $\overline{v_1v_n}$ (Figura 2.6(b)). Em seguida, encontra o vértice v_i $(1 < i < n)$ mais distante de $\overline{v_1v_n}$. Se v_i estiver a uma distância $d \leq \epsilon$ de $\overline{v_1v_n}$, então a linha resultante será apenas $\mathcal{L}' = v_1v_n$. Por outro lado, se $d > \epsilon$, então $\overline{v_1v_n}$ é substituído pelos segmentos $\overline{v_1v_i}$ e $\overline{v_iv_n}$ (Figura 2.6(c)). De maneira análoga, $\overline{v_1v_i}$ e $\overline{v_iv_n}$ são processados em relação às suas respectivas sublinhas $\mathcal{L}_{[1,i]}$ e $\mathcal{L}_{[i,n]}$ (Figuras 2.6(d)). Este processo continua (Figura 2.6(e)) até que o critério de parada da tolerância ϵ seja alcançado. O vértices inseridos até então determinam a linha $\mathcal{L}'$ (Figura 2.6(f)).

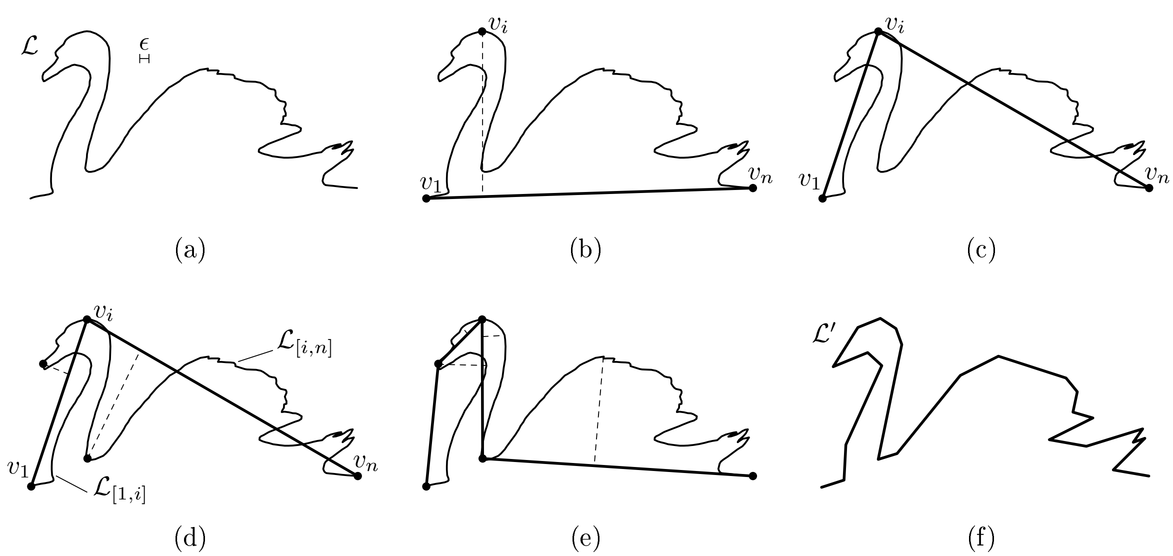


Figura 2.6: Aplicação do algoritmo RDP.

O RDP tem ordem de complexidade no pior caso de $O(n^2)$, em função do número de vértices de entrada n, ou ainda $O(mn)$, em função do número de vértices de entrada n e de saída m. Em 1992, Hershberger e Snoeyink [18] aperfeiçoaram o RDP, que passou a ter complexidade $\Theta(n \log n)$. O RDP é um algoritmo extrator, pois só mantém, na linha simplificada, vértices da original. Além disso, ele dito incremental, porque, a cada passo, insere vértices na simplificação. Por este motivo, ele pode ser facilmente adaptado à produção de mapas independentes de escala (veja Seção A.1). Note que, quando o valor de ϵ é relativamente grande, alguns segmentos da linha simplificada podem se sobrepor, configurando auto-interseções, como ilustra a Figura 2.7.

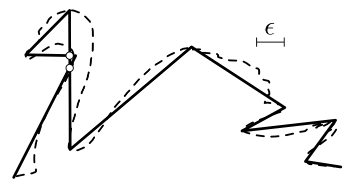


Figura 2.7: O algoritmo RDP gera as auto-interseções indicadas com pontos brancos.

2.2.2 Algoritmo de Visvalingam

Em 1993, Visvalingam e Whyatt [46] introduziram um ASI denominado por eles algoritmo de Visvalingam. A idéia básica do algoritmo é eliminar iterativamente os vértices da linha, em função de uma medida de área calculada para cada vértice. Segundo eles, o seu algoritmo alcança resultados mais satisfatórios que o RDP, quando comparados a generalizações realizadas manualmente por cartógrafos. A razão disto é que o algoritmo apresenta a capacidade implícita de remoção hierárquica dos subfenômenos da linha. Eles mostraram que os algoritmos anteriores, entre eles o RDP, eram incapazes de realizar generalizações cartográficas de boa qualidade, por não se preocuparem com a semântica dos subfenômenos, e eram apenas satisfatórios para simplificações mínimas.

Considere a linha $\mathcal{L} = v_1 v_2 \cdots v_n$ da Figura 2.8(a). A simplificação do algoritmo de Visvalingam sobre $\mathcal{L}$ a uma tolerância ϵ é dividida em duas etapas e pode ser resumida da seguinte maneira. Na primeira etapa, ele determina, para cada vértice v_i $(1 < i < n)$, a área do triângulo $\triangle v_{i-1} v_i v_{i+1}$, denominada área efetiva de v_i (Figura 2.8(b)). Na segunda etapa, o algoritmo elimina iterativamente os vértices, um por vez, até que o critério de tolerância seja satisfeito ou que só restem os extremos v_1 e v_n na simplificação. A cada passo, o algoritmo encontra o vértice v_i de menor área efetiva. Se esta área for menor do que ϵ, v_i é eliminado e as áreas efetivas dos seus vértices vizinhos v_{i-1} e v_{i+1} são recalculadas. Em caso contrário, o algoritmo pára a simplificação com a linha $\mathcal{L}'$, formada pelos vértices não eliminados até então (Figura 2.8(c)).

O algoritmo de Visvalingam tem ordem de complexidade no pior caso de $O(n^2)$, em função apenas da entrada, ou ainda $O(n(n-m))$, em função da entrada e da saída. Tal qual o RDP, ele é um algoritmo extrator, só que decremental, porque, ao contrário daquele, ele elimina um vértice a cada iteração. Além disso, ele já foi concebido com

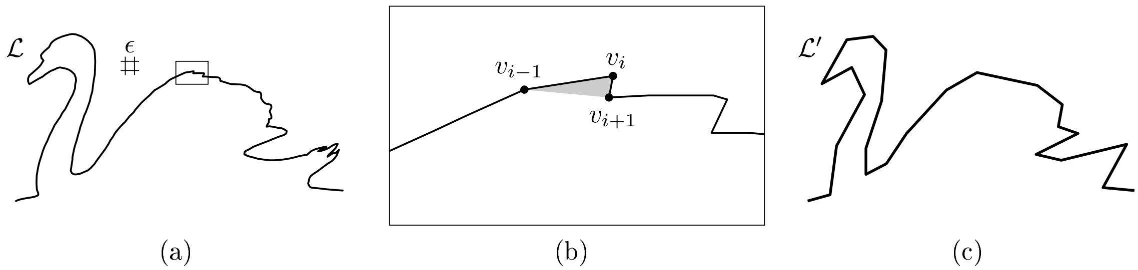


Figura 2.8: Aplicação do algoritmo de Visvalingam.

o intuito de produzir mapas independentes de escala (veja Seção A.2). O algoritmo de Visvalingam é, em geral, mais lento do que o RDP e, tal qual ele, não é capaz de evitar auto-interseções. No entanto, ele ganhou grande notoriedade na literatura, devido à sua capacidade de remover hierarquicamente os subfenômenos da linha. Por esta razão, Visvalingam e Whyatt consideram que, embora não evite qualquer tipo de inconsistência, o seu algoritmo é capaz de realizar a verdadeira generalização cartográfica.

2.2.3 Métodos de Recuperação da Consistência

Após a aplicação de um ASI nas linhas do mapa, as inconsistências introduzidas devem ser corrigidas por um método de recuperação da consistência. Distinguem-se duas situações. Na primeira situação, adquire-se o mapa original cheio de detalhes, que é processado por um ASI, gerando um mapa simplificado com possíveis inconsistências. Neste caso, a remoção das inconsistências é efetuada com base nos dados do mapa original. Na segunda situação, adquire-se diretamente o mapa simplificado, sem qualquer conhecimento dos dados originais. Neste caso, a remoção das inconsistências só pode ser realizada com base nos dados do mapa simplificado. O processo de recuperação, nestas duas situações, difere-se basicamente na utilização ou não dos dados originais.

Com os dados originais, o método mais simples de recuperar a consistência é substituir as linhas ou segmentos inconsistentes pelas suas versões originais. A identificação das linhas inconsistentes é feita, por exemplo, com algoritmos que calculam a interseção de segmentos e a pertinência de pontos em áreas. Embora simples e eficaz na remoção de inconsistências, as desvantagens deste método são inúmeras. Primeiramente, os trechos substituídos retêm todos os vértices e, portanto, não são generalizados. Além disso, a generalização do mapa como um todo torna-se desbalanceada, isto é, enquanto algumas linhas aparentam certa simplicidade, outras apresentam grande complexidade. Por fim, este método é lento, devido à maneira como as inconsistências são detectadas. Exemplos deste método são os algoritmos Edwardes *et al.* [13] e de McKeown *et al.* [24].

Existe outro método de recuperar a consistência da simplificação a partir dos dados originais, embora não se conheça algoritmos na literatura que o utilizem. A idéia deste método é reaplicar o ASI sobre as linhas inconsistentes a uma tolerância menor. Cada linha inconsistente é, então, substituída por uma versão com mais vértices. Se as inconsistências persistirem, estas linhas são novamente submetidas à simplificação a uma tolerância ainda menor, gerando-se linhas com mais vértices do que as anteriores. Este processo continua até a remoção de todas as inconsistências. Embora este método diminua o desbalanceamento do mapa simplificado e aumente o grau de simplificação das linhas substituídas, o seu desempenho é ainda menor do que o do método anterior, visto que cada linha pode ser processada e substituída inúmeras vezes.

Sem os dados originais, a correção torna-se notavelmente mais complicada. Algumas inconsistências, tais como um objeto dentro da área incorreta, não podem ser detectadas de maneira automatizada. Para as outras inconsistências, as soluções resumem-se, em geral, a deslocamentos nas linhas conflitantes pela inserção ou reposicionamento de vértices. O deslocamento de objetos é uma das questões mais difíceis da generalização cartográfica, por envolver o conhecimento subjetivo dos cartógrafos. Além disso, como não se tem conhecimento das linhas originais, os deslocamentos são realizados às cegas e podem, conseqüentemente, tornar as linhas simplificadas relativamente diferentes dos fenômenos reais que elas representam. Exemplos deste método são os algoritmos de Ware e Jones [48], de Lonergan e Jones [22] e de Ware *et al.* [49].

2.3 Simplificação em Contexto

A simplificação em contexto foi a primeira abordagem da simplificação consistente de linha. Um Algoritmo de Simplificação em Contexto (ASC) recebe como entrada a linha a ser simplificada e os outros objetos do mapa que se encontram na sua vizinhança e retorna como saída a linha consistentemente simplificada em relação a si e a tais objetos. Para obter o mapa simplificado consistente completo, o ASC deve ser aplicado sobre todas as suas linhas, conforme ilustra a Figura 2.9. Um caminho tracejado na figura indica que o objeto que o percorre só deve ser considerado como entrada do ASC em questão, se estiver na vizinhança da linha que está sendo simplificada. Perceba que o conjunto $\mathfrak{L}$ também pode conter contornos de áreas, que são tratadas implicitamente.

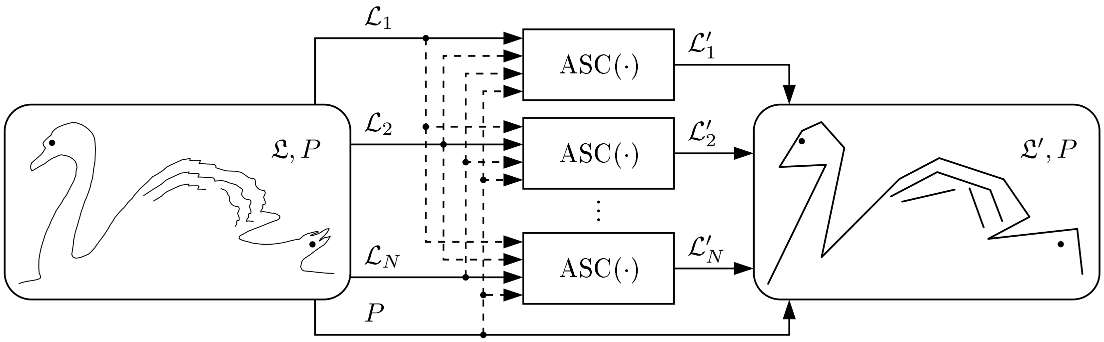


Figura 2.9: Fluxo de dados na aplicação de um ASC sobre um mapa.

Os ASCs mais conhecidos atualmente são o algoritmo de De Berg *et al.* [8] e o algoritmo de Saalfeld [37]. Ambos os algoritmos baseiam-se em ASIs extratores presentes na literatura e apenas lhes adicionam o controle sobre a consistência do mapa. Para limitar a busca por inconsistências, ambos consideram apenas os objetos no fecho convexo da linha, determinado com o algoritmo de Melkman [28]. De Berg *et al.* apresentam um importante passo na teorização do problema da consistência na simplificação. O seu algoritmo, no entanto, é de pouco uso prático devido à má qualidade da generalização. Saalfeld apresenta um solução prática a este problema, ao adaptar o RDP à simplificação consistente. Esta seção descreve em detalhes estes dois algoritmos.

2.3.1 Algoritmo de De Berg *et al.*

Em 1995, De Berg *et al.* [7, 8] apresentaram a primeira proposta de simplificação em contexto com o intuito de simplificar consistentemente as linhas de uma subdivisão planar. Uma subdivisão planar é um mapa no qual todo o espaço é subdividido em áreas, como exemplifica a Figura 2.10. Toda e qualquer linha de uma subdivisão pertence ao contorno de duas áreas, uma de cada lado da linha. O algoritmo de De Berg *et al.*

recebe como entrada uma linha poligonal $\mathcal{L}$ da subdivisão, um conjunto de pontos P na vizinhança de $\mathcal{L}$ e uma tolerância ϵ e retorna como saída uma linha simplificada $\mathcal{L}'$ que atende três metas: (1) $\mathcal{L}'$ está a no máximo uma distância ϵ de $\mathcal{L}$; (2) os pontos de P permanecem do mesmo lado em relação a $\mathcal{L}$ e $\mathcal{L}'$; e (3) $\mathcal{L}'$ não se auto-intersecta.

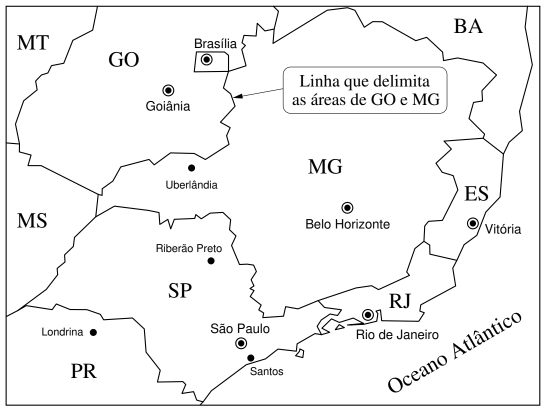


Figura 2.10: Subdivisão planar da região Sudeste do Brasil.

Antes de descrever o algoritmo, é interessante esclarecer dois pontos importantes. Primeiramente, o que De Berg *et al.* pretendem com a meta (2) é, na realidade, preservar os pontos de P nas suas devidas áreas. A consistência das áreas com respeito aos pontos é, portanto, preservada de maneira implícita tratando-se as suas linhas. Para validar esta idéia, De Berg *et al.* formalizaram o conceito de consistência de uma linha, especificado na Definição 5 e ilustrado pela Figura 2.11. Segundo eles, pode-se mostrar que, se duas linhas $\mathcal{L}$ e $\mathcal{L}'$ estão de acordo com a Definição 5, então $\mathcal{L}'$ é consistente a $\mathcal{L}$ com respeito a P independentemente das áreas que elas delimitam. Esta afirmação ratifica a idéia do tratamento implícito das áreas considerando-se apenas as linhas que as compõem.

Definição 5. Sejam $\mathcal{L}$ e $\mathcal{L}'$ linhas poligonais orientadas do vértice v_1 ao vértice v_n e seja P um conjunto de pontos no plano. A linha $\mathcal{L}'$ é dita **consistente** a $\mathcal{L}$ com respeito a P, se existir uma linha poligonal $\mathcal{C}$ orientada de v_n a v_1 que feche $\mathcal{L}$ e $\mathcal{L}'$ em polígonos simples contendo o mesmo subconjunto de pontos de P no seu interior.

O raciocínio desta definição é relativamente simples. Considere que, depois da simplificação da configuração dada na Figura 2.11, o polígono $\mathcal{P}$, formado pelas linhas $\mathcal{L}$ e $\mathcal{C}$ (Figura 2.12(a)), seja substituído pelo polígono $\mathcal{P}'$, formado pelas linhas $\mathcal{L}'$ e $\mathcal{C}$

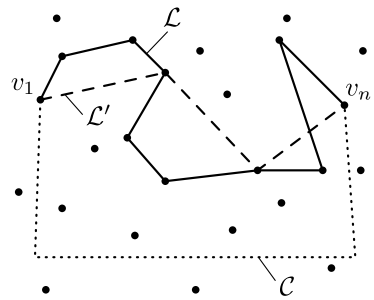
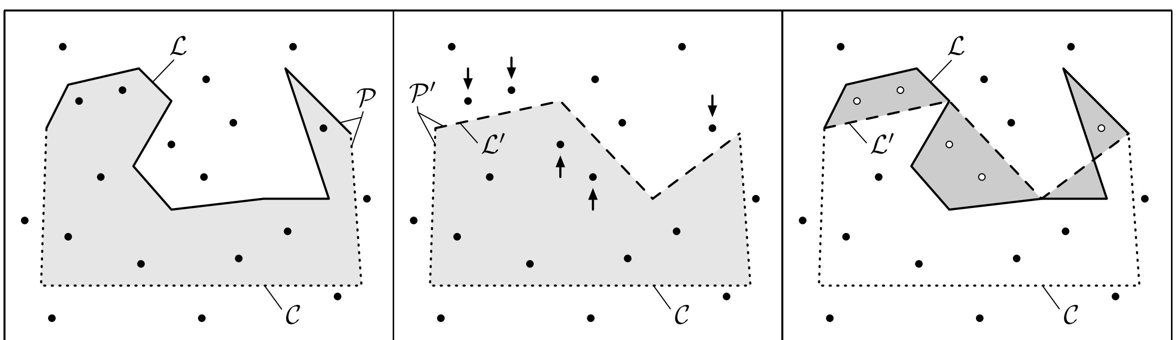


Figura 2.11: Consistência das linhas $\mathcal{L}$ e $\mathcal{L}'$ com respeito a um conjunto de pontos.

(Figura 2.12(b)). Sabe-se que um ponto está consistentemente localizado com respeito a $\mathcal{P}$ e $\mathcal{P}'$, se estiver dentro de ambos os polígonos ou fora de ambos os polígonos. Na Figura 2.12(b), os pontos que estão dentro de $\mathcal{P}$ e fora $\mathcal{P}'$ são indicados com setas para baixo e os que estão fora de $\mathcal{P}$ e dentro $\mathcal{P}'$ são indicados com setas para cima. A partir da Figura 2.12(c), pode-se inferir que os pontos que estão entre as linhas $\mathcal{L}$ e $\mathcal{L}'$ são os únicos que mudaram de lado com respeito aos polígonos $\mathcal{P}$ e $\mathcal{P}'$.



(a) Polígono original $\mathcal{P}$ (b) Polígono simplificado $\mathcal{P}'$ (c) Região de inconsistência

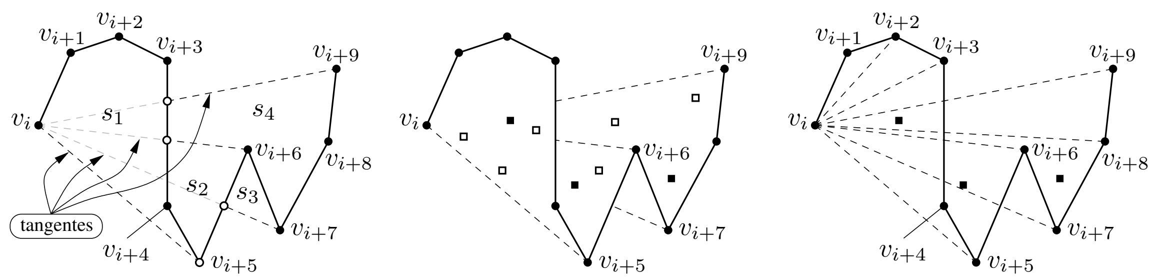
Figura 2.12: Consistência de um conjunto de pontos com respeito aos polígonos $\mathcal{P}$ e $\mathcal{P}'$. Os únicos pontos que causam inconsistências estão entre as linhas $\mathcal{L}$ e $\mathcal{L}'$.

O segundo ponto importante refere-se à meta (3). Segundo De Berg *et al.*, a ausência de auto-interseções na linha simplificada $\mathcal{L}'$ pode ser alcançada apenas preservando-se os lados dos vértices da própria linha $\mathcal{L}$. As metas (2) e (3), então, podem ser atingidas de modo similar, preservando-se os lados dos pontos de P e dos vértices de $\mathcal{L}$, respectivamente. Embora não seja especificada nas metas do algoritmo, esta estratégia também é utilizada para evitar interseções entre diferentes linhas da subdivisão. Isto é feito adicionando-se os vértices das outras linhas do mapa ao conjunto P. Este tratamento uniforme de pontos e vértices, no entanto, não está de acordo com o princípio da Definição 5 e será discutido após a descrição do algoritmo.

O algoritmo de De Berg *et al.* baseia-se no ASI publicado por Imai e Iri [19] em 1988, que atende a meta (1). Dada uma linha poligonal $\mathcal{L} = v_1 v_2 \cdots v_n$ e uma tolerância ϵ, o algoritmo de Imai e Iri gera uma linha simplificada $\mathcal{L}'$ com o menor número de vértices possível e que esteja a no máximo uma distância ϵ de $\mathcal{L}$. Basicamente, ele cria um grafo dirigido $\mathcal{G}$, cujos nós são os vértices de $\mathcal{L}$ e cujas arestas, os segmentos válidos para $\mathcal{L}'$. Um segmento $\overline{v_i v_j}$ é dito válido para $\mathcal{L}'$, se, para todo vértice v_k com $i < k < j$, a distância de v_k a $\overline{v_i v_j}$ for menor ou igual a ϵ. A partir do grafo $\mathcal{G}$, a linha $\mathcal{L}'$ é obtida juntando-se as arestas do menor caminho de v_1 a v_n. O desempenho do algoritmo foi aperfeiçoado em alguns trabalhos [4, 3] e está, hoje, na ordem de complexidade $\Theta(n^2)$.

Para atender as suas três metas, o algoritmo de De Berg *et al.* trabalha não com um, mas com dois grafos dirigidos, denominados $\mathcal{G}_1$ e $\mathcal{G}_2$. O grafo $\mathcal{G}_1$ é idêntico ao grafo $\mathcal{G}$ do algoritmo de Imai e Iri e, portanto, já atende a meta (1). O grafo $\mathcal{G}_2$, por sua vez, atende as duas metas adicionais. Ele é composto pelo mesmo conjunto de nós de $\mathcal{G}_1$, porém por um conjunto diferente de arestas. Um segmento $\overline{v_i v_j}$ compõe o conjunto de arestas de $\mathcal{G}_2$, se preservar os lados dos pontos do conjunto P, conforme a meta (2), e de todo vértice v_k, para $1 \leq k < i$ e $j < k \leq n$, conforme a meta (3). O algoritmo gera, então, o grafo $\mathcal{G} = \mathcal{G}_1 \cap \mathcal{G}_2$ e obtém a linha simplificada $\mathcal{L}'$ juntando-se as arestas do menor caminho de v_1 a v_n em $\mathcal{G}$, de maneira similar ao algoritmo de Imai e Iri.

Na determinação do grafo $\mathcal{G}_2$, De Berg *et al.* propõem, a princípio, um algoritmo para linhas x-monotônicas, mas que pode ser adaptado para linhas arbitrárias. Para cada vértice v_i de $\mathcal{L}$, o algoritmo determina, em três passos, os segmentos válidos $\overline{v_i v_j}$ para $\mathcal{L}'$ com $i + 1 < j \leq n$, como ilustra a Figura 2.13. No primeiro passo, ilustrado pela Figura 2.13(a), ele calcula os segmentos tangentes e as faces indicadas na figura. São tangentes os segmentos que ligam o vértice v_i a vértices v_j, tal que v_{j-1} e v_{j+1} estejam do mesmo lado de $\overline{v_i v_j}$. As faces s_i, então, são delimitadas pela linha $\mathcal{L}$ e por um ou mais segmentos tangentes. Nesta situação, a seqüência de vértices de $\mathcal{L}$ que forma uma face (por exemplo, $v_i, v_{i+1}, \ldots, v_{i+5}$ para s_1) é sempre θ-monotônica com respeito a v_i e qualquer raio que parte de v_i sempre cruza as faces na ordem crescente dos índices.



(a) Cálculo de tangentes e faces (b) Distribuição de pontos (c) Busca por segmentos válidos

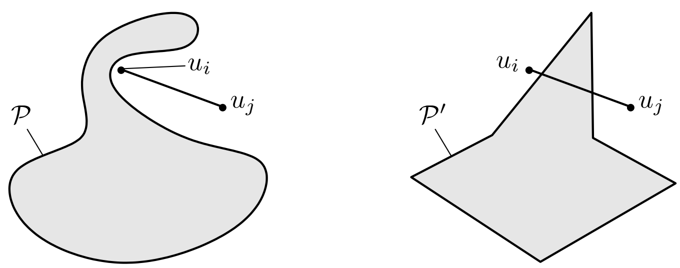
Figura 2.13: Passos do algoritmo de De Berg *et al.*.

No segundo passo, ilustrado pela Figura 2.13(b)), o algoritmo distribui os pontos de P entre as faces com uso de algoritmos de varredura do plano ou de localização de pontos. Para cada face, apenas o ponto com maior distância angular do seu segmento tangente é considerado. Os outros pontos, representados pelos quadrados brancos, são descartados. Com os pontos distribuídos, o algoritmo segue para o terceiro passo, ilustrado pela Figura 2.13(c), no qual busca os segmentos válidos para $\mathcal{L}'$ a partir v_i. Primeiramente, ele ordena os segmentos $\overline{v_i v_{i+1}}, \overline{v_i v_{i+2}}, \ldots, \overline{v_i v_n}$ pela inclinação. Em seguida, para cada face, ele avalia a consistência dos segmentos na ordem calculada em relação ao único ponto daquela face. Os segmentos aceitos formam as arestas de $\mathcal{G}_2$.

De Berg *et al.* sugerem um maneira simples de generalizar este algoritmo para linhas arbitrárias. Segundo eles, para que o algoritmo funcione, não é preciso que a linha seja x-monotônica, mas que apresente as duas propriedades destacadas anteriormente: a θ-monotonicidade em cada uma das faces e a ordem crescente das faces em relação aos raios que partem de v_i. Assim sendo, na busca por segmentos válidos em linhas arbitrárias, é suficiente repartir a linha $\mathcal{L}$ em sublinhas com estas duas propriedades. Com esta estratégia, no entanto, é muito improvável que o algoritmo encontre todos os segmentos válidos para $\mathcal{L}'$. Portanto, mesmo determinando o menor caminho no grafo $\mathcal{G}$, o algoritmo não garante que este caminho corresponda a menor linha $\mathcal{L}'$.

Um vez descrito o algoritmo, pode-se retomar a questão do tratamento uniforme de pontos e vértices, que diz respeito à Definição 5. Esta definição refere-se unicamente à consistência de uma linha e da sua simplificação com respeito a pontos na sua vizinhança. Sem outras restrições, ela não pode ser usada para evitar interseções e auto-interseções.

Para entender isto, observe o exemplo da Figura 2.14. Suponha que, ao substituir o polígono $\mathcal{P}$ pelo polígono $\mathcal{P}'$, o algoritmo preserve os lados dos vértices u_i e u_i de outra linha, como ilustra a figura. Mesmo estando os vértices no exterior de ambos os polígonos, como exige a definição, o segmento $\overline{u_i u_j}$ intersecta $\mathcal{P}'$. Isto acontece porque, embora u_i e u_j estejam do lado correto, os pontos intermediários de $\overline{u_i u_j}$ mudaram de lado.



(a) Polígono original $\mathcal{P}$ (b) Polígono simplificado $\mathcal{P}'$

Figura 2.14: Caso típico de interseção com arestas de outras linhas.

Embora a Definição 5 não possa ser utilizada no tratamento uniforme de pontos e vértices, o algoritmo de De Berg *et al.* consegue eliminar efetivamente interseções e auto-interseções. Isto acontece por acaso, graças ao seu método de escolha de segmentos válidos, que analisa individualmente cada segmento válido $\overline{v_i v_j}$ em relação à sua sublinha correspondente $\mathcal{L}_{[i,j]}$. Esta análise, como será demonstrado no Capítulo 3, é mais restritiva do que a da Definição 5 e consegue evitar interseções e auto-interseções. A importância de avaliar esta definição reside, na realidade, no fato de que algoritmo de Saalfeld a utiliza juntamente com o tratamento uniforme de pontos e vértices e não consegue evitar todas as inconsistências no mapa, como será mostrado na Seção 2.3.2.

O algoritmo de De Berg *et al.* tem, no pior caso, complexidade $O(n(n+p)\log n)$, onde n é o número de vértices da linha $\mathcal{L}$ e p é o número de pontos do conjunto P. O algoritmo, no entanto, é bastante lento, porque trata o problema como uma questão de otimização e leva em consideração todos os vértices das outras linhas originais do mapa, contrariando a hipótese deste trabalho. Dentre outras limitações, ele só pode ser usado em mapas codificados sob a forma de subdivisões planares, o que reduz bastante a sua aplicabilidade. Ele não efetua nenhum tratamento de proximidade e, portanto, os objetos do mapa podem aparecer indevidamente próximos. Além disso, nenhum estudo foi apresentado sobre a sua adaptação à produção de mapas independentes de escala.

Nenhum destas é, no entanto, a sua maior limitação. O que mais prejudica a aceitação deste algoritmo é o fato de que ele não efetua uma generalização propriamente dita sobre a linha. Isto porque ele não utiliza um critério visual apropriado para a escolha dos vértices das linhas, nem remove hierarquicamente os seus subfenômenos. Para amenizar este problema, De Berg *et al.* sugeriram uma implementação consistente do RDP, que daria à sua teoria um uso cartográfico adequado. A mudança no RDP seria apenas no critério de parada da recursão. Além de verificar se a tolerância ϵ foi atingida em cada segmento, a recursão teria que verificar se a consistência foi alcançada. Esta idéia traduziu-se no algoritmo de Saalfeld, descrito na Seção 2.3.2.

2.3.2 Algoritmo de Saalfeld

Em 1999, Saalfeld [37], seguindo a sugestão de De Berg *et al.*, concebeu um algoritmo que adiciona ao RDP a capacidade de simplificar consistentemente uma linha. As suas metas de consistência são as mesmas das do algoritmo de De Berg *et al.*, porém não se restringem a subdivisões. Elas podem ser aplicadas a fenômenos lineares, isto é, a linhas que não fazem parte de contornos de fenômenos de área. O algoritmo é composto por duas etapas, como ilustra a Figura 2.15. Na primeira etapa, ele simplesmente aplica o RDP sobre a linha. Na segunda etapa, ele corrige localmente os erros, inserindo vértices adicionais apenas nos trechos que causam inconsistências. Saalfeld apresentou uma maneira eficiente de identificar os trechos que causam inconsistências e de atualizar o estado da consistência dos trechos após a inserção um vértice adicional.

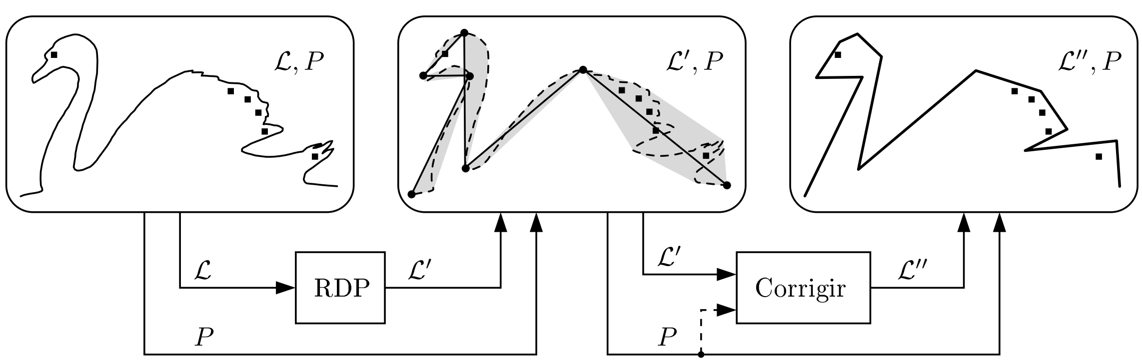


Figura 2.15: Fluxo de dados na aplicação do algoritmo de Saalfeld sobre uma linha.

A etapa de correção é realizada da seguinte maneira. Após aplicar o RDP sobre a linha original $\mathcal{L} = v_1 v_2 \cdots v_n$, o algoritmo obtém uma linha simplificada $\mathcal{L}'$. Para cada segmento $\overline{v_i v_j}$ de $\mathcal{L}'$, o algoritmo primeiramente determina o fecho convexo da sublinha $\mathcal{L}_{[i,j]}$. Em seguida, determina quais pontos de P e quais dentre os vértices das sublinhas $\mathcal{L}_{[1,i)}$ e $\mathcal{L}_{(j,n]}$ estão na vizinhança de $\mathcal{L}_{[i,j]}$. Estes pontos e vértices são armazenados juntamente com $\mathcal{L}_{[i,j]}$ e $\overline{v_i v_j}$ em um objeto, denominado, nesta dissertação, Unidade de Recuperação de Consistência (URC) de $\mathcal{L}_{[i,j]}$, denotada por $\mathcal{U}_{\mathcal{L}_{[i,j]}}$. A determinação e o armazenamento destes pontos e vértices em $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ é denominada coleta dos pontos externos de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$. A URC $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ tem como objetivo, além de armazenar estes dados, monitorar a consistência do segmento $\overline{v_i v_j}$ com respeito à sublinha $\mathcal{L}_{[i,j]}$.

Segundo Saalfeld, os pontos externos que se encontram entre o segmento $\overline{v_i v_j}$ e a sublinha $\mathcal{L}_{[i,j]}$ estão do lado incorreto com respeito a $\mathcal{L}$ e $\mathcal{L}'$ e sempre causam inconsistências. Para determinar se um ponto externo de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ se encontra do lado incorreto, o algoritmo computa o número de cruzamentos de um raio partindo do ponto com $\overline{v_i v_j}$ e $\mathcal{L}_{[i,j]}$. O ponto externo estará do lado incorreto se este número for ímpar, conforme exemplificam os pontos brancos na Figura 2.16. Para que o segmento $\overline{v_i v_j}$ seja considerado inconsistente, é suficiente que um dos pontos externos de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ esteja do lado incorreto. Esta idéia de lado incorreto é uma extensão do conceito de consistência de De Berg *et al.* que abrange também a consistência de fenômenos lineares com respeito a pontos.

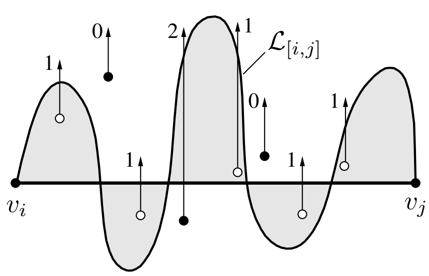


Figura 2.16: Determinação dos pontos externos que causam inconsistências em $\mathcal{U}_{\mathcal{L}_{[i,j]}}$.

Para cada URC $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ com algum ponto externo do lado incorreto, o algoritmo escolhe um vértice adicional v_k e a quebra em duas novas URCs $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ e $\mathcal{U}_{\mathcal{L}_{[k,j]}}$. O vértice v_k é escolhido segundo o critério de seleção do RDP, isto é, v_k é o vértice mais distante do segmento $\overline{v_i v_j}$. Antes de criar as URCs $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ e $\mathcal{U}_{\mathcal{L}_{[k,j]}}$, o algoritmo atualiza a classificação de lado dos pontos externos de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$. Em outras palavras, ele determina quais pontos inverteram de lado, isto é, passaram do lado correto para o incorreto, ou vice-versa. Para fazer isto de maneira eficiente, Saalfeld utiliza uma propriedade por ele descoberta, que indica que apenas os pontos que estão dentro do triângulo $\triangle v_i v_j v_k$ mudam de lado. Assim sendo, um simples teste, denominado teste de inversão do triângulo, é suficiente para atualizar o estado de cada ponto externo de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$, conforme ilustra a Figura 2.17.

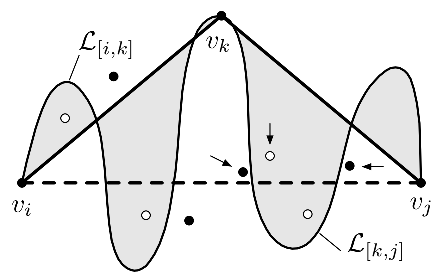
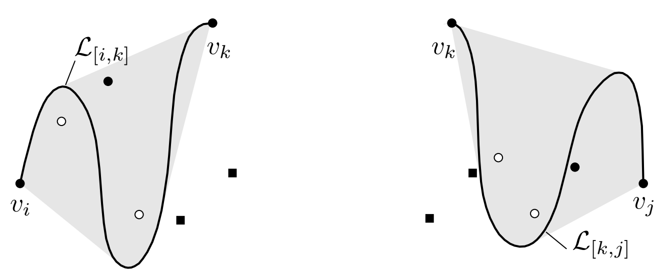


Figura 2.17: Teste de inversão do triângulo.

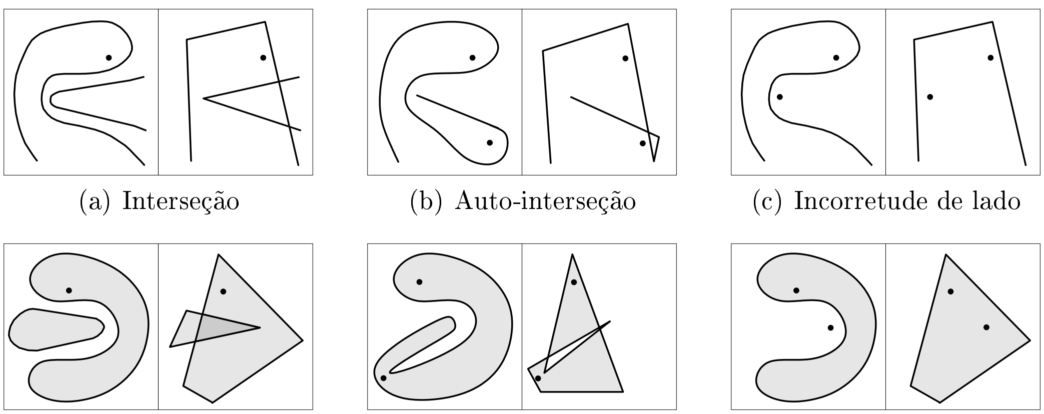
Após a atualização, o algoritmo cria as URCs $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ e $\mathcal{U}_{\mathcal{L}_{[k,j]}}$, determinando os fechos convexos das suas respectivas sublinhas $\mathcal{L}_{[i,k]}$ e $\mathcal{L}_{[k,j]}$, como ilustra a Figura 2.18. Em seguida, o algoritmo coleta os pontos externos de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ nas duas novas URCs. Parte destes pontos poderá estar apenas em $\mathcal{U}_{\mathcal{L}_{[i,k]}}$, outra parte, apenas em $\mathcal{U}_{\mathcal{L}_{[i,k]}}$, uma terceira parte, em ambas as URCs, e uma última parte, em nenhuma delas. Para evitar que $\mathcal{L}_{[i,k]}$ intersecte a simplificação de $\mathcal{L}_{[k,j]}$, ou vice-versa, o algoritmo também tem que coletar os vértices da sublinha de uma URC na outra. Repare que isto só é necessário se os fechos convexos das sublinhas se sobrepuserem, o que não é o caso ilustrado.



(a) URC $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ (b) URC $\mathcal{U}_{\mathcal{L}_{[k,j]}}$

Figura 2.18: Quebra de uma URC e coleta dos seus pontos externos nas URCs resultantes.

Embora a quebra de uma URC possa passar alguns pontos do lado correto para o incorreto, o algoritmo de Saalfeld é sempre convergente, pois, no pior caso[3], ele insere todos os vértices das linhas e recupera a geometria e, conseqüentemente, a topologia do mapa original. Na realidade, a maior limitação do algoritmo de Saalfeld é que, por seguir o princípio da Definição 5, ele não consegue evitar, em certos casos, interseções, auto-interseções e também incorretudes de lados, como ilustra a Figura 2.19. As incorretudes de lado acontecem porque o algoritmo é incoerente. A incoerência está no fato de que o algoritmo calcula o lado de um ponto externo da URC $\mathcal{U}_{\mathcal{L}_{[k,j]}}$ com base no segmento $\overline{v_i v_j}$ e na sublinha $\mathcal{L}_{[k,j]}$ e o atualiza com base no triângulo $\triangle v_i v_j v_k$, considerando o intervalo $[i, j]$ como um todo, em vez de tratar os intervalos $[i, k]$ e $[k, j]$ separadamente.



(a) Interseção (b) Auto-interseção (c) Incorretude de lado

(d) Interseção (e) Auto-interseção (f) Incorretude de lado

Figura 2.19: Casos típicos de inconsistências remanescentes do algoritmo de Saalfeld.

O algoritmo de Saalfeld tem, no pior caso, complexidade $O(n(n + p) \log m)$, onde n é o número de vértices da linha $\mathcal{L}$, m é o número de vértices da linha $\mathcal{L}'$ e p é o número de pontos do conjunto P. O seu desempenho é superior ao do algoritmo de De Berg *et al.*, em virtude da sua menor complexidade e da sua simplicidade, mas ainda é baixo, porque o algoritmo considera todos os vértices das linhas originais (o que se opõe a hipótese deste trabalho). Por ter sido concebido sobre o RDP, o algoritmo de Saalfeld tem maior uso cartográfico do que o de De Berg *et al.*, mesmo não sendo capaz de remover hierarquicamente subfenômenos. Por outro lado, assim como o algoritmo de De Berg *et al.*, ele não é adaptado à produção de mapas independentes de escala.

[3] É praticamente impossível acontecer o pior caso com dados reais [37].

2.4 Simplificação Global

A simplificação global é a abordagem da simplificação consistente que processa todas as linhas do mapa em conjunto. A visão global permite que se tenha conhecimento de todas as linhas simplificadas durante o processamento. Em razão disto, é possível simplificar uma determinada linha levando-se em consideração apenas os vértices incluídos até então nas outras linhas. Um Algoritmo de Simplificação Global (ASG) recebe como entrada o mapa completo, ou seja, o seu conjunto de linhas $\mathfrak{L}$ e de pontos P, e retorna como saída um mapa consistentemente simplificado, contendo um novo conjunto de linhas simplificadas $\mathfrak{L}'$, conforme ilustra a Figura 2.20. Esta seção descreve o algoritmo de van der Poorten e Jones [41, 42], que, pelo breve conhecimento do autor desta dissertação, é o único algoritmo desta abordagem que se destaca na literatura.

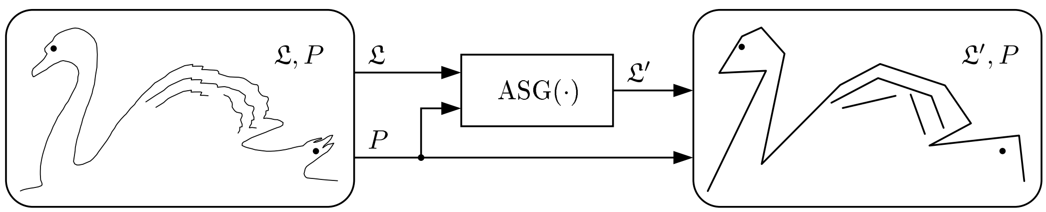
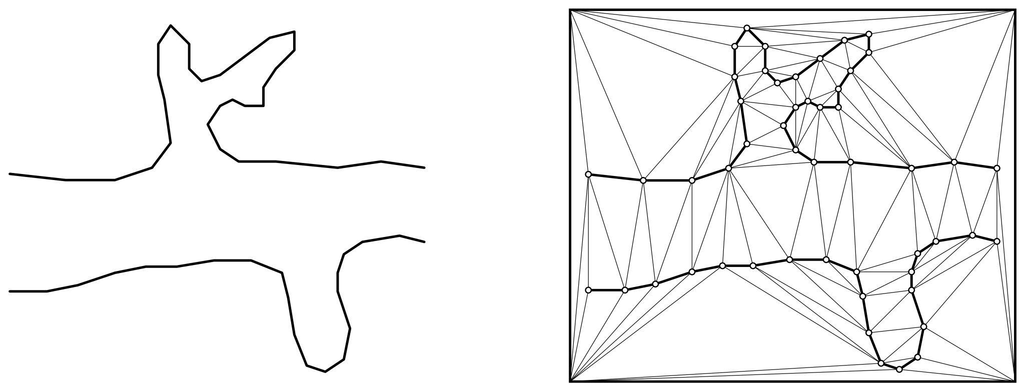


Figura 2.20: Fluxo de dados na aplicação de um ASG sobre um mapa.

2.4.1 Algoritmo de van der Poorten e Jones

Em 1999, van der Poorten e Jones descreveram um algoritmo [41] capaz de remover hierarquicamente os subfenômenos das linhas de uma mapa sem modificar a sua topologia. No seu algoritmo, a remoção dos subfenômenos é realizada de maneira explícita, baseando-se em métricas calculadas a partir da Triangulação de Delaunay Restrita (TDR) do mapa. O algoritmo identifica os subfenômenos, classifica-os pela métrica e remove os de métricas menores. A TDR também tem um papel fundamental na preservação da topologia do mapa. É com base nela que o mapa é organizado hierarquicamente em subfenômenos que contêm outros subfenômenos. Os menores subfenômenos, que não contêm nenhum outro, são sempre eliminados primeiro. Desta maneira, a remoção de um subfenômeno não afeta os outros e a topologia do mapa é preservada implicitamente.

A TDR difere-se da Triangulação de Delaunay (TD) tradicional por exigir que as arestas das linhas poligonais façam parte da triangulação, conforme ilustra a Figura 2.21.[4] O algoritmo de van der Poorten e Jones, publicado em 1999, aplica a TDR sobre todo o mapa original em uma etapa de pré-processamento e a reaplica novamente sobre todo o mapa após cada remoção de subfenômeno. Em 2002, no entanto, van der Poorten e Jones [42] publicaram uma segunda versão do algoritmo que, após uma remoção, reaplica a TDR apenas sobre a região do subfenômeno removido, aumentando o desempenho do algoritmo. A hierarquia, determinada a partir da TDR do mapa, é construída sob a forma de ramos e sub-ramos, que representam os subfenômenos das linhas.

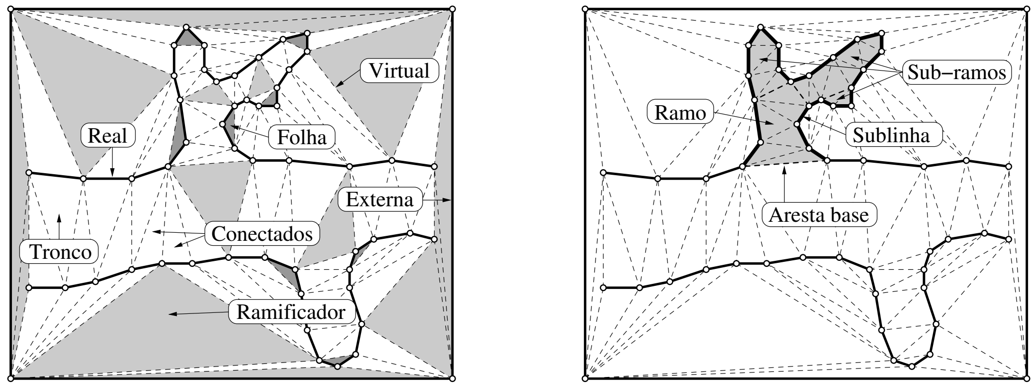


(a) Mapa original (b) TDR do mapa

Figura 2.21: Aplicação da TDR sobre um mapa.

Para determinar os ramos, o algoritmo classifica as arestas e os triângulos da TDR da seguinte maneira. Uma aresta é denominada real, se pertencer a uma das linhas do mapa; externa, se pertencer ao retângulo que o envolve; e virtual, se não pertencer às linhas nem ao retângulo. Um triângulo é denominado folha, se contiver duas arestas reais; tronco, se contiver uma única aresta real; e ramificador, se não contiver nenhuma aresta real. Triângulos que compartilham uma aresta virtual são ditos conectados. A Figura 2.22(a) ilustra esta classificação. Um ramo de uma linha é definido como a região formada por um conjunto contíguo de triângulos conectados, delimitada por uma seqüência de arestas reais (uma sublinha) desta linha e por uma aresta virtual de um triângulo ramificador, dita aresta base, conforme ilustra a Figura 2.22(b).

[4] A inclusão do retângulo que envolve o mapa original (Figura 2.21(b)) faz parte do algoritmo de van der Poorten e Jones. Os lados do retângulo devem ser considerados arestas da TDR.



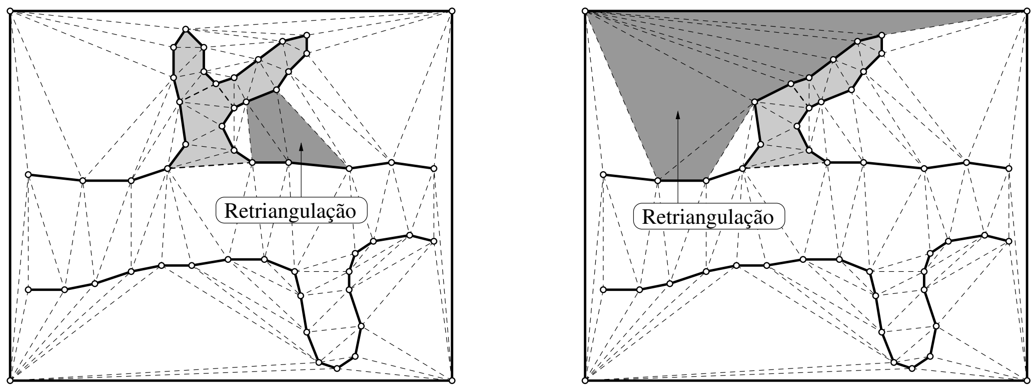
(a) Classificação de arestas e triângulos (b) Exemplo de ramos e sub-ramos

Figura 2.22: Determinação dos ramos das linhas de um mapa a partir da sua TDR.

Uma sublinha de uma linha do mapa que delimita um ramo da TDR define um subfenômeno daquela linha. Assim sendo, remover um subfenômeno é o mesmo que remover um ramo. A remoção de ramos é realizada da seguinte maneira. Primeiramente, o algoritmo calcula uma determinada métrica (envolvendo, por exemplo, a área, tamanho ou largura dos triângulos) para todos os ramos de todas as linhas.[5] O algoritmo, então, remove o ramo de menor métrica, substituindo a sublinha que o delimita pela sua aresta base. Este processo repete-se até que uma tolerância pré-determinada seja alcançada. Após cada remoção, a região do ramo excluído é retriangulada e as métricas dos ramos afetados são recalculadas. A Figura 2.23 ilustra as duas primeiras remoções.

Mesmo a segunda versão do algoritmo [42] possui muitas limitações. A primeira delas é a de que o algoritmo trata apenas linhas abertas e desconexas, não levando em consideração linhas fechadas, pontos, nem redes de linhas conectadas pelos seus extremos. Isto restringe bastante a sua aplicabilidade. Outra limitação é que, ao substituir a sublinha que delimita um ramo pela sua aresta base, o algoritmo elimina a suavidade da linha naquela região. Este procedimento também torna a simplificação desbalanceada, já que nas regiões em que ramos foram removidos há bem menos vértices do que nas que ficaram intactas. Uma terceira limitação é o baixo desempenho do algoritmo, que é o pior dentre os algoritmos citados, devido à inúmeras aplicações da TDR.

[5] A métrica de um ramo é, em geral, a soma das métricas dos seus triângulos. Ela é, portanto, sempre maior do que a dos seus sub-ramos, pois aquele contém todos os triângulos destes e mais alguns.



(a) Remoção do primeiro ramo (b) Remoção do segundo ramo

Figura 2.23: Remoção de ramos, seguida de retriangulação.

Em 2002, van der Poorten *et al.* [43] publicaram uma terceira versão do algoritmo adaptada à produção de mapas independentes de escala, mas que também corrigia algumas das limitações citadas. Para preservar a suavidade das regiões de ramos removidos, o algoritmo passou a inserir novos vértices e deslocar ligeiramente os existentes. O desbalanceamento das linhas foi amenizado com a aplicação do RDP a uma baixa tolerância, com o intuito único de remover o excesso de vértices. No que se refere ao tratamento de proximidade, os autores apenas sugerem uma forma de adicioná-la ao algoritmo [20]. O algoritmo passou a ser aplicado a linhas fechadas, mas ainda não inclui pontos, nem redes de linhas. Além disso, o seu baixo desempenho em nada foi melhorado.

2.5 Considerações Finais

Embora muito se tenha evoluído, os trabalhos em simplificação consistente ainda apresentam diversas limitações. Do ponto de vista prático, além do baixo desempenho e da falta de tratamento à proximidade, comum aos três algoritmos, despontam outras limitações. O algoritmo de De Berg *et al.* restringe-se à consistência de subdivisões planares e tem uso cartográfico bastante limitado. O algoritmo de Saalfeld, por sua vez, limita-se apenas a corrigir simplificações feitas pelo RDP e não é capaz, em certos casos, de eliminar todas as inconsistências topológicas. Já o algoritmo de van der Poorten e Jones não trata pontos, nem redes de linhas. Em compensação, é o único adaptado à produção de mapas independentes de escala. Nenhum destes algoritmos reúne todas as metas da consistência e todos tipos de objetos almejados neste trabalho.

Do ponto de vista teórico, apenas o trabalho de De Berg *et al.* desenvolveu um estudo sobre o problema da simplificação consistente. Este estudo resume-se, contudo, ao conceito de consistência, dado na Definição 5. Como demonstrado na Seção 2.3.1, a imposição das condições da Definição 5 aos vértices das linhas não impede o aparecimento de interseções e auto-interseções. Esta definição limita-se à consistência de pontos com respeito à áreas e não pode, portanto, ser utilizada no tratamento uniforme de pontos e vértices. Para que todas as metas de consistência sejam atingidas, condições mais restritivas devem ser impostas às linhas. Espera-se que, se tais condições também forem aplicados aos pontos do mapa, é possível tratar pontos e vértices uniformemente.

A Tabela 2.1 exibe um quadro que resume as limitações dos três trabalhos apresentados. O algoritmo proposto neste trabalho visa contornar estas limitações reunindo as vantagens destes trabalhos. O algoritmo utiliza URCs em uma estratégia de simplificação global e aplica a hipótese apresentada na Seção 1.3.2. Com isto ele reduz o número de vértices inseridos pela simplificação, melhorando o aspecto visual do mapa e o desempenho do processo. O algoritmo proposto pode ser utilizado sobre diferentes ASIs. Com isto ele é capaz de realizar uma remoção hierárquica de subfenômenos ao ser utilizado sobre o algoritmo de Visvalingam. Antes de descrever o algoritmo proposto no Capítulo 4, um estudo teórico da simplificação consistente, envolvendo tanto a questão da topologia, quanto a questão da proximidade, é apresentando no Capítulo 3.

	De Berg *et al.* [8]	Saalfeld [37]	van der Poorten e Jones [41, 42, 43]
Abordagem	Contexto	Contexto	Global
Objetos tratados	Pontos e áreas	Todos	Linhas e áreas
Pres. da topologia	Sim	Falha	Sim
Cons. de espaçamentos	Não	Não	Não
Mapas indep. de escala	Não	Não	Sim
Desempenho	Baixo	Baixo	Baixo

Tabela 2.1: Quadro comparativo dos principais trabalhos presentes na literatura.

Capítulo 3

Simplificação Consistente de Linha

Como mencionado no Capítulo 2, o estado da arte de simplificação consistente não apresenta um estudo teórico do problema que envolva todas as inconsistências discutidas nesta dissertação. O trabalho de De Berg *et al.*, do ponto de vista teórico, é apenas um pequeno passo nesta direção, pois resume-se à corretude de lado de pontos com respeito a áreas. Este capítulo introduz um estudo completo sobre a simplificação consistente em mapas não redundantes, admitindo como base o tratamento uniforme de pontos e vértices. O estudo demonstra que, para garantir a consistência do mapa, é suficiente analisar os vértices das linhas simplificadas, o que reforça o uso da abordagem de simplificação global. Em resumo, este capítulo compõe-se de um conjunto de condições que, se respeitadas pelos pontos e pelos vértices das linhas simplificadas, garantem: (1) a ausência de novas interseções entre os objetos, (2) a ausência de auto-interseções nas linhas, (3) a corretude de lado dos objetos e (4) a conservação de espaçamentos entre eles.

Este capítulo está organizado da seguinte maneira. A Seção 3.1 discute as vantagens do tratamento uniforme de pontos e vértices. Em seguida, a Seção 3.2 apresenta condições que evitam o aparecimento de novas interseções no mapa simplificado e trata as auto-interseções como casos particulares de interseções. A Seção 3.3, por sua vez, demonstra que estas condições também podem ser utilizadas na preservação dos lados dos objetos. Mais adiante, a Seção 3.4 introduz condições adicionais para a conservação de espaçamentos entre os objetos. Por fim, a Seção 3.5 apresenta as considerações finais deste capítulo, discutindo os pontos teóricos que serão utilizados no algoritmo.

3.1 Tratamento Uniforme de Pontos e Vértices

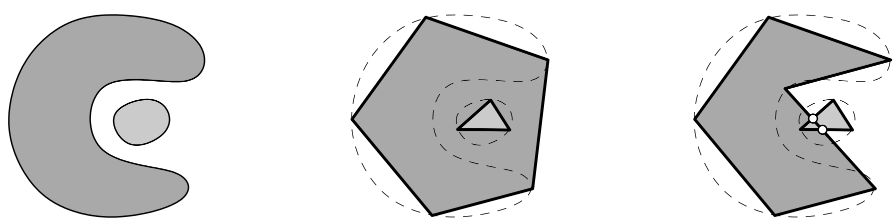
Tratar pontos e vértices de linhas uniformemente significa impor as mesmas restrições a estes para atingir uma ou mais metas de consistência. Antes de mais nada, é preciso saber se o tratamento uniforme é realmente possível. Será que a conservação de espaçamentos, por exemplo, pode ser atingida impondo-se restrições idênticas aos pontos e aos vértices das linhas? A resposta a esta pergunta é sim. Mesmo sabendo que pontos e linhas são objetos de dimensionalidades diferentes, a natureza das metas de consistência é bastante similar para ambos. Neste capítulo, será mostrado que, embora não seja bem sucedido no algoritmo de Saalfeld, o tratamento uniforme de pontos e vértices pode e deve ser utilizado na simplificação consistente de linha, por trazer apenas benefícios.

O maior destes benefícios é a homogeneidade da codificação de pontos externos. Na avaliação da consistência de uma linha simplificada, pontos, vértices de outras linhas e os vértices da própria linha podem ser vistos simplesmente como pontos externos. O algoritmo não precisa saber que vértices formam que segmentos, nem sequer precisa diferenciar vértices de pontos. Em outras palavras, eles podem ser codificados exatamente da mesma maneira. A consistência é alcançada impondo-se as mesmas restrições a todos os pontos externos, sem ser preciso tornar explícitos os objetos avaliados. Uma estratégia alternativa seria estruturar os segmentos explicitamente, armazenando os seus extremos juntos. Neste caso, seria preciso detectar as interseções e verificar os lados dos segmentos separadamente, além, é claro, de codificar segmentos e pontos de maneiras distintas.

É importante diferenciar o tratamento uniforme de pontos e vértices do tratamento unificado de metas de consistência. Tratar metas de maneira unificada significa atingir metas diferentes com restrições similares. Este é o caso dos algoritmos de De Berg *et al.* e de Saalfeld, que tratam a corretude de lado, a interseção e a auto-interseção de modo similar. Será mostrado ao longo deste capítulo que estas metas podem, de fato, ser atingidas com um tratamento unificado. A conservação de espaçamentos, no entanto, só pode ser atingida impondo-se restrições adicionais. Para dar ao estudo uma seqüência lógica, a ausência de interseções e auto-interseções será analisada primeiro, em seguida, a corretude de lado, cujas demonstrações se baseiam na questão anterior, e, por último, a conservação de espaçamentos, que não tem conexão com nenhuma das duas.

3.2 Ausência de Interseções

Sabe-se que as únicas interseções presentes em mapas sem redundâncias ocorrem entre extremos de linhas. Estas interseções sempre são preservadas pela simplificação, porque os algoritmos mantêm os extremos.[1] É suficiente, portanto, atentar-se a novas interseções que possam aparecer. Como base à teoria, considera-se que tais interseções podem envolver pontos, linhas e contornos de áreas, mas não as áreas em si. Casos em que objetos estão totalmente inseridos no exterior de uma área e, após a simplificação, aparecem no seu interior, ou vice-versa, seriam rigorosamente vistos como interseções, mas serão considerados aqui incorretudes de lado, como ilustra a Figura 3.1. Nestas circunstâncias, linhas e contornos de áreas podem ser vistos simplesmente como linhas.



(a) Original (b) Incorretude de lado (c) Interseção

Figura 3.1: Distinção entre incorretude de lado e interseção envolvendo uma área.

Visto que não há interseções entre pontos no mapa original e que a simplificação não os modifica, não haverá interseções entre eles no mapa simplificado. Por outro lado, interseções entre pontos e linhas, somente entre linhas ou em uma única linha podem ocorrer. As interseções entre pontos e linhas podem ser tratadas diretamente, verificando-se se os pontos do mapa não estão sobre às linhas simplificadas. As interseções entre linhas, por outro lado, precisam ser estudadas cuidadosamente, porque o tratamento uniforme de pontos e vértices não permite a verificação direta. Já as interseções em uma única linha ou auto-interseções podem ser estudadas como casos particulares de interseções, por serem facilmente mapeadas em casos equivalentes destas.

[1]A remoção ou o reposicionamento de um vértice extremo de uma linha pode ser aplicado na correção de inconsistências. Evidentemente, a remoção só pode ser realizada, se tal extremo não for compartilhado com outras linhas. O reposicionamento, por outro lado, pode ser aplicado sem problemas, se as linhas que o compartilham forem modificadas consistentemente. Em ambos os casos, deve-se ter conhecimento da conectividade das linhas, o que, em geral, não é o caso dos algoritmos de simplificação.

Esta seção analisa as interseções e auto-interseções envolvendo as respectivas simplificações $\mathcal{L}'_1$ e $\mathcal{L}'_2$ de duas linhas $\mathcal{L}_1$ e $\mathcal{L}_2$ do mapa original. Para aumentar a didática das demonstrações, o estudo é abordado de forma progressiva. Considera-se, a princípio, que as condições descritas devem ser respeitadas pelos vértices de $\mathcal{L}_1$ e $\mathcal{L}_2$ e que estas não se intersectam nem nos extremos. Nestas circunstâncias, são analisados casos de simplificação trivial, que são, em seguida, generalizados para casos de simplificação não trivial. Demonstra-se, então, que é suficiente que as condições sejam respeitadas apenas pelos vértices de $\mathcal{L}'_1$ e $\mathcal{L}'_2$. Por fim, são analisados dois casos particulares: as configurações em que $\mathcal{L}_1$ e $\mathcal{L}_2$ se conectam pelos extremos e as auto-interseções.

3.2.1 Casos de Simplificação Trivial

Suponha que, após a simplificação, uma dada linha $\mathcal{L}_1 = v_1 v_2 \cdots v_{n_1}$ seja substituída pela sua simplificação trivial $\mathcal{L}'_1 = v_1 v_{n_1}$ e que elas formem juntas o polígono simples $\mathcal{P}$, como ilustra a Figura 3.2(a). Seja $\overline{u_i u_{i+1}}$ uma aresta da linha $\mathcal{L}_2 = u_1 u_2 \cdots u_{n_2}$ que não intersecta $\mathcal{L}_1$ e cujos extremos u_i e u_{i+1} não estão sobre $\overline{v_1 v_{n_1}}$. Deseja-se saber se $\overline{u_i u_{i+1}}$ intersecta a linha simplificada $\mathcal{L}'_1$. Nesta situação, é possível determinar a existência de interseção apenas analisando-se o posicionamento relativo de u_i e u_{i+1} com respeito a $\mathcal{P}$. Há três posicionamentos possíveis: (1) u_i e u_{i+1} encontram-se ambos no interior de $\mathcal{P}$ (Figura 3.2(b)); (2) u_i e u_{i+1} encontram-se ambos no exterior de $\mathcal{P}$ (Figura 3.2(c)); e (3) u_i e u_{i+1} encontram-se em lados distintos de $\mathcal{P}$ (Figura 3.2(d)).

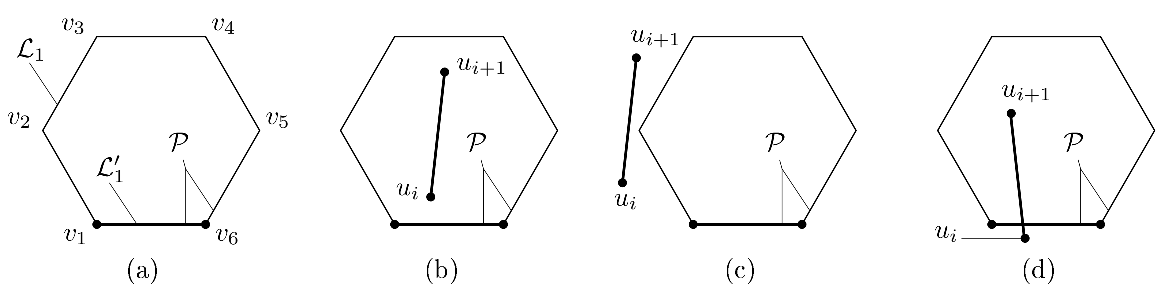


Figura 3.2: Análise do posicionamento dos extremos do segmento $\overline{u_i u_{i+1}}$ com respeito ao polígono $\mathcal{P}$, formado pela linha original $\mathcal{L}_1$ e pela sua simplificação trivial $\mathcal{L}'_1 = v_1 v_{n_1}$.

Observando-se a Figura 3.2, é fácil perceber que só haverá interseção, se u_i e u_{i+1} estiverem um no interior e o outro no exterior do polígono $\mathcal{P}$. De fato, nesta configuração, sempre haverá interseção, porque, para que u_i e u_{i+1} se encontrem em lados distintos, o segmento $\overline{u_i u_{i+1}}$ precisa necessariamente cruzar $\mathcal{P}$. Como $\overline{u_i u_{i+1}}$ não intersecta a linha original $\mathcal{L}_1$, este cruzamento só pode ser com o segmento $\overline{v_1 v_{n_1}}$. Além disso, visto que $\overline{u_i u_{i+1}}$ e $\overline{v_1 v_{n_1}}$ não podem ser colineares (em virtude das restrições descritas), eles só poderão se intersectar em um único ponto. Esta dedução ratifica a idéia de que, em casos similares ao da Figura 3.2, a existência de interseção pode ser determinada apenas analisando-se o posicionamento dos pontos u_i e u_{i+1} com respeito ao polígono $\mathcal{P}$.

Observe agora os exemplos da Figura 3.3. As Figuras 3.3(a) e 3.3(b) ilustram casos comuns de linhas em zig-zag e as Figuras 3.3(c) e 3.3(d), casos típicos de linhas em espiral. Em todos estes casos, a linha original $\mathcal{L}_1$ cruza a simplificada $\mathcal{L}'_1 = v_1 v_{n_1}$, formando o polígono complexo $\mathcal{P}$. Nestes casos, também é possível determinar a existência de interseção entre $\mathcal{L}'_1$ e um segmento $\overline{u_i u_{i+1}}$ de $\mathcal{L}_2$ com base no posicionamento de u_i e u_{i+1} com respeito a $\mathcal{P}$. No entanto, sendo $\mathcal{P}$ um polígono complexo, é preciso que os seus lados, interior e exterior, sejam definidos por uma regra que permita tal determinação. Para tanto, esta regra deve basear-se na Propriedade 1, conhecida como propriedade da paridade. A Figura 3.4 ilustra exemplos da aplicação desta propriedade.

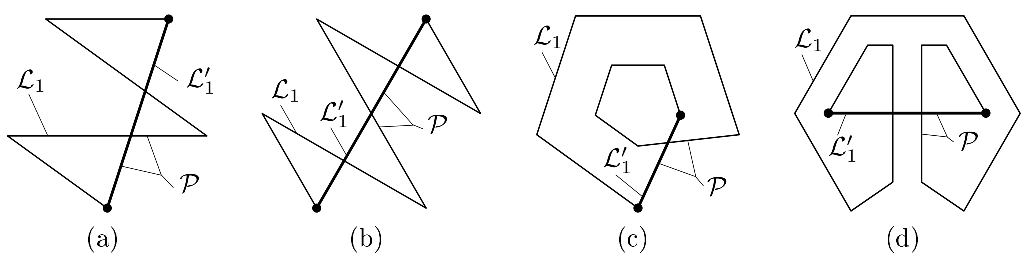


Figura 3.3: Simplificações triviais que produzem polígonos complexos.

Propriedade 1 (paridade)**.** Sejam $\mathcal{P}$ um polígono simples ou complexo e p_1 e p_2 um par de pontos no plano que não estão sobre $\mathcal{P}$. Os pontos p_1 e p_2 estarão do mesmo lado de $\mathcal{P}$, isto é, ambos no seu interior ou ambos no seu exterior, se um segmento de curva (ou de reta) qualquer que os liga cruzar $\mathcal{P}$ um número par de vezes. Se o número de cruzamentos for ímpar, os pontos p_1 e p_2 estarão em lados distintos de $\mathcal{P}$.[2]

[2]Observe que, se a curva que liga p_1 a p_2 cruzar $\mathcal{P}$ em pontos em que este se auto-intersecta (Figura 3.4(c)), o cruzamento da curva com cada segmento de $\mathcal{P}$ deve ser levado em conta. Além disso, não são contadas interseções que apenas tangenciam o polígono (Figura 3.4(d)).

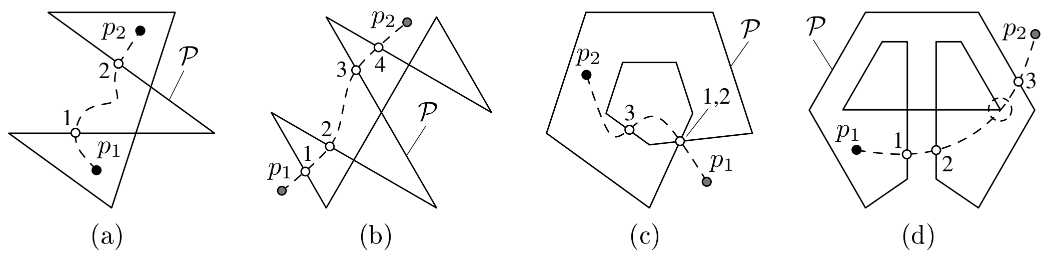


Figura 3.4: Aplicação da propriedade de paridade: (a, b) p_1 e p_2 encontram-se do mesmo lado de $\mathcal{P}$; (c, d) p_1 e p_2 encontram-se em lados distintos de $\mathcal{P}$.

A regra que define o interior e o exterior de um polígono $\mathcal{P}$ é a seguinte. Assume-se primeiramente que todo ponto no infinito está sempre no exterior de $\mathcal{P}$. Esta assunção serve de base para que se possa determinar, com uso da propriedade da paridade, os lados dos pontos que não estão no infinito. Assim sendo, um ponto p não sobreposto a $\mathcal{P}$ estará no seu exterior, se uma semi-curva (ou semi-reta) qualquer que parte de p e tende a (um ponto no) infinito cruzar $\mathcal{P}$ um número par de vezes (Figuras 3.5(a) e 3.5(b)), e no seu interior, caso contrário (Figuras 3.5(c) e 3.5(d)). Esta regra é válida tanto para polígonos simples quanto para complexos. Ela é bastante utilizada na área da Computação Gráfica e em áreas relacionadas para o preenchimento de polígonos [15].

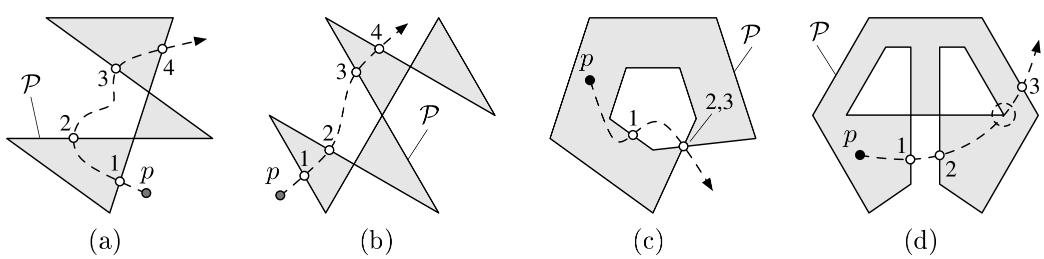


Figura 3.5: Determinação do interior e do exterior de polígonos complexos.

Para entender como $\mathcal{P}$ pode ser utilizado na determinação de interseções, é preciso atentar-se a uma conseqüência da propriedade da paridade. A fronteira entre o interior e o exterior do polígono $\mathcal{P}$ é definida pelas suas arestas. Portanto, ao passar do interior para o exterior de $\mathcal{P}$ e vice-versa, sempre se cruzará pelo menos uma aresta de $\mathcal{P}$. Assim sendo, mesmo nos casos de polígonos complexos, se u_i e u_{i+1} estiverem em lados distintos de $\mathcal{P}$, $\overline{u_i u_{i+1}}$ deverá necessariamente cruzar $\mathcal{P}$ pelo menos uma vez. Como este cruzamento não pode ser com $\mathcal{L}_1$, certamente será com a linha simplificada $\mathcal{L}'_1 = v_1 v_{n_1}$.

Esta dedução é válida apenas para os casos de simplificação trivial, em que $\mathcal{L}_1'$ consiste apenas do segmento $\overline{v_1 v_{n_1}}$, e é demonstrada formalmente pelo Lema 1.

Lema 1. *Sejam $\mathcal{L}_1 = v_1 v_2 \cdots v_{n_1}$ uma linha poligonal e $\mathcal{P}$ o polígono formado por $\mathcal{L}_1$ e pelo segmento $\overline{v_1 v_{n_1}}$. Seja $\overline{u_i u_{i+1}}$ uma aresta de uma linha $\mathcal{L}_2$ que está na vizinhança de $\mathcal{L}_1$, mas não a intersecta, e cujos extremos u_i e u_{i+1} não estão sobre $\overline{v_1 v_{n_1}}$. Se $\overline{u_i u_{i+1}}$ intersectar $\overline{v_1 v_{n_1}}$, então u_i e u_{i+1} estarão um no interior e o outro no exterior de $\mathcal{P}$.*

Demonstração. Suponha que u_i e u_{i+1} estejam ambos no interior ou ambos no exterior de $\mathcal{P}$. Neste caso, pela propriedade da paridade, $\overline{u_i u_{i+1}}$ cruzará $\mathcal{P}$ um número par de vezes. Visto que u_i e u_{i+1} não estão sobre $\overline{v_1 v_{n_1}}$ e que $\overline{u_i u_{i+1}}$ não intersecta $\mathcal{L}_1$, mas intersecta $\overline{v_1 v_{n_1}}$, tem-se que $\overline{u_i u_{i+1}}$ e $\overline{v_1 v_{n_1}}$ não são colineares. Em conseqüência disto, a interseção entre $\overline{u_i u_{i+1}}$ e $\overline{v_1 v_{n_1}}$ só pode ser em um único ponto. Para que o número de cruzamentos seja par, o outro ou outros cruzamentos de $\overline{u_i u_{i+1}}$ com $\mathcal{P}$ deverão ser necessariamente com $\mathcal{L}_1$, o que contraria a condição do lema de que $\mathcal{L}_1$ e $\mathcal{L}_2$ não se intersectam. $\square$

Com uso do Lema 1, é possível garantir que não há interseção entre a simplificação $\mathcal{L}_1' = v_1 v_{n_1}$ e o segmento $\overline{u_i u_{i+1}}$ da linha $\mathcal{L}_2$ que não intersecta $\mathcal{L}_1$, se u_i e u_{i+1} estiverem ambos no interior ou ambos no exterior de $\mathcal{P}$. É evidente que, se esta condição for obedecida por todos os segmentos de $\mathcal{L}_2$ que estão na vizinhança de $\mathcal{L}_1$, $\mathcal{L}_1'$ não intersectará $\mathcal{L}_2$. Na prática, no entanto, verificar esta condição não é factível. Isto porque a uniformidade no tratamento de pontos e vértices exige que a informação de quais pontos formam quais segmentos não seja utilizada. Além disso, é bem provável que, para muitos segmentos, um dos seus extremos não esteja na vizinhança de $\mathcal{L}_1$, de maneira a não ser considerado na análise de consistência, como destaca a Figura 3.6.

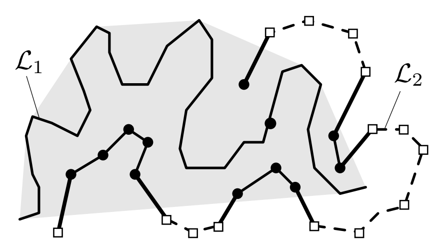


Figura 3.6: Alguns segmentos da linha $\mathcal{L}_2$ podem ter um extremo na vizinhança da linha $\mathcal{L}_1$ e o outro não, sendo este desconsiderado na análise de consistência.

Na prática, este problema pode ser reapresentado da seguinte maneira. Dados a linha original $\mathcal{L}_1 = v_1 v_2 \cdots v_{n_1}$, a sua simplificação $\mathcal{L}'_1 = v_1 v_{n_1}$ e o conjunto de pontos Q que estão na vizinhança de $\mathcal{L}_1$, deseja-se saber se os segmentos que não intersectam $\mathcal{L}_1$ e que têm, pelo menos, um extremo em Q podem intersectar $\mathcal{L}'_1$. Por não conhecer estes segmentos, o conjunto Q deve ser tratado como um todo. Distinguem-se, então, duas configurações. Na primeira, todos os pontos de Q estão no interior do polígono $\mathcal{P}$. Neste caso, ainda é possível haver interseção, pois um dos segmentos pode ter um extremo que não esteja em Q e que, por conseguinte, estará no exterior de $\mathcal{P}$. Na segunda, todos os pontos de Q estão no exterior de $\mathcal{P}$. Neste caso, os extremos que não estão em Q também estarão no exterior de $\mathcal{P}$. Portanto, pode-se garantir que não há interseção.

3.2.2 Casos de Simplificação Não Trivial

Até então, observou-se que a solução encontrada para os casos de simplificação trivial se fundamenta no polígono formado pela linha original $\mathcal{L}_1$ e pelo segmento $\overline{v_1 v_{n_1}}$. Para aplicar esta solução para uma linha simplificada $\mathcal{L}'_1$ não trivial, isto é, com mais de dois vértices, é preciso analisar cada segmento $\overline{v_i v_j}$ de $\mathcal{L}'_1$ e a sua sublinha correspondente $\mathcal{L}_{1[i,j]}$ individualmente. Por exemplo, para o caso da Figura 3.7(a), na qual $\mathcal{L}_1 = v_1 v_2 \cdots v_6$ e $\mathcal{L}'_1 = v_1 v_4 v_6$, esta análise é feita da seguinte maneira. Para o segmento $\overline{v_1 v_4}$, tem-se que o ponto u_i está no interior do polígono formado por $\overline{v_1 v_4}$ e $\mathcal{L}_{1[1,4]}$, como ilustra a Figura 3.7(b). O mesmo acontece para o segmento $\overline{v_4 v_6}$ e a sublinha $\mathcal{L}_{1[4,6]}$, como ilustra a Figura 3.7(c). Portanto, analisando-se o ponto u_i, não se pode garantir que o segmento $\overline{u_i u_{i+1}}$ não intersecte o segmento $\overline{v_1 v_4}$, nem o segmento $\overline{v_4 v_6}$.

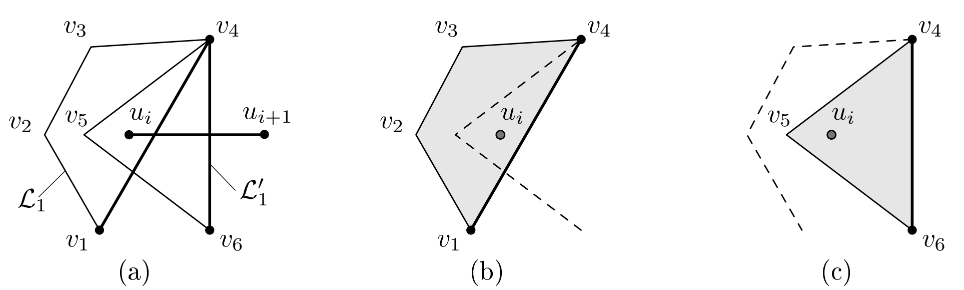


Figura 3.7: Análise da interseção entre (a) o segmento $\overline{u_i u_{i+1}}$ e a linha $\mathcal{L}'_1$ (b) para o polígono formado por $\overline{v_1 v_4}$ e $\mathcal{L}_{1[1,4]}$ e (c) para o polígono formado por $\overline{v_4 v_6}$ e $\mathcal{L}_{1[4,6]}$.

Com o intuito de simplificar posteriores explicações, assume-se, a partir deste ponto, a notação $\mathcal{P}_{\mathcal{L}_{1[i,j]}}$ para designar o polígono formado pela união da sublinha $\mathcal{L}_{1[i,j]}$ com o segmento $\overline{v_i v_j}$, denominado polígono associado a $\mathcal{L}_{1[i,j]}$. A solução para os casos de simplificação não trivial pode, então, ser apresentada da seguinte maneira. Para garantir que não haja interseções entre $\mathcal{L}'_1$ e os segmentos de $\mathcal{L}_2$ que tenham pelo menos um dos extremos no conjunto de pontos Q na vizinhança de $\mathcal{L}_1$, é preciso que, para cada sublinha $\mathcal{L}_{1[i,j]}$ substituída em $\mathcal{L}'_1$ pelo segmento $\overline{v_i v_j}$, o seu polígono associado $\mathcal{P}_{\mathcal{L}_{1[i,j]}}$ tenha todos os pontos de Q no seu exterior. Esta generalização direta do Lema 1 para casos de simplificação não trivial é apresentada formalmente no Lema 2.

Lema 2. *Sejam $\mathcal{L}_1 = v_1 v_2 \cdots v_{n_1}$ uma linha poligonal e $\mathcal{L}'_1 = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m_1)}$ uma linha simplificada de $\mathcal{L}_1$. Sejam $\mathcal{L}_2$ uma linha poligonal que não intersecta $\mathcal{L}_1$ e Q o conjunto dos vértices de $\mathcal{L}_2$ que estão na vizinhança de $\mathcal{L}_1$. Se, para cada sublinha $\mathcal{L}_{1[\alpha(i),\alpha(i+1)]}$ com $1 \leq i < m_1$, os pontos de Q estiverem todos no exterior do polígono $\mathcal{P}_{\mathcal{L}_{1[\alpha(i),\alpha(i+1)]}}$, então $\mathcal{L}'_1$ não intersectará $\mathcal{L}_2$.*

Demonstração. Sabe-se, a partir das condições descritas, que tanto os vértices de $\mathcal{L}_2$ que pertencem a Q, quanto os que não pertencem, estão todos no exterior de $\mathcal{P}_{\mathcal{L}_{1[\alpha(i),\alpha(i+1)]}}$ para um determinado i. Pelo Lema 1, tem-se, então, que os segmentos de $\mathcal{L}_2$ não intersectam o segmento $\overline{v_{\alpha(i)} v_{\alpha(i+1)}}$. Visto que isto é válido para todo intervalo $1 \leq i < m_1$ de vértices de $\mathcal{L}'_1$, tem-se que os segmentos de $\mathcal{L}_2$ não intersectam os segmentos de $\mathcal{L}'_1$. □

Um exemplo da aplicação prática do Lema 2 é o seguinte. A Figura 3.8(a) ilustra a linha $\mathcal{L}_1 = v_1 v_2 \cdots v_{10}$ em forma de espiral, a sua simplificação $\mathcal{L}'_1 = v_1 v_3 v_5 v_{10}$ e cinco pontos na vizinhança de $\mathcal{L}_1$. Sabe-se pelo Lema 2 que, para não haver interseções entre $\mathcal{L}'_1$ e segmentos que não intersectam $\mathcal{L}_1$ e que têm pelo menos um destes pontos como extremo, é preciso que estes pontos estejam no exterior dos polígonos $\mathcal{P}_{\mathcal{L}_{1[1,3]}}$, $\mathcal{P}_{\mathcal{L}_{1[3,5]}}$ e $\mathcal{P}_{\mathcal{L}_{1[5,10]}}$. As Figuras 3.8(b), 3.8(c) e 3.8(d) ilustram o interior destes polígonos em cinza claro e destacam os pontos que não respeitam as condições do Lema 2. Conclui-se que não é possível garantir que não há interseção entre $\mathcal{L}'_1$ e segmentos que tenham pelo menos um extremo dentre os pontos p_1, p_4 e p_5. Por outro lado, segmentos com extremos apenas dentre os pontos p_2 e p_3 só intersectarão $\mathcal{L}'_1$, se também intersectarem $\mathcal{L}_1$.

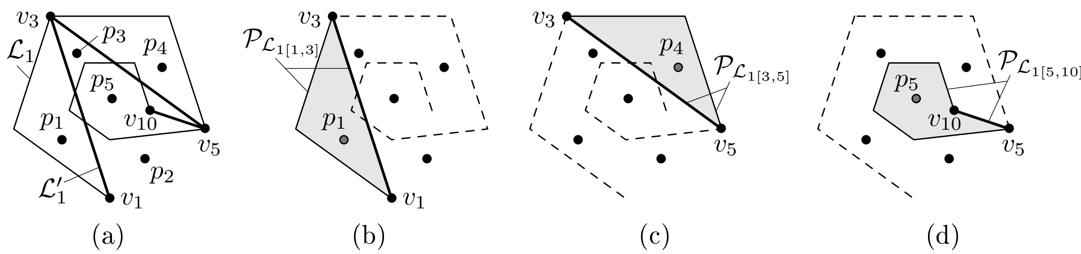


Figura 3.8: Análise de interseção entre a (a) linha $\mathcal{L}_1'$ e segmentos cujos extremos estão na vizinhança de $\mathcal{L}_1$ para os polígonos (b) $\mathcal{P}_{\mathcal{L}_{1[1,3]}}$, (c) $\mathcal{P}_{\mathcal{L}_{1[3,5]}}$ e (d) $\mathcal{P}_{\mathcal{L}_{1[5,10]}}$.

3.2.3 Ausência de Interseção no Mapa Simplificado

Dadas duas linhas $\mathcal{L}_1$ e $\mathcal{L}_2$ do mapa original que não se intersectam, o Lema 2 garante que, se os vértices de $\mathcal{L}_2$ respeitarem as suas condições, a simplificação $\mathcal{L}_1'$ não intersectará $\mathcal{L}_2$. De maneira análoga, se os vértices de $\mathcal{L}_1$ respeitarem as condições do Lema 2, a simplificação $\mathcal{L}_2'$ não intersectará $\mathcal{L}_1$. Em uma simplificação consistente, entretanto, o que se deseja é que $\mathcal{L}_1'$ e $\mathcal{L}_2'$ não se intersectem no mapa simplificado, já que $\mathcal{L}_1$ e $\mathcal{L}_2$ não se intersectam no mapa original. Assim como nos trabalhos de De Berg *et al.* e Saalfeld, espera-se que isto seja garantido pelo fato de que a simplificação é realizada por um algoritmo extrator. Em outras palavras, as linhas $\mathcal{L}_1'$ e $\mathcal{L}_2'$ contêm apenas vértices das suas versões originais correspondentes $\mathcal{L}_1$ e $\mathcal{L}_2$, respectivamente.

O que torna esta expectativa não tão evidente é o fato de que a demonstração do Lema 1 e, conseqüentemente, a do Lema 2 sustentam-se na condição de que as linhas $\mathcal{L}_1$ e $\mathcal{L}_2$ não se intersectam. Como esta condição influencia diretamente na contagem dos cruzamentos de segmentos de $\mathcal{L}_2$ com os polígonos associados de $\mathcal{L}_1'$, e vice-versa, não se pode garantir, por estes lemas, que $\mathcal{L}_1'$ e $\mathcal{L}_2'$ não se intersectam. Felizmente, as condições no Lema 2 são, de fato, suficientes para garantir que $\mathcal{L}_1'$ e $\mathcal{L}_2'$ não se intersectam, mas uma demonstração adicional é necessária, como apresenta formalmente o Teorema 1. Se, para cada par de linhas do mapa original, as condições do Teorema 1 forem respeitadas, então poder-se-á garantir que as linhas do mapa simplificado não se intersectarão.

Teorema 1. *Sejam $\mathcal{L}_1 = v_1 v_2 \cdots v_{n_1}$ e $\mathcal{L}_2 = u_1 u_2 \cdots u_{n_2}$ linhas poligonais que não se intersectam e sejam $\mathcal{L}_1' = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m_1)}$ e $\mathcal{L}_2' = u_{\beta(1)} u_{\beta(2)} \cdots u_{\beta(m_2)}$ linhas simplificadas de $\mathcal{L}_1$ e $\mathcal{L}_2$, respectivamente. Se (1) os vértices de $\mathcal{L}_2$ estiverem todos no exterior* *dos polígonos* $\mathcal{P}_{\mathcal{L}_{1[\alpha(i),\alpha(i+1)]}}$ *para* $1 \leq i < m_1$ *e (2) os vértices de* $\mathcal{L}_1$ *estiverem todos no exterior dos polígonos* $\mathcal{P}_{\mathcal{L}_{2[\beta(j),\beta(j+1)]}}$ *para* $1 \leq j < m_2$*, então* $\mathcal{L}'_1$ *e* $\mathcal{L}'_2$ *não se intersectarão.*

Demonstração. A partir do Lema 2 e da condição (1), tem-se que $\mathcal{L}'_1$ e $\mathcal{L}_2$ não se intersectam. A partir da condição (2) e do fato de que $\mathcal{L}'_1$ contém apenas vértices de $\mathcal{L}_1$, tem-se que os vértices de $\mathcal{L}'_1$ estão todos no exterior dos polígonos $\mathcal{P}_{\mathcal{L}_{2[\beta(j),\beta(j+1)]}}$ para $1 \leq j < m_2$. Visto que $\mathcal{L}'_1$ e $\mathcal{L}_2$ não se intersectam e que os vértices de $\mathcal{L}'_1$ estão todos no exterior dos polígonos $\mathcal{P}_{\mathcal{L}_{2[\beta(j),\beta(j+1)]}}$ para $1 \leq j < m_2$, tem-se, diretamente a partir do Lema 2, que $\mathcal{L}'_1$ e $\mathcal{L}'_2$ não se intersectam. □

3.2.4 Restrição das Condições às Linhas Simplificadas

Segundo o Teorema 1, para que $\mathcal{L}'_1$ e $\mathcal{L}'_2$ não se intersectem no mapa simplificado, é suficiente que os vértices das linhas originais $\mathcal{L}_1$ e $\mathcal{L}_2$ respeitem as suas condições. Por outro lado, de acordo com a hipótese deste trabalho (veja Seção 1.3.2), é suficiente observar apenas os vértices das próprias simplificações $\mathcal{L}'_1$ e $\mathcal{L}'_2$ para garantir que elas não se intersectem. Nesta seção, será demonstrado que, para que as linhas originais $\mathcal{L}_1$ e $\mathcal{L}_2$ não se intersectem, é suficiente que apenas os vértices de $\mathcal{L}'_1$ e $\mathcal{L}'_2$ respeitem as condições do Teorema 1. Em outras palavras, pode-se restringir as condições do Teorema 1 às linhas simplificadas. Para entender a base desta demonstração, é interessante analisar primeiramente o caso mais simples com simplificações triviais.

São dadas duas linhas $\mathcal{L}_1 = v_1v_2 \cdots v_{n_1}$ e $\mathcal{L}_2 = u_1u_2 \cdots u_{n_2}$ que não se intersectam. Considere que, após a simplificação, elas foram substituídas respectivamente pelas suas simplificações triviais $\mathcal{L}'_1 = v_1v_{n_1}$ e $\mathcal{L}'_2 = u_1u_{n_2}$ e que $\mathcal{L}_1$ e $\mathcal{L}'_1$ não se cruzam e $\mathcal{L}_2$ e $\mathcal{L}'_2$ não se cruzam. Suponha que $\mathcal{L}'_1$ e $\mathcal{L}'_2$ se intersectem e que os vértices u_1 e u_{n_2} estejam ambos no exterior do polígono $\mathcal{P}_{\mathcal{L}_1}$, como ilustra a Figura 3.9(a). As únicas configurações possíveis para a linha $\mathcal{L}_2$ são a da Figura 3.9(b), em que $\mathcal{L}_1$ cruza $\mathcal{L}_2$, e a da Figura 3.9(c), em que v_1 ou v_{n_1} está no interior de $\mathcal{P}_{\mathcal{L}_2}$. A primeira configuração não pode ocorrer, já que $\mathcal{L}_1$ e $\mathcal{L}_2$ não se intersectam. A única configuração possível, então, é a segunda. Ao se tentar traçar uma linha $\mathcal{L}_2$ que não cruze $\mathcal{L}_1$, pelo menos um dos extremos de $\mathcal{L}'_1$ ficará no interior de $\mathcal{P}_{\mathcal{L}_2}$. Se $\mathcal{L}_2$ fosse traçado primeiro, a situação seria análoga. Esta idéia é a base da demonstração do Teorema 2, que abrange simplificações $\mathcal{L}'_1$ e $\mathcal{L}'_2$ não triviais.

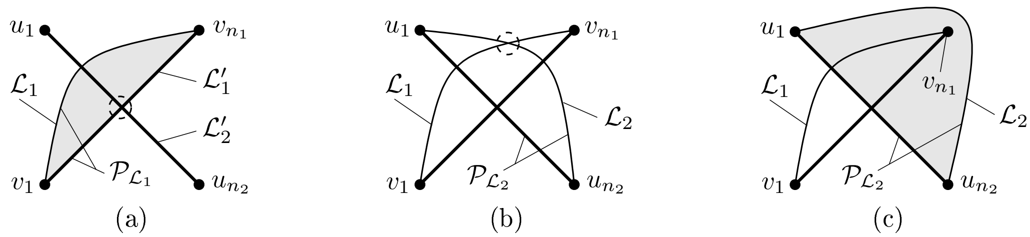


Figura 3.9: As configurações de $\mathcal{L}_2$, quando (a) as simplificações $\mathcal{L}'_1 = v_1 v_{n_1}$ e $\mathcal{L}'_2 = u_2 u_{n_2}$ se cruzam: (b) $\mathcal{L}_1$ e $\mathcal{L}_2$ intersectam-se e (c) $\mathcal{L}_1$ e $\mathcal{L}_2$ não se intersectam.

Teorema 2. *Sejam $\mathcal{L}_1 = v_1 v_2 \cdots v_{n_1}$ e $\mathcal{L}_2 = u_1 u_2 \cdots u_{n_2}$ linhas poligonais que não se intersectam e sejam $\mathcal{L}'_1 = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m_1)}$ e $\mathcal{L}'_2 = u_{\beta(1)} u_{\beta(2)} \cdots u_{\beta(m_2)}$ linhas simplificadas de $\mathcal{L}_1$ e $\mathcal{L}_2$, respectivamente. Se (1) os vértices de $\mathcal{L}'_2$ estiverem todos no exterior dos polígonos $\mathcal{P}_{\mathcal{L}_{1[\alpha(i),\alpha(i+1)]}}$ para $1 \leq i < m_1$ e (2) os vértices de $\mathcal{L}'_1$ estiverem todos no exterior dos polígonos $\mathcal{P}_{\mathcal{L}_{2[\beta(j),\beta(j+1)]}}$ para $1 \leq j < m_2$, então $\mathcal{L}'_1$ e $\mathcal{L}'_2$ não se intersectarão.*

Demonstração. Suponha que haja um cruzamento entre um segmento $\overline{v_{i_1} v_{j_1}}$ de $\mathcal{L}'_1$ e um segmento $\overline{u_{i_2} u_{j_2}}$ de $\mathcal{L}'_2$, que substituem respectivamente as sublinhas $\mathcal{L}_{1[i_1,j_1]}$ e $\mathcal{L}_{2[i_2,j_2]}$. Pela condição (1), tem-se que os vértices u_{i_2} e u_{j_2} estão ambos no exterior do polígono $\mathcal{P}_{\mathcal{L}_{1[i_1,j_1]}}$. Assim sendo, tem-se, pela propriedade da paridade, que o $\overline{u_{i_2} u_{j_2}}$ cruza $\mathcal{P}_{\mathcal{L}_{1[i_1,j_1]}}$ um número par de vezes. Visto que $\overline{v_{i_1} v_{j_1}}$ e $\overline{u_{i_2} u_{j_2}}$ se cruzam uma única vez, tem-se que $\overline{u_{i_2} u_{j_2}}$ cruza $\mathcal{L}_{1[i_1,j_1]}$ um número ímpar de vezes.

Pela condição (2), tem-se que os vértices v_{i_1} e v_{j_1}, extremos de $\mathcal{L}_{1[i_1,j_1]}$, estão ambos no exterior do polígono $\mathcal{P}_{\mathcal{L}_{2[i_2,j_2]}}$. Assim sendo, tem-se, pela propriedade da paridade, que a sublinha $\mathcal{L}_{1[i_1,j_1]}$ cruza $\mathcal{P}_{\mathcal{L}_{2[i_2,j_2]}}$ um número par de vezes. Disto e do fato de que $\overline{u_{i_2} u_{j_2}}$ cruza $\mathcal{L}_{1[i_1,j_1]}$ um número ímpar de vezes, tem-se que $\mathcal{L}_{1[i_1,j_1]}$ deve cruzar $\mathcal{L}_{2[i_2,j_2]}$ também um número ímpar de vezes. Portanto, $\mathcal{L}_{1[i_1,j_1]}$ e $\mathcal{L}_{2[i_2,j_2]}$ cruzam-se pelo menos uma vez, o que contraria a condição de que $\mathcal{L}_1$ e $\mathcal{L}_2$ não se intersectam.

Se $\overline{v_{i_1} v_{j_1}}$ e $\overline{u_{i_2} u_{j_2}}$ não se cruzarem, eles ainda podem se intersectar se forem colineares. Nestas circunstâncias, só há três configurações possíveis nas quais $\mathcal{L}_{1[i_1,j_1]}$ e $\mathcal{L}_{2[i_2,j_2]}$ não se intersectam, ilustradas na Figura 3.10. Nas três configurações, pelo menos um dos vértices u_{i_2} e u_{j_2} está sobre $\mathcal{P}_{\mathcal{L}_{1[i_1,j_1]}}$ ou pelo menos um dos vértices v_{i_1} e v_{j_1} está sobre $\mathcal{P}_{\mathcal{L}_{2[i_2,j_2]}}$, o que contraria respectivamente as condições (1) e (2). □

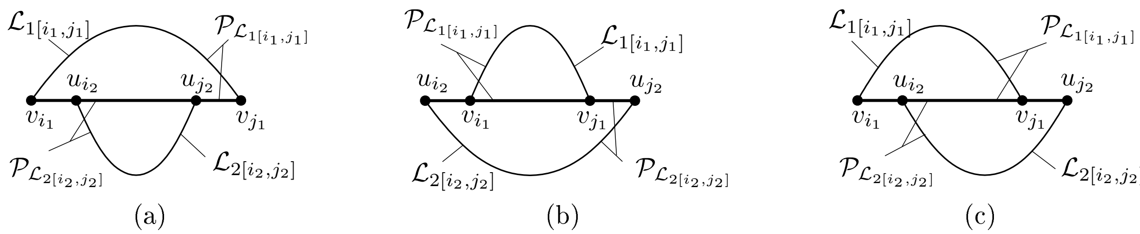


Figura 3.10: As configurações de colinearidade dos segmentos $\overline{v_{i_1}v_{j_1}}$ e $\overline{u_{i_2}u_{j_2}}$, quando as sublinhas $\mathcal{L}_{1[i_1,j_1]}$ e $\mathcal{L}_{2[i_2,j_2]}$ não se intersectam: (a) u_{i_2} e u_{j_2} estão sobre $\overline{v_{i_1}v_{j_1}}$; (b) v_{i_1} e v_{j_1} estão sobre $\overline{u_{i_2}u_{j_2}}$; e (c) u_{i_2} está sobre $\overline{v_{i_1}v_{j_1}}$ e v_{i_j} está sobre $\overline{u_{i_2}v_{j_2}}$.

3.2.5 Casos Particulares

Resta ainda analisar dois casos particulares. O primeiro ocorre quando as linhas $\mathcal{L}_1$ e $\mathcal{L}_2$ estão conectadas pelos extremos. Suponha que esta conexão seja entre o extremo v_1 de $\mathcal{L}_1$ e o extremo u_1 de $\mathcal{L}_2$, gerando uma situação como a da Figura 3.11(a). O que se deseja, então, é garantir que o segmento $\overline{u_1u_i}$ não intersecte $\mathcal{L}'_1$ em outro ponto além de v_1. Na realidade, devido à coincidência de v_1 e u_1, u_1 está sobre o polígono $\mathcal{P}_{\mathcal{L}_{1[1,3]}}$, como ilustra a Figura 3.11(b). Em relação a $\mathcal{P}_{\mathcal{L}_{1[1,3]}}$, é suficiente verificar apenas se o vértice u_i está no seu exterior. Em relação aos outros polígonos associados, no entanto, a verificação é a usual, isto é, os dois vértices u_1 e u_i devem estar no exterior dos polígonos, o que não é o caso de u_i com respeito ao polígono $\mathcal{P}_{\mathcal{L}_{1[3,5]}}$, como mostra a Figura 3.11(c).

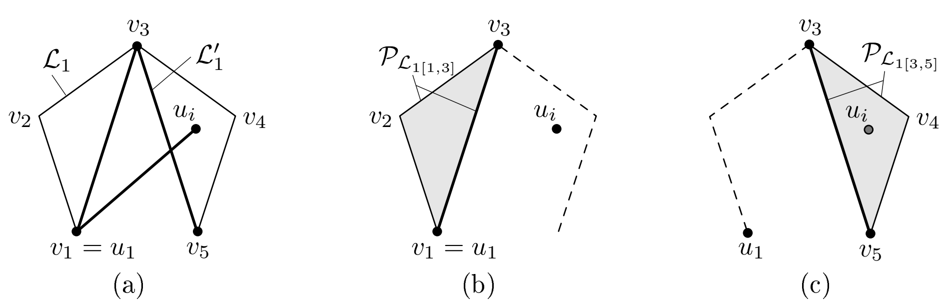


Figura 3.11: Análise do caso em que as linhas $\mathcal{L}_1$ e $\mathcal{L}_2$ estão conectadas por um extremo.

As outras situações em que $\mathcal{L}_1$ e $\mathcal{L}_2$ se conectam por um dos extremos são similares à da Figura 3.11. Existe, no entanto, a situação em que $\mathcal{L}_1$ e $\mathcal{L}_2$ se conectam pelos seus dois extremos, isto é, $v_1 = u_1$ e $v_{n_1} = u_{n_2}$, ou vice-versa. No caso da Figura 3.12(a), em relação a $\mathcal{P}_{\mathcal{L}_{1[1,3]}}$, é suficiente verificar o vértice u_{n_2} (Figura 3.12(b)) e, em relação a

$\mathcal{P}_{\mathcal{L}_{1[3,5]}}$, é suficiente verificar o vértice u_1 (Figura 3.12(c)). Na situação em que $\mathcal{L}'_1 = v_1 v_{n_1}$ e $\mathcal{L}'_2 = u_1 u_{n_2}$, ilustrada pela Figura 3.12(d), em relação a $\mathcal{P}_{\mathcal{L}_{1[1,5]}}$, ambos os vértices u_1 e u_{n_2} não são verificados, o que acaba resultando na coincidência dos segmentos $\overline{v_1 v_5}$ e $\overline{u_1 u_{n_2}}$. Este, no entanto, é o único tipo de interseção aceito no mapa simplificado, por ser considerado um colapso da área formada por $\mathcal{L}_1$ e $\mathcal{L}_2$ em uma linha.

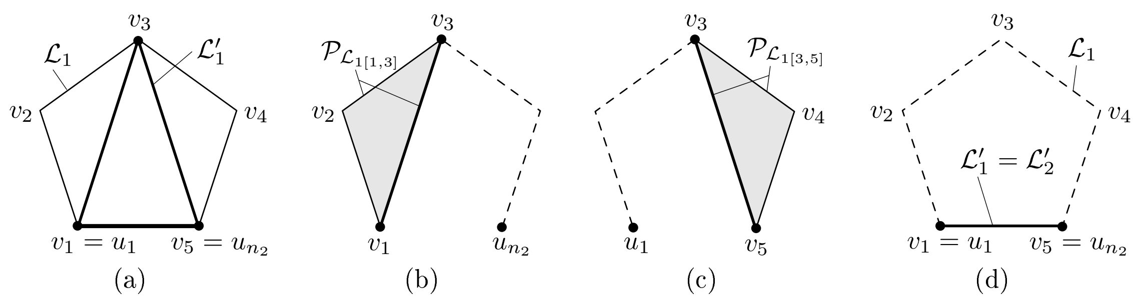


Figura 3.12: Análise do caso em que as linhas $\mathcal{L}_1$ e $\mathcal{L}_2$ estão conectadas por dois extremos.

O segundo caso particular é o de auto-interseção na linha $\mathcal{L}'_1$. As auto-interseções são casos particulares de interseções, porque podem ser reduzidas em casos similares de interseções. Observando-se a Figura 3.13, percebe-se que garantir que não há interseção entre os segmentos $\overline{v_1 v_4}$ e $\overline{v_5 v_6}$ de $\mathcal{L}'_1$ (Figura 3.13(a)) é equivalente a garantir que não há interseção entre os segmentos $\overline{u_1 u_4}$ e $\overline{t_2 t_3}$ das suas respectivas linhas $\mathcal{L}'_2$ e $\mathcal{L}'_3$, que se conectam por um dos extremos (Figura 3.13(b)). Portanto, para garantir que um dado segmento $\overline{v_{\alpha(i)} v_{\alpha(i+1)}}$ da simplificação $\mathcal{L}'_1 = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m_1)}$ não intersecte o restante da linha, é suficiente que os outros vértices de $\mathcal{L}'_1$, isto é, os vértices $v_{\alpha(k)}$, para todo $1 \leq k < i$ e $i+1 < k \leq m_1$, estejam no exterior do polígono $\mathcal{P}_{\mathcal{L}_{[\alpha(i),\alpha(i+1)]}}$.

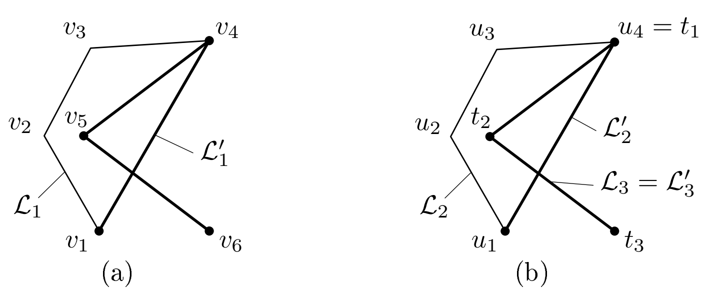


Figura 3.13: Redução da (a) auto-interseção entre os segmentos $\overline{v_1 v_4}$ e $\overline{v_5 v_6}$ da linha $\mathcal{L}'_1$ na (b) interseção entre os segmentos $\overline{u_1 u_4}$ e $\overline{t_2 t_3}$ das linhas $\mathcal{L}'_2$ e $\mathcal{L}'_3$.

3.3 Corretude de Lado

Quando as condições do Teorema 2 forem respeitadas por todas as linhas e pontos do mapa simplificado, pode-se garantir que novas interseções não foram introduzidas pela simplificação. Como mencionado na Seção 3.1, estas mesmas condições também garantem a corretude de lado dos objetos com respeito a áreas e linhas. Esta seção apresenta demonstrações formais que comprovam esta afirmação. Em particular, será demonstrado que a corretude de lado de um objeto com respeito a uma área pode ser alcançada de maneira implícita, apenas analisando as linhas que compõem o seu contorno. Assim sendo, linhas e áreas podem, na prática, ser tratadas de maneira similar, mesmo que a noção intuitiva de lado de uma linha e de uma área seja diferente.

Sabe-se que um fenômeno linear ou de área pode ser codificado tanto por uma única linha poligonal, quanto pelo encadeamento de várias delas, como ilustra a Figura 3.14. No contexto da corretude de lado, assim como no da auto-interseção, casos que envolvem uma única linha (Figuras 3.14(a) e 3.14(b)) podem ser facilmente tratados como casos que envolvem várias linhas (Figuras 3.14(c) e 3.14(d)), e vice-versa. Nesta situação, é suficiente analisar os casos que envolvem apenas uma linha, pois são mais simples. Assim sendo, distinguem-se três casos, de acordo com o objeto do qual se deseja preservar o lado: a corretude de lado de um ponto, de uma linha ou de uma área. A corretude de lado de um ponto será examinado primeiro, pois é a base para os outros dois casos.

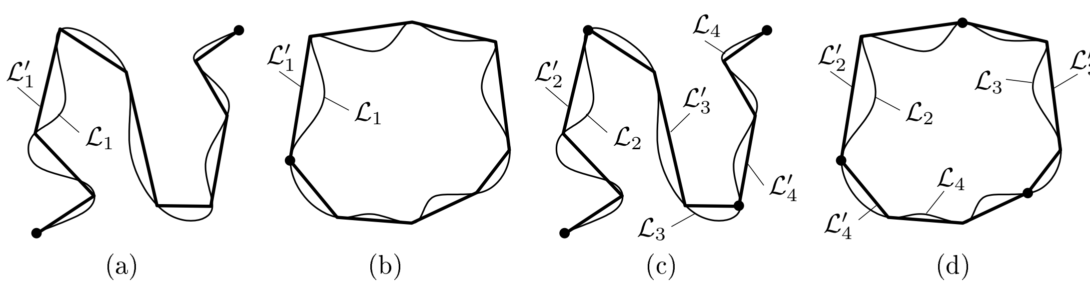


Figura 3.14: Exemplos de casos equivalentes: (a,b) uma única linha poligonal $\mathcal{L}_1$ e a sua simplificação $\mathcal{L}'_1$; e (c, d) três linhas poligonais $\mathcal{L}_1$, $\mathcal{L}_2$ e $\mathcal{L}_3$, conectadas pelos seus extremos, e as suas respectivas simplificações $\mathcal{L}'_1$, $\mathcal{L}'_2$ e $\mathcal{L}'_3$.

3.3.1 Corretude de Lado de um Ponto

Considere primeiramente o caso da corretude de lado de um ponto p com respeito a uma área. Suponha que esta área seja representada pela linha $\mathcal{L}_1 = v_1 v_2 \cdots v_{n_1}$, ilustrada pela Figura 3.15(a), e que a sua simplificação gera a linha $\mathcal{L}'_1 = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m_1)}$, ilustrada pela Figura 3.15(b). Pela regra da paridade, para que p esteja do mesmo lado de $\mathcal{L}_1$ e $\mathcal{L}'_1$, uma semi-reta qualquer partindo de p deve cruzar ambas as linhas um número par de vezes ou cruzar ambas as linhas um número ímpar de vezes. Em outras palavras, o número de cruzamentos não precisa ser o mesmo, mas a paridade sim. Uma condição suficiente para isto é que o ponto p esteja no exterior dos polígonos $\mathcal{P}_{\mathcal{L}_{1[\alpha(i),\alpha(i+1)]}}$ para todo $1 \leq i < m_1$, como ilustra a Figura 3.15(c). O Teorema 3 formaliza esta dedução.

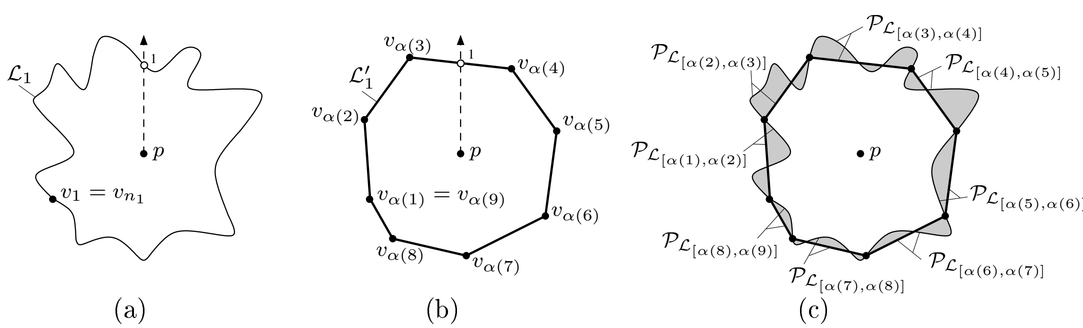


Figura 3.15: Corretude de lado do ponto p com respeito às linhas $\mathcal{L}_1$ e $\mathcal{L}'_1$.

Teorema 3. *Sejam $\mathcal{L}_1 = v_1 v_2 \cdots v_{n_1}$ uma linha fechada, $\mathcal{L}'_1 = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m_1)}$ uma linha simplificada de $\mathcal{L}_1$ e p um ponto no plano. Se p estiver no exterior dos polígonos $\mathcal{P}_{\mathcal{L}_{1[\alpha(i),\alpha(i+1)]}}$, para todo $1 \leq i < m_1$, então p estará do mesmo lado de $\mathcal{L}_1$ e $\mathcal{L}'_1$.*

Demonstração. Tome uma semi-reta r partindo de p que não intersecte nenhum vértice de $\mathcal{L}'_1$. Sabe-se, então, que r não intersecta nenhuma aresta de $\mathcal{L}'_1$ em mais de um ponto. Em outras palavras, ou r cruza a aresta ou sequer a intersecta. Como p está no exterior dos polígonos $\mathcal{P}_{\mathcal{L}_{1[\alpha(i),\alpha(i+1)]}}$, para todo $1 \leq i < m_1$, r cruzará cada polígono um número par de vezes. Assim sendo, se r cruzar $\overline{v_{\alpha(i)} v_{\alpha(i+1)}}$, ele cruzará $\mathcal{L}_{1[\alpha(i),\alpha(i+1)]}$ um número ímpar de vezes e, se r não cruzar $\overline{v_{\alpha(i)} v_{\alpha(i+1)}}$, ele cruzará $\mathcal{L}_{1[\alpha(i),\alpha(i+1)]}$ um número par de vezes. Nas duas situações, a paridade do número de cruzamentos será a mesma. Portanto, a soma dos cruzamentos de r com $\mathcal{L}_1$ terá a mesma paridade que a soma dos cruzamentos de r com $\mathcal{L}'_1$ e, conseqüentemente, p estará do mesmo lado de $\mathcal{L}_1$ e $\mathcal{L}'_1$. $\square$

Observando-se o caso da Figura 3.16, é fácil perceber que a condição do Teorema 3 é suficiente, mas não necessária à corretude de lado de um ponto. O ponto p encontra-se do mesmo lado de $\mathcal{L}_1$ e $\mathcal{L}'_1$ (Figuras 3.16(a) e 3.16(b)), muito embora não esteja no exterior de todos os polígonos (Figura 3.16(c)), como exige a condição. Isto ocorre porque a sublinha $\mathcal{L}_{1[\alpha(4),\alpha(5)]}$ de $\mathcal{L}_1$ cruza o segmento $\overline{v_{\alpha(3)}v_{\alpha(4)}}$ de $\mathcal{L}'_1$, formando a região na qual p se encontra (Figura 3.16(d)). Qualquer ponto que estivesse nesta região estaria no exterior de ambas as linhas. Casos similares a este acontecem sempre que uma sublinha de $\mathcal{L}_1$ cruza um segmento de $\mathcal{L}'_1$, correspondente a outra sublinha de $\mathcal{L}_1$. O uso desta condição tem uma importância, no entanto, que será discutida mais adiante.

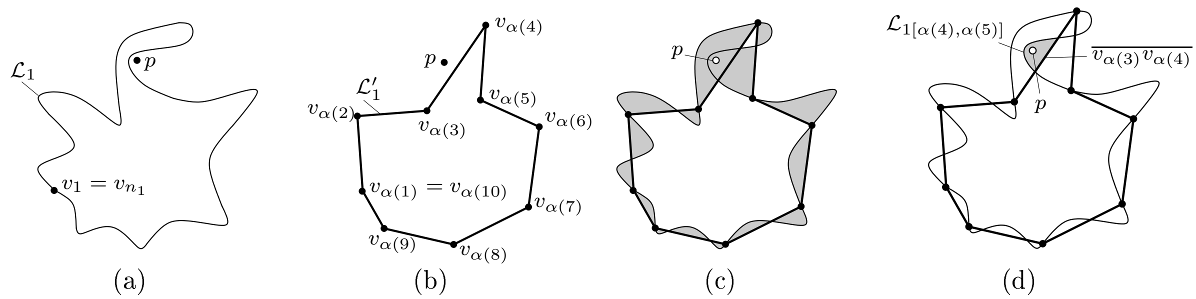


Figura 3.16: Caso que demonstra que a condição do Teorema 3 é suficiente, mas não necessária à corretude de lado do ponto p com respeito às linhas $\mathcal{L}_1$ e $\mathcal{L}'_1$.

Antes de analisar o caso da corretude de lado de um ponto com respeito a uma linha, é preciso ter em mente que a noção de lado de uma linha não é tão intuitiva quanto a de uma área. Embora seja simples estabelecer os lados da linha nas suas proximidades, em função da sua orientação, como ilustra a Figura 3.17(a), não é tão intuitivo dividir todo o espaço à sua volta em regiões bem delimitadas. Por exemplo, observando a Figura 3.17(b), não é possível dizer, de maneira intuitiva, em que lado da linha estão os pontos ao seu redor. Esta situação agrava-se no exemplo da Figura 3.17(c), em que a linha tem forma de espiral. Assim sendo, em vez de definir os lados de uma linha explicitamente, optou-se aqui por definir a corretude de lado de um ponto com respeito a uma linha e à sua simplificação, conforme apresentado na Definição 6.

Definição 6. Sejam $\mathcal{L}_1 = v_1 v_2 \cdots v_{n_1}$ uma linha poligonal, $\mathcal{L}'_1 = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m_1)}$ uma linha simplificada de $\mathcal{L}_1$ e p um ponto no plano. O ponto p é dito do mesmo lado de $\mathcal{L}_1$ e $\mathcal{L}'_1$, se e somente se estiver no exterior dos polígonos $\mathcal{P}_{\mathcal{L}_{[\alpha(i),\alpha(i+1)]}}$, para todo $1 \leq i < m_1$.

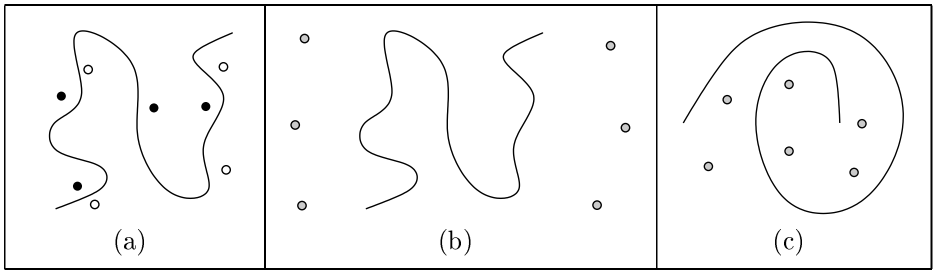


Figura 3.17: Exemplos de que a noção de lado de linhas não é tão intuitiva.

Esta definição pode ser comparada ao conceito de consistência dado na Definição 5. No caso de linhas sem espirais (Figuras 3.18(a) e 3.18(b)), percebe-se que esta definição é mais restritiva, porque, ao contrário da Definição 5, ela exige não apenas que o ponto p mantenha-se do lado correto de $\mathcal{L}_1$ como um todo, mas também das sublinhas $\mathcal{L}_{1[1,i]}$ e $\mathcal{L}_{1[i,n_1]}$ individualmente. No caso de espirais (Figuras 3.18(c) e 3.18(d)), por outro lado, esta definição acaba sendo menos restritiva, porque permite, em alguns casos, que p apareça no meio do espiral. Embora, à primeira vista, a Definição 5 exiba um resultado mais agradável para espirais, ela exige, na prática, um número elevado de vértices, que mais prejudicaria do que melhoraria a legibilidade do mapa em uma escala reduzida.

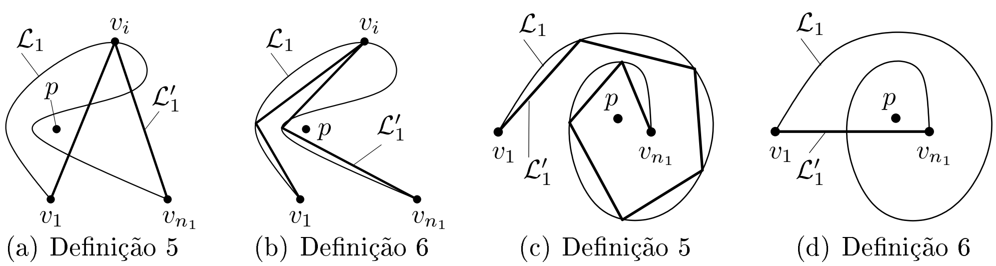
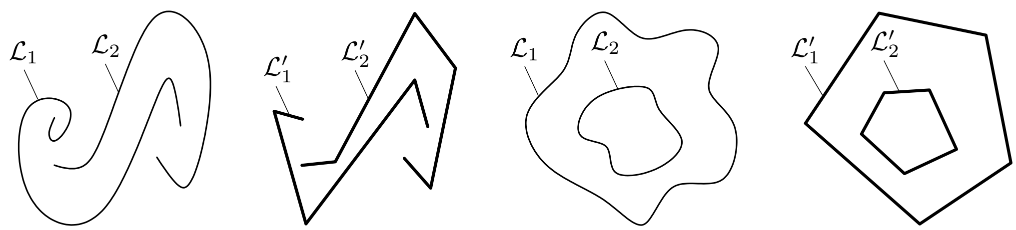


Figura 3.18: Comparação da noção de lado das Definições 5 e 6.

Fica claro que é possível tratar linhas e áreas de maneira similar, visto que a condição do Teorema 3, que é semelhante à da Definição 6, é imposta apenas à linha que compõe o contorno da área, mas não a área em si. Portanto, embora a condição do Teorema 3 seja suficiente, mas não necessária à corretude de lado de pontos com respeito a áreas, como ilustrado pela Figura 3.16, ela permite que as áreas sejam tratadas de maneira implícita. Esta condição também possibilita o tratamento uniforme de pontos e vértices na corretude de lado. A razão disso, como será demonstrado na Seção 3.3.2, é que esta condição também é a base para a corretude de lado de linhas e áreas.

3.3.2 Corretude de Lado de Linhas e Áreas

Ao analisar a corretude de lado de um ponto p com respeito a uma linha ou área $\mathcal{L}_1$ e à sua simplificação $\mathcal{L}'_1$, não foi preciso se preocupar com a localização de p no mapa simplificado, já que pontos não são deslocados, muito menos com o lado de $\mathcal{L}'_1$ com respeito a p, já que a idéia de lado não se aplica a pontos. A relação de lado entre linhas e áreas, no entanto, além de ser mútua, deve considerar que o processo simplificador pode alterar a forma destes objetos. Nos casos das Figuras 3.19(a) e 3.19(c), deseja-se saber se as suas simplificações, ilustradas respectivamente pelas Figuras 3.19(b) e 3.19(d), mantêm a mesma relação de lado. Como esta relação é mútua, é preciso verificar tanto o lado de $\mathcal{L}'_1$ com respeito a $\mathcal{L}_2$ e $\mathcal{L}'_2$, quanto o lado de $\mathcal{L}'_2$ com respeito a $\mathcal{L}_1$ e $\mathcal{L}'_1$.



(a) (b) (c) (d)

Figura 3.19: Corretude de lado entre as linhas $\mathcal{L}_1$ e $\mathcal{L}_2$ e as suas simplificações $\mathcal{L}'_1$ e $\mathcal{L}'_2$ (a, b) quando ambas as linhas são fechadas e (c, d) quando ambas as linhas são abertas.

Quando ambas as linhas são fechadas, uma sempre estará inserida totalmente em um lado da outra. Neste caso, no qual a noção de lado é intuitiva, a mútua corretude de lado é garantida pelas mesmas condições do Teorema 2, conforme formaliza o Corolário 1. A razão disso é que tais condições garantem tanto que os vértices de uma linha estejam do lado correto com respeito à outra, quanto que as linhas simplificadas não se intersectem. Quando ambas as linhas são abertas, a noção de lado não é tão intuitiva. No entanto, ainda é possível tratar a corretude de lado de linhas e áreas de maneira similar. Para isto, a corretude de lado entre linhas abertas é dada pela Definição 7, que tem as mesmas condições do Corolário 1. Casos em que uma linha é aberta e a outra fechada são tratados de maneira similar, novamente impondo as mesmas condições do Corolário 1.

Corolário 1. *Sejam* $\mathcal{L}_1 = v_1 v_2 \cdots v_{n_1}$ *e* $\mathcal{L}_2 = u_1 u_2 \cdots u_{n_1}$ *linhas fechadas que não se intersectam e sejam* $\mathcal{L}'_1 = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m_1)}$ *e* $\mathcal{L}'_2 = u_{\beta(1)} u_{\beta(2)} \cdots u_{\beta(m_2)}$ *linhas simplifi-*

cadas de $\mathcal{L}_1$ e $\mathcal{L}_2$, respectivamente. Se (1) os vértices de $\mathcal{L}'_2$ estiverem todos no exterior dos polígonos $\mathcal{P}_{\mathcal{L}_{[\alpha(i),\alpha(i+1)]}}$, para todo $1 \leq i < m_1$, e (2) os vértices de $\mathcal{L}'_1$ estiverem todos no exterior dos polígonos $\mathcal{P}_{\mathcal{L}_{2[\beta(j),\beta(j+1)]}}$, para todo $1 \leq j < m_2$, então $\mathcal{L}'_1$ estará do mesmo lado de $\mathcal{L}_2$ e $\mathcal{L}'_2$, assim como $\mathcal{L}'_2$ estará do mesmo lado de $\mathcal{L}_1$ e $\mathcal{L}'_1$.

Demonstração. Tem-se, pelas condições (1) e (2) e pelo Teorema 3, que os vértices $\mathcal{L}'_1$ estão do mesmo lado de $\mathcal{L}_2$ e $\mathcal{L}'_2$ e os vértices $\mathcal{L}'_2$ estão do mesmo lado de $\mathcal{L}_1$ e $\mathcal{L}'_1$. Tem-se, pelas condições (1) e (2) e pelo Teorema 2, que $\mathcal{L}'_1$ e $\mathcal{L}'_2$ que não se intersectam. Assim sendo, não só os vértices de $\mathcal{L}'_1$, mas também os pontos intermediários das suas arestas estão do mesmo lado de $\mathcal{L}_1$ e $\mathcal{L}'_1$, assim como não só os vértices de $\mathcal{L}'_2$, mas também os pontos intermediários das suas arestas estão do mesmo lado de $\mathcal{L}_1$ e $\mathcal{L}'_1$. Portanto, $\mathcal{L}'_1$ está do mesmo lado de $\mathcal{L}_2$ e $\mathcal{L}'_2$, assim como $\mathcal{L}'_2$ está do mesmo lado de $\mathcal{L}_1$ e $\mathcal{L}'_1$. □

Definição 7. Seja $\mathcal{L}_1 = v_1 v_2 \cdots v_{n_1}$ e $\mathcal{L}_2 = u_1 u_2 \cdots u_{n_1}$ linhas abertas que não se intersectam e sejam $\mathcal{L}'_1 = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m_1)}$ e $\mathcal{L}'_2 = u_{\beta(1)} u_{\beta(2)} \cdots u_{\beta(m_2)}$ linhas simplificadas de $\mathcal{L}_1$ e $\mathcal{L}_2$, respectivamente. Diz-se que $\mathcal{L}'_1$ está do mesmo lado de $\mathcal{L}_2$ e $\mathcal{L}'_2$ e que $\mathcal{L}'_2$ está do mesmo lado de $\mathcal{L}_1$ e $\mathcal{L}'_1$, se e somente se (1) os vértices de $\mathcal{L}'_2$ estiverem todos no exterior dos polígonos $\mathcal{P}_{\mathcal{L}_{[\alpha(i),\alpha(i+1)]}}$, para todo $1 \leq i < m_1$, e (2) os vértices de $\mathcal{L}'_1$ estiverem todos no exterior dos polígonos $\mathcal{P}_{\mathcal{L}_{2[\beta(j),\beta(j+1)]}}$, para todo $1 \leq j < m_2$,

3.3.3 Casos Particulares

Resta ainda analisar dois casos particulares. No primeiro caso, as linhas $\mathcal{L}_1$ e $\mathcal{L}_2$ conectam-se por um dos extremos, como ilustra a Figura 3.20(a). Neste exemplo, fica claro que não é preciso verificar a corretude de lado dos vértices $v_{\alpha(2)}$ de $\mathcal{L}'_1$ em relação a $\mathcal{L}'_2$ e $u_{\beta(1)}$ de $\mathcal{L}'_2$ em relação a $\mathcal{L}'_1$, já que estes vértices coincidem. Isto também é válido para casos envolvendo linhas fechadas. No segundo caso, deseja-se saber se parte uma linha $\mathcal{L}'_1$ mudou de lado com respeito a outras partes dela mesma, acarretando em uma mudança de orientação de $\mathcal{L}'_1$ em relação a $\mathcal{L}_1$, como ilustra a Figura 3.20(b). Este caso, envolvendo uma única linha, pode ser tratado como o caso de duas linhas conectadas, ilustrado pela Figura 3.20(a). Assim sendo, deve-se verificar o lado de cada vértice $v_{\alpha(i)}$ de $\mathcal{L}_1$ com cada segmento $\overline{v_{\alpha(j)} v_{\alpha(j+1)}}$, para todo $1 \leq i \leq m_1$, $1 \leq j < i-1$ e $i < j < m_1$.

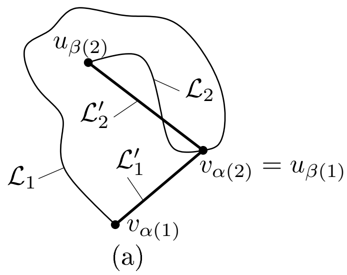


Figura 3.20: Casos particulares de corretude de lado: (a) as linhas $\mathcal{L}_1$ e $\mathcal{L}_2$ conectam-se por um extremo; (b) a linha simplificada $\mathcal{L}'_1$ muda de orientação em relação à original $\mathcal{L}_1$.

3.4 Conservação de Espaçamentos

A conservação de espaçamentos é o último aspecto tratado pela simplificação consistente proposta neste trabalho. Para que seja alcançada, condições adicionais às discutidas até aqui precisam ser respeitadas. A conservação de espaçamentos é uma restrição geométrica que tem como propósito permitir a distinção entre as fronteiras de objetos desconexos, evitando que estes, quando muito próximos, pareçam se intersectar. Na realidade, visto que, até então, parâmetros importantes como a espessura das linhas e o tamanho dos pontos não foram considerados, é bem provável que vários objetos possam, de fato, se intersectar, modificando a topologia do mapa. Esta situação agrava-se quando o mapa vetorial é exibido em um dispositivo matricial de baixa resolução. A Figura 3.21 ilustra exemplos de interseções em virtude da proximidade dos objetos.

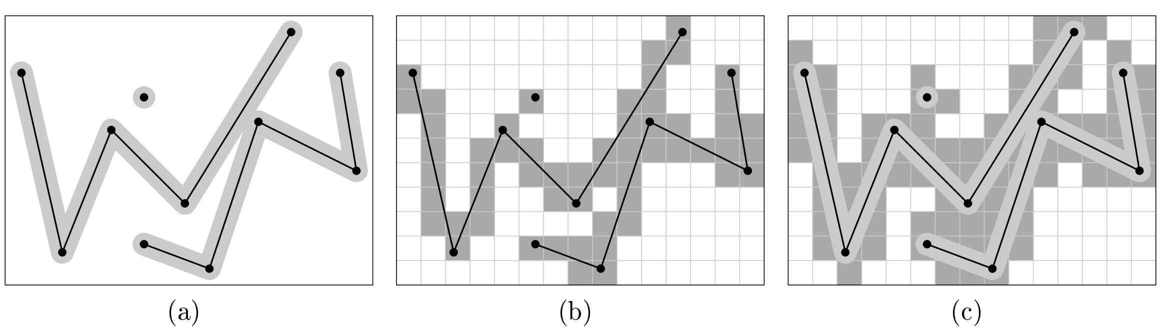


Figura 3.21: Casos em que a proximidade dos objetos pode causar interseções (a) devido à espessura das linhas e ao tamanho dos pontos, (b) devido à baixa resolução do dispositivo matricial de exibição ou (c) devido a ambos os fatores.

Assume-se, para fins teóricos, que os objetos do mapa original estão separados por um espaçamento perceptível e que tais conflitos de proximidade só podem ser gerados pela simplificação. O que se deseja, então, é manter entre os objetos uma distância mínima, que já existia antes da simplificação. A determinação desta distância, entretanto, não faz parte do escopo deste trabalho. Considera-se aqui que a simplificação deve conservar entre pontos, linhas e contornos de áreas, no mínimo, uma distância δ previamente determinada. Na conservação de espaçamentos, como nas questões anteriores, contornos de áreas podem ser tratados simplesmente como linhas. Assim sendo, a conservação de espaçamentos pode ser dividida em casos entre ponto e linha e somente entre linhas.

3.4.1 Conservação de Espaçamentos entre Ponto e Linha

Sejam $\mathcal{L}_1 = v_1 v_2 \cdots v_{n_1}$ uma linha poligonal, $\mathcal{L}'_1 = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m_1)}$ uma linha simplificada de $\mathcal{L}_1$ e p um ponto no plano a uma distância perceptível de $\mathcal{L}_1$. Deseja-se saber se, após a simplificação, $\mathcal{L}'_1$ se mantém a pelo menos uma dada distância δ de p. É fácil perceber que a condição necessária e suficiente para que isto aconteça é que p se conserve a uma distância maior ou igual a δ dos segmentos de $\mathcal{L}'_1$. Em outras palavras, o espaçamento entre $\mathcal{L}'_1$ e p será conservado, se e somente se a distância entre o ponto p e o segmento $\overline{v_{\alpha(i)} v_{\alpha(i+1)}}$ for maior ou igual a δ para todo $1 \leq i < m_1$. No exemplo da Figura 3.22, apenas os pontos p_1 e p_4 não satisfazem esta condição, pois se encontram na região próxima a $\mathcal{L}'_1$, isto é, a uma distância menor do que δ de $\mathcal{L}'_1$.

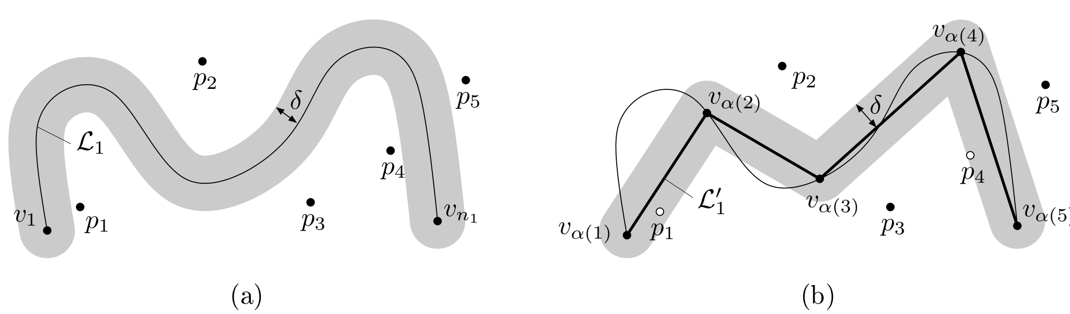


Figura 3.22: Análise do espaçamento entre ponto e linha: (a) a linha original $\mathcal{L}_1$ está espaçada de todos os pontos e (b) a simplificação $\mathcal{L}'_1$ está próxima de p_1 e p_4.

3.4.2 Conservação de Espaçamentos entre Linhas

Sejam $\mathcal{L}_1 = v_1 v_2 \cdots v_{n_1}$ e $\mathcal{L}_2 = u_1 u_2 \cdots u_{n_2}$ linhas poligonais que estão a uma distância perceptível uma da outra e $\mathcal{L}'_1 = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m_1)}$ e $\mathcal{L}'_2 = v_{\beta(1)} v_{\beta(2)} \cdots v_{\beta(m_2)}$ linhas simplificadas de $\mathcal{L}_1$ e $\mathcal{L}_2$, respectivamente. Deseja-se saber se, após a simplificação, $\mathcal{L}'_1$ e $\mathcal{L}'_2$ se mantêm a pelo menos uma distância δ uma da outra. Para que isto aconteça, é necessário e suficiente que cada vértice de $\mathcal{L}'_1$ se conserve a uma distância maior ou igual a δ dos segmentos de $\mathcal{L}'_2$, assim como cada vértice de $\mathcal{L}'_2$ se conserve a uma distância maior ou igual a δ dos segmentos de $\mathcal{L}'_1$, como estabelece o Teorema 4. No exemplo da Figura 3.23, apenas os vértices $v_{\alpha(5)}$ e $u_{\beta(2)}$ não satisfazem estas condições, pois se encontram nas regiões próximas a $\mathcal{L}'_2$ e $\mathcal{L}'_1$, respectivamente.

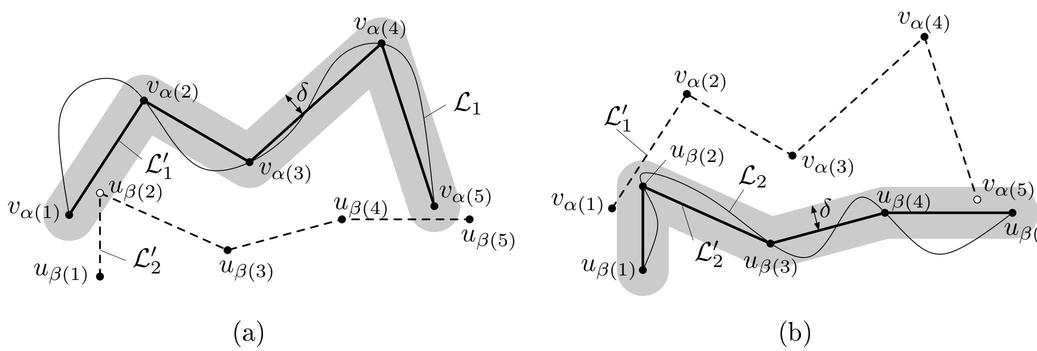


Figura 3.23: Análise do espaçamento entre linhas: (a) o vértice $u_{\beta(2)}$ de $\mathcal{L}'_2$ está próximo à aresta $\overline{v_{\alpha(1)} v_{\alpha(2)}}$ de $\mathcal{L}'_1$; e (b) o vértice $v_{\alpha(5)}$ de $\mathcal{L}'_1$ está próximo à aresta $\overline{u_{\beta(4)} u_{\beta(5)}}$ de $\mathcal{L}'_2$.

Teorema 4. *Sejam $\mathcal{L}_1 = v_1 v_2 \cdots v_{n_1}$ e $\mathcal{L}_2 = u_1 u_2 \cdots u_{n_2}$ linhas poligonais que não se intersectam e sejam $\mathcal{L}'_1 = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m_1)}$ e $\mathcal{L}'_2 = u_{\beta(1)} u_{\beta(2)} \cdots u_{\beta(m_2)}$ linhas simplificadas de $\mathcal{L}_1$ e $\mathcal{L}_2$, respectivamente. As linhas $\mathcal{L}'_1$ e $\mathcal{L}'_2$ estarão a pelo menos uma distância δ uma da outra, se e somente se (1) os vértices $v_{\alpha(i)}$ estiverem a pelo menos uma distância δ de $\overline{u_{\beta(j)} u_{\beta(j+1)}}$ para todo $1 \le i \le m_1$ e $1 \le j < m_2$ e (2) os vértices $u_{\beta(i)}$ estiverem a pelo menos uma distância δ de $\overline{v_{\alpha(j)} v_{\alpha(j+1)}}$ para todo $1 \le i \le m_2$ e $1 \le j < m_1$.*

Demonstração. A menor distância entre os segmentos $\overline{v_{\alpha(i)} v_{\alpha(i+1)}}$ de $\mathcal{L}'_1$ e $\overline{u_{\beta(j)} u_{\beta(j+1)}}$ de $\mathcal{L}'_2$ é a menor dentre as distâncias de $v_{\alpha(i)}$ a $\overline{u_{\beta(j)} u_{\beta(j+1)}}$, de $v_{\alpha(i+1)}$ a $\overline{u_{\beta(j)} u_{\beta(j+1)}}$, de $u_{\beta(j)}$ a $\overline{v_{\alpha(i)} v_{\alpha(i+1)}}$ e de $u_{\beta(j+1)}$ a $\overline{v_{\alpha(i)} v_{\alpha(i+1)}}$. De acordo com as condições (1) e (2), estas distâncias são todas maiores ou iguais a δ. Assim sendo, a distância entre $\overline{v_{\alpha(i)} v_{\alpha(i+1)}}$ e

$\overline{u_{\beta(j)}u_{\beta(j+1)}}$ é necessariamente maior ou igual a δ. Como isto vale para todo $1 \leq i < m_1$ e $1 \leq j < m_2$, os segmentos de $\mathcal{L}'_1$ estão a pelo menos uma distância δ dos segmentos de $\mathcal{L}'_2$. Portanto, $\mathcal{L}'_1$ e $\mathcal{L}'_2$ estão a pelo menos uma distância δ uma da outra. $\square$

3.4.3 Casos Particulares

Resta ainda analisar dois casos particulares. No primeiro caso, as linhas $\mathcal{L}_1$ e $\mathcal{L}_2$ conectam-se pelos extremos, como ilustra a Figura 3.24(a). Neste exemplo, fica claro que não é preciso verificar os espaçamentos entre o vértice $v_{\alpha(5)}$ de $\mathcal{L}'_1$ e o segmento $\overline{u_{\beta(4)}u_{\beta(5)}}$ de $\mathcal{L}'_2$ e entre o vértice $u_{\beta(5)}$ de $\mathcal{L}'_2$ e o segmento $\overline{v_{\alpha(4)}v_{\alpha(5)}}$ de $\mathcal{L}'_1$, já que $v_{\alpha(5)}$ e $u_{\beta(5)}$ coincidem. No segundo caso, deseja-se saber se partes de uma linha $\mathcal{L}'_1$ aparecem próximas de outras partes dela mesma, como ilustra a Figura 3.24(b). O caso de uma única linha pode ser, mais um vez, tratado como o caso de duas linhas conectadas, ilustrado pela Figura 3.24(a). Este caso reduz-se a verificar o espaçamento de cada vértice $v_{\alpha(i)}$ de $\mathcal{L}_1$ com cada segmento $\overline{v_{\alpha(j)}v_{\alpha(j+1)}}$ para todo $1 \leq i \leq m_1$, $1 \leq j < i-1$ e $i < j < m_1$.

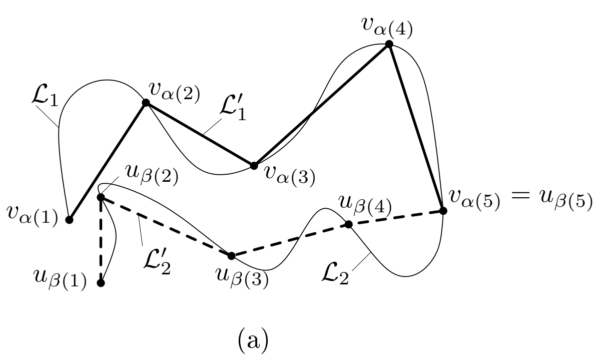


Figura 3.24: Casos particulares de espaçamento entre linhas: (a) as linhas $\mathcal{L}_1$ e $\mathcal{L}_2$ conectam-se pelos extremos; e (b) a linha simplificada $\mathcal{L}'_1$ está próxima de si mesma.

3.5 Considerações Finais

Neste capítulo, demonstrou-se que, se todas as condições descritas forem respeitadas pelos pontos e pelos vértices das linhas simplificadas, o processo simplificador será consistente. Um ponto importante disto é que tais condições são avaliadas nas sublinhas das linhas e não nas linhas como um todo, como feito na Definição 5. Assim sendo, é possível analisar a consistência do mapa simplificado apenas verificando se as condições foram respeitadas por cada sublinha e pelo seu segmento correspondente, não sendo necessário ter conhecimento de que sublinha ou segmento faz parte de que linha. Esta abordagem foi propositadamente escolhida, para permitir o uso das condições descritas como condições de parada nas URCs, já que elas monitoram apenas as sublinhas.

Outro ponto importante é que, para todas as metas de consistência, ficou bem claro que apenas os vértices das linhas simplificadas precisam ser avaliados e não os vértices das linhas originais, como o fizeram De Berg *et a.* e Saalfeld (veja Seção 2.3), confirmando a hipótese deste trabalho sob o ponto de vista teórico. Sob o ponto de vista prático, o algoritmo proposto, através da simplificação global, pode limitar a análise da consistência nas URCs apenas sobre os vértices das linhas simplificadas. Portanto, resta somente mostrar os benefícios desta abordagem. Em outras palavras, ainda é preciso avaliar se os mapas resultantes do algoritmo realmente contêm menos vértices adicionais e se o desempenho do algoritmo é melhorado por causa disto.

Para isto, ainda é preciso responder algumas questões de implementação. A primeira delas refere-se ao método de atualização do estado dos pontos externos de uma URC, quando esta é quebrada. O teste de inversão do triângulo, utilizado no algoritmo de Saalfeld, não mantém corretamente o estado dos pontos externos e não poderá ser usado. As outras questões referem-se à conservação de espaçamentos. É possível que, se a distância mínima exigida for relativamente grande em relação aos espaçamentos do mapa original, alguns dos seus objetos possam ser considerados próximos antes da simplificação. Além disso, pontos externos podem aparecer próximos ao segmento monitorado por uma URC, mesmo que não estejam na sua vizinhança. Estas questões serão respondidas no Capítulo 4, que apresenta detalhadamente o algoritmo proposto.

Capítulo 4

Algoritmo

A idéia básica do algoritmo proposto é eliminar localmente as inconsistências do mapa simplificado, inserindo vértices adicionais nas linhas por meio de URCs (Unidades de Recuperação de Consistência). A consistência é garantida em cada URC com base nas condições apresentadas no Capítulo 3, que abrangem tanto a preservação da topologia, quanto a conservação de espaçamentos. O algoritmo adéqua-se a uma diversidade de ASIs que podem ser utilizados em diferentes tipos de fenômenos. Ele apresenta um bom desempenho, fruto de considerar apenas os vértices das linhas simplificadas em uma abordagem global e de utilizar métodos alternativos na coleta de pontos externos. O mapa de entrada pode conter pontos, linhas e áreas, desde que não contenha redundâncias. O mapa resultante é invariante em relação à ordem de processamento das URCs. O algoritmo também é capaz de produzir mapas independentes de escala.

Este capítulo está organizado da seguinte maneira. A Seção 4.1 apresenta brevemente a estrutura geral e as três etapas do algoritmo. A Seção 4.2, por sua vez, discute os pré-requisitos que os ASIs devem atender para serem utilizados no algoritmo. Em seguida, as Seções 4.3, 4.4 e 4.5 descrevem, em detalhes, as três etapas do algoritmo. Por se tratar de um ponto que merece atenção especial, a conservação de espaçamentos é descrita separadamente na Seção 4.6. Mais adiante, a Seção 4.7 demonstra que o resultado do algoritmo é invariante em relação à ordem de processamento das URCs. Já a Seção 4.8 explana como o algoritmo é adaptado à produção de mapas independentes de escala. Por fim, a Seção 4.9 apresenta as considerações finais deste capítulo.

4.1 Estrutura Geral do Algoritmo

Em uma abordagem de simplificação global, o algoritmo recebe como entradas uma tolerância ϵ e um mapa, formado pelos conjuntos de linhas $\mathfrak{L}$[1] e de pontos P, e retorna como saída um mapa simplificado e consistente ao original, formado pelos conjuntos de linhas simplificadas $\mathfrak{L}''$ e de pontos P, como ilustra a Figura 4.1. O algoritmo é dividido em três etapas. Na primeira etapa, ele simplifica as linhas do conjunto $\mathfrak{L}$ à tolerância ϵ com uso de um ou mais ASIs, produzindo um conjunto intermediário de linhas $\mathfrak{L}'$. Na segunda etapa, ele inicializa, a partir de $\mathfrak{L}'$ e P, o conjunto de URCs $\mathfrak{U}$ que será utilizado no tratamento da consistência. Na terceira e última etapa, ele corrige as inconsistências introduzidas em $\mathfrak{L}'$, gerando o conjunto de linhas de saída $\mathfrak{L}''$.

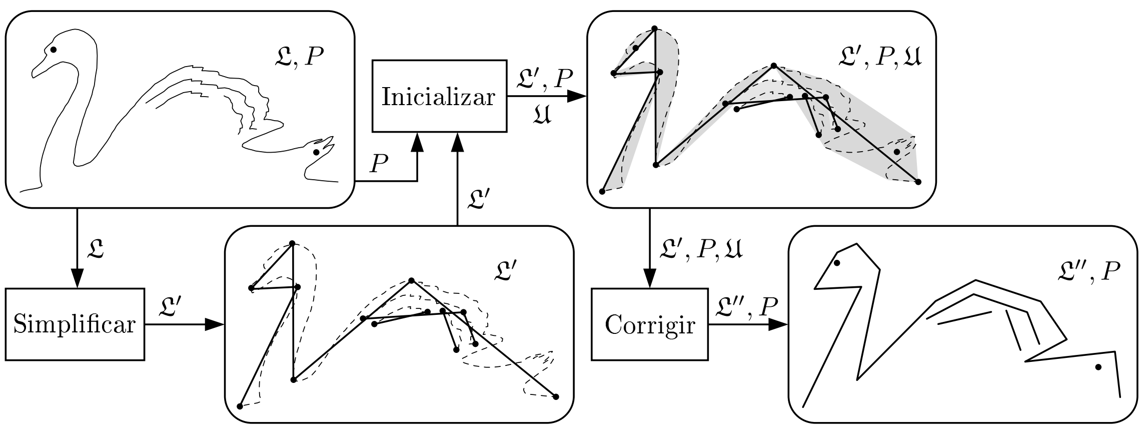


Figura 4.1: Fluxo de dados na aplicação do algoritmo proposto.

A etapa de simplificação pode aplicar diferentes ASIs em linhas distintas do mapa, desde que estes ASIs atendam certos pré-requisitos. A etapa de inicialização, além de criar as URCs, determina o grafo de interferência das URCs (definido na Seção 4.4.1) e faz uma coleta inicial de pontos externos. O grafo de interferência das URCs é necessário para aumentar a eficiência no processo de coleta. A etapa de correção é realizada de maneira iterativa. A cada passo, o algoritmo corrige uma URC, inserindo um vértice adicional no trecho da linha simplificada que ela monitora, e coleta e classifica este vértice adicional nas suas URCs interferentes. Este processo continua até que as condições de consistência sejam respeitadas e a tolerância ϵ seja alcançada em todas as URCs.

[1] O conjunto $\mathfrak{L}$ inclui tanto as linhas propriamente ditas, quanto os contornos das áreas.

4.2 Pré-requisitos dos ASIs

Para que um ASI possa ser integrado no algoritmo proposto, ela precisa atender dois pré-requisitos. O primeiro pré-requisito é que ele seja extrator, isto é, que as linhas por ele simplificadas contenham apenas vértices das suas linhas originais. Como as condições de consistência do Capítulo 3 não prevêem o aparecimento de novos vértices, nem o deslocamento dos existentes, ASIs irrestritos poderiam introduzir inconsistências que, muitas vezes, não seriam corrigidas apenas pela inserção de vértices adicionais durante a correção. Além disso, com um ASI extrator, garante-se que a inserção de vértices adicionais sempre gera um mapa simplificado consistente. Isto porque, no pior dos casos, todos os vértices são inseridos e a geometria do mapa original é recuperada.

O segundo pré-requisito é que o ASI insira vértices nas linhas simplificadas em uma ordem bem definida e independente da tolerância ϵ e que, uma vez inserido em uma linha, um vértice não possa mais ser dela removido. Este pré-requisito permite que o algoritmo proposto determine qual o próximo vértice adicional a ser inserido em uma URC durante a etapa de correção. Se o ASI for do tipo que insere vários vértices em um único passo, é preciso estabelecer um critério de desempate na escolha do próximo vértice. Este pré-requisito também é a base para que o resultado do algoritmo seja invariante em relação à ordem de processamento das URCs. Adicionalmente, ele é fundamental à adaptação do algoritmo para a produção de mapas independentes de escala.

Embora somente se venha comentando em inserção de vértices e não na remoção, não há nenhum pré-requisito quanto ao fato de o ASI ser incremental. ASIs decrementais, como o de Visvalingam [46], também funcionam corretamente com o algoritmo proposto. Eles devem, entretanto, ser adaptados a um modo especial de funcionamento. Durante a etapa de simplificação, um ASI decremental deve, além de remover os vértices, armazenar a ordem de remoção utilizada. Isto permite que, durante a etapa de correção, o algoritmo proposto possa, invertendo esta ordem, determinar o próximo vértice adicional a ser inserido. Se isto não for feito, o algoritmo não terá como saber qual é o próximo vértice, pois, muitas vezes, o método de escolha do vértice pelo ASI não é reversível.

Sabe-se que o algoritmo proposto recebe uma única tolerância ϵ como entrada para os vários tipos de ASIs. É necessário, então, estimar adequadamente esta tolerância para a métrica de escolha de vértices de cada ASI utilizado. Por outro lado, não há nenhum pré-requisito quanto ao fato de o ASI utilizar uma tolerância geométrica como critério de parada. ASIs que utilizam o número de vértices como critério de parada também funcionam corretamente com o algoritmo proposto. Eles devem, no entanto, estimar o no número de vértices adequado ao seu critério a partir da tolerância ϵ. Estes ASIs não precisam se preocupar com a tolerância ϵ durante a etapa de correção, porque o número mínimo de vértices nas linhas é alcançado durante a etapa de simplificação.

4.3 Etapa de Simplificação

Na etapa de simplificação, o algoritmo proposto aplica os ASIs sobre as linhas do conjunto $\mathfrak{L} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_N\}$ à tolerância ϵ e obtém um conjunto de linhas simplificadas $\mathfrak{L}' = \{\mathcal{L}'_1, \mathcal{L}'_2, \ldots, \mathcal{L}'_N\}$, como ilustra a Figura 4.2. Cada linha pode ser processada por um ASI específico, como indica a notação $\mathrm{ASI}_i(\cdot)$ utilizada na figura. Para que o algoritmo saiba que ASI deve ser aplicado sobre que linha, a cada linha de entrada é associado o identificador do ASI com o qual ela deverá ser processada. A grande vantagem desta abordagem é que fenômenos de diferentes tipos, como, por exemplo, rios, rodovias e curvas de nível, podem ser simplificadas com um ASI apropriado para o seu tipo.

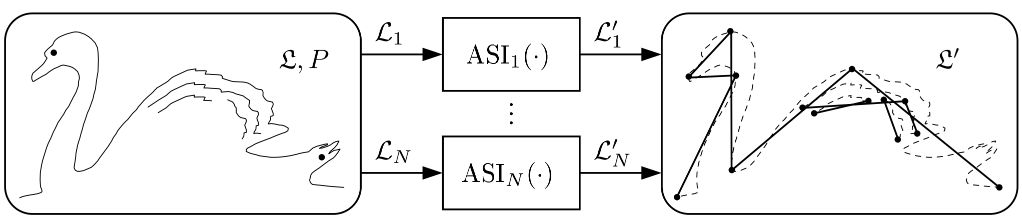
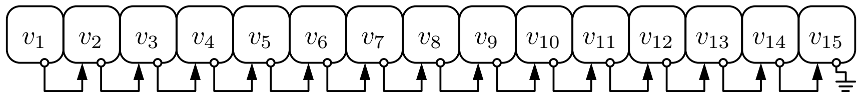


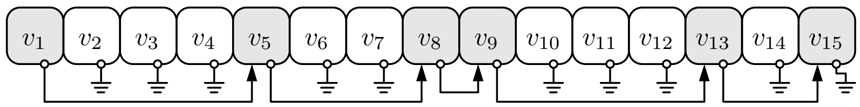
Figura 4.2: Fluxo de dados na etapa de simplificação.

O algoritmo armazena as linhas sob a forma de vetores de vértices. Quando a etapa de simplificação termina, os vértices descartados pelos ASIs não são removidos dos vetores. Eles permanecem mascarados, para serem possivelmente utilizados na etapa de correção. Para identificar quais vértices foram descartados pela simplificação, o algoritmo associa a cada vértice o índice do próximo vértice não descartado na linha. Observe o exemplo da Figura 4.3. Antes da simplificação, cada vértice da linha $\mathcal{L} = v_1 v_2 \cdots v_{15}$

armazenada aponta para o seu subseqüente (Figura 4.3(a)). Após a simplificação, os vértices não descartados passam a apontar para o próximo vértice não descartado de $\mathcal{L}$, isto é, o seu subseqüente em $\mathcal{L}' = v_1 v_5 v_8 v_9 v_{13} v_{15}$ (Figura 4.3(b)).



(a) Estrutura de dados da linha original $\mathcal{L} = v_1 v_2 \cdots v_{15}$



(b) Estrutura de dados da linha simplificada $\mathcal{L}' = v_1 v_5 v_8 v_9 v_{13} v_{15}$

Figura 4.3: Estrutura de dados de uma linha antes e após a etapa de simplificação.

4.4 Etapa de Inicialização

A etapa de inicialização é composta por dois passos, como ilustra a Figura 4.4. No primeiro passo, o algoritmo cria o conjunto de URCs $\mathfrak{U}$ para monitorar a consistência de cada sublinha das linhas de $\mathfrak{L}$ e do seu segmento correspondente nas linhas de $\mathfrak{L}'$. Ainda neste passo, o algoritmo determina o grafo de interferência das URCs. No segundo passo, o algoritmo coleta os pontos externos em cada uma das URCs. Os pontos externos de uma URC correspondem aos pontos de P e aos vértices das linhas de $\mathfrak{L}'$ que estão na vizinhança da sua sublinha. Após a coleta, os pontos externos são classificados de acordo com a sua posição com respeito ao polígono associado à sublinha da URC.

4.4.1 Criação das URCs

Sabe-se que a sublinha $\mathcal{L}_{[\alpha(i),\alpha(i+1)]}$ pode ser vista como o trecho da linha original $\mathcal{L}$ que foi substituído pelo segmento $\overline{v_{\alpha(i)} v_{\alpha(i+1)}}$ na linha simplificada $\mathcal{L}'$. Assim sendo, para cada linha $\mathcal{L}' = v_{\alpha(1)} v_{\alpha(2)} \cdots v_{\alpha(m)}$ de $\mathfrak{L}'$, resultante da etapa de simplificação, o algoritmo cria as URCs $\mathcal{U}_{\mathcal{L}_{[\alpha(i),\alpha(i+1)]}}$ para todo $1 \leq i < m$. Por exemplo, no caso da Figura 4.3(b), o algoritmo cria as URCs $\mathcal{U}_{\mathcal{L}_{[1,5]}}$, $\mathcal{U}_{\mathcal{L}_{[5,8]}}$, $\mathcal{U}_{\mathcal{L}_{[9,13]}}$ e $\mathcal{U}_{\mathcal{L}_{[13,15]}}$. Como não há vértices entre v_8

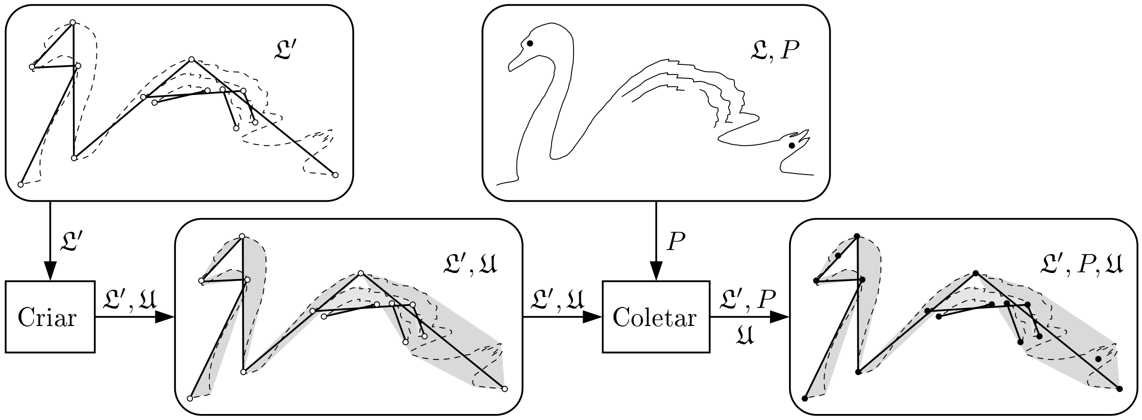


Figura 4.4: Fluxo de dados na etapa de inicialização.

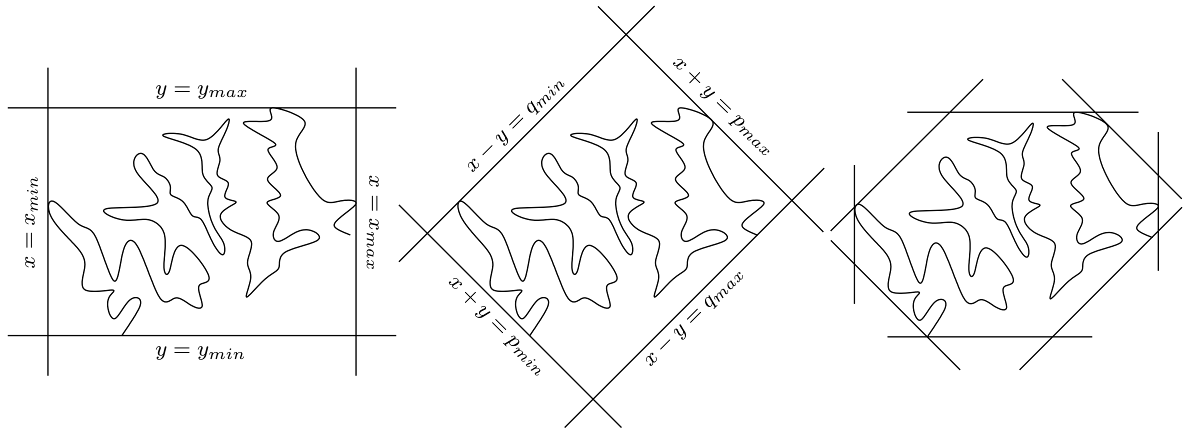
e v_9, não é preciso criar a URC $\mathcal{U}_{\mathcal{L}_{[8,9]}}$. Para cada URC, o algoritmo determina o fecho convexo da sua sublinha, assim como outras regiões de vizinhança explicadas a seguir. A estrutura de dados de uma URC armazena um apontador para a linha original, os índices inicial e final da sua sublinha e as geometrias das regiões de vizinhança.

Ainda no passo de criação, o algoritmo determina o grafo de interferência das URCs, cujos nós são as URCs de $\mathfrak{U}$ e cujas arestas ligam as URCs interferentes. Duas URCs são ditas interferentes, se as suas regiões de vizinhança se sobrepõem. Como o cálculo da sobreposição dos fechos convexos de duas URCs envolve um grande número de operações, o algoritmo utiliza regiões de vizinhança de geometria mais simples. Para implementar o grafo de interferência, cada URC armazena, além dos dados já descritos, uma lista de apontadores às suas URCs interferentes. O grafo de interferência é muito importante por questões de eficiência. Isto porque ele permite que um vértice de uma URC seja verificado quanto à coleta apenas nas URCs que ela interfere.

Para verificar se duas URCs são interferentes, o algoritmo utiliza o Retângulo Envolvente Alinhado Mínimo (REAM)[2] e o Retângulo Envolvente Inclinado Mínimo (REIM), que têm geometria bem simples e de fácil determinação. O REAM de uma linha é o retângulo de menor área que envolve os seus vértices e cujos lados são alinhados aos eixos x e y (Figura 4.5(a)). O REIM, por sua vez, é o retângulo de menor área que envolve os seus vértices, mas cujos lados são alinhados às retas $x + y = 0$ e $x - y = 0$

[2]O REAM, sendo destes retângulos o mais empregado, é geralmente referido apenas por Retângulo Envolvente Mínimo (REM). A sigla REAM é aqui utilizada, para enfatizar a diferença entre eles.

(Figura 4.5(b)). A verificação da sobreposição de dois REAMs ou REIMs requer no máximo quatro comparações. A verificação da sobreposição de dois octógonos formados pelos REAMs e REIMs de duas URCs requer no máximo oito comparações.



(a) REAM (b) REIM (c) REAM+REIM

Figura 4.5: Retângulos envolventes alinhado e inclinado mínimos de um linha.

O algoritmo proposto atenta-se primeiramente em determinar as URCs cujas REAMs se sobrepõem. A determinação destas sobreposições é um problema bem conhecido na literatura: o problema da interseção de retângulos. Foi utilizada uma solução ótima para este problema publicada por Edelsbrunner [12] em 1980. Nesta solução, uma reta vertical se movimenta da esquerda para esquerda, varrendo o plano, e, com base em uma árvore de intervalos, verifica quais dos retângulos que ela cruza se intersectam, como ilustra a Figura 4.6. Após constatar que os REAMs de duas URCs se sobrepõem, o algoritmo proposto verifica se os seus REIMs também se sobrepõem. Se sim, estas URCs são consideradas interferentes e uma aresta é inserida entre elas no grafo de interferência.[3]

4.4.2 Coleta de Pontos Externos

A coleta de pontos externos é necessária também por questões de eficiência. Para não ter que avaliar todos os pontos e vértices do mapa, o algoritmo coleta, para cada URC, apenas o pequeno grupo de pontos externos que se encontram na sua vizinhança. O fecho

[3] Se os dados do mapa já estiverem dispostos em estruturas de localização espacial, como *quadtrees* [14] ou *R-trees* [17], é possível utilizar tais estruturas para aumentar o desempenho do algoritmo.

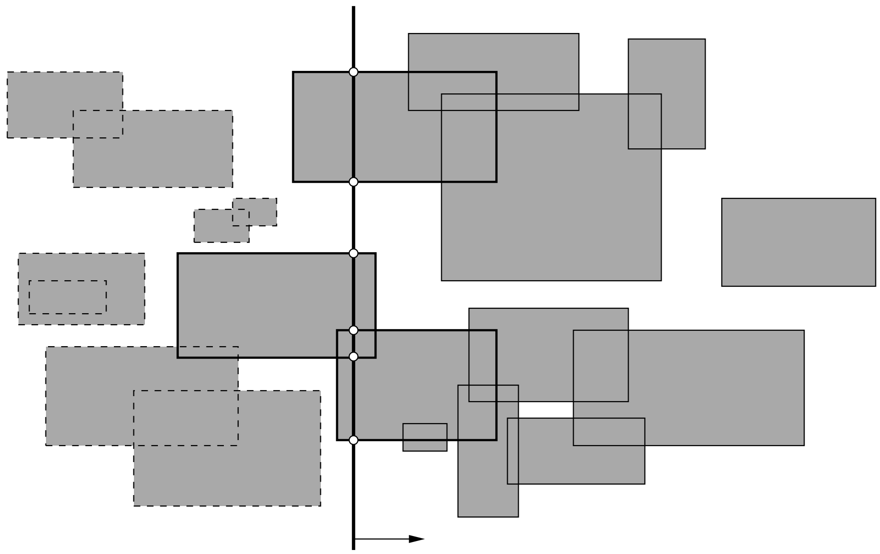
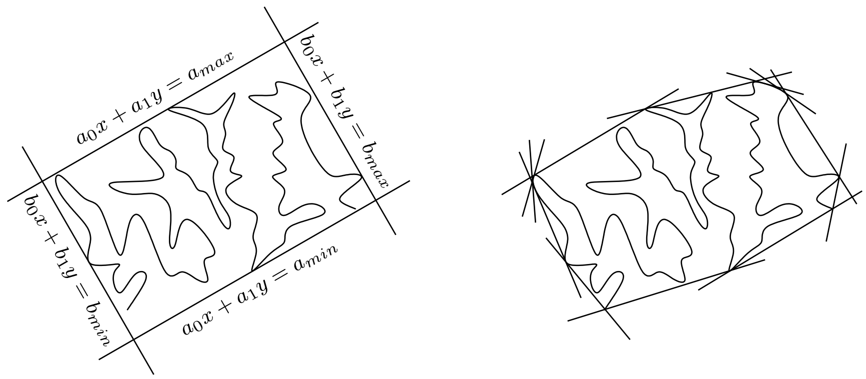


Figura 4.6: Aplicação do algoritmo de Edelsbrunner [12].

convexo, sendo o menor polígono convexo que envolve a sua sublinha, representa a região que contém o menor número de pontos externos. O ganho em eficiência do fecho convexo, no entanto, é comprometido pelo baixo desempenho do processo de determinação do fecho e do teste de pertinência de um ponto. Na prática, uma maior eficiência pode ser alcançada utilizando-se o REAM e o REIM. Mesmo contendo alguns pontos externos a mais, a eficiência é compensada devido à simplicidade das suas geometrias.

Para posteriores comparações de ganho em eficiência, além do fecho convexo, do REAM e do REIM, outra geometria tradicional será utilizada: o Retângulo Envolvente Orientado Mínimo (REOM). O REOM de uma linha é o retângulo de menor área que envolve os seus vértices (Figura 4.7(a)). A vantagem do REOM é que a sua área é, geralmente, bem próxima à do fecho convexo, como se pode observar na Figura 4.7(b). Além disso, o teste de pertinência de um ponto a um REOM envolve quatro comparações, como para os outros retângulos. A desvantagem é que algoritmos que determinam o REOM de uma sublinha, precisam antes computar o seu fecho convexo. O tempo de determinação do REOM é, portanto, ligeiramente superior ao do fecho convexo.

O algoritmo utiliza as regiões de vizinhança e o grafo de interferência, para coletar os pontos externos das URCs. Uma vez coletados, os pontos externos de uma URC são classificados em função da sua posição com respeito ao polígono associado à sublinha



(a) REOM (b) Fecho convexo

Figura 4.7: Comparação entre as áreas do fecho convexo e do REOM de uma linha.

que ela monitora. Quando um ponto se encontra no interior ou sobre o polígono, ele é classificado como inconsistente e, quando se encontra no exterior de polígono, como consistente. Para fazer esta verificação, o algoritmo utiliza o algoritmo de Melkman [28], de maneira similar ao algoritmo de Saalfeld. Ele armazena, no entanto, os cruzamentos calculados pelo algoritmo de Melkman em um vetor associado a cada ponto externo, que é utilizado durante a correção, para atualizar a sua classificação (veja Seção 4.5.2).

4.5 Etapa de Correção

Após a etapa de inicialização, o algoritmo inicia um processo iterativo de recuperação da consistência do mapa, como ilustra a Figura 4.8. Neste processo, o algoritmo basicamente verifica se as condições de consistência estão sendo respeitadas em cada uma das URCs de $\mathfrak{U}$ e insere vértices adicionais nas que não respeitam. Para que novas inconsistências sejam identificadas, ele verifica se os vértices inseridos em uma URC devem ser coletados pelas suas interferentes. O algoritmo também verifica se o critério de tolerância ϵ, alcançado na etapa de simplificação, foi perdido com a inserção dos vértices adicionais. Quando não há mais erros, o algoritmo pára com um conjunto de linhas simplificadas $\mathfrak{L}''$ e de pontos P, que formam juntos um mapa consistente ao original.

Para não ter que verificar todas as URCs de $\mathfrak{U}$ a cada iteração, o algoritmo as dispõe em duas listas: a primeira, denotada por $\mathfrak{U}_s$, contém as URCs com erros, ditas

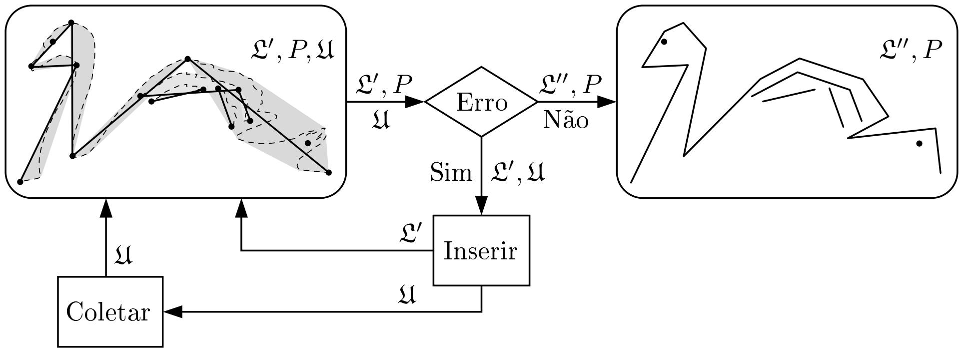


Figura 4.8: Fluxo de dados da etapa de correção.

sujas[4]; e a segunda, denotada por $\mathfrak{U}_l$, contém as URCs sem erros, ditas limpas. A cada iteração, o algoritmo retira uma URC suja de $\mathfrak{U}_s$ e a quebra em duas novas URCs, inserindo um vértice adicional. Com a quebra, o algoritmo precisa atualizar o grafo de interferência das URCs. Além disso, como o vértice adicional é coletado pelas URCs interferentes, é possível que algumas destas que não eram sujas passem a ser. Estas URCs são retiradas de $\mathfrak{U}_l$ e colocadas em $\mathfrak{U}_s$. O algoritmo também analisa os erros nas duas novas URCs criadas e as adiciona nas listas correspondentes. O algoritmo termina, quando o conjunto $\mathfrak{U}_s$ está vazio, isto é, quando não há mais erros no mapa.

4.5.1 Quebra de uma URC

Observe a URC $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ ilustrada pela Figura 4.9(a). Como alguns dos seus pontos externos são inconsistentes, isto é, estão dentro ou sobre $\mathcal{P}_{\mathcal{L}_{[i,j]}}$, o algoritmo mantém $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ na lista $\mathfrak{U}_s$. Em uma dada iteração do laço de correção, o algoritmo retira $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ de $\mathfrak{U}_s$ e insere um vértice v_k $(i < k < j)$ na linha simplificada $\mathcal{L}'$ entre v_i e v_j. Em seguida, ele quebra $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ nas URCs $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ e $\mathcal{U}_{\mathcal{L}_{[k,j]}}$ e determina as suas regiões de vizinhança, conforme mostra a Figura 4.9(b). O algoritmo utiliza estas regiões para coletar os pontos externos de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$, como ilustra a Figura 4.9(c). Perceba que uma parte destes pontos externos é coletada apenas em $\mathcal{U}_{\mathcal{L}_{[i,k]}}$, outra parte, apenas em $\mathcal{U}_{\mathcal{L}_{[k,j]}}$, uma terceira parte, em ambas as URCs e uma última parte não é coletada em nenhuma das duas.

[4]É importante salientar que uma URC é considerada suja ou com erros, quando as condições de consistência não são respeitadas ou quando o critério de tolerância ϵ não é alcançado.

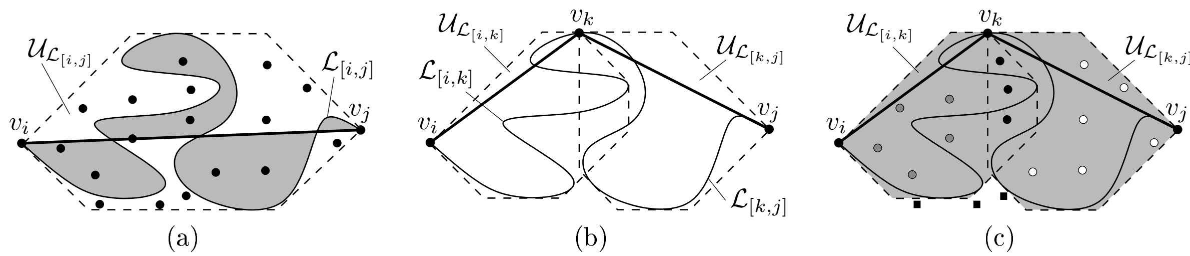


Figura 4.9: Quebra de uma URC: (a) URC original $\mathcal{U}_{\mathcal{L}_{[i,j]}}$; (b) URCs resultantes da quebra $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ e $\mathcal{U}_{\mathcal{L}_{[k,j]}}$; e (c) coleta dos pontos externos de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ em $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ e $\mathcal{U}_{\mathcal{L}_{[k,j]}}$.

É possível que o vértice v_k cause novas inconsistências no mapa. Assim sendo, o algoritmo verifica se v_k deve ser coletado em outras URCs. Graças ao grafo de interferência das URCs, só é preciso checar a coleta de v_k nas URCs interferentes de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$, como ilustra a Figura 4.10(a). Após este passo, o algoritmo substitui $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ por $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ e $\mathcal{U}_{\mathcal{L}_{[k,j]}}$ no grafo de interferência. Esta substituição é realizada checando-se a sobreposição dos REAMs e REIMs de $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ e de $\mathcal{U}_{\mathcal{L}_{[k,j]}}$ com os das URCs interferentes de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$, como mostra a Figura 4.10(b). O algoritmo considera que $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ e $\mathcal{U}_{\mathcal{L}_{[k,j]}}$ sempre são interferentes. De maneira similar à coleta de pontos externos, uma parte das URCs interferentes de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ interfere apenas em $\mathcal{U}_{\mathcal{L}_{[i,k]}}$, outra parte, apenas em $\mathcal{U}_{\mathcal{L}_{[k,j]}}$, uma terceira parte, em ambas as URCs e uma última parte não interfere em nenhuma das duas.

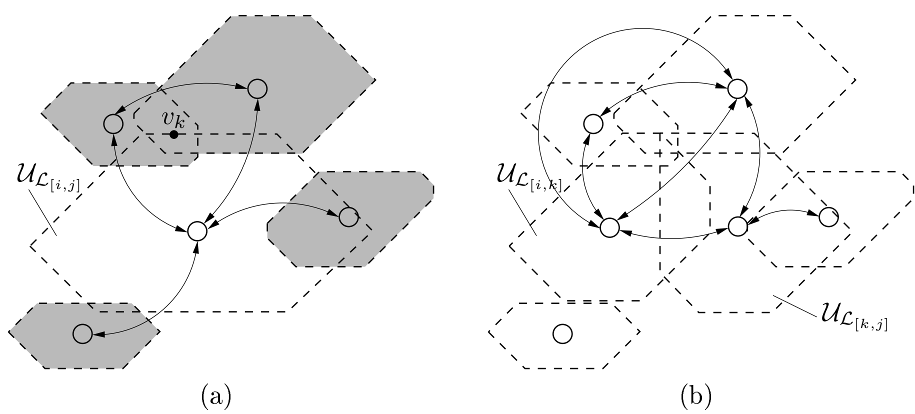


Figura 4.10: (a) Coleta do vértice de quebra v_k da URC $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ nas suas interferentes; e (b) substituição de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ pelas URCs $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ e $\mathcal{U}_{\mathcal{L}_{[k,j]}}$ no grafo de interferência.

4.5.2 Atualização da Classificação dos Pontos Externos

Após coletar, em $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ e $\mathcal{U}_{\mathcal{L}_{[k,j]}}$, alguns dos pontos externos de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$, o algoritmo atualiza a sua classificação. O teste de inversão do triângulo, presente no algoritmo de Saalfeld, não pode ser utilizado, porque não permite que as URCs monitorem corretamente as condições de consistência. Para verificar se tais condições estão sendo respeitadas em $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ e $\mathcal{U}_{\mathcal{L}_{[k,j]}}$, é preciso determinar se os seus pontos externos estão no exterior dos polígonos $\mathcal{P}_{\mathcal{L}_{[i,k]}}$ e $\mathcal{P}_{\mathcal{L}_{[k,j]}}$, respectivamente. Para isto, o algoritmo emprega uma estratégia de atualização alternativa. Nesta estratégia, os cruzamentos de semi-retas partindo dos pontos externos com a sublinha $\mathcal{L}_{[i,j]}$, que foram computados em um passo anterior, são usados para determinar os cruzamentos com as sublinhas $\mathcal{L}_{[i,k]}$ e $\mathcal{L}_{[k,j]}$.

Observe o exemplo da Figura 4.11. Considere que o algoritmo de Melkman tenha sido usado para determinar os cruzamentos de uma semi-reta partindo do ponto p com a sublinha $\mathcal{L}_{[5,18]}$ (Figura 4.11(a)). Considere também que os índices dos primeiros vértices dos segmentos cruzados tenham sido armazenados em um vetor, denominado vetor de cruzamentos (Figura 4.11(d)). As variáveis c_i e c_f apontam, respectivamente, para o primeiro elemento do vetor e para o elemento logo depois do último. O número de cruzamentos é obtido pela subtração $(c_f - c_i)$. Para determinar o número de cruzamentos desta mesma semi-reta com $\mathcal{L}_{[5,12]}$, é suficiente modificar c_f (Figuras 4.11(b) e 4.11(e)). Já no caso de $\mathcal{L}_{[12,18]}$, deve-se modificar c_i (Figuras 4.11(c) e 4.11(f)).

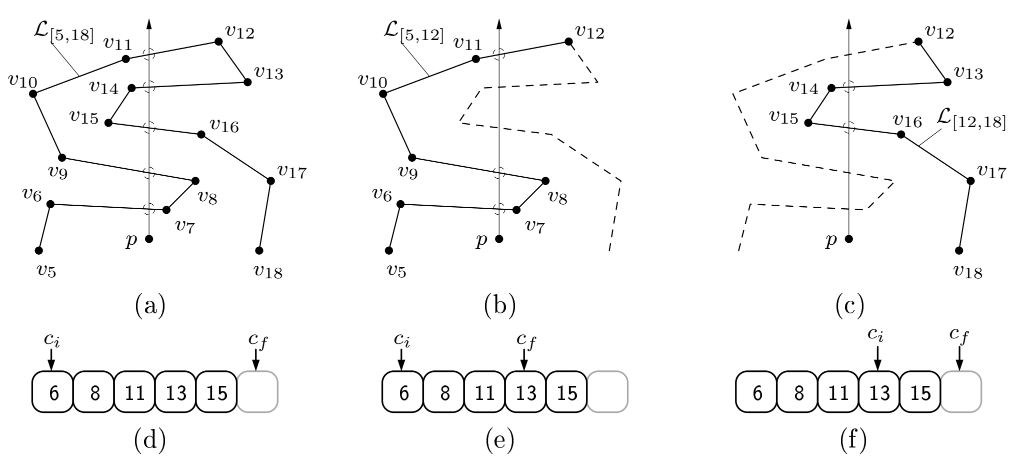


Figura 4.11: (a, b, c) Cruzamentos de uma semi-reta partindo do ponto p com as sublinhas $\mathcal{L}_{[5,18]}$, $\mathcal{L}_{[5,12]}$ e $\mathcal{L}_{[12,18]}$ e (d, e, f) os seus respectivos vetores de cruzamentos.

Na realidade, as variáveis c_f em relação a $\mathcal{L}_{[5,12]}$ e c_i em relação a $\mathcal{L}_{[12,18]}$ apontam ambas para o mesmo elemento do vetor de cruzamentos. Este elemento é determinado com base no vértice de quebra v_k, que, no caso da Figura 4.11, é o vértice v_{12}. O algoritmo busca o primeiro elemento do vetor de cruzamentos com valor maior do que k. Como, no caso da Figura 4.11, $k = 12$, este elemento é o de valor 13, que representa o cruzamento da semi-reta com o segmento $\overline{v_{13}v_{14}}$ de $\mathcal{L}_{[5,18]}$. Visto que os elementos do vetor de cruzamentos são sempre organizados em ordem crescente, o algoritmo pode efetuar uma busca binária por este elemento. Assim sendo, de um modo geral, a modificação das variáveis c_f em relação $\mathcal{L}_{[i,k]}$ e c_i em relação $\mathcal{L}_{[k,j]}$ tem complexidade logarítmica.

Em $\mathcal{U}_{\mathcal{L}_{[5,18]}}$, o ponto p tem a si associado o seu vetor de cruzamentos com $\mathcal{L}_{[5,18]}$ e as suas variáveis c_i e c_f. Para classificar p, o algoritmo soma o número de cruzamentos com $\mathcal{L}_{[5,18]}$ ao possível cruzamento com o segmento $\overline{v_5v_{18}}$ (Figura 4.12(a)). Durante a quebra de $\mathcal{U}_{\mathcal{L}_{[5,18]}}$, o algoritmo encontra o elemento 13 no vetor de cruzamentos. Se p for coletado em $\mathcal{U}_{\mathcal{L}_{[5,12]}}$, o algoritmo faz uma cópia de p com o valor de c_f corrigido. De maneira análoga, se p for coletado em $\mathcal{U}_{\mathcal{L}_{[12,18]}}$, ele faz outra cópia de p com valor de c_i corrigido. O vetor de cruzamentos não é copiado, mas uma referência ao vetor original é mantida nas duas cópias. Para atualizar a classificação de p em $\mathcal{U}_{\mathcal{L}_{[5,12]}}$, o algoritmo soma o número de cruzamentos de $\mathcal{L}_{[5,12]}$ com o possível cruzamento com o segmento $\overline{v_5v_{12}}$ (Figura 4.12(b)). O análogo é feito para $\mathcal{U}_{\mathcal{L}_{[12,18]}}$ (Figura 4.12(c)).

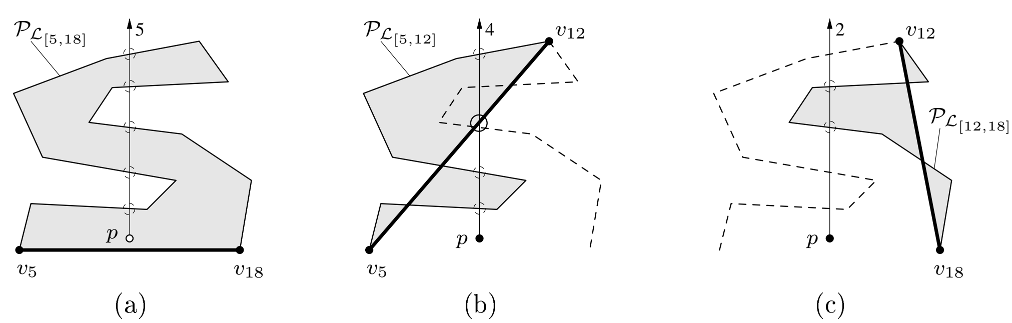
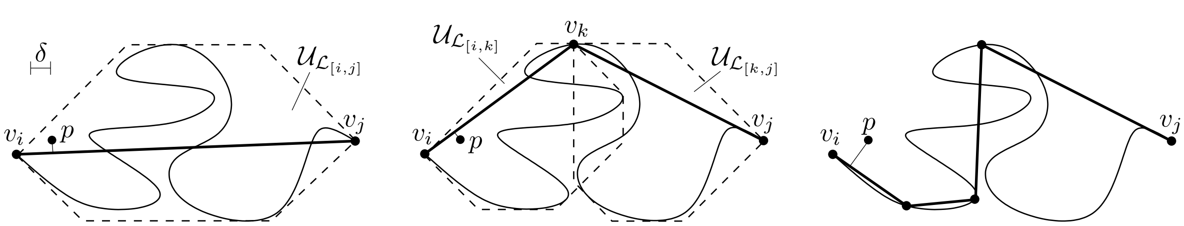


Figura 4.12: (a, b, c) Cruzamentos de uma semi-reta partindo do ponto p com os polígonos $\mathcal{P}_{\mathcal{L}_{[5,18]}}$, $\mathcal{P}_{\mathcal{L}_{[5,12]}}$ e $\mathcal{P}_{\mathcal{L}_{[12,18]}}$, correspondentes às sublinhas da Figura 4.11.

O vetor de cruzamentos de p é criado no momento em que os seus cruzamentos são determinados pelo algoritmo de Melkman, isto é, quando ele é coletado em $\mathcal{U}_{\mathcal{L}_{[5,18]}}$. A partir daí, todas as URCs originadas a partir de $\mathcal{U}_{\mathcal{L}_{[5,18]}}$ criam apenas uma referência ao vetor de cruzamentos original, que é associada à sua cópia de p. Já que as cópias de p guardam apenas referências, não há sobrecarga com a cópia. A complexidade desta estratégia é, portanto, determinada pela modificação das variáveis c_i e c_f, que tem complexidade logarítmica. Na prática, o número de cruzamentos é desprezível em relação ao número total de segmentos da sublinha. Assim sendo, o desempenho desta estratégia é praticamente equivalente ao do teste de inversão do triângulo.

4.6 Adição da Conservação de Espaçamentos

Tudo que se apresentou até então está relacionado apenas à preservação da topologia no mapa simplificado. O algoritmo também é capaz de conservar espaçamentos entre os objetos. Tal qual a preservação da topologia, a conservação de espaçamentos é realizada dentro das URCs. Considere o caso da URC $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ e do seu ponto externo p, ilustrado pela Figura 4.13(a). O algoritmo verifica se p se encontra a uma distância maior ou igual a δ do segmento $\overline{v_iv_j}$. Como este não é o caso, ele insere o vértice v_k e quebra $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ nas URCs $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ e $\mathcal{U}_{\mathcal{L}_{[k,j]}}$, como ilustra a Figura 4.13(b). Como p ainda está próximo ao segmento $\overline{v_iv_k}$, o processo de quebra continua até que, em todas as URCs criadas, p se mantenha à distância δ dos segmentos, como mostra a Figura 4.13(c).



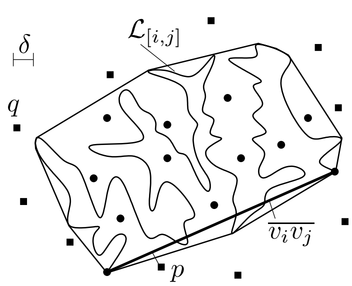
(a) Problema em $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ (b) Problema em $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ (c) Distância δ alcançada

Figura 4.13: Conservação da distância mínima δ entre o ponto p e a linha simplificada $\mathcal{L}'$ no trecho entre os vértices v_i e v_j, monitorado pela URC $\mathcal{U}_{\mathcal{L}_{[i,j]}}$.

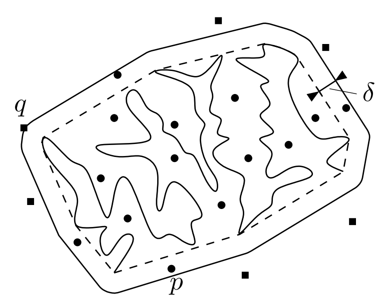
Dois pontos importantes ainda precisam ser analisados em detalhes. O primeiro diz respeito à região de vizinhança de uma URC. A vizinhança de uma URC deve ser pelo menos ligeiramente maior do que o fecho convexo da sua sublinha, já que a simplificação pode tornar próximos, até mesmo, objetos fora do fecho. Só assim, o algoritmo poderá coletar os pontos externos que estão ao redor do fecho. O segundo ponto diz respeito à convergência do algoritmo. Na prática, sabe-se que objetos podem estar muito próximos já no mapa original. Nestes casos, o algoritmo poderia divergir, isto é, ele inseriria um grande número de vértices, mas não conseguiria espaçar os objetos adequadamente. Uma solução é empregada para garantir que algoritmo não insira vértices sem necessidade.

4.6.1 Dilatação das Regiões de Vizinhança

Sabe-se que, para preservar a consistência topológica em uma dada URC $\mathcal{U}_{\mathcal{L}_{[i,j]}}$, é suficiente tratar os pontos externos que se encontram no fecho convexo da sublinha $\mathcal{L}_{[i,j]}$, como ilustra a Figura 4.14(a). Para conservar espaçamentos, no entanto, o fecho convexo não é o bastante. Perceba, por exemplo, que o ponto p, mesmo estando fora do fecho convexo, encontra-se a uma distância menor do que δ do segmento $\overline{v_i v_j}$. Na realidade, os pontos externos que podem aparecer a uma distância menor do que δ de $\overline{v_i v_j}$ incluem não somente os que estão no fecho convexo de $\mathcal{L}_{[i,j]}$, mas também os que estão a uma distância menor do que δ do fecho. A região de vizinhança de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$, então, passa a ser o fecho convexo de $\mathcal{L}_{[i,j]}$ dilatado de uma distância δ, como ilustra a Figura 4.14(b).



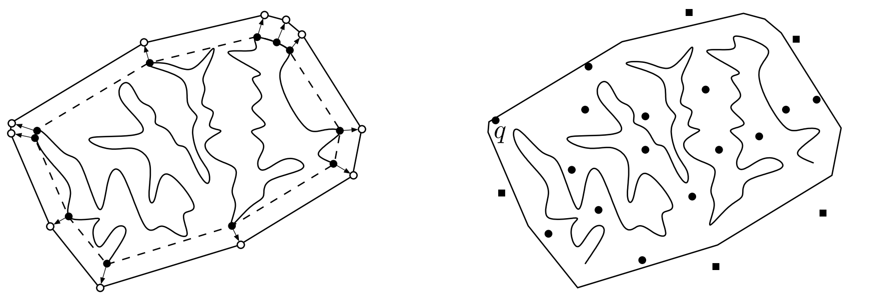
(a) Fecho convexo original



(b) Fecho convexo dilatado

Figura 4.14: Dilatação do fecho convexo da sublinha $\mathcal{L}_{[i,j]}$.

Para coletar um ponto externo na região da Figura 4.14(b), o algoritmo verifica se o ponto está dentro do fecho original. Se não estiver, ele verifica se o ponto está a uma distância menor do que δ de alguma aresta do fecho. Devido ao cálculo das distâncias, este método é ineficiente. Para aumentar a eficiência, o algoritmo desloca os vértices do fecho, gerando a região ilustrada pela Figura 4.15(a). Cada vértice é deslocado com base no ângulo entre as suas duas arestas. Como esta região já inclui o espaço ao redor do fecho original, não é preciso calcular as distâncias. Em contrapartida, os cantos desta região não são arredondados como os da região da Figura 4.14(b). Em virtude disto, alguns pontos, como o ponto q, são coletados sem necessidade, como ilustra a Figura 4.15(b). A quantidade de pontos a mais, no entanto, é desprezível na prática.



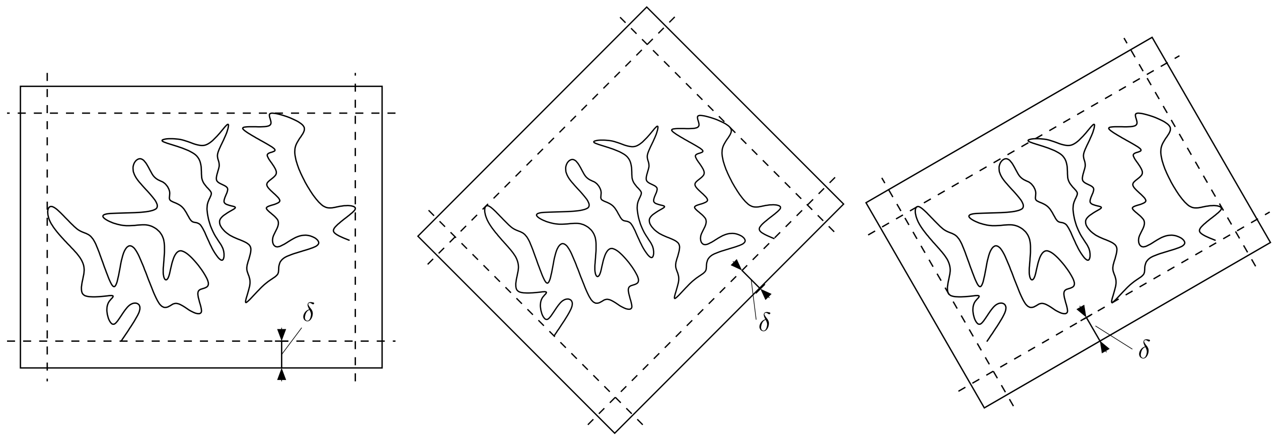
(a) Deslocamento dos vértices (b) Coleta desnecessária do ponto q

Figura 4.15: Dilatação do fecho convexo pelo deslocamento dos seus vértices.

A dilatação também é feita nos retângulos envolventes, conforme ilustra a Figura 4.16. Para tanto, o algoritmo precisa apenas modificar as constantes mínimas e máximas que governam as equações dos retângulos. A dilatação do REAM é realizada de maneira direta, somando-se δ a x_{max} e a y_{max} e subtraindo-se δ de x_{min} e de y_{min}. A dilatação do REIM é feita de modo similar, porém as somas e as subtrações devem ser feitas com o valor δ multiplicado pelo fator $\sqrt{2}$, devido à inclinação de 45^o do REIM. No caso do REOM, os fatores de multiplicação das constantes são determinados de acordo com a sua orientação, de modo semelhante ao caso do fecho convexo dilatado com cantos. A Tabela 4.1 exibe os cálculos das modificações nas constantes.

REAM	REIM	REOM
$x_{min} \leftarrow x_{min} - \delta$	$p_{min} \leftarrow p_{min} - \delta\sqrt{2}$	$a_{min} \leftarrow a_{min} - \delta\,\lvert\frac{a_0 a_1}{\sqrt{a_0^2+a_1^2}}\rvert$
$x_{max} \leftarrow x_{max} + \delta$	$p_{max} \leftarrow p_{max} + \delta\sqrt{2}$	$a_{max} \leftarrow a_{max} + \delta\,\lvert\frac{a_0 a_1}{\sqrt{a_0^2+a_1^2}}\rvert$
$y_{min} \leftarrow y_{min} - \delta$	$q_{min} \leftarrow q_{min} - \delta\sqrt{2}$	$b_{min} \leftarrow b_{min} - \delta\,\lvert\frac{b_0 b_1}{\sqrt{b_0^2+b_1^2}}\rvert$
$y_{max} \leftarrow y_{max} + \delta$	$q_{max} \leftarrow q_{max} + \delta\sqrt{2}$	$b_{max} \leftarrow b_{max} + \delta\,\lvert\frac{b_0 b_1}{\sqrt{b_0^2+b_1^2}}\rvert$

Tabela 4.1: Modificação das constantes mínimas e máximas dos retângulos envolventes.

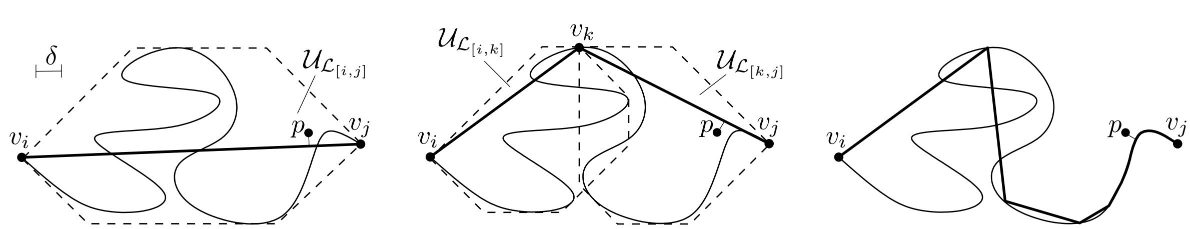


(a) REAM dilatado (b) REIM dilatado (c) REOM dilatado

Figura 4.16: Dilatação dos retângulos envolventes.

4.6.2 Convergência do Algoritmo

Para validar as condições de conservação de espaçamentos, assumiu-se que os objetos do mapa original se encontravam a uma distância maior ou igual a δ uns dos outros (veja Seção 3.4). Na prática, sabe-se que isto nem sempre é verdade. No exemplo da Figura 4.17, o ponto externo p da URC $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ está a uma distância menor do que δ do segmento $\overline{v_i v_j}$. Mesmo quebrando $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ sucessivamente, a proximidade não seria evitada. Uma tentativa de correção com esta levaria o algoritmo a divergir, inserindo vértices em excesso, e poderia comprometer a legibilidade da linha simplificada na escala pretendida. Casos como este não podem ser resolvidos apenas com a inserção de vértices adicionais. Outros métodos, como operadores de deslocamento, devem ser utilizados.



(a) Problema em $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ (b) Problema em $\mathcal{U}_{\mathcal{L}_{[k,j]}}$ (c) Distância δ não alcançada

Figura 4.17: Tentativa fracassada de manter uma distância δ entre o ponto p e a linha simplificada $\mathcal{L}'$ com a inserção de vértices adicionais entre os vértices v_i e v_j.

A melhor estratégia nestes casos seria só inserir vértices, se houvesse a certeza de que a proximidade seria evitada. No caso da Figura 4.17(a), o algoritmo só deveria quebrar $\mathcal{U}_{\mathcal{L}_{[i,j]}}$, se estivesse certo de que p se encontra a uma distância maior do que δ de cada uma das arestas de $\mathcal{L}_{[i,j]}$. Isto porque, no pior dos casos, todos os vértices de $\mathcal{L}_{[i,j]}$ seriam inseridos e a proximidade seria evitada. Para saber as relações de proximidade entre p e as arestas de $\mathcal{L}_{[i,j]}$, o algoritmo utiliza um vetor, similar ao vetor de cruzamentos, denominado vetor de proximidades. Ele armazena os índices das arestas de $\mathcal{L}_{[i,j]}$ das quais p mantém uma distância menor do que δ. O algoritmo também utiliza as variáveis p_i e p_f, para determinar eficientemente as proximidades nas URCs derivadas de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$.

Observe o exemplo da Figura 4.18. O algoritmo determina as arestas da sublinha $\mathcal{L}_{[5,18]}$ que estão próximas a p (Figura 4.18(a)) e armazena os seus índices no vetor de proximidades (Figura 4.18(d)). De maneira análoga aos cruzamentos, o número de proximidades pode ser calculado pela subtração $(p_f - p_i)$. Embora p não esteja próximo a $\overline{v_5 v_{18}}$, ele está no interior de $\mathcal{P}_{\mathcal{L}_{[5,18]}}$. O algoritmo, então, quebra $\mathcal{U}_{\mathcal{L}_{[5,18]}}$ e gera duas novas URCs (Figuras 4.18(b) e 4.18(c)). Para determinar o número de proximidades p com as sublinhas $\mathcal{L}_{[5,15]}$ e $\mathcal{L}_{[15,18]}$, o algoritmo atualiza as variáveis p_i e p_f (Figuras 4.18(e) e 4.18(f)) e calcula o valor da subtração $(p_f - p_i)$ para cada URC. Embora p esteja próximo a $\overline{v_5 v_{15}}$, o algoritmo não quebra $\mathcal{U}_{\mathcal{L}_{[5,15]}}$, porque p também está próximo a $\mathcal{L}_{[5,15]}$. Por outro lado, como p está próximo a $\overline{v_{15} v_{18}}$, mas longe de $\mathcal{L}_{[5,18]}$, o algoritmo quebra $\mathcal{U}_{\mathcal{L}_{[5,18]}}$.

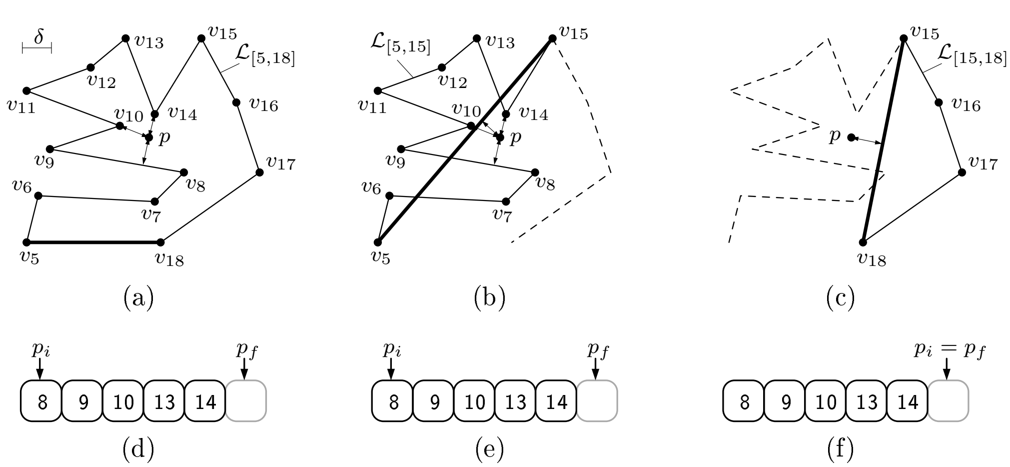


Figura 4.18: (a, b, c) Proximidades do ponto p com as arestas das sublinhas $\mathcal{L}_{[5,18]}$, $\mathcal{L}_{[5,12]}$ e $\mathcal{L}_{[12,18]}$ e (d, e, f) os seus respectivos vetores de proximidades.

Ao adicionar a conservação de espaçamentos à preservação da topologia, o algoritmo deve quebrar uma dada URC $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ sempre que a tolerância ϵ não for alcançada ou pelo menos um dos seus pontos externos (1) estiver no interior ou sobre $\mathcal{P}_{\mathcal{L}_{[i,j]}}$, ou (2) estiver próximo a $\overline{v_i v_j}$, mas longe de $\mathcal{L}_{[i,j]}$. Para simplificar este teste, a cada ponto externo de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ são associadas duas variáveis booleanas, uma que monitora a primeira condição e outra que monitora a segunda. Quando quebra uma URC, o algoritmo atualiza ambas as variáveis de cada um dos seus pontos externos, utilizando os vetores de cruzamentos e proximidades. O algoritmo só pára quando a tolerância ϵ for alcançada em todas as URCs e as duas variáveis dos pontos externos de todas as URC forem falsas, ou seja, quando as metas de simplificação e consistência forem alcançadas.

4.7 Invariância do Resultado do Algoritmo

Esta seção demonstra que o resultado do algoritmo proposto é invariante em relação à ordem de processamento das URCs. Isto quer dizer que, mesmo com a escolha aleatória de uma URC suja da lista $\mathfrak{U}_s$ durante a correção, o algoritmo sempre gera o mesmo conjunto de linhas simplificadas $\mathfrak{L}''$. A razão disto é que, embora a ordem de processamento das URCs possa variar, o próximo vértice a ser inserido em uma dada URC $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ é sempre bem definido, pois é determinado pelo ASI associado à linha $\mathcal{L}$. A invariância do resultado é muito importante, pois permite que várias instâncias do algoritmo executando em diferentes máquinas gerem um resultado único para os mesmos dados de entrada, mesmo que os dados sejam dispostos em ordens distintas.

Considere que os ASIs utilizados pelo algoritmo proposto sejam determinísticos, de forma a gerar sempre o mesmo conjunto de linhas $\mathfrak{L}'$ na etapa de simplificação. Sabe-se que a etapa de inicialização não modifica este conjunto. Resta, então, demonstrar que a escolha aleatória das URCs na etapa de correção do algoritmo não interfere no conjunto final de linhas $\mathfrak{L}''$. Para demonstrar formalmente que este conjunto é invariante, o processo de correção é apresentando no Algoritmo 1. Esta é apenas uma versão sintética da etapa de correção, que não entra nos pequenos detalhes de implementação. Basicamente, ela preenche as listas de URCs sujas $\mathfrak{U}_s$ e limpas $\mathfrak{U}_l$ e procede com o laço de correção, no qual, a cada iteração, retira aleatoriamente uma URC de $\mathfrak{U}_s$, até que $\mathfrak{U}_s$ esteja vazia.

Procedimento corrigirURCs(**var** $\mathfrak{U}$: **URC**[]; **var** $\mathfrak{L}'$: **Linha**[]; P: **Ponto**[]; ϵ: **Real**)

Var
    $\mathfrak{U}_l$, $\mathfrak{U}_s$: **URC**[];
Início
    Insira as URCs limpas de $\mathfrak{U}$ na lista $\mathfrak{U}_l$ e as sujas na lista $\mathfrak{U}_s$;
    Enquanto $\mathfrak{U}_s$ não estiver vazia **faça**
        Retire aleatoriamente uma URC $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ de $\mathfrak{U}_s$;
        Insira, em $\mathcal{L}'$, o vértice v_k, determinado pelo ASI associado a $\mathcal{L}$;
        Para cada URC $\mathcal{U}$ interferente de $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ **faça**
            Colete v_k em $\mathcal{U}$;
            Se $\mathcal{U}$ tornou-se suja com a inserção de v_k **então**
                Retire $\mathcal{U}$ de $\mathfrak{U}_l$ e a insira em $\mathfrak{U}_s$;
            Fim se
        Fim para
        Quebre $\mathcal{U}_{\mathcal{L}_{[i,j]}}$ em $\mathcal{U}_{\mathcal{L}_{[i,k]}}$ e $\mathcal{U}_{\mathcal{L}_{[k,j]}}$ e as insira nas listas adequadas;
        Atualize o grafo de interferências das URCs;
    Fim enquanto
Fim

Algoritmo 1: Correção das URCs do conjunto $\mathfrak{U}$.

Fica evidente que o Algoritmo 1 pode proceder todas as possíveis ordens de processamento das URCs, já que, a cada iteração, ele considera aleatória a escolha da URC suja. O Teorema 5, no entanto, garante que o resultado do Algoritmo 1 é invariante em relação a esta ordem. Para demonstrar o Teorema 5, é preciso ter em mente que a quebra de uma URC pode influenciar na consistência de uma das suas URCs interferentes, assim como a quebra desta URC interferente pode influenciar na consistência daquela URC. Portanto, para cada ordem, o estado de consistência das URCs é diferente. Para facilitar a demonstração, utiliza-se a notação $\mathfrak{L}'_{(i)}$ para designar o estado do conjunto $\mathfrak{L}'$ na i-ésima iteração do laço de correção. Sendo assim, a notação $\mathfrak{L}'_{(0)}$ indica o estado inicial de $\mathfrak{L}'$, isto é, o conjunto de linhas proveniente da etapa de simplificação.

Teorema 5. *A aplicação do Algoritmo 1 à tolerância ϵ sobre um mapa proveniente da etapa de simplificação, formado pelo conjunto de linhas poligonais $\mathfrak{L}' = \{\mathcal{L}'_1, \mathcal{L}'_2, \ldots, \mathcal{L}'_N\}$ e pelo conjunto de pontos P, gera sempre o mesmo mapa simplificado, formado pelo conjunto de linhas simplificadas $\mathfrak{L}'' = \{\mathcal{L}''_1, \mathcal{L}''_2, \ldots, \mathcal{L}''_N\}$ e pelo conjunto de pontos P, independentemente da ordem de processamento das URCs do conjunto $\mathfrak{U}$.*

Demonstração. Suponha que duas instâncias do Algoritmo 1 processem as URCs em ordens distintas e gerem os conjuntos $\hat{\mathfrak{L}}'' = \{\hat{\mathcal{L}}''_1, \hat{\mathcal{L}}''_2, \ldots, \hat{\mathcal{L}}''_N\}$ e $\tilde{\mathfrak{L}}'' = \{\tilde{\mathcal{L}}''_1, \tilde{\mathcal{L}}''_2, \ldots, \tilde{\mathcal{L}}''_N\}$, que diferem em pelo menos uma linha. Considere que, na i-ésima iteração do laço, as duas instâncias tenham os mesmos conjuntos de linhas, isto é, $\hat{\mathfrak{L}}'_{(i)}$ e $\tilde{\mathfrak{L}}'_{(i)}$ sejam idênticos. Suponha que, na $(i+1)$-ésima iteração, a primeira instância escolha a URC $\mathcal{U}_{\mathcal{L}_{K[i,j]}}$ e a quebre, inserindo o vértice v_k na linha $\hat{\mathcal{L}}'_K$ (Figuras 4.19(a) e 4.19(b)). Suponha, por hipótese, que v_k esteja na simplificação final $\hat{\mathcal{L}}''_K$ da primeira instância, mas não esteja na simplificação final $\tilde{\mathcal{L}}''_K$ da segunda. Nestas circunstâncias, v_k é o primeiro vértice a ser inserido em uma linha de $\hat{\mathfrak{L}}''$ que não está na linha correspondente de $\tilde{\mathfrak{L}}''$.

Sabe-se que a segunda instância também contém uma URC $\mathcal{U}_{\mathcal{L}_{K[i,j]}}$ na i-ésima iteração, já que as linhas $\hat{\mathcal{L}}''_{(i)}$ e $\tilde{\mathcal{L}}''_{(i)}$ são idênticas e têm URCs monitorando cada um dos seus trechos simplificados. Perceba também que a URC $\mathcal{U}_{\mathcal{L}_{K[i,j]}}$ da segunda instância coletou, até a i-ésima iteração, os mesmos pontos externos que a da primeira, pois os conjuntos $\hat{\mathfrak{L}}'_{(i)}$ e $\tilde{\mathfrak{L}}'_{(i)}$ são idênticos e o conjunto P é o mesmo para as duas instâncias. Sendo assim, quaisquer que sejam os motivos para que a URC $\mathcal{U}_{\mathcal{L}_{K[i,j]}}$ da primeira instância seja suja (Figuras 4.19(c), 4.19(c) e 4.19(e)), estes mesmos motivos serão aplicados à da segunda, que também será considerada suja. Logo, a URC $\mathcal{U}_{\mathcal{L}_{K[i,j]}}$ da segunda instância será quebrada, mesmo que em uma iteração posterior, e v_k será inserido na linha $\tilde{\mathcal{L}}''_K$, o que contraria a hipótese assumida. Portanto, quando um vértice é inserido em uma linha de uma instância, ele é necessariamente inserido na linha correspondente da outra. $\square$

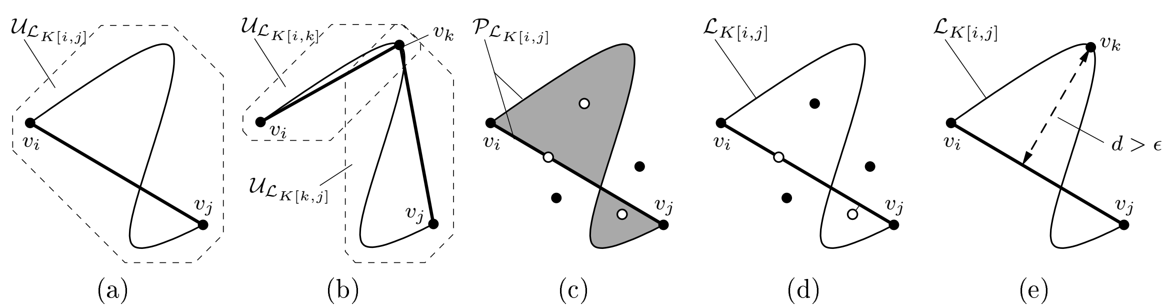


Figura 4.19: (a, b) A quebra da URC $\mathcal{U}_{\mathcal{L}_{K[i,j]}}$ pode acontecer por três motivos: (c) a presença de pelo menos um ponto externo no interior ou sobre o polígono $\mathcal{P}_{\mathcal{L}_{K[i,j]}}$; (d) a presença de pelo menos um ponto externo próximo ao segmento $\overline{v_i v_j}$, mas distante da sublinha $\mathcal{L}_{K[i,j]}$; e (e) a tolerância ϵ não ter sido alcançada no trecho entre v_i e v_j.

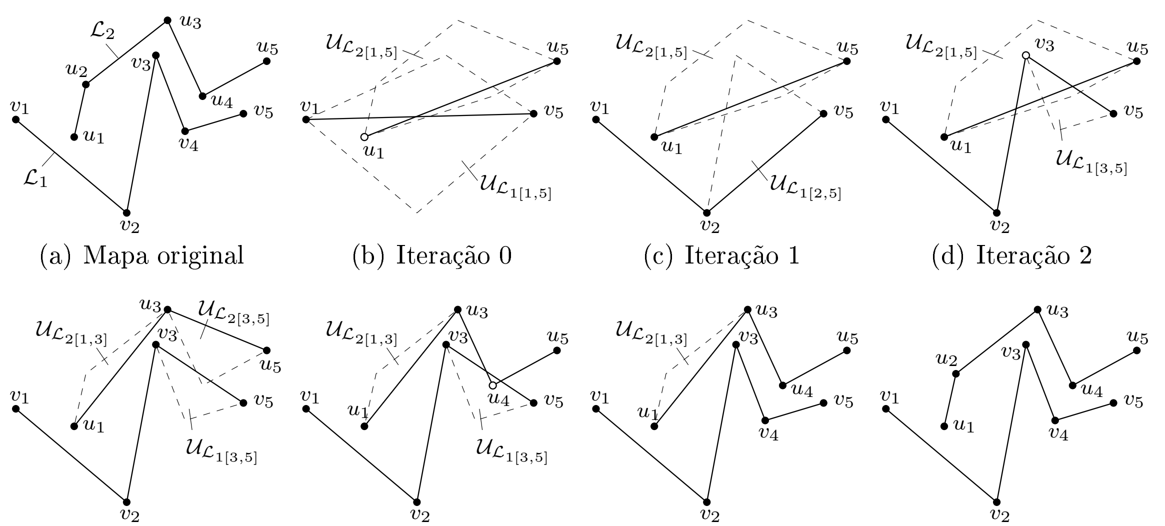
4.8 Produção de Mapas Independentes de Escala

Até o presente momento, mostrou-se que o algoritmo proposto é capaz de produzir um mapa simplificado consistente ao original, conforme as condições de consistência do Capítulo 3. Este é o modo de funcionamento usual do algoritmo, no qual o mapa resultante é adequado a uma única escala. O algoritmo tem um segundo modo de funcionamento, no qual é adaptado para produzir mapas independentes de escala com todos os níveis possíveis de detalhamento. Neste modo de funcionamento, ele não remove os vértices das linhas, apenas associa-lhes valores de tolerâncias, que podem ser usadas mais tarde para filtrar eficientemente os vértices necessários a uma dada escala. Este modo de funcionamento baseia-se na simples alteração dos valores de tolerâncias já fornecidos pelos ASIs, com o intuito de corrigir as inconsistências de cada nível.

Na etapa de simplificação, o algoritmo aplica os ASIs sobre as linhas, mas como se estivesse produzindo linhas independentes de escala (veja Apêndice A). Ao final desta etapa, cada vértice têm a si associado uma tolerância determinada pelo ASI da sua linha, mas a filtragem dos vértices ainda pode introduzir inconsistências. O algoritmo, então, coloca as tolerâncias encontradas, juntamente com a tolerância ∞ (infinita), em uma lista de tolerâncias T e as organiza em ordem decrescente. Na etapa de inicialização, para cada linha $\mathcal{L} = v_1 v_2 \cdots v_n$ do mapa original, o algoritmo cria a URC $\mathcal{U}_{\mathcal{L}_{[1,n]}}$ para monitorar a sua simplificação trivial $\mathcal{L}_{[1,n]}$. Estas URCs referem-se a uma hipotética simplificação do mapa à tolerância ∞, representando o nível com menor detalhamento.

Na etapa de correção, o algoritmo recupera a consistência de cada um dos níveis na ordem em que eles estão na lista T. O algoritmo avalia primeiramente a tolerância ∞ e determina, com base nas URCs criadas, se há inconsistências neste nível. Se houver, ele insere os vértices adicionais necessários para corrigi-las e associa-lhes esta tolerância, para que apareçam neste nível no momento da filtragem. Quando um vértice adicional é inserido, a sua tolerância é retirada de T, para que o algoritmo não tenha que processá-la em vão. Ele passa, então, à próxima tolerância de T e insere o vértice referente a ela. Ele repete o processo de correção, isto é, verifica se há inconsistências neste nível e, se houver, insere os vértices adicionais necessários, associando a eles a tolerância atual. Ao final do processo, os vértices de todas as linhas estarão associados às tolerâncias adequadas.

Observe a aplicação da etapa de correção no exemplo da Figura 4.20, cujos dados de cada iteração são exibidos pela Tabela 4.20. A iteração 0 representa o estado das linhas após a etapa de inicialização. Na iteração 1, cuja tolerância atual é ∞, o algoritmo detecta uma inconsistência na URC $\mathcal{U}_{\mathcal{L}_{1[1,5]}}$ e insere o vértice v_2 para corrigi-la. Ao fazer isso, ele associa a v_2 o valor ∞ e retira a tolerância ϵ_{v_2} de T. Na iteração 2, como o nível ∞ foi corrigido, o algoritmo retira ∞ de T, passa para a tolerância seguinte ϵ_{v_3} e insere o vértice v_3. Na iteração 3, o algoritmo detecta uma inconsistência na URC $\mathcal{U}_{\mathcal{L}_{2[1,5]}}$ e insere o vértice u_3 para corrigi-la. Ao fazer isto, ele associa a u_3 o valor ϵ_{v_3} e retira a tolerância ϵ_{u_3} de T. Este processo continua até que a última tolerância seja avaliada.



(a) Mapa original (b) Iteração 0 (c) Iteração 1 (d) Iteração 2

(e) Iteração 3 (f) Iteração 4 (g) Iteração 5 (h) Iteração 6

Figura 4.20: Aplicação da etapa de correção do algoritmo sobre o mapa formado pelas linhas $\mathcal{L}_1$ e $\mathcal{L}_2$ na produção de um mapa independente de escala para todos os níveis.

i	Lista de tolerâncias	v_1	v_2	v_3	v_4	v_5	u_1	u_2	u_3	u_4	u_5
0	$T = \{\infty, \epsilon_{v_2}, \epsilon_{v_3}, \epsilon_{u_3}, \epsilon_{u_4}, \epsilon_{v_4}, \epsilon_{u_2}\}$	∞	ϵ_{v_2}	ϵ_{v_3}	ϵ_{v_4}	∞	∞	ϵ_{u_2}	ϵ_{u_3}	ϵ_{u_4}	∞
1	$T = \{\infty, \epsilon_{v_3}, \epsilon_{u_3}, \epsilon_{u_4}, \epsilon_{v_4}, \epsilon_{u_2}\}$	∞	∞	ϵ_{v_3}	ϵ_{v_4}	∞	∞	ϵ_{u_2}	ϵ_{u_3}	ϵ_{u_4}	∞
2	$T = \{\epsilon_{v_3}, \epsilon_{u_3}, \epsilon_{u_4}, \epsilon_{v_4}, \epsilon_{u_2}\}$	∞	∞	ϵ_{v_3}	ϵ_{v_4}	∞	∞	ϵ_{u_2}	ϵ_{u_3}	ϵ_{u_4}	∞
3	$T = \{\epsilon_{v_3}, \epsilon_{u_4}, \epsilon_{v_4}, \epsilon_{u_2}\}$	∞	∞	ϵ_{v_3}	ϵ_{v_4}	∞	∞	ϵ_{u_2}	ϵ_{v_3}	ϵ_{u_4}	∞
4	$T = \{\epsilon_{u_4}, \epsilon_{v_4}, \epsilon_{u_2}\}$	∞	∞	ϵ_{v_3}	ϵ_{v_4}	∞	∞	ϵ_{u_2}	ϵ_{v_3}	ϵ_{u_4}	∞
5	$T = \{\epsilon_{u_4}, \epsilon_{u_2}\}$	∞	∞	ϵ_{v_3}	ϵ_{u_4}	∞	∞	ϵ_{u_2}	ϵ_{v_3}	ϵ_{v_4}	∞
6	$T = \{\epsilon_{u_2}\}$	∞	∞	ϵ_{v_3}	ϵ_{u_4}	∞	∞	ϵ_{u_2}	ϵ_{v_3}	ϵ_{u_4}	∞

Tabela 4.2: Dados das iterações da etapa de correção da Figura 4.20.

Ainda que o algoritmo utilize as URCs de um nível de detalhamento no seguinte, a produção de todos os níveis de escala é uma tarefa custosa. Muitas vezes, é suficiente que o mapa esteja disponível em um número finito e pequeno de escalas. Para ganhar eficiência nestes casos, o algoritmo proposto é adaptado a um terceiro modo de funcionamento. Neste modo, o algoritmo recebe como entrada, em vez de uma única tolerância ϵ, um conjunto de tolerâncias $T = \{\epsilon_0, \epsilon_1, \epsilon_2, \ldots, \epsilon_n\}$, organizado em ordem decrescente e na qual a tolerância ϵ_0 vale infinito. O algoritmo, então, procede com as três etapas para cada tolerância de T, como se estivesse no modo de funcionamento usual. Em termos de tempo, seria equivalente a executar o algoritmo $(n + 1)$ vezes.[5] A única diferença é que, ao final deste modo, os vértices estão todos associados às tolerâncias adequadas.

Uma limitação destes dois modos de funcionamento é que nenhum operador de seleção de objetos é utilizado. No modo de funcionamento usual, como só há uma tolerância, espera-se que um operador de seleção seja aplicado no mapa antes da simplificação. Ao produzir mapas independentes de escala, no entanto, é bem provável que muitos detalhes do mapa original acabem restringindo demais a simplificação e prejudiquem a remoção de vértices e, conseqüentemente, a legibilidade das linhas. Se, ao invés disso, um algoritmo de seleção fosse integrado ao algoritmo proposto e aplicado à medida que os níveis de tolerância fossem processados, os objetos seriam eliminados gradativamente e o algoritmo proposto seria certamente mais rápido e de maior qualidade.

4.9 Considerações Finais

Neste capítulo, apresentou-se o algoritmo proposto neste trabalho, detalhando-se as suas etapas e principais características. Muito embora o sucesso de boa parte destas características só possa ser comprovado com os resultados práticos, já é possível afirmar que o algoritmo proposto tem maior aplicabilidade que os algoritmos de De Berg *et al.* e de van der Poorten e Jones. Isto porque ele é capaz de simplificar consistentemente mapas contendo os três tipos de objetos (pontos, linhas e áreas), desde que tais mapas

[5]Perceba que, para um pequeno número de níveis, não há ganho em eficiência ao utilizar as URCs remanescentes de um nível de tolerância ϵ_i no próximo nível ϵ_{i+1}, porque a etapa de simplificação do nível ϵ_{i+1} teria que quebrar várias vezes as URCs do nível ϵ_i, tornando o processo lento.

não contenham redundâncias. Em comparação ao algoritmo de Saalfeld, o algoritmo proposto mostra-se mais eficaz, visto que, com base nas condições de consistência do Capítulo 3, ele elimina efetivamente todas as inconsistências do mapa resultante.

Outro ponto interessante é que o algoritmo proposto pode ser aplicado sobre diferentes ASIs. Neste contexto, ele funciona com uma espécie de camada de controle da consistência do mapa que age sobre a camada de simplificação das linhas, composta pelos ASIs. Embora a remoção hierárquica de subfenômenos não faça parte do escopo deste trabalho, o algoritmo proposto, ao ser aplicado sobre o algoritmo de Visvalingam, torna-se capaz de realizá-la de maneira consistente. Com isto, ele passa a estar um patamar acima dos algoritmos de De Berg *et al.* e de Saalfeld e se equipara ao algoritmo de van der Poorten e Jones, que também tem esta capacidade. Além disso tudo, é possível criar novos ASIs adequados a tipos específicos de fenômenos e utilizá-los com o algoritmo proposto, desde que eles tenham os pré-requisitos mencionados na Seção 4.2.

Demonstrou-se que o resultado do algoritmo é invariante em relação à ordem de processamento das URCs. Em virtude disso, o algoritmo sempre gera o mesmo mapa simplificado para um dado mapa de entrada, independentemente de como os seus dados estão dispostos. O algoritmo proposto, assim como o algoritmo de van der Poorten, é capaz de produzir mapas independentes de escala. A diferença é que, além de produzir mapas em todos os níveis possíveis de detalhamento, ele também produz mapas em um número finito de níveis. Além destas características, resta validar as melhorias do algoritmo na prática. Espera-se comprovar a boa legibilidade dos mapas resultantes e o bom desempenho do algoritmo com os resultados apresentados no Capítulo 5.

Capítulo 5

Resultados

Este capítulo apresenta uma seleção de resultados obtidos com o algoritmo proposto, utilizando, em grande parte, o RDP como ASI, e compara-os principalmente aos do algoritmo de Saalfeld. Para demonstrar a capacidade do algoritmo proposto de utilizar diferentes ASIs, ele também é aplicado sobre o algoritmo de Visvalingam, para o qual o algoritmo de Saalfeld não é adaptado. Para validar cada um dos objetivos deste trabalho, as vantagens do algoritmo proposto são avaliadas uma a uma. Os resultados são analisados em termos do percentual de retenção de vértices intermediários das linhas. Por exemplo, uma simplificação com retenção de 25% de um mapa com 1500 vértices intermediários, gera um mapa com 325 vértices intermediários. Uma simplificação com retenção de 0%, por sua vez, gera uma mapa sem vértices intermediários, ou seja, apenas com simplificações triviais. O total de vértices das linhas no mapa simplificado é a soma dos vértices intermediários retidos com os vértices extremos das linhas.

Este capítulo está organizado da seguinte maneira. A Seção 5.1 apresenta os mapas de teste utilizados. Em seguida, a Seção 5.2 analisa o aspecto visual dos mapas simplificados em função da preservação efetiva da topologia, do número reduzido de vértices adicionais, da conservação de espaçamentos, do uso de diferentes ASIs e da produção de mapas independentes de escala. Mais adiante, a Seção 5.3 analisa o desempenho do algoritmo em função da simplificação global, do uso de diferentes regiões de vizinhança, do uso de diferentes ASIs, do grafo de interferência e da produção de mapas independentes de escala. Por fim, a Seção 5.4 apresenta as considerações finais deste capítulo.

5.1 Mapas de Teste

Na aquisição dos resultados, foram utilizados 9 mapas de teste em diferentes escalas e com números variados de linhas, vértices e pontos, listados na Tabela 5.1. Os mapas de 1 a 4 representam com curvas de nível o relevo de diferentes regiões do Canadá (fonte: *Centre for Topographic Information* [5]) e os mapas de 5 a 9 representam diferentes camadas geográficas de dois países da Europa e de três estados brasileiros (fonte: *Digital Chart of the World Server* [6]). Todos os mapas foram submetidos a uma operação de remoção de redundâncias, tornando-os adequados à aplicação dos algoritmos. Os mapas serão referenciados no restante do capítulo pelos seus números, fornecidos na coluna #.

#	Nome	Escala	Cam.	#Linhas	#Vértices	#Pontos	#Total
1	Crownest	1:50.000	T	1.485	454.751	0	454.751
2	Ark Mountain	1:50.000	T	1.258	342.206	0	342.206
3	North Bay	1:250.000	T	1.632	131.241	0	131.241
4	Swan Lake	1:250.000	T	305	74.545	0	74.545
5	Amazonas	1:1.000.000	PCH	8.464	160.863	1.273	162.136
6	França	1:1.000.000	PCH	6.764	90.173	1.803	91.976
7	Itália	1:1.000.000	PCR	5.391	47.990	1.729	49.719
8	Paraná	1:1.000.000	PCR	1.718	17.922	459	18.381
9	São Paulo	1:1.000.000	PC	13	3.192	444	3.636

Tabela 5.1: Mapas utilizados na aquisição dos resultados com as seguintes camadas: cidades (C), hidrográfica (H), política (P), relevo (T) e rodoviária (R).

Os mapas são agrupados em quatro conjuntos. O primeiro conjunto é formado pelos mapas 1 e 2, compostos por um número relativamente pequeno de curvas de nível em relação à grande quantidade de vértices. O segundo conjunto é formado pelos mapas 3 e 4, também compostos por curvas de nível, porém em uma escala menor e com um número menor de vértices. O terceiro conjunto é formado pelos mapas de 5 a 8, que, mesmo contendo tipos distintos de fenômenos (rios e rodovias), possuem em comum a característica de os pontos (que representam as cidades) estarem muito próximos às linhas. O quarto conjunto é formado unicamente pelo mapa 9, que é composto apenas por cidades e contornos políticos. Estes conjuntos foram escolhidos, porque possuem diferentes tipos de fenômenos em graus distintos de detalhamento.

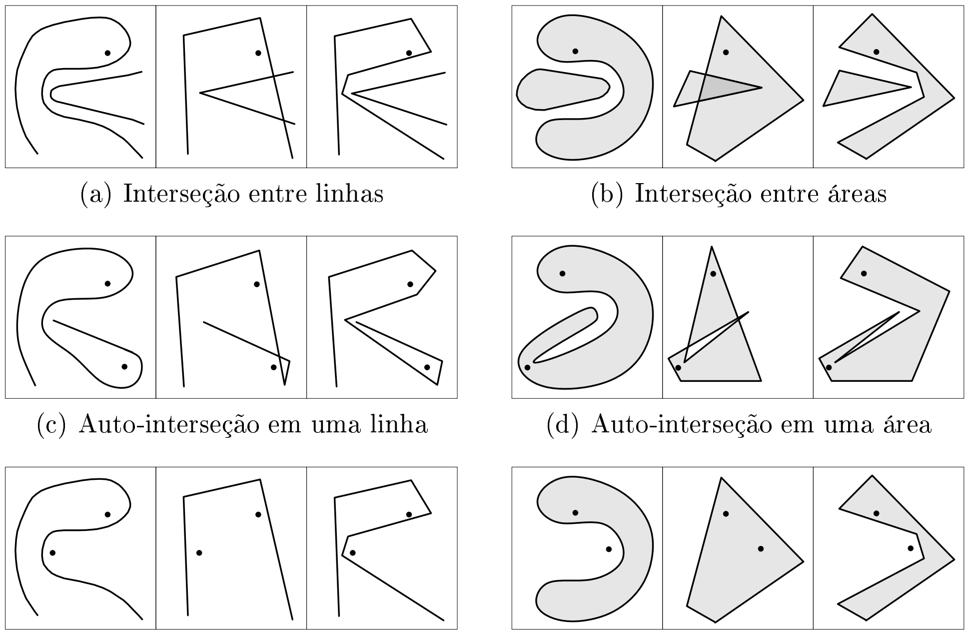
5.2 Aspecto Visual dos Mapas Simplificados

Esta seção analisa o aspecto visual dos mapas simplificados em função de diversos fatores. Compara-se primeiramente o algoritmo de Saalfeld e o proposto em termos da preservação da topologia e do número de vértices adicionais inseridos durante a etapa de correção. Em seguida, demonstra-se o benefício prático trazido pela conservação de espaçamentos entre os objetos e a utilidade do vetor de proximidades na convergência do algoritmo. Mais adiante, ilustra-se o efeito alcançado com o uso de diferentes ASIs em tipos distintos de fenômenos. Posteriormente, é feita uma análise geral do aspecto visual, envolvendo os quatro pontos discutidos até então. Por fim, analisa-se o aspecto visual de um mapa independente de escala produzido pelo algoritmo proposto.

5.2.1 Preservação Efetiva da Topologia

A Figura 5.1 ilustra resultados típicos com dados sintéticos em que o algoritmo de Saalfeld falha, devido ao teste de inversão do triângulo, e o algoritmo proposto (sobre o RDP) consegue preservar a topologia com uso do vetor de cruzamentos. Para cada resultado, o quadrado da esquerda apresenta o mapa original, o do centro, a simplificação do algoritmo de Saalfeld e o da direita, a simplificação do algoritmo proposto. Estes resultados representam casos comuns de interseção (Figuras 5.1(a) e 5.1(b)), auto-interseção (Figuras 5.1(c) e 5.1(d)) e incorretude de lado (Figuras 5.1(e) e 5.1(f)) envolvendo pontos, linhas e áreas. Em dados reais, casos como estes acontecem com maior freqüência, quando, ainda na etapa de simplificação, o RDP retém poucos vértices.

A Figura 5.2 exibe um exemplo deste problema para o mapa 1, na qual a etapa de simplificação com o RDP tem retenção de 0%, ou seja, apenas os extremos das linhas foram preservados. A Figura 5.2(a) ilustra o trecho original do mapa 1. A Figura 5.2(b) ilustra a simplificação inconsistente do algoritmo de Saalfeld. Observe que o número de interseções é bastante elevado, comprometendo a legibilidade do mapa. A Figura 5.2(c) ilustra a simplificação do algoritmo proposto, na qual todas as interseções foram eliminadas. Perceba que ambos os algoritmos inseriram um grande número de vértices adicionais durante a etapa de correção. Isto acontece porque o mapa 1 tem altíssima densidade e, para a retenção de 0% do RDP, o número de interseções torna-se muito grande.

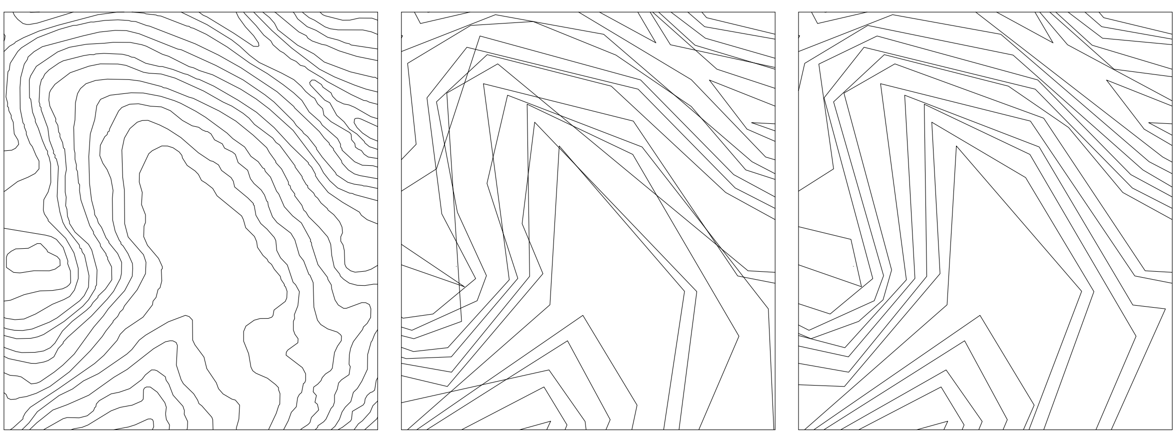


(a) Interseção entre linhas (b) Interseção entre áreas

(c) Auto-interseção em uma linha (d) Auto-interseção em uma área

(e) Incorretude de lado com uma linha (f) Incorretude de lado com uma área

Figura 5.1: Casos típicos de inconsistências topológicas do algoritmo de Saalfeld (centro), que são corretamente eliminadas pelo algoritmo proposto (direita).

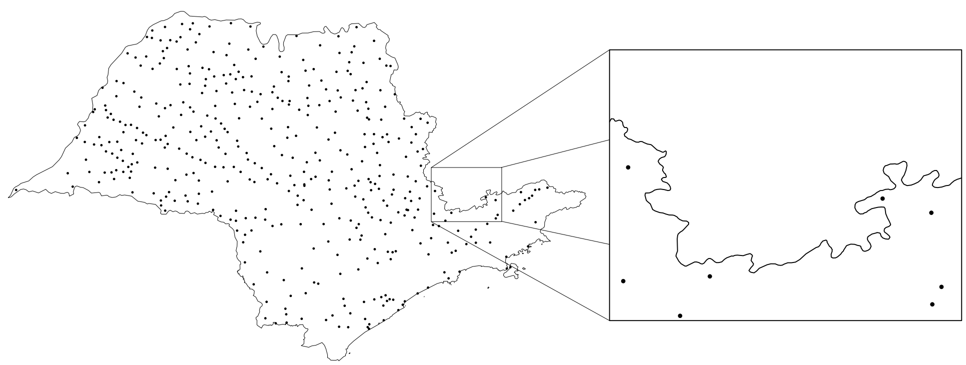


(a) Original (b) Saalfeld (c) Proposto

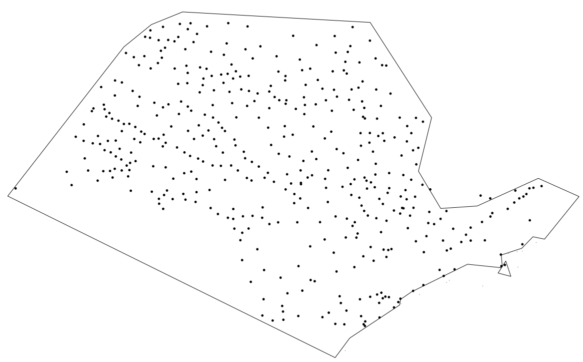
Figura 5.2: Trecho da simplificação do mapa 1 com retenção de 0% do RDP, na qual o algoritmo de Saalfeld deixa inúmeras interseções, que são eliminadas pelo proposto.

A Figura 5.3 exibe outro exemplo deste problema, mas agora no trecho do mapa 9 destacado na Figura 5.3(a). A Figura 5.3(b) ilustra a simplificação do algoritmo de Saalfeld, que mantém erroneamente algumas cidades no exterior do estado de São Paulo.

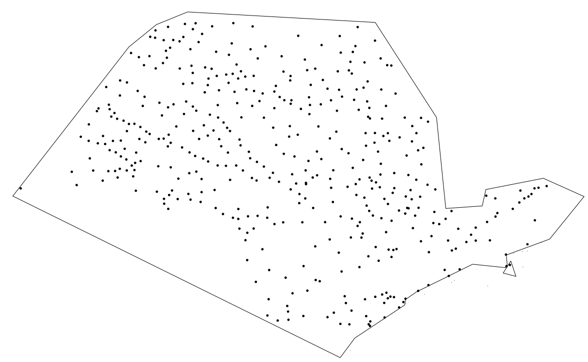
A Figura 5.3(c) ilustra a simplificação do algoritmo proposto, que preserva corretamente todas as cidades no interior do estado. Neste exemplo, é possível perceber que o uso incoerente do teste de inversão do triângulo dentro do algoritmo de Saalfeld, além de causar interseções e auto-interseções, pode também causar a incorretude de lado, além de inserir vértices desnecessários. No algoritmo proposto, este problema não acontece, pois ele insere vértices sempre e apenas nos trechos que realmente precisam.



(a) Original



(b) Saalfeld com 24 vértices



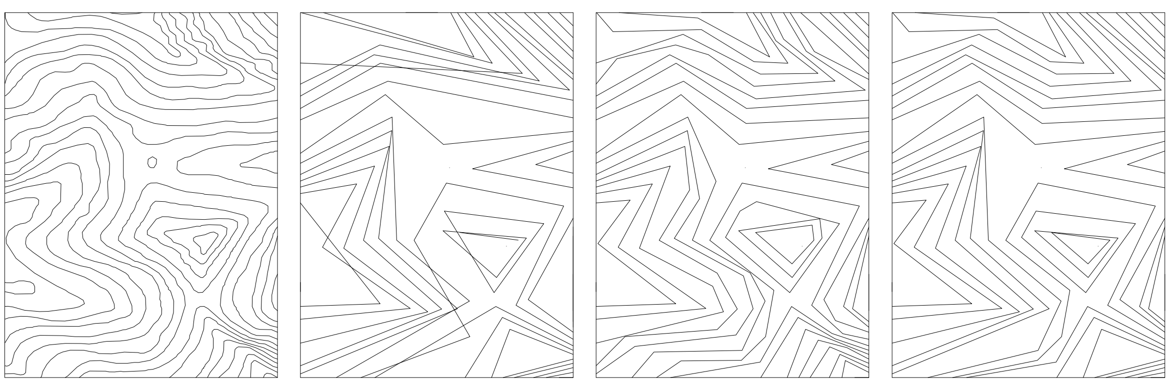
(c) Proposto (RDP) com 22 vértices

Figura 5.3: Simplificação do mapa 9 com baixa retenção do RDP, na qual o algoritmo de Saalfeld causa incorretude de lado e o algoritmo proposto preserva a topologia.

5.2.2 Número Reduzido de Vértices Adicionais

Como discutido anteriormente, assume-se que o nível adequado de generalização das linhas (isto é, a forma das linhas) do mapa simplificado é satisfatoriamente alcançado com uso dos ASIs na etapa de simplificação. Assim sendo, é interessante que a etapa de correção recupere a consistência do mapa simplificado inserindo o menor número possível de vértices adicionais. Desta maneira, ela tenta não comprometer o nível de generalização já alcançado na etapa de simplificação. Neste contexto, dois aspectos precisam ser considerados. Por um lado, as condições de consistência do Capítulo 3 são, muitas vezes, mais restritivas do que as do algoritmo de Saalfeld. Por outro lado, tais condições são aplicadas apenas aos vértices das linhas simplificadas, em vez de serem aplicadas a todos os vértices das linhas originais, como no algoritmo de Saalfeld.

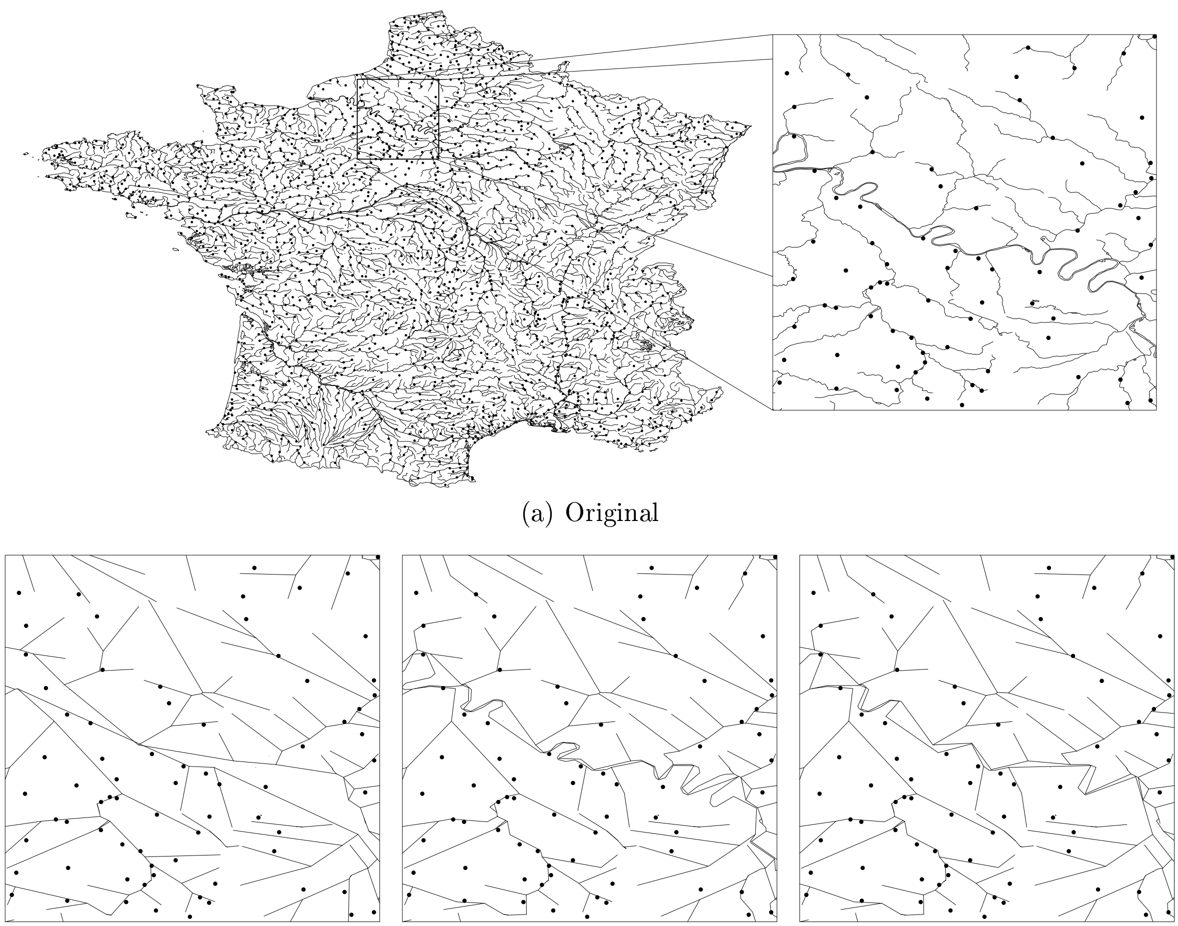
A Figura 5.4 ilustra um exemplo em que o algoritmo proposto insere menos vértices do que o de Saalfeld. A aplicação do RDP sobre o trecho do mapa 2 ilustrado pela Figura 5.4(a) retém 4% dos vértices intermediários e resulta no mapa simplificado exibido na Figura 5.4(b). Até este ponto, considera-se que o nível de generalização desejado já foi alcançado com o RDP e que as interseções remanescentes devem ser resolvidas com o menor número de vértices possível. Perceba, pela Figura 5.4(c), que o algoritmo de Saalfeld, por levar em consideração os vértices das linhas originais, insere vértices que não têm influência alguma nas interseções. O algoritmo proposto, por outro lado, insere vértices apenas para solucionar as interseções, conforme ilustra a Figura 5.4(d).



(a) Original (b) RDP (c) Saalfeld (d) Proposto

Figura 5.4: Trecho da simplificação do mapa 2 com retenção de 4% do RDP, em que o algoritmo de Saalfeld insere mais vértices adicionais do que o algoritmo proposto.

A Figura 5.5 compara o número de vértices adicionais inseridos no trecho do mapa 6 destacado na Figura 5.5(a). Na etapa de simplificação, o RDP gera um mapa com retenção de 0% e que contém inúmeras cidades do lado incorreto dos rios afluentes e também dentro do leito do rio principal, como mostra a Figura 5.5(b). O algoritmo de Saalfeld corrige todas estas inconsistências, porém ao custo de inserir um grande número de vértices adicionais no rio principal, como indica a Figura 5.5(c). O algoritmo proposto, por outro lado, reduz bastante o número de vértices adicionais, conforme ilustra a Figura 5.5(d). Este exemplo é muito comum em mapas hidrográficos, pois, nestes casos, os rios principais com leitos largos são geralmente representados por duas linhas bastante próximas e sinuosas, similares às curvas de nível.

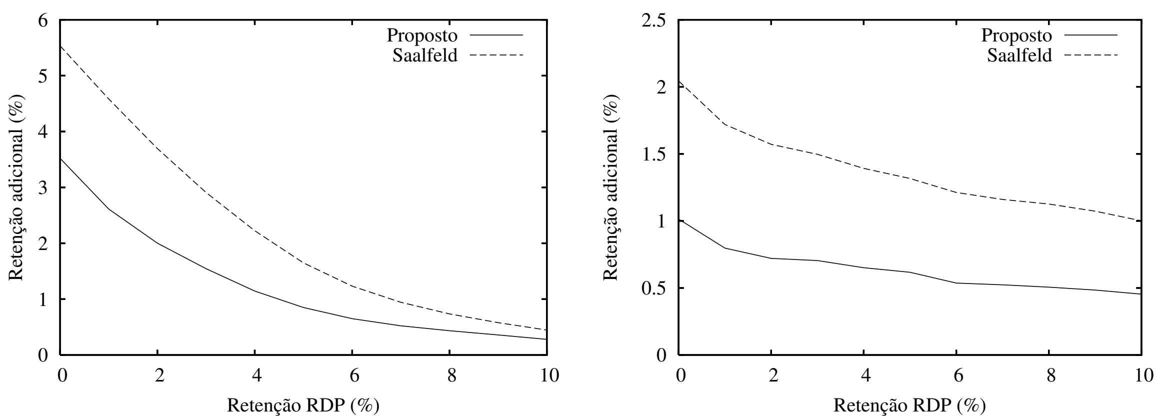


(a) Original

(b) RDP (c) Saalfeld (d) Proposto

Figura 5.5: Trecho da simplificação do mapa 6 com retenção de 0% do RDP, em que o algoritmo de Saalfeld insere mais vértices adicionais do que o algoritmo proposto.

A Figura 5.6 compara o número de vértices adicionais inseridos pelo algoritmo de Saalfeld e pelo proposto para simplificações dos mapas 1 e 3 com retenções do RDP entre 0% e 10%. O termo retenção adicional indica o percentual de vértices adicionais que foram inseridos durante a etapa de correção. Perceba que o início das curvas é caracterizado por uma alta taxa de retenção adicional, isto é, um grande número vértices adicionais, devido às inúmeras inconsistências presentes em baixas retenções do RDP. Para a curva do mapa 2, ilustrada pela Figura 5.6(a), a retenção adicional do algoritmo proposto é, em média, 1,6 vezes menor do que a do algoritmo de Saalfeld. Já para a curva do mapa 6, ilustrada pela Figura 5.6(b), a retenção adicional do algoritmo proposto é, em média, duas vezes menor do que a do algoritmo de Saalfeld.



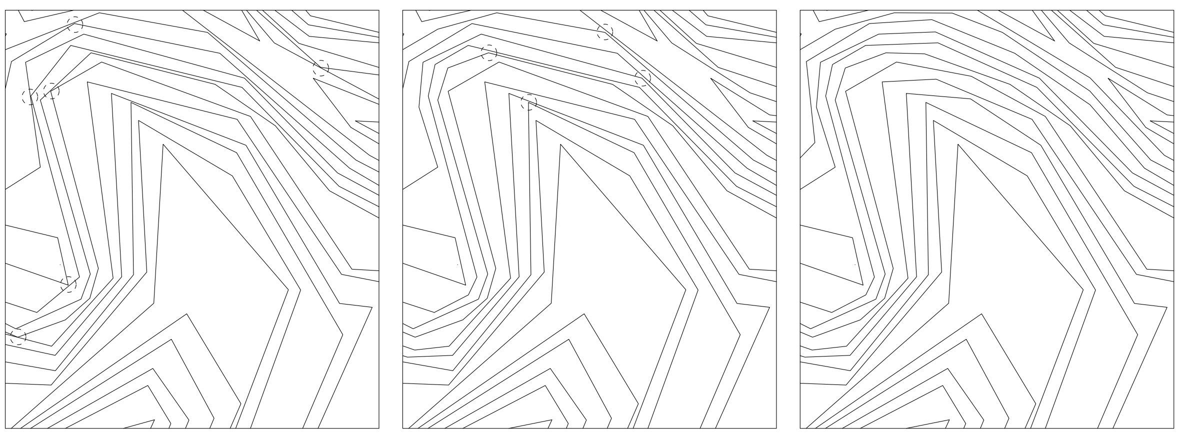
(a) Mapa 2 (b) Mapa 6

Figura 5.6: Variação do número de vértices adicionais inseridos pelo algoritmo de Saalfeld e pelo proposto para simplificações com retenção do RDP entre 0% e 10%.

Observe que, à medida que a retenção do RDP aumenta, a retenção adicional dos algoritmos diminui e tende a zero. Isto porque quanto mais vértices o RDP retém, menos inconsistências aparecem e menos vértices adicionais são inseridos. Na realidade, a partir de certo ponto, o algoritmo proposto insere poucos vértices adicionais ou sequer insere, enquanto o de Saalfeld ainda insere um número considerável de vértices. No geral, o algoritmo proposto insere de 1,5 a 2 vezes menos vértices adicionais do que o de Saalfeld. Vale salientar que a curva do mapa 2 começa com uma retenção adicional maior e desce mais bruscamente do que a do mapa 6. Esta é uma tendência das curvas de mapas com alta densidade de dados. Experimentalmente, constatou-se que quanto menor a densidade de dados do mapa, menor a inclinação da sua curva.

5.2.3 Conservação de Espaçamentos

Algumas das simplificações apresentadas até então contêm objetos próximos, porque a conservação de espaçamentos não foi utilizada. A Figura 5.7 ilustra a conservação de espaçamentos no trecho do mapa 1 exibido pela Figura 5.2(a). A simplificação com distância mínima nula permite que algumas linhas se toquem em razão das suas espessuras, como destaca a Figura 5.7(a). Já a simplificação com distância mínima $\delta = 5 \cdot 10^{-5}$ elimina estes contatos, conforme mostra a Figura 5.7(b), porém permite que linhas, que embora não se toquem, encontrem-se muito próximas. Esta proximidade é evitada com a distância mínima $\delta = 1 \cdot 10^{-4}$, como ilustra a Figura 5.7(c). O algoritmo insere um baixo número de vértices adicionais, cerca de 1 ou 2 vértices por proximidade eliminada.



(a) Proposto ($\delta = 0$) (b) Proposto ($\delta = 5 \cdot 10^{-5}$) (c) Proposto ($\delta = 1 \cdot 10^{-4}$)

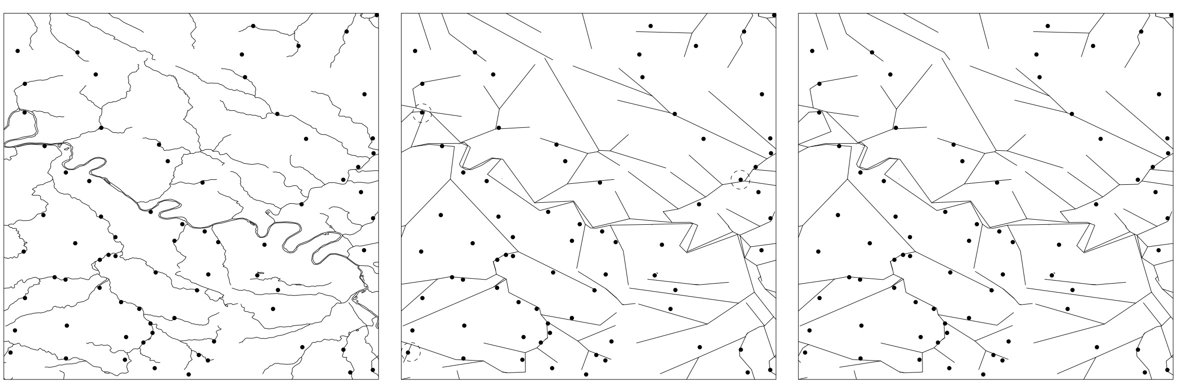
Figura 5.7: Conservação de espaçamentos no trecho do mapa 1 da Figura 5.2(a).

O algoritmo proposto utiliza o vetor de proximidades para evitar a inserção desnecessária de vértices adicionais, quando alguns objetos já estão muito próximos no mapa original. A Figura 5.8 ilustra a utilidade do vetor de proximidades sobre o trecho do mapa 3 exibido pela Figura 5.8(a). Observe que, na região destacada, as linhas se encontram tão próximas, que chegam a se tocar. A simplificação com distância mínima nula gera o mapa ilustrado pela Figura 5.8(b), que contém outras linhas próximas, além das do mapa original. A Figura 5.8(c) ilustra a simplificação com distância mínima $\delta = 1 \cdot 10^{-4}$. Perceba que as proximidades da Figura 5.8(b) foram eliminadas, enquanto que as do mapa original continuam, sem que nenhum vértice desnecessário seja inserido.

(a) Original (b) Proposto ($\delta = 0$) (c) Proposto ($\delta = 1 \cdot 10^{-4}$)

Figura 5.8: Conservação de espaçamentos em um trecho do mapa 3.

A Figura 5.9 ilustra a conservação de espaçamentos no trecho do mapa 6 ilustrado pela Figura 5.9(a). Neste trecho, quase todas as cidades já se encontram muito próximas aos rios. A simplificação com distância mínima nula, ilustrada pela Figura 5.9(b), permite que, nos três casos destacados, um rio se aproxime de uma cidade da qual não estava próximo. A simplificação com distância mínima $\delta = 1 \cdot 10^{-2}$ corrige estas inconsistências, inserindo um ou dois vértices adicionais para cada, conforme mostra a Figura 5.9(c). Este exemplo demonstra que o vetor de proximidades garante a convergência do algoritmo, isto é, evita a inserção desnecessária de vértices, pois apenas os três casos destacados, nos quais os objetos não estão próximos no mapa original, são resolvidos.

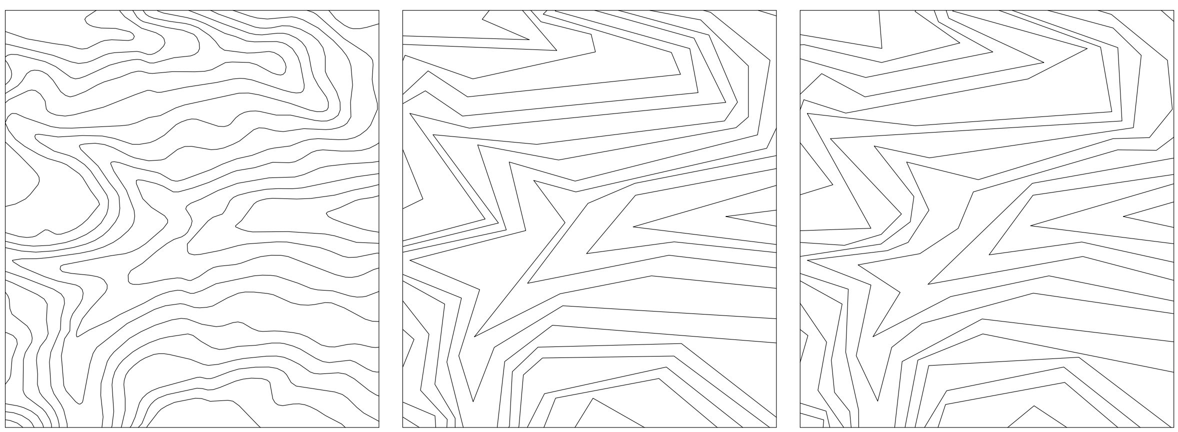


(a) Original (b) Proposto ($\delta = 0$) (c) Proposto ($\delta = 1 \cdot 10^{-2}$)

Figura 5.9: Conservação de espaçamentos em um trecho do mapa 6.

5.2.4 Uso de Diferentes ASIs

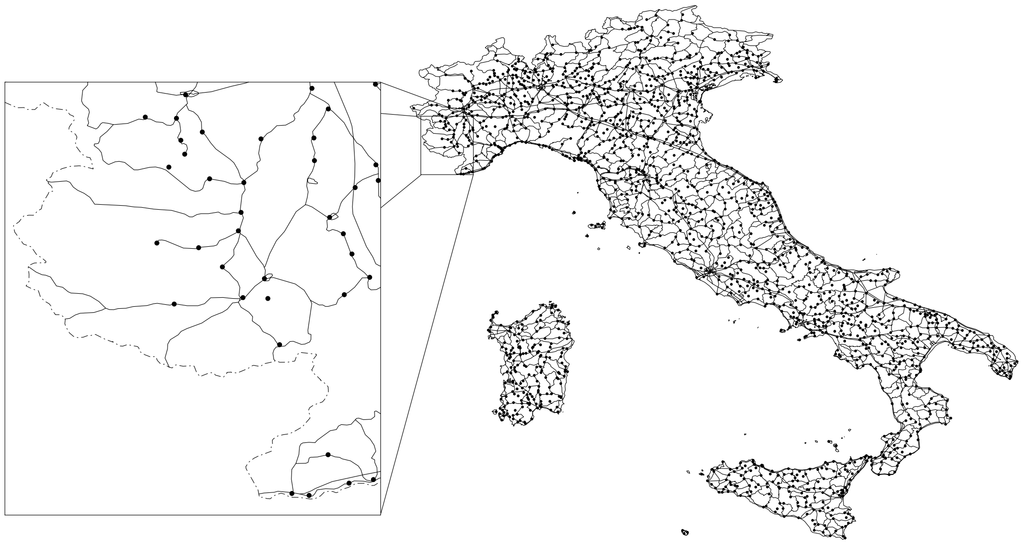
Em todas as simplificações apresentadas até então, o algoritmo proposto foi utilizado apenas sobre o RDP. Sabe-se que o algoritmo também pode ser utilizado sobre outros ASIs. A Figura 5.10 demonstra esta capacidade na simplificação do trecho do mapa 4 ilustrado pela Figura 5.10(a). O algoritmo é utilizado tanto sobre o RDP, como mostra a Figura 5.10(b), quanto sobre o de Visvalingam, como ilustra a Figura 5.10(c). Observe que, em ambos os casos, a consistência é alcançada. As diferenças entre os ASIs resumem-se apenas à ordem de inserção dos vértices, que resulta em linhas simplificadas com diferentes formas. É importante reiterar que não faz parte deste trabalho comparar os ASIs, muito menos informar em que tipo de fenômenos eles devem ser usados.



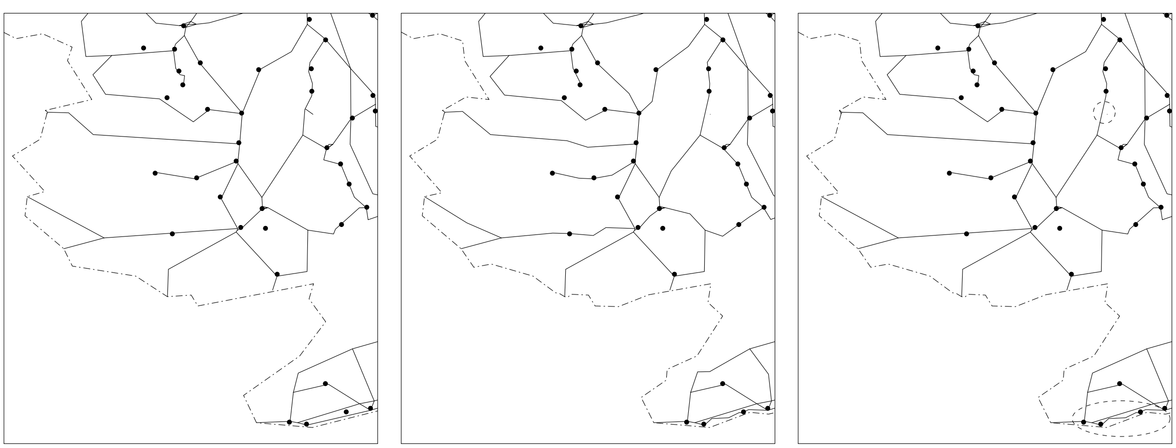
(a) Original (b) Proposto (RDP) (c) Proposto (Visvalingam)

Figura 5.10: Uso de diferentes ASIs no algoritmo proposto em um trecho do mapa 4.

A Figura 5.11 ilustra a aplicação do algoritmo proposto utilizando estes ASIs em diferentes tipos de fenômeno no trecho do mapa 7 ilustrado pela Figura 5.11(a). As linhas tracejadas representam as fronteiras e os contornos das cidades e as linhas inteiras, as rodovias. A Figura 5.11(b) ilustra a simplificação de todas as linhas com o RDP como ASI. A Figura 5.11(c) ilustra, por sua vez, a simplificação de todas as linhas com o algoritmo de Visvalingam como ASI. A Figura 5.11(d) exibe a simplificação com os dois ASIs ao mesmo tempo, sendo o de Visvalingam para as fronteiras e contornos de cidades e o RDP para as rodovias. Observe, nas regiões destacadas, que o uso dos dois ASIs implica em mudanças em relação às simplificações anteriores para a garantia da consistência.



(a) Original

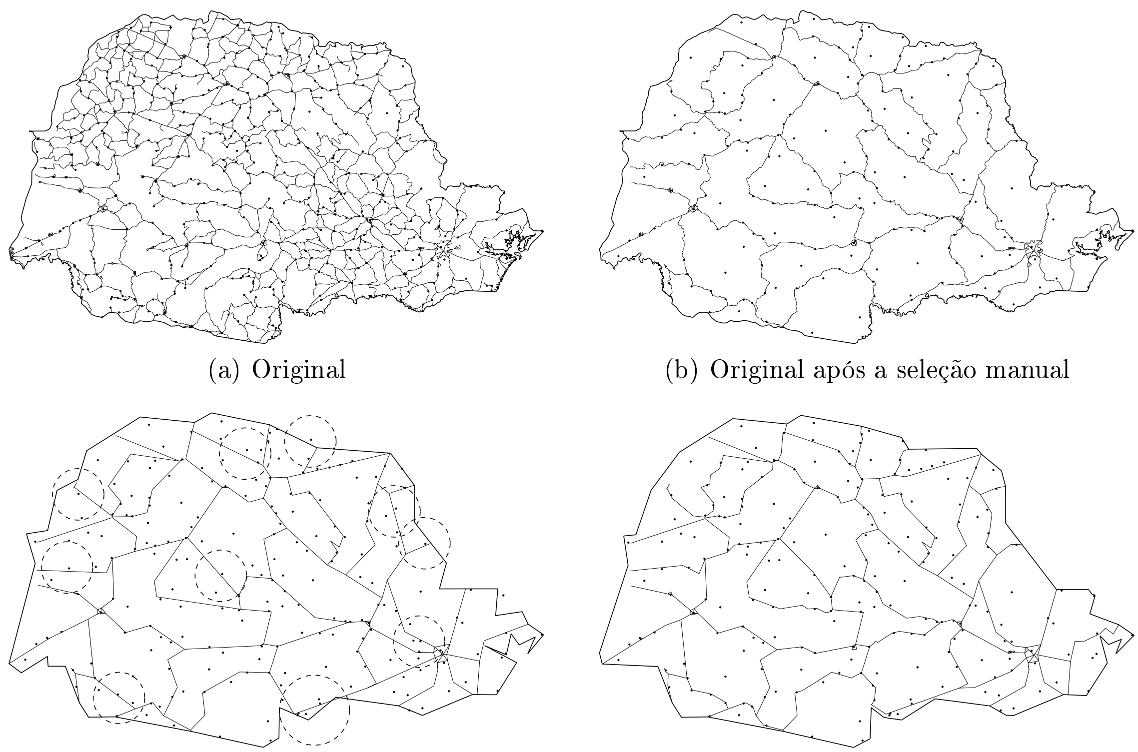


(b) Proposto (RDP) (c) Proposto (Visvalingam) (d) Proposto (ambos)

Figura 5.11: Uso de diferentes ASIs no algoritmo proposto em um trecho do mapa 7.

5.2.5 Análise Geral

A Figura 5.12 ilustra uma análise geral do aspecto visual, envolvendo os quatro pontos discutidos até aqui. A Figura 5.12(a) ilustra o mapa 8, contendo todos os seus objetos. Para melhor representá-lo na escala dada, ele foi submetido a uma seleção manual dos seus principais objetos, como ilustra a Figura 5.12(b). A Figura 5.12(c) ilustra a simplificação do RDP com baixa retenção e destaca as inconsistências que mais prejudicam a legibilidade do mapa. A Figura 5.12(d), por sua vez, ilustra a simplificação do algoritmo proposto. Além de tornar o mapa consistente com a inserção de poucos vértices adicionais, o algoritmo mostra-se capaz de aplicar ambos os ASIs, sendo o RDP sobre as rodovias e o algoritmo de Visvalingam sobre o contorno político.



(a) Original

(b) Original após a seleção manual

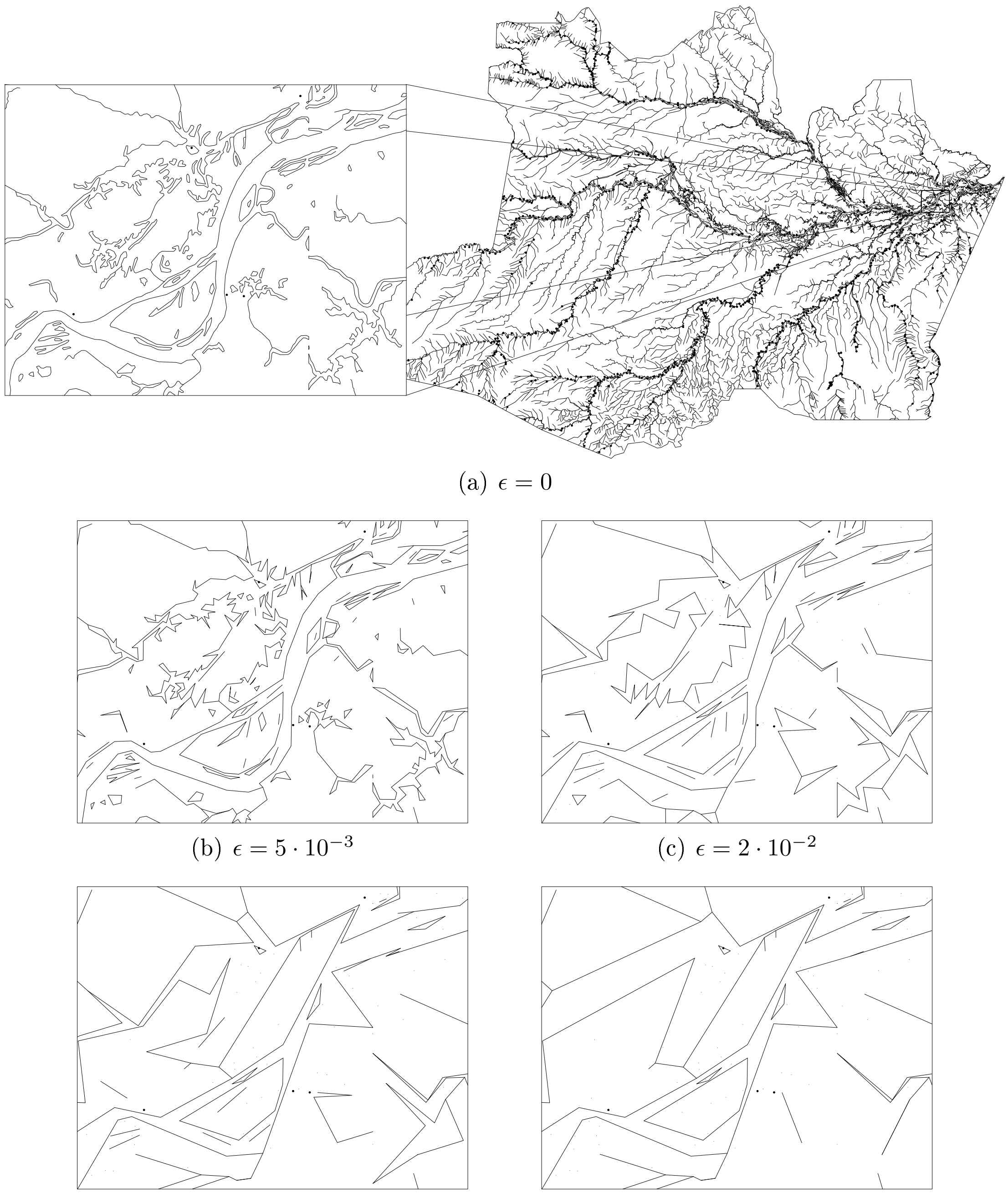
(c) RDP

(d) Proposto ($\delta = 1 \cdot 10^{-3}$)

Figura 5.12: Análise geral do aspecto visual do mapa 8 nas simplificações feitas pelo RDP e pelo algoritmo proposto após a seleção manual dos seus principais objetos.

5.2.6 Produção de Mapas Independentes de Escala

Como mencionado na Seção 4.8, o algoritmo proposto pode ser utilizado em três modos de funcionamento. Nos exemplos ilustrados até então, ele foi utilizado apenas no primeiro modo, no qual produz um mapa simplificado em um único nível de detalhamento, adequado a uma escala pré-determinada. Nos outros modos de funcionamento, o algoritmo produz mapas independentes de escala. Para exemplificar esta capacidade do algoritmo, a Figura 5.13 ilustra os cinco níveis de detalhamento de um trecho do mapa 5 resultantes da aplicação do algoritmo proposto no terceiro modo de funcionamento para o conjunto de tolerâncias $T = \{\infty, 5 \cdot 10^{-2}, 2 \cdot 10^{-2}, 5 \cdot 10^{-3}, 0\}$. Observe que, à medida que a tolerância aumenta, o nível de detalhamento das linhas diminui.



(a) $\epsilon = 0$

(b) $\epsilon = 5 \cdot 10^{-3}$

(c) $\epsilon = 2 \cdot 10^{-2}$

(d) $\epsilon = 5 \cdot 10^{-2}$

(e) $\epsilon = \infty$

Figura 5.13: Trecho do mapa 5 em 5 níveis pré-estabelecidos de detalhamento, produzidos pelo algoritmo proposto no terceiro modo de funcionamento.

Neste exemplo, é possível perceber a limitação do algoritmo na produção de mapas independentes de escala (veja Seção 4.8). Nos níveis com menor detalhamento, algumas linhas simplificadas mantêm apenas os vértices extremos das suas versões originais. O resultado é que estas linhas acabam se tornando simples segmentos ou mesmo pontos, os quais poderiam ser eliminados do mapa. Estes segmentos e pontos, no entanto, não são eliminados, porque o algoritmo não aplica nenhum operador de seleção. Assim sendo, eles ainda são considerados pelo algoritmo no tratamento da consistência e fazem com que ele insira mais vértices adicionais. Isto demonstra a importância da integração de um operador de seleção à produção de mapas independentes de escala.

5.3 Desempenho do Algoritmo

Esta seção comprova o bom desempenho do algoritmo em virtude das vantagens descritas nos capítulos anteriores. Primeiramente, discute-se o benefício da simplificação global, em detrimento à simplificação em contexto, ao se considerar apenas os vértices das linhas simplificadas na aplicação das condições de consistência. Em seguida, compara-se o desempenho do algoritmo com o uso de diferentes regiões de vizinhança. Mais adiante, compara-se o desempenho do algoritmo sobre diferentes ASIs. Em seguida, demonstra-se o ganho em desempenho com a determinação do grafo de interferência das URCs. Por fim, apresenta-se o desempenho do algoritmo nos dois modos de funcionamento voltados à produção de mapas independentes de escala. Todos os resultados foram adquiridos em um computador Pentium IV 2GHz com 512MB de RAM.

5.3.1 Simplificação Global

Sabe-se que a abordagem de simplificação global é a única que permite que as condições de consistência do Capítulo 3 sejam aplicadas apenas aos vértices das linhas simplificadas. Mostrou-se, na Seção 5.2.2, que esta estratégia diminui consideravelmente o número de vértices adicionais inseridos durante a etapa de correção. Espera-se naturalmente que, com menos vértices adicionais, o algoritmo tenha um desempenho melhor. A Tabela 5.2 compara os tempos do algoritmo proposto, que utiliza a abordagem global, com uma versão alternativa do algoritmo proposto, que utiliza a abordagem em contexto, aplicando as condições de consistência aos vértices das linhas originais. Perceba que, para os mapas 1, 2, 3 e 4, que têm alta densidade de dados, a diferença de desempenho é significativa, principalmente na retenção de 0%.

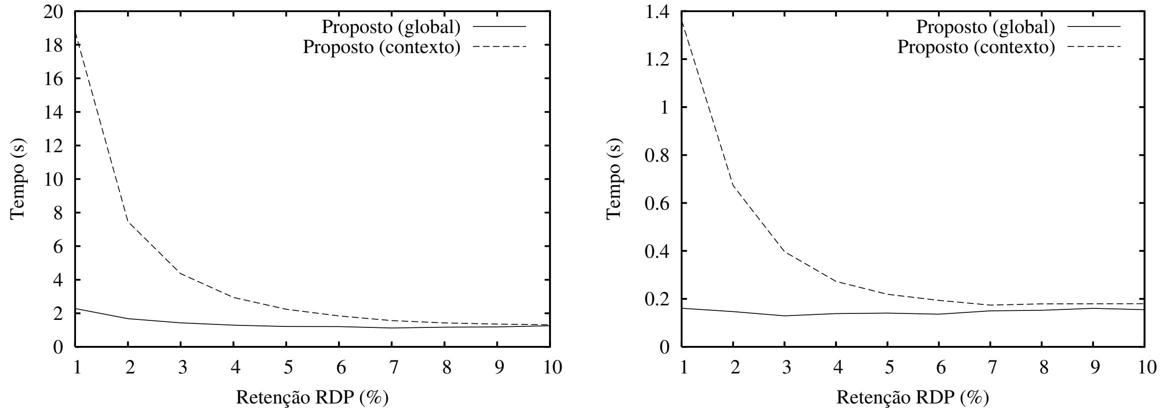
	Proposto (global)			Proposto (em contexto)		
#	0%	0,5%	1%	0%	0,5%	1%
1	10,694s	2,994s	2,282s	20m04,125s	45,574s	18,777s
2	6,389s	2,189s	1,438s	9m16,779s	35,146s	12,700s
3	1,823s	0,208s	0,180s	50,618s	1,187s	0,623s
4	1,248s	0,209s	0,161s	1m39,270s	3,004s	1,361s
5	0,371s	0,325s	0,315s	1,021s	0,663s	0,580s
6	0,164s	0,169s	0,171s	0,211s	0,191s	0,185s
7	0,158s	0,129s	0,123s	0,242s	0,137s	0,132s
8	0,042s	0,037s	0,037s	0,061s	0,039s	0,038s
9	0,086s	0,007s	0,005s	0,146s	0,012s	0,006s

Tabela 5.2: Tempos de processamento do algoritmo proposto nas abordagens global e em contexto para simplificações com retenção do RDP de 0%, 0,5% e 1%.

A Figura 5.14 ilustra a evolução das duas abordagens para retenções do RDP entre 1% e 10%. À medida que a retenção do RDP aumenta, isto é, à medida que mais vértices são retidos na etapa de simplificação, os tempos das estratégias se aproximam. Isto acontece porque quanto mais vértices o RDP retém, menores são as regiões de vizinhança das URCs e menor é o número de vértices coletados nas duas abordagens. Além disso, a abordagem global despensa certo tempo na atualização do grafo de interferência das URCs, que não está presente na abordagem em contexto. De um modo geral, pode-se dizer que a abordagem global é bem mais eficiente, quando o número de inconsistências é elevado, e é equivalente, quando este número é pequeno.

5.3.2 Uso de Diferentes Regiões de Vizinhança

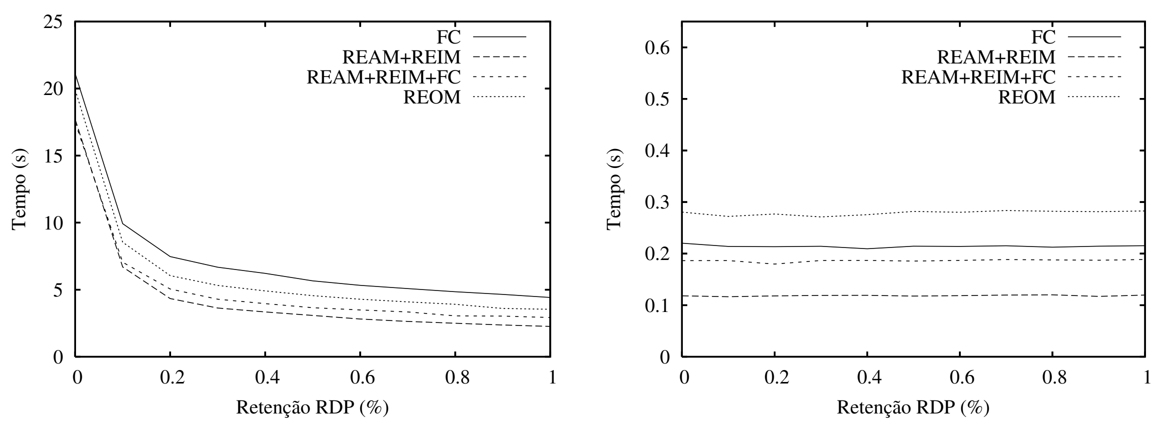
Como discutido na Seção 4.4.2, o desempenho do algoritmo também depende da região de vizinhança adotada. Esta dependência reside no tempo gasto na determinação da geometria, no teste de pertinência de um ponto e no número de pontos externos coletados. A Figura 5.15 ilustra os tempos de processamento do algoritmo proposto para



(a) Mapa 1 (b) Mapa 4

Figura 5.14: Variação do tempo de processamento do algoritmo proposto nas abordagens global e em contexto para simplificações com retenção do RDP entre 1% e 10%.

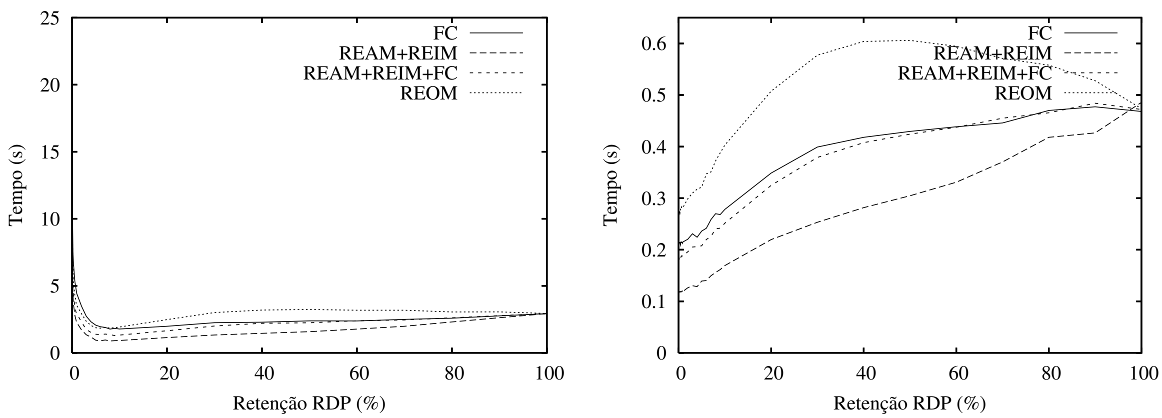
os mapas 1 e 6 com retenções do RDP entre 0% e 1%. São comparados o fecho convexo (FC), a combinação do REAM e do REIM, a combinação do REAM, do REIM e do fecho convexo e o REOM. Perceba que, embora o fecho convexo seja a região que coleta menos pontos externos, a combinação do REAM e do REIM forma a região mais eficiente, por ter uma geometria mais simples. O REOM, embora também seja simples, exige que o fecho convexo seja determinado previamente, comprometendo o seu desempenho.



(a) Mapa 1 (b) Mapa 6

Figura 5.15: Variação do tempo de processamento do algoritmo proposto com uso de diferentes regiões de vizinhança com retenções do RDP entre 0% e 1%.

A Figura 5.16 ilustra os tempos de processamento do algoritmo proposto para os mesmos mapas só que com todo o intervalo de retenções do RDP. Observe que a inclinação das curvas aumenta no trecho com retenções entre 20% e 40%. Isto acontece porque, particularmente neste trecho, o número de URCs aumenta consideravelmente, assim como o número de pontos externos coletados por elas. Perceba que a inclinação é mais acentuada, quando o algoritmo utiliza o REOM, devido à complexidade da sua determinação. De um modo geral, em todos os mapas de teste, o uso do algoritmo proposto com a combinação do REAM com o REIM mostrou-se mais eficiente.

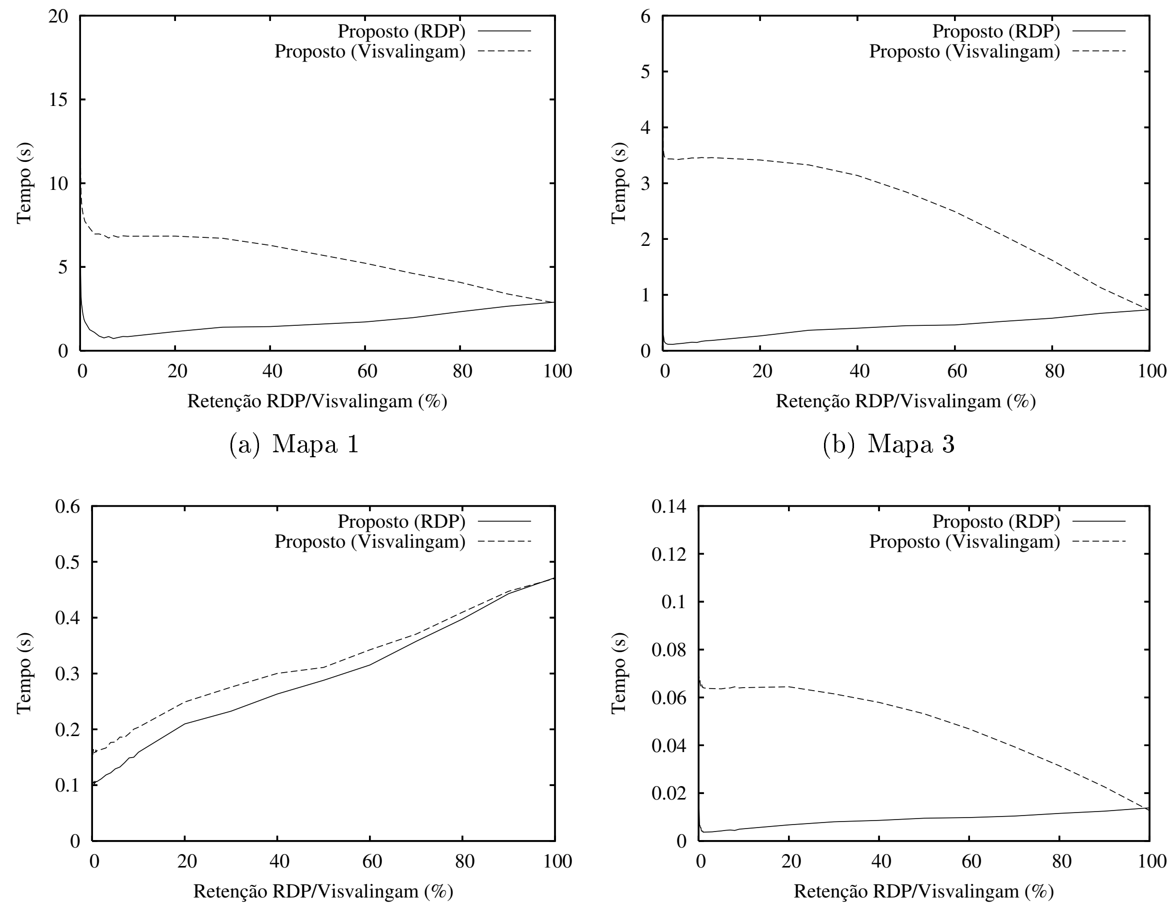


(a) Mapa 1 (b) Mapa 6

Figura 5.16: Variação dos tempos de processamento do algoritmo com uso de diferentes regiões de vizinhança com retenções do RDP entre 0% e 100%.

5.3.3 Uso de Diferentes ASIs

O desempenho do algoritmo proposto depende diretamente dos ASIs sobre os quais ele é utilizado. Os ASIs influenciam tanto na etapa de simplificação, na qual são aplicados diretamente sobre as linhas, quanto na etapa de correção, na qual decidem qual o próximo vértice a ser inserido. A Figura 5.17 ilustra o tempos de processamento do algoritmo proposto sobre o RDP e sobre o algoritmo de Visvalingam com todas as retenções. Observe que, como se esperava, o desempenho sobre RDP é superior ao sobre o algoritmo de Visvalingam, que, por ser decremental, só se torna mais rápido com altas retenções. Ao utilizar os dois ASIs, o desempenho é intermediário.



(a) Mapa 1

(b) Mapa 3

(c) Mapa 6

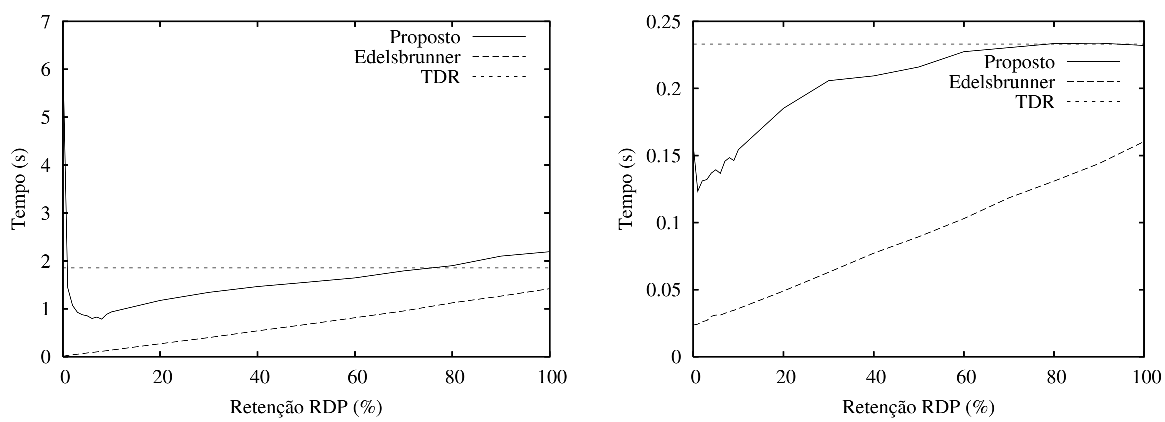
(d) Mapa 9

Figura 5.17: Variação dos tempos de processamento do algoritmo proposto sobre diferentes ASIs com retenções na etapa de simplificação entre 0% e 100%.

5.3.4 Determinação do Grafo de Interferência das URCs

Durante a etapa de inicialização, o algoritmo proposto determina o grafo de interferência das URCs. Neste processo, o algoritmo de Edelsbrunner tem papel fundamental, porque encontra os pares de URCs cujos REAMs se intersectam. A determinação do grafo de interferência funciona, na realidade, como um estágio de pré-processamento, porque organiza as URCs, para que a coleta de pontos externos tenha um desempenho satisfatório. No algoritmo de van der Poorten e Jones, pode-se considerar pré-processamento a determinação da TDR do mapa. Uma diferença importante entre estes dois processos é que a TDR é sempre aplicada sobre o mapa original, enquanto que o algoritmo de Edelsbrunner é aplicado sobre o mapa resultante da etapa de simplificação.

A Figura 5.18 ilustra os tempos de processamento dos algoritmos proposto e de Edelsbrunner para simplificações dos mapas 2 e 7 sobre todo o intervalo de retenções. Ela também mostra o tempo gasto na determinação da TDR destes mapas, realizada com o eficiente programa Triangle [38]. Como esperado, enquanto o tempo do algoritmo de Edelsbrunner cresce em função da retenção, o tempo da TDR é constante, visto que ela é sempre aplicada sobre o mapa original. Observe que, em todo o intervalo, o tempo do algoritmo de Edelsbrunner é inferior ao da TDR, mesmo para a retenção de 100%. Isto acontece não só com os mapas 2 e 7, mas também com todos outros.



(a) Mapa 2 (b) Mapa 7

Figura 5.18: Variação dos tempos de processamento do algoritmo proposto e do de Edelsbrunner com todas as retenções e o tempo gasto na determinação da TDR.

Observe que, a partir de certa retenção, o tempo do algoritmo proposto passa a sofrer mais influência do algoritmo de Edelsbrunner. Isto acontece porque o número de inconsistências diminui, mas o de retângulos aumenta. Esta influência, no entanto, não traz prejuízos significativos ao desempenho do algoritmo. No caso mapa 7, por exemplo, o algoritmo proposto é mais rápido do que a TDR em quase todas as retenções. Perceba que a TDR é a base do algoritmo de van der Poorten e Jones, já que é aplicada uma vez sobre o mapa original e toda vez que um subfenômeno é removido. Portanto, com o exemplo da Figura 5.18, é possível ter uma idéia da eficiência do algoritmo proposto, que chega a ser mais rápido do que uma única aplicação da TDR.

5.3.5 Produção de Mapas Independentes de Escala

A Tabela 5.3 ilustra o tempo gasto na produção de mapas independentes de escala com todos os níveis de detalhamento para os 9 mapas de teste. Como esperado, o tempo do algoritmo neste modo de funcionamento é relativamente elevado para os mapas mais complexos. Este processo é, no entanto, realizado uma única vez e, quando uma escala for requerida, ela poderá ser obtida com uma simples filtragem. Este modo de funcionamento é muito interessante em servidores de dados geográficos, nos quais não se tem conhecimento do nível de escala que será requerido. Outro detalhe importante é que o desempenho do algoritmo proposto sobre o RDP mostra-se maior do que sobre o algoritmo de Visvalingam em todas as instâncias, confirmando a eficiência do RDP.

#	RDP	Visvalingam
1	2h01m44,063s	2h47m56,417s
2	1h00m59,620s	1h20m30,603s
3	14m05,094s	15m19,904s
4	3m55,497s	4m34,967s
5	16m04,644s	17m20,363s
6	4m09,499s	4m45,358s
7	56,120s	1m00,311s
8	6,610s	7,826s
9	0,100s	0,176s

Tabela 5.3: Tempos de processamento do algoritmo proposto para a produção de mapas independentes de escala com todos os níveis possíveis de detalhamento.

Sabe-se que, quando um número limitado de níveis é suficiente, o algoritmo proposto pode ser utilizado em um terceiro modo de funcionamento. A Tabela 5.4 ilustra os tempos gastos neste modo de funcionamento na produção de mapas independentes de escala com um número pré-estabelecido de níveis: 5, 20 e 100 níveis. Observe que os tempos são bem inferiores aos da Tabela 5.3, mesmo para 100 níveis de detalhamento. Este modo de funcionamento torna-se, então, muito útil para SIGs, nos quais já se sabe os níveis desejados. É interessante observar que, neste modo de funcionamento, o tempo de produção de um determinado número de níveis é aproximadamente igual à soma dos tempos gastos pelo algoritmo no modo usual executando uma vez para cada nível.

	RDP			Visvalingam		
#	5	20	100	5	20	100
1	21,182s	50,919s	3m23,734s	27,258s	55,701s	3m25,176s
2	13,632s	34,984s	2m23,173s	15,707s	36,883s	2m25,338s
3	4,569s	11,457s	48,973s	7,931s	14,665s	49,108s
4	2,505s	6,318s	26,815s	3,613s	7,272s	26,928s
5	3,548s	13,141s	1m05,249s	3,613s	12,587s	1m03,974s
6	1,880s	6,808s	35,794s	1,913s	6,835s	34,151s
7	1,065s	3,841s	19,208s	1,064s	3,787s	19,099s
8	0,323s	1,176s	5,709s	0,329s	1,142s	5,788s
9	0,104s	0,199s	0,700s	0,165s	0,257s	0,741s

Tabela 5.4: Tempos de processamento do algoritmo proposto para a produção de mapas independentes de escala com um número pré-estabelecido de níveis de detalhamento.

5.4 Considerações Finais

Neste capítulo, os resultados indicam que o algoritmo proposto apresenta vantagens práticas em relação aos algoritmos presentes na literatura, tanto em termos do aspecto visual do mapa resultante, quanto em termos do desempenho do algoritmo. Ao comparar os resultados do algoritmo proposto a simplificações feitas pelo algoritmo de Saalfeld, observou-se, em particular, que a preservação efetiva da topologia e o número reduzido de vértices adicionais inseridos trazem grandes benefícios ao aspecto visual do mapa. Os dois outros fatores do aspecto visual avaliados, isto é, a conservação de espaçamentos e o uso de diferentes ASIs, trazem benefícios em relação aos três algoritmos – os algoritmos de De Berg *et al.*, de Saalfeld e de van der Poorten e Jones.

O bom desempenho também é resultante de diversos fatores. Observou-se que a aplicação da simplificação global, em oposição à abordagem em contexto (base dos algoritmos de De Berg *et al.* e de Saalfeld), tem grande influência no bom desempenho do algoritmo, principalmente quando o número de inconsistências é elevado. Observou-se também que o uso da combinação do REAM com o REIM como região de vizinhança, além de tornar a implementação mais simples, torna o algoritmo mais eficiente. É importante perceber que, embora não comparado diretamente aos três algoritmos em termos do desempenho, o algoritmo proposto leva grande vantagem, pois supera o desempenho de vários aspectos do algoritmo de Saalfeld, que é dentre os três o mais eficiente.

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho tratou da simplificação consistente de linha em mapas cartográficos. Sob o foco teórico, apresentou-se um conjunto de condições de consistência que, quando aplicadas aos vértices das linhas simplificadas e aos pontos do mapa, garantem a ausência de interseções e auto-interseções, a corretude de lado e a conservação de espaçamentos entre os objetos. Sob o foco prático, propôs-se um algoritmo capaz de gerar, com base nestas condições, um mapa consistente ao original. O algoritmo proposto basicamente utiliza a abordagem de simplificação global e corrige os erros por meio de URCs, inserindo vértices adicionais nos trechos inconsistentes. Mostrou-se que ele alcançou excelentes avanços, quando comparado aos principais algoritmos presentes na literatura, tanto no aspecto visual dos mapas simplificados, quanto no desempenho.

Até então, nenhum dos trabalhos em simplificação consistente tinha estudado tão profundamente a questão da consistência dentro do processo simplificador. O máximo a que se tinha chegado era o estudo feito por De Berg *et al.* sobre a consistência de um conjunto de pontos com respeito a uma área, que foi equivocadamente generalizado para linhas por Saalfeld. Em todo caso, nenhum dos trabalhos demonstrou formalmente a consistência do método de simplificação que propunham. Este trabalho apresentou um estudo teórico envolvendo não só a preservação da topologia do mapa, mas também a conservação de espaçamentos entre os seus objetos. Ele demonstrou que é suficiente analisar os vértices das linhas simplificadas, em vez de todos os vértices das linhas originais, o que culminou em um algoritmo eficiente e que gera melhores resultados.

O algoritmo proposto é capaz de utilizar diferentes ASIs sobre linhas de diferentes tipos de fenômeno, como, por exemplo, rios, rodovias, contornos políticas e curvas de nível. Ao fazer isto, ele consegue separar o controle sobre a forma da linha, isto é, a generalização propriamente dita, do controle de consistência do mapa. A remoção hierárquica de subfenômenos, embora não seja o objetivo deste trabalho, pôde ser alcançada com o uso do algoritmo de Visvalingam, tornando o método comparável ao algoritmo de van der Poorten e Jones. Sugere-se que seja analisado com maior profundidade um novo paradigma de simplificação, no qual os ASIs seriam desenvolvidos para tipos específicos de fenômenos e deixariam a questão da consistência para o algoritmo proposto, que poderia ser visto como uma camada superior de controle de consistência.

Ao utilizar a simplificação global para aplicar as condições de consistência apenas aos vértices das linhas simplificadas, o algoritmo proposto reduz consideravelmente o número de vértices adicionais inseridos em comparação com o de Saalfeld. Isto pôde ser comprovado nos resultados que mostraram uma redução de 1,5 a 2 vezes neste número. Este é um fator importante do aspecto visual, já que se considera que, com o uso dos ASIs, as linhas já tenham atingido o grau de simplificação desejado. Assim sendo, reduzindo-se o número de vértices adicionais, espera-se não comprometer este grau de simplificação. Outra questão importante é que, como os vértices inseridos são escolhidos pelos ASIs, o número de vértices adicionais depende diretamente do ASI utilizado.

No que diz respeito à conservação de espaçamentos, observou-se que mesmo objetos que estão fora da região de vizinhança de uma URC poderiam aparecer próximos do segmento tratado por ela. Para agregar esta observação, o algoritmo aplica uma dilatação nas regiões de vizinhança. Outra questão importante, que não foi considerada na teoria, é que, nos casos em que objetos já se encontram próximos antes da simplificação, não é possível garantir, apenas com a inserção de vértices adicionais, que estes objetos se distanciem. Para evitar que o algoritmo insira muitos vértices tentando evitar problemas como este, introduziu-se o vetor de proximidades, que armazena a informação necessária para que o algoritmo saiba que casos podem ser resolvidos.

O algoritmo proposto foi adaptado à produção de mapas independentes de escala. Esta adaptação requer apenas pequenas modificações na sua estrutura, para garantir um bom desempenho. É possível produzir mapas tanto para todos os níveis de escala, adequados às necessidades de servidores de dados geográficos, quanto para um número finito de níveis, adequados às necessidades de SIGs. Nos dois casos, as tolerâncias são armazenadas em um vetor em ordem decrescente e a simplificação é aplicada gradativamente, seguindo-se a ordem do vetor. Uma questão importante é que, ao se utilizar diferentes ASIs, deve-se fazer uma correspondência entre os seus critérios de tolerância, para que o grau de simplificação seja equivalente para todas as linhas do mapa.

O bom desempenho do algoritmo foi alcançado graças a três fatores importantes. O primeiro é a união da simplificação global à aplicação das condições de consistência apenas sobre os vértices das linhas simplificadas. Ela reduz o número de vértices adicionais e, conseqüentemente, o tempo gasto na etapa de correção. O segundo é a adoção, como região de vizinhança de uma URC, da combinação do REAM e do REIM da sua sublinha. Por ter geometria mais simples do que a do fecho convexo, esta combinação tem baixos tempos de determinação e de teste de pertinência de um ponto. O terceiro é o grafo de interferência das URCs, que reduz consideravelmente o tempo de coleta de pontos externos, tanto na etapa de inicialização, quanto na de correção.

Embora tenha introduzido várias melhorias em comparação aos outros algoritmos, o algoritmo proposto ainda apresenta certas limitações. Uma destas limitações ocorre na produção de mapas independentes de escala, na qual ele utiliza todos os objetos do mapa original em todos os níveis de detalhamento, como ilustrado pela Figura 5.13. Em razão disto, os níveis mais baixos, além de terem uma quantidade excessiva de objetos, contém muitas inconsistências, que só são corrigidas com a inserção de um número elevado de vértices adicionais. Quando um determinado nível de detalhamento é requerido, é possível aplicar um operador de seleção para diminuir a quantidade de objetos no mapa. Esta solução, entretanto, não reduz o número de vértices adicionais deste nível.

O ideal é integrar o operador de seleção ao algoritmo proposto, permitindo que cada nível já seja gerado com a quantidade de objetos adequada. O algoritmo poderia utilizar um ou mais operadores de seleção, associando a cada linha do mapa o operador de seleção mais adequado ao tipo de fenômeno que ela representa. A simplicidade na integração com operadores de seleção reside no fato de que eles apenas eliminam objetos do mapa e, portanto, não promovem qualquer deslocamento. Assim sendo, a seleção, tal qual a simplificação de ASIs extratores, poderia ser tratada seguindo-se as condições de consistência discutidas neste trabalho. A integração da seleção seria a alternativa mais simples, para contornar a limitação do algoritmo na produção de mapas independentes de escala e é, portanto, sugerida como trabalho futuro.

A integração da seleção poderia ser realizada da seguinte maneira. Suponha que as tolerâncias tenham sido calculadas ou fornecidas e estão ordenadas em um vetor. Para a tolerância infinita, a seleção deverá manter no mapa apenas os objetos de maior importância, cujas linhas serão consistentemente simplificadas. À medida que o algoritmo percorre o vetor de tolerâncias, mais e mais objetos deverão aparecer no mapa. Em outras palavras, objetos de menor importância vão sendo considerados pelo algoritmo gradativamente. Ao chegar na tolerância zero, todos os objetos serão selecionados e nenhuma simplificação será feita. Perceba que, com a diminuição das tolerâncias, tanto a seleção quanto a simplificação aumentam o número de pontos e vértices no mapa.

Outra limitação do algoritmo é a já mencionada proximidade de objetos do mapa original. Quando objetos já se encontram próximos no mapa original, o algoritmo proposto não é capaz de garantir que eles apareçam devidamente espaçados no mapa simplificado, como ilustrado pela Figura 5.8. De certo modo, esta limitação é aceitável, porque ocorre em um pequeno grau na grande maioria dos mapas. Além disso, em casos em que, por exemplo, cidades estão próximas a rios, o tratamento é feito com a utilização de cores diferentes. Por outro lado, quando uma proximidade é realmente problemática e não pode ser resolvida com a inserção de vértices, a única solução é a utilização de um operador de deslocamento, que pode ser integrado ao algoritmo em trabalhos futuros.

A integração do deslocamento é, no entanto, mais complicada do que a da seleção. Isto porque as condições de consistência discutidas neste trabalho assumem que são mantidos apenas pontos e vértices das linhas originais, sem modificá-los. Ao permitir que pontos e vértices sejam deslocados, estas condições passam a ser inválidas e a consistência não pode mais ser garantida apenas com a inserção de vértices. É possível, no entanto, integrar o deslocamento de maneira parcial, permitindo pequenos deslocamentos dos vértices para solucionar os problemas de proximidade. Por outro lado, se a integração fosse total, o algoritmo certamente inseriria um número de vértices ainda menor. Isto, no entanto, exigiria uma completa reformulação do estudo de consistência.

Referências Bibliográficas

[1] BICANIC, Z., AND SOLARIC, R. Cartographic generalization and contributions to its automation. *Geoadria 7*, 2 (2002), 5–21.

[2] BUTTENFIELD, B. P. Transmitting vector geospatial data across the internet. In *GIScience 2002* (2002), M. J. Egenhofer and D. M. Mark, Eds., Springer Verlag, pp. 51–64.

[3] CHAN, W. S., AND CHIN, F. Approximation of polygonal curves with minimum number of line segments. In *The 3rd International Symposium on Algorithms and Computation* (1992), Springer-Verlag, pp. 378–387. Computer Science 650.

[4] CORMEN, T. H., LEISERSON, C. E., AND RIVEST. *Introduction to Algorithms.* MIT Press, Cambridge, Massachusetts, 1990.

[5] Centre for Topographic Information. Site internet `http://maps.nrcan.gc.ca/`. Data do último acesso: 13/06/2007.

[6] Digital Chart of the World Server. Site internet `http://www.maproom.psu.edu/dcw/`. Data do último acesso: 13/06/2007.

[7] DE BERG, M., VAN KREVELD, M., AND SCHIRRA, S. A new approach to subdivision simplification. In *The 12th Interational Symposium of Computer-Assisted Cartography* (1995), vol. 12, pp. 79–88.

[8] DE BERG, M., VAN KREVELD, M., AND SCHIRRA, S. Topologically correct subdivision simplification using the bandwidth criterion. *Cartography and Geographic Information Systems 25*, 4 (1998), 243–257.

[9] Dettori, G., and Falcidieno, B. An algorithm for selecting main points on a line. *Computers and Geosciences 8* (1982), 3–10.

[10] Douglas, D. H., and Peucker, T. K. Algorithms for the reduction of the number of points required for represent a digitized line or its caricature. *Canadian Cartographer 10*, 2 (1973), 112–122.

[11] Dutton, G. Scale, sinuosity and point selection in digital line generalization. *Cartography and Geographic Information Science 26*, 1 (1999), 33–53.

[12] Edelsbrunner, H. Dynamic data structures for orthogonal intersections queries. Tech. rep., Tech. Univ. Graz, Institute für Informationsverarbeitung, 1980.

[13] Edwardes, A., Mackaness, W., and Urvin, T. Self evaluating generalization algorithms to automatically derive multi scale boundary sets. In *The 8th International Symposium on Spatial Data Handling* (Vancouver, Canada, 1998), pp. 361–372.

[14] Finkel, R. A., and Bentley, J. L. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica 4* (1974), 1–9.

[15] Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. *Computer Graphics: Principles and Practice (2nd ed.).* Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA, 1990.

[16] Galanda, M. *Automated Polygon Generalization in a Multi Agent System.* PhD thesis, Universität Zürich, 2003.

[17] Guttman, A. R-trees: a dynamic index structure for spatial searching. In *SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 1984), ACM Press, pp. 47–57.

[18] Hershberger, J., and Snoeyink, J. Speeding up the Douglas-Peucker line simplification algorithm. In *The 5th International Symposium on Spatial Data Handling* (1992), pp. 134–143.

[19] Imai, H., and Iri, M. *Computational morphology.* Elsevier, 1988, ch. Polygonal approximations of a curve-formulations and algorithms, pp. 71–86.

[20] Jones, C. B., and Ware, J. M. Proximity search with a triangulated data model. *The Computer Journal 41*, 2 (1998), 71–83.

[21] Justo, S. C. *Geometric problems in cartographic networks*. PhD thesis, Universiteit Utrecht, Março 2004.

[22] Lonergan, M., and Jones, C. B. An iterative displacement method for conflict resolution in map generalization. *Algorithmica 30*, 2 (2001), 287–301.

[23] Marino, J. S. Identification of characteristic points along naturally occurring lines: an empirical study. *The Canadian Cartographer 13*, 1 (1979), 70–80.

[24] McKeown, D., McMahill, J., and Caldwell, D. The use of spatial context awareness in feature simplification. In *GeoComputation Conference Procedings* (July 1999), pp. 25–28.

[25] McMaster, R. B. Automated line generalization. *Cartographica 24*, 2 (1987), 74–111.

[26] McMaster, R. B. The geometric properties of numerical generalization. *Geographical Analysis 19*, 4 (1987), 330–346.

[27] McMaster, R. B., and Shea, K. S. *Generalization in Digital Cartography*. Association of American Geographers, 1710 16th Street NW, Washington D.C. 20009-3198, 1992.

[28] Melkman, A. A. On-line construction of the convex hull of a simple polyline. *Information Processing Letters 25* (1987), 11–12.

[29] Monmonier, M. S. Towards a practicable model of cartographic generalization. In *Auto Carto London* (1986), M. Blakemore, Ed., ICA, pp. 257–266.

[30] Morrison, J. L. Map generalization: theory, practice and economics. In *Auto Carto* (1975), vol. 2, pp. 99–112.

[31] Müller, J. C. Fractal and automated line generalization. *Cartographic Journal 24* (1987), 27–34.

[32] Müller, J. C. Optimum point density and compaction rates for the representation of geographic lines. In *Auto Carto* (1987), N. R. Chrisman, Ed., vol. 8, pp. 221–230.

[33] Müller, J. C., Weibel, R., Lagrange, J.-P., and Salgé, F. Generalization: state of the art and issues. In *Generalization and GIS. Methodology and Practice*, J. C. Müller, J.-P. Lagrange, and R. Weibel, Eds., GISDATA 1. Taylor & Francis, 1995, pp. 3–17.

[34] Project AGENT. State of the art and selection of basic algorithms. Tech. Rep. ESPRIT LTR 24,1999, Institut Géographique National, 1999.

[35] Ramer, U. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing 1* (1972), 224–256.

[36] Rieger, M., and Coulson, M. Consensus or confusion: Carographer's knowledge of generalization. *Cartographica 2&3*, 20 (1993), 69–80.

[37] Saalfeld, A. Topologically consistent line simplification with the Douglas-Peucker algorithm. *Cartography and Geographic Information Science 26*, 1 (1999), 7–18.

[38] Shewchuk, J. R. Triangle: A two-dimensional quality mesh generator and delaunay triangulator. Site internet `http://www.cs.cmu.edu/~quake/triangle.html`. Data do último acesso: 13/06/2007.

[39] Spiess, E. The need for generalization in gis environment. In *GIS and generalization — Methodology and Practice* (1989), GISDATA, pp. 31–46.

[40] Thapa, K. Automatic line generalization using zero crossings. *Photogrammetric Engineering and Remote Sensing 54*, 4 (1988), 511–517.

[41] van der Poorten, P. M., and Jones, C. B. Customisable line generalisation using Delaunay triangulation. In *The 19th International Cartographic Association Conference* (August 1999).

[42] van der Poorten, P. M., and Jones, C. B. Characterisation and generalisation of cartographic lines using Delaunay triangulation. *International Journal of Geographical Information Science 16*, 8 (2002), 773–795.

[43] van der Poorten, P. M., Zhou, S., and Jones, C. B. Topologically-consistent map generalization procedures and multi-scale spatial databases. In *GIScience 2002* (2002), M. J. Egenhofer and D. M. Mark, Eds., Springer Verlag, pp. 209–227.

[44] van Horn, E. K. Generalizing cartographic data bases. In *Auto Carto* (1985), vol. 7, pp. 532–540.

[45] Visvalingam, M., and Whyatt, J. D. The Douglas-Peucker algorithm for line simplification: re-evaluation trough visualization. *Computer Graphics Forum 9*, 3 (1991), 213–228.

[46] Visvalingam, M., and Whyatt, J. D. Line generalisation by repeated elimination of points. *Cartographic Journal 30*, 1 (1993), 46–51.

[47] Wang, Z., and Müller, J. C. Line generalization based on analysis of shape characteristics. *Cartography and Geographic Information Systems 22*, 4 (1998), 264–275.

[48] Ware, J. M., and Jones, C. B. Conflict reduction in map generalization using iterative improvement. *GeoInformatica 2*, 4 (1998), 383–407.

[49] Ware, J. M., Jones, C. B., and Thomas, N. Automated map generalization with multiple operators: a simulated annealing approach. *International Journal of Geographic Information Science 17*, 8 (2003), 743–769.

[50] Weibel, R. *Algorithmic Foundations of Geographic Information Systems*. Springer Verlag, 1997, ch. Generalization of Spatial Data: Principles and Selected Algorithms, pp. 99–152.

[51] Weibel, R., and Dutton, G. *Geographical Information Systems*, vol. 1. John Wiley & Sons, Inc., 1999, ch. Generalising spatial data and dealing with multiple representations, pp. 125–155.

[52] White, E. R. Perceptual evaluation of line generalisation algorithms. Master's thesis, University of Oklahoma, 1983.

[53] Williams, R. Preserving the area regions. *Computer Graphics Forum 6* (1987), 43–48.

[54] Wu, S.-T., da Silva, A. C. G., and Márquez, M. R. G. The Douglas-Peucker algorithm: sufficient conditions for non-self-intersecting. *Journal of the Brazilian Computer Society 9*, 3 (2004), 67–79.

[55] Wu, S.-T., and Márquez, M. R. G. A non-self-intersection Douglas-Peucker algorithm. In *The 16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)* (2003), IEEE Press, pp. 60–66.

[56] Zhang, Y.-Y. A modified Douglas-Peucker simplification algorithm. Master's thesis, The Ohio State University, 1996.

[57] Zhou, M., and Bertolotto, M. Efficiently generating multiple representations for web mapping. In *The 5th International Workshop on Web and Wireless Geographical Information Systems* (2005), K.-J. Li and C. Vangenot, Eds., Springer Verlag, pp. 54–65.

[58] Zhou, S., and Jones, C. B. A multi-representation spatial data model and database. In *The 8th International Symposium on Spatial and Temporal Databases* (2004), T. Hadzilacos and Y. T. Y. Manolopoulos, J. F. Roddick, Eds., Springer Verlag, pp. 394–411.

Apêndice A

Adaptação de ASIs à Produção de Mapas Independentes de Escala

[1]O termo independente de escala refere-se a um mapa cujos vértices das linhas são rotulados com valores de tolerância, conforme ilustra a Figura A.1. Estes valores correspondem normalmente às métricas calculadas por um ASI extrator para estes vértices. Eles podem ser utilizados em um processo de filtragem para a obtenção eficiente de uma escala desejada. Por exemplo, quando uma escala é requerida a uma tolerância ϵ, é suficiente percorrer os vértices das linhas, buscando-se aqueles com valor de tolerância maior ou igual a ϵ. O resultado desta filtragem é um mapa simplificado cujas linhas são exatamente as mesmas que as do mapa original processado pelo ASI à tolerância ϵ. Isto porque este ASI estabelece uma ordem bem definida para os vértices das linhas, baseada na sua métrica geométrica. Este apêndice descreve como os algoritmos RDP e de Visvalingam podem ser adaptados à produção de mapas independentes de escala.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}
∞	ϵ_2	ϵ_3	ϵ_4	ϵ_5	ϵ_6	ϵ_7	ϵ_8	ϵ_9	ϵ_{10}	ϵ_{11}	ϵ_{12}	ϵ_{13}	ϵ_{14}	ϵ_{15}	∞

Figura A.1: Armazenamento da linha $\mathcal{L} = v_1 v_2 \cdots v_{16}$ sob a forma de um vetor em que cada vértice é rotulado com um valor de tolerância especificado pelo ASI.

[1]É interessante ler, antes deste apêndice, a Seção 2.1, que introduz notações e conceitos que serão utilizados, e a Seção 2.2, que descreve os algoritmos que serão discutidos.

A.1 Adaptação do Algoritmo RDP

O algoritmo RDP adaptado à produção de mapas independentes de escala é equivalente ao RDP clássico (veja Seção 2.2.1) executado a uma tolerância ϵ nula. No entanto, em vez de gerar uma linha simplificada $\mathcal{L}'$, ele apenas rotula os vértices da linha original $\mathcal{L} = v_1 v_2 \cdots v_n$, para que eles possam ser filtrados posteriormente. O algoritmo inicia o processo avaliando o segmento $\overline{v_1 v_n}$ e só pára quando avalia todos os segmentos gerados durante o processo de rotulação dos vértices. Para cada segmento $\overline{v_i v_j}$ avaliado, ele encontra o vértice v_k $(i < k < j)$ mais distante de $\overline{v_i v_j}$, rotula-o com a sua distância ao segmento $\overline{v_i v_j}$, que é armazenada em ϵ_k, e continua o processo recursivamente para os novos segmentos $\overline{v_i v_k}$ e $\overline{v_k v_j}$. O Algoritmo 2 ilustra este procedimento.

Procedimento simplificarRDP(**var** $\mathcal{L}$: **Linha**; i, j: **Inteiro**; ϵ_{ant}: **Real**)

01 **Var**
02   k, l: **Inteiro**;
03   d_{atual}, d_{max}: **Real**;
04 **Início**
05   $d_{max} \leftarrow 0$;
06   **Para** $l \leftarrow i + 1$ **até** $j - 1$ **faça**
07     $d_{atual} \leftarrow$ distância entre $\overline{v_i v_j}$ e v_l;
08   **Se** $d_{atual} > d_{max}$ **então**
09       $d_{max} \leftarrow d_{atual}$;
10       $k \leftarrow l$;
11   **Fim se**
12   **Fim para**
13   $\epsilon_k \leftarrow$ menor entre d_{max} e ϵ_{ant};
14   **Se** $d_{max} > 0$ **então**
15   simplificarRDP($\mathcal{L}$, i, k, ϵ_k);
16   simplificarRDP($\mathcal{L}$, k, j, ϵ_k);
17   **Fim se**
18 **Fim**

Algoritmo 2: Adaptação do RDP à produção de mapas independentes de escala.

O trecho da linha 05 à linha 12 encontra o vértice v_k mais distante do segmento $\overline{v_i v_j}$ e armazena esta distância em d_{max}. Suponha que, após avaliar este segmento, o algoritmo avalie o segmento $\overline{v_i v_k}$ e encontre o vértice v_l $(i < l < k)$ como o mais distante. É possível que a distância de v_l a $\overline{v_i v_k}$ seja maior do que a distância v_k a $\overline{v_i v_j}$. Neste caso, ϵ_l seria maior do que ϵ_k, fazendo com que v_l fosse escolhido erroneamente antes de v_k. Para que isto não aconteça, a tolerância atribuída a um vértice nunca deve ser maior que a do seu antecessor. O algoritmo, então, sempre rotula o vértice atual com o valor menor entre a distância calculada d_{max} e a tolerância anterior ϵ_{ant}, conforme a linha `13`. A condição de parada $d_{max} > 0$, dada na linha `14`, evita que a recursividade seja chamada para os casos degenerados, em que $j = i + 1$, nos quais não há vértices intermediários.

A.2 Adaptação do Algoritmo de Visvalingam

O algoritmo de Visvalingam, na realidade, foi originalmente concebido para produzir mapas independentes de escala. A Seção 2.2.2, no entanto, descreve uma versão do algoritmo que simplifica a linha original $\mathcal{L} = v_1 v_2 \cdots v_n$ da maneira usual, removendo alguns dos seus vértices e gerando uma linha simplificada $\mathcal{L}'$. O algoritmo de Visvalingam original é bastante similar a esta versão executada a uma tolerância ϵ infinita. Primeiramente, ele determina a área efetiva de cada vértice v_i de $\mathcal{L}$, isto é, a área do triângulo $\triangle v_{i-1} v_i v_{i+1}$, que é armazenada no rótulo ϵ_i. Em seguida, ele inicia um processo iterativo de atualização destes rótulos. A cada iteração, ele encontra o vértice v_i de menor área efetiva, marca-o, para que ele não seja reprocessado, e atualiza as áreas efetivas dos seus dois vizinhos. O Algoritmo 3 ilustra este procedimento.

O trecho da linha `05` à linha `07` determina as áreas efetivas dos vértices de $\mathcal{L}$. O algoritmo de Visvalingam, assim como o algoritmo RDP adaptado, também deve tratar os casos em que as tolerâncias dos vértices não seguem a ordem imposta pelo algoritmo. A diferença é que, como o algoritmo de Visvalingam é decremental, alguns vértices podem ter área efetiva menor do que a dos marcados antes dele. Para contornar esta situação, o algoritmo verifica se a tolerância ϵ_i do vértice atual é menor que a anterior ϵ_{ant}. Se for, ele rotula este vértice com ϵ_{ant}, conforme o trecho da linha `11` à linha `13`. Quando o algoritmo marca v_i na linha `14`, ele apenas indica que, nas iterações seguintes, este vértice não deverá ser reprocessado. Os vizinhos de v_i, referenciados na linha `15`, são os dois vértices mais próximos a ele que ainda não foram marcados.

Procedimento simplificarVisvalingam(**var** $\mathcal{L}$: **Linha**)

Var
  i: **Inteiro**;
  ϵ_{ant}: **Real**;
Início
  Para $i \leftarrow 2$ **até** $n - 1$ **faça**
    $\epsilon_i \leftarrow$ área do triângulo $\triangle v_{i-1} v_i v_{i+1}$;
  Fim para
  $\epsilon_{ant} \leftarrow 0$;
  Enquanto houver vértices não marcados em $\mathcal{L}$ **faça**
    Encontre v_i com menor área efetiva;
    Se $\epsilon_i < \epsilon_{ant}$ **então**
      $\epsilon_i \leftarrow \epsilon_{ant}$;
    Fim se
    Marque v_i;
    Atualize as áreas efetivas dos vizinhos de v_i;
    $\epsilon_{ant} \leftarrow \epsilon_i$;
  Fim Enquanto
Fim

Algoritmo 3: Procedimento original da simplificação de Visvalingam, já concebido para a produção de mapas independentes de escala.