Elementos de SIG

Geometria computacional para SIG

Clodoveu Davis

Estruturas de dados em informática

- Organização da memória do computador
- Memória primária x secundária
- Memória virtual
- Endereçamento de memória
- Apontadores

Memória

Conceitos

"Ideally, one would desire an indefinitely large memory capacity such that any particular (...) word would be immediately available. (...) We are (...) forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible."

A. W. Burks, H. H. Goldstine and J. von Neumann

Preliminary Discussion of the Logical Design of an Electronic Computing Instrument (1946)

Hierarquia de memórias

Conceitos

- Programadores e usuários querem ter acesso a grandes volumes de memória
- Uma solução econômica para isso é a hierarquia de memórias, que aproveita o *princípio da localidade*
- Cada nível da hierarquia é menor, mais rápido e mais caro que o nível anterior
- O objetivo é construir um sistema com custo quase tão baixo quanto o da memória mais barata e quase tão rápido quanto a memória mais rápida

Hierarquia de memórias

Conceitos

- Cada nível da hierarquia é mapeado para o inferior, sendo que o último nível deve conter sempre todos os dados relevantes
- O ideal é ter um sistema de endereçamento único para todos os níveis
- A velocidade da memória não tem crescido tão rápido quanto a velocidade dos processadores

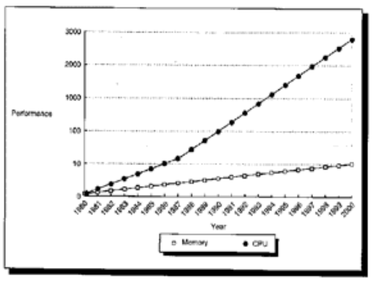


FIGURE 5.1 Starting with 1980 performance as a baseline, the performance of memory and CPUs are plotted over time. The memory baseline is 64-KB DRAM in 1980, with three years to the next generation and a 7% per year performance improvement in latency (see Figure 5.30 on page 429). The CPU line assumes a 1.35 improvement per year until 1986, and a 1.55 improvement thereafter. Note that the vertical axis must be on a logarithmic scale to record the size of the CPU-DRAM performance gap.

Hierarquia de memórias

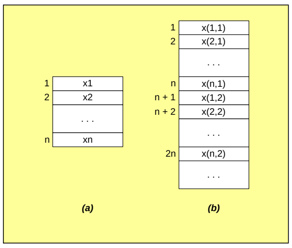
Conceitos

- Níveis da hierarquia (~2005):
  - registradores: 512-1024 bytes, na CPU (1-2 ns)
  - cache primária: 8-16 KB, no chip da CPU (5 ns)
  - cache secundária: 128-512 KB, também no chip da CPU (Pentium IV / Athlon64) (10-15 ns)
  - memória principal: 256 MB-2 GB (30-40 ns) ($0,20/MB)
  - memória em disco: 20-300 GB (5-10 ms) ($0,0016/MB)
  - memória em fita (100 ms e acima) / CD-ROM / DVD-ROM
  - rede (? ms)

Estruturas de dados em informática

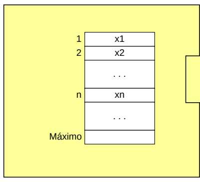
- Vetores, matrizes, arranjos
- Listas lineares
- Filas
- Pilhas
- Grafos
- Redes
- Árvores
- Objetos
  - classes, instâncias
  - hierarquia, herança
  - encapsulamento
  - polimorfismo

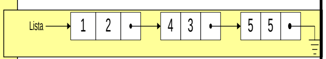
Vetores e matrizes



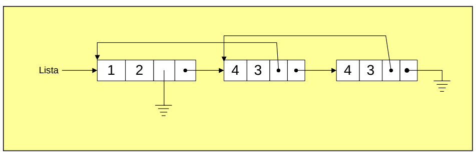
Listas lineares

- Implementadas como arranjo
- Implementadas com apontadores

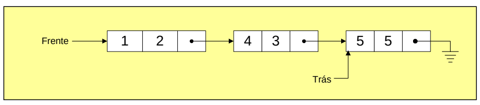




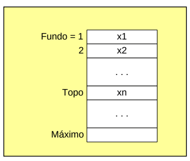
Listas lineares duplamente encadeadas



Fila

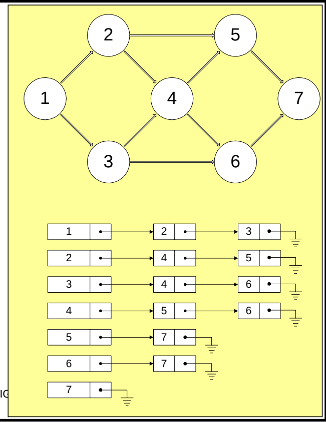


Pilha



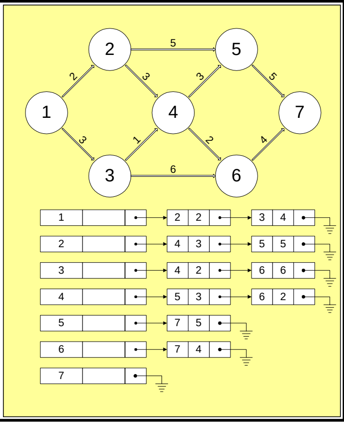
Grafos

- Conjuntos de arcos e nós
- Os arcos podem ser direcionados ou não
- A cada arco correspondem *sempre* dois nós



Redes

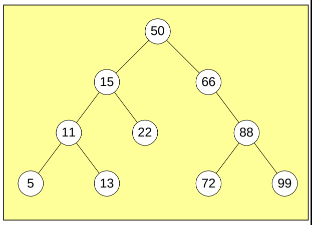
- Redes são grafos com custos ou pesos associados aos arcos
- Utilização intensiva em problemas de otimização



Árvores

- Uma árvore é um *grafo acíclico não direcionado*
- Raiz, folhas
- Propriedades:
  - grau (de cada nó)
  - profundidade
  - altura

- Árvore binária



Algoritmos básicos em SIG

- Algoritmos básicos: triângulos, áreas, detecção de interseções
- Simplificação de poligonais
- Cálculo de centróide
- Ponto em polígono
- Diagrama de Voronoi
- Triangulação de Delaunay
- Caminho mínimo em redes

Tipos abstratos para dados vetoriais

- Definem estruturas de memória para comportar as três entidades geométricas básicas para a representação vetorial
- É acrescentada uma estrutura para segmentos de reta
- É acrescentada uma estrutura para o retângulo, útil em várias situações

Tipos abstratos para dados vetoriais

```
estrutura Ponto
início
   inteiro x;
   inteiro y;
fim;

estrutura Segmento
início
   Ponto p1;
   Ponto p2;
fim;

estrutura Retângulo
início
   inteiro x1;
   inteiro y1;
   inteiro x2;
   inteiro y2;
fim;

estrutura Poligonal
início
   inteiro numPontos;
   Retângulo
   retânguloEnvolventeMínimo;
   Ponto[] vertice;
fim;
```

Problemas de precisão numérica

- A representação de números no computador é limitada (finita), restrita a um dado número de bits, mesmo quando são usadas variáveis reais
- Para minimizar o problema, alguns SIG representam coordenadas com valores inteiros, fixando o número de casas decimais
- Ainda assim podem ocorrer problemas de *overflow*

Problemas de precisão numérica

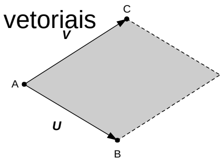
- Coordenadas UTM
  - em cada fuso, N varia de 0 a 10.000.000, e E varia de 0 a 1.000.000
  - não há valores negativos
- Inteiros de 32 bits
  - não-negativos: variam de 0 a 4.294.967.295
  - com sinal: variam de -2.147.483.647 a +2.147.483.647
- Codificação
  - usar “fator de precisão” fixo, ex. 100
  - correspondencia entre unidades internas e unidades reais
  - 0 a 42.949.672,95 é mais que suficiente para a UTM, com precisão de centímetro

Problemas de precisão numérica

- Produtos entre valores de coordenadas podem levar a overflow
- Usar números reais não resolve o problema e ainda introduz erros de arredondamento
- Foi proposta a geometria computacional de precisão finita para resolver o problema, mas ainda não está presente em SIG comerciais

Algoritmos básicos

- Triângulos e produtos vetoriais



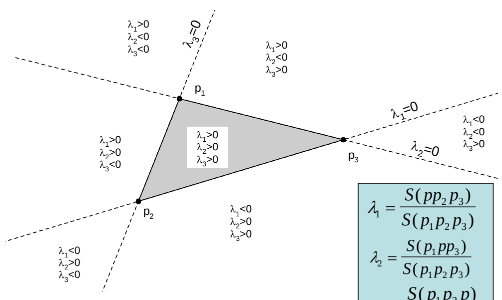
$$|U \times V| = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ x_U & y_U & z_U \\ x_V & y_V & z_V \end{vmatrix} = (y_U z_V - z_U y_V)\hat{i} + (z_U x_V - x_U z_V)\hat{j} + (x_U y_V - y_U x_V)\hat{k}$$

$$S = \frac{(x_U y_V - y_U x_V)}{2} \Rightarrow S = \frac{1}{2}(x_A y_B - y_A x_B + y_A x_C - x_A y_C + x_B y_C - y_B x_C)$$

Triângulos

```
função áreaOrientadaTriângulo(Ponto A, Ponto B, Ponto C):
   real

início
   retorne ((A.x*C.y - A.y*C.x + A.y*B.x - A.x*B.y +
                 C.x*B.y - C.y*B.x) / 2);
fim;


função áreaTriângulo(Ponto A, Ponto B, Ponto C): real

início

   retorne abs(áreaOrientadaTriângulo(A, B, C));
fim.
```


Coordenadas baricêntricas

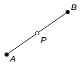


$$\lambda_1 = \frac{S(pp_2p_3)}{S(p_1p_2p_3)}$$

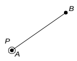
$$\lambda_2 = \frac{S(p_1pp_3)}{S(p_1p_2p_3)}$$

$$\lambda_3 = \frac{S(p_1p_2p)}{S(p_1p_2p_3)}$$

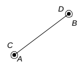
Pontos e segmentos



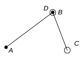
`em(P, AB)` — Ponto é interior ao segmento; pontos extremos são excluídos.



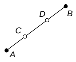
`emExtremo(P, AB)` — Ponto coincide com um ponto extremo do segmento



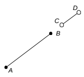
`iguais(AB, CD)` — Ambos os pontos extremos coincidem



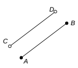
`seEncontram(AB, CD)` — Compartilham exatamente um ponto extremo



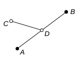
`superpostos(AB, CD)` — São colineares e compartilham um trecho comum



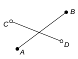
`alinhados(AB, CD)` — São colineares e não têm ponto em comum



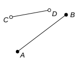
`paralelos(AB, CD)` — Têm a mesma inclinação e não são iguais nem superpostos



`seTocam(AB, CD)` — Não são superpostos e um dos pontos extremos de um segmento pertence ao outro segmento

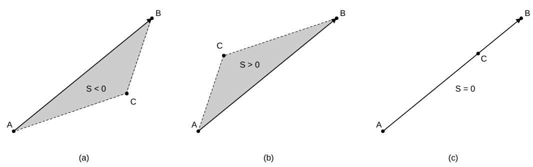


`seInterceptam(AB, CD)` — Têm um ponto em comum e não se encontram nem se tocam



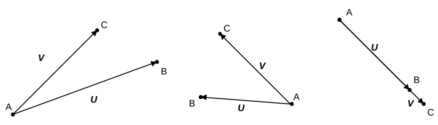
`disjuntos(AB, CD)` — Não são iguais, nem se encontram, nem se tocam, nem são paralelos, nem se interceptam, nem se superpõem

Ponto e segmento orientado



À direita | À esquerda | Alinhado

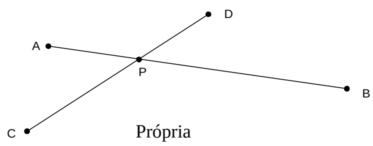
Posicionamento relativo entre vetores



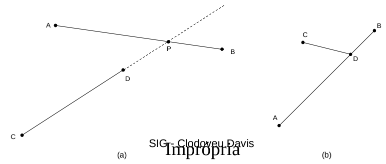
(a) $U \times V > 0$ | (b) $U \times V < 0$ | (c) $U \times V = 0$

U->V anti-horário | U->V horário | U e V alinhados

Interseção de segmentos

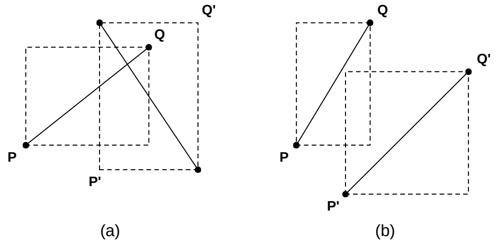


Própria

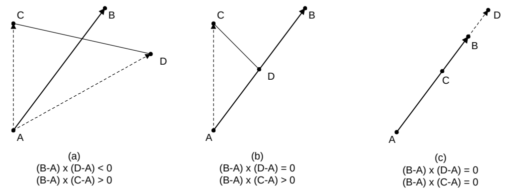


Imprópria

Interseção entre retângulos envolventes mínimos



Verificação da interseção



Simplificação de poligonais

- Muitas entidades do mundo real podem ser modeladas como poligonais
- Em BD geográficos, compõem cerca de 80% do volume de dados vetoriais
- A complexidade de representação varia de acordo com a geomorfologia e com o processo de captura de dados
- Existe a necessidade de visualização com menor detalhamento

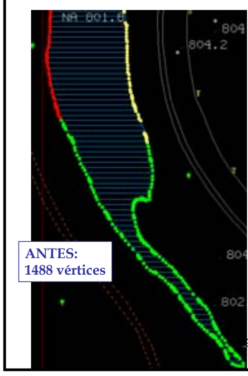
Definições

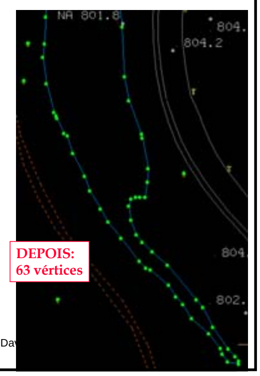
- **Poligonal:** coleção finita de segmentos de reta no plano, formando uma *curva simples*
  - *Curva:* segmentos conectados seqüencialmente
  - *Simples:* não há interseções de segmentos não adjacentes
- **Vértices:** pontos extremos dos segmentos
- Codificada tipicamente como um vetor de vértices

Caracterização do Problema

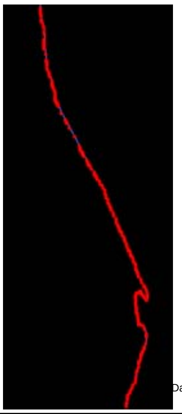
- Produzir uma representação *aproximada* de uma poligonal dada, que seja *mais compacta* (mais grosseira, formada por menos vértices) que a original
- Finalidades:
  - Eliminar excesso de vértices (*weed*)
  - Generalização: gerar representações utilizáveis em escalas menores
    - Evitar redundância no banco de dados geográfico
- Manter uma forma gráfica aceitável

Simplificação de linhas

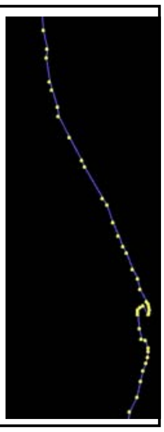




RIO:
786 vértices
(390 metros)



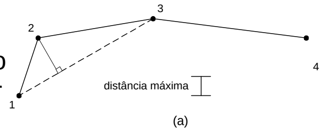
Corrigido:
44 vértices



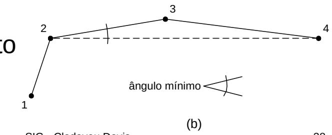
Davis

Medidas de Proximidade

- Máximo deslocamento perpendicular



- Mínimo deslocamento angular



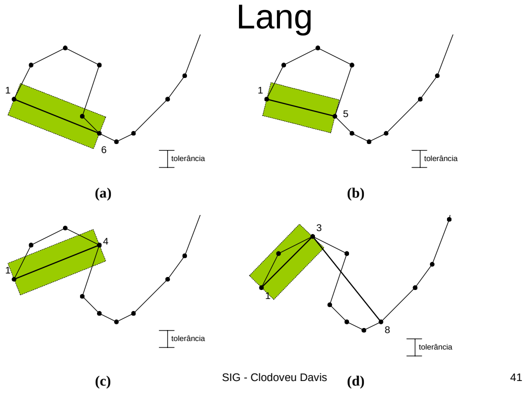
Algoritmos Hierárquicos e Não-Hierárquicos

- O resultado da simplificação pode ser:
  - uma poligonal formada por um subconjunto dos vértices da poligonal original, ou
  - uma poligonal formada por vértices distintos dos que formam a poligonal original, à exceção do primeiro e do último
- No primeiro caso, se a redução da tolerância causa apenas a inclusão de novos vértices, o algoritmo é dito *hierárquico*

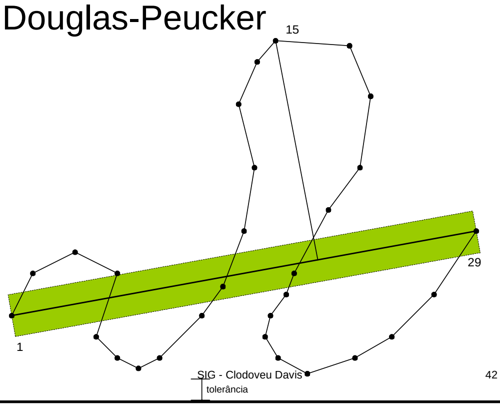
Principais Algoritmos

- *k*-ésimo vértice
- Vértice aleatório
- Jenks
- Reumann-Witkam / Opheim
- Lang
- Douglas-Peucker
- Zhao-Saalfeld
- Visvalingam-Whyatt

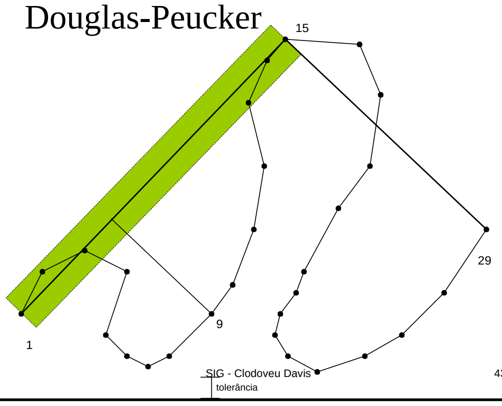
Lang



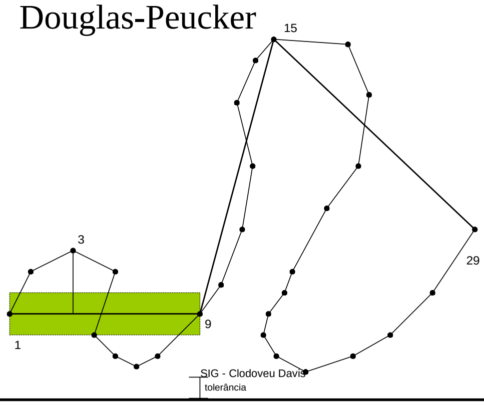
Douglas-Peucker



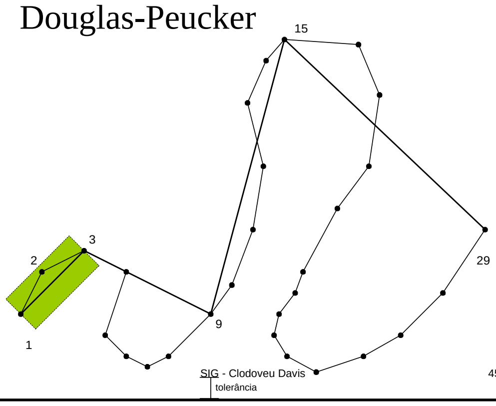
Douglas-Peucker



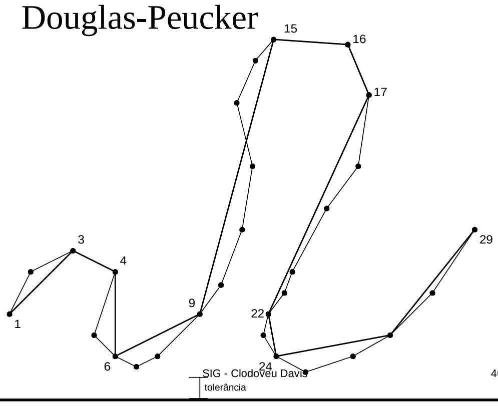
Douglas-Peucker



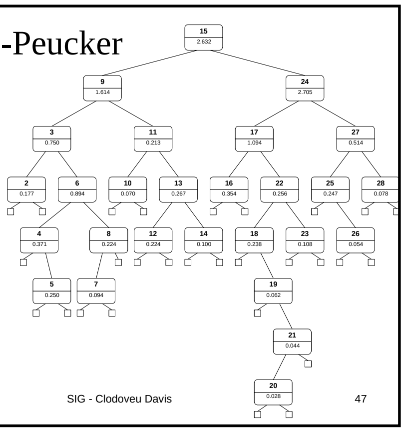
Douglas-Peucker



Douglas-Peucker



Douglas-Peucker



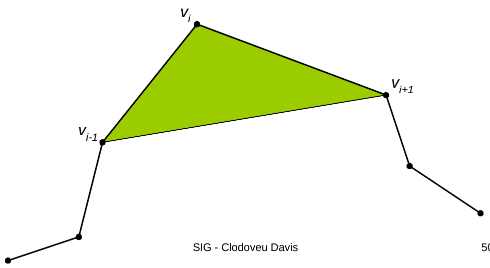
Douglas-Peucker

- Reconhecidamente o melhor em termos de preservação das características da poligonal original
- Unanimidade no processamento (*weeding*) pós-digitalização
- Comportamento geométrico inadequado em simplificação mais radical
- O(*n log n*) no melhor caso; O(n^2) no pior caso, O(*n log n*) se usada a técnica de *path hull*
- Paralelizável

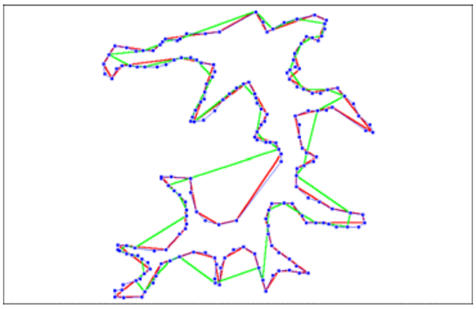
Visvalingam-Whyatt

- Eliminação progressiva do vértice ao qual corresponde a menor *área efetiva* em cada passo
- Forma um *ranking* dos vértices que pode ser organizado em uma fila de prioridades
- Eliminação de “cantos”
- Indefinição do critério de parada (tolerância)

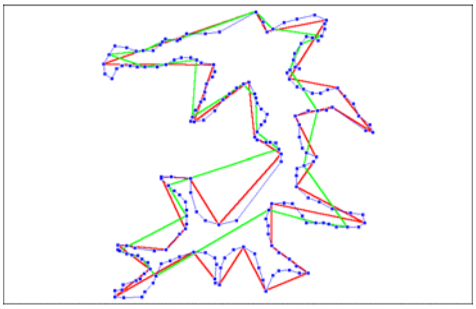
Visvalingam-Whyatt



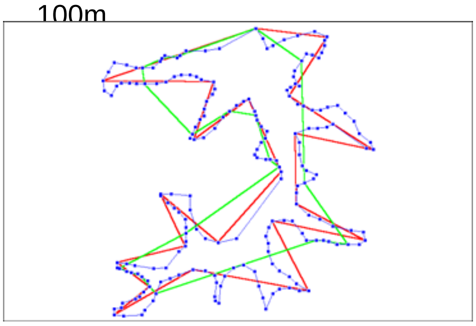
DP x VW, 20m



DP x VW, 50m

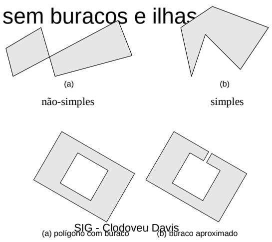


DP x VW, 100m



Geometria de polígonos

- Propriedades básicas de polígonos
  - curva fechada simples
  - com ou sem buracos e ilhas



Cálculo de área

- Baseada na divisão em triângulos
  - resultado positivo: vértices em sentido anti-horário
  - resultado negativo: vértices em sentido horário
  - a fórmula funciona com buracos e ilhas codificados corretamente

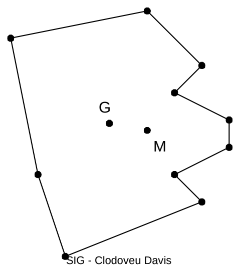
$$A(P) = \frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - y_i x_{i+1})$$

Determinação do centróide

- Centróide = centro de gravidade = centro de massa
- Alguns SIG consideram como centróide qualquer ponto interior ao polígono
- Em alguns casos, o centróide real pode ficar fora do polígono, o que é indesejável como apoio à representação
- Alguns SIG calculam o centróide simplesmente pela média das coordenadas dos vértices
- Outras alternativas: centro do retângulo envolvente mínimo, centro de um círculo circunscrito.

Determinação do centróide

- Cálculo pela média dos vértices, problema



Determinação do centróide

- Centróide de um triângulo

$$x_G = \frac{x_A + x_B + x_C}{3} \qquad y_G = \frac{y_A + y_B + y_C}{3}$$

- No polígono: como no cálculo da área, formar triângulos entre cada segmento e um ponto fixo, então usar sinais negativos e ponderações pela distância para obter o centróide

Determinação do centróide

- Formulação completa

$$A(P) = \frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - y_i x_{i+1})$$

$$x_C = \frac{\sum_{i=0}^{n-1}(x_{i+1} + x_i) \times (x_i y_{i+1} - y_i x_{i+1})}{3A(P)}$$

$$y_C = \frac{\sum_{i=0}^{n-1}(y_{i+1} + y_i) \times (x_i y_{i+1} - y_i x_{i+1})}{3A(P)}$$

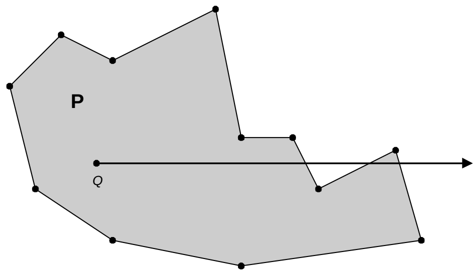
Determinação do centróide

- Problema: quando o centróide cai fora do polígono
  - detectar essa situação usando o teste de ponto em polígono
  - determinar uma diagonal do polígono (segmento entre vértices não-consecutivos que não cruza nenhuma aresta)
  - usar o ponto médio dessa diagonal

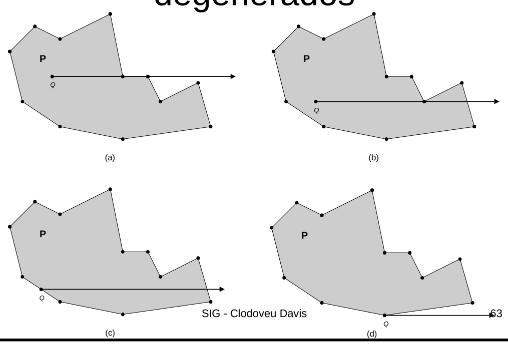
Ponto em polígono

- Talvez o problema resolvido mais freqüentemente na operação de SIG
- Algoritmo sensível a casos especiais e situações limite, precisa ser muito bem implementado para garantir o funcionamento correto
- Princípio:
  - traçar a partir do ponto dado, uma semi-reta em qualquer direção
  - se a semi-reta interceptar a fronteira do polígono um número par de vezes, o ponto está **fora**; do contrário, o ponto está **dentro** do polígono

Ponto em polígono



Ponto em polígono - casos degenerados

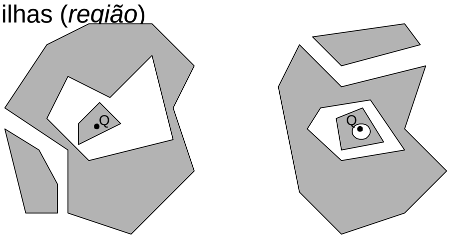


Ponto em polígono

- Para estes casos, a solução está em adotar um critério para a contagem de interseções de modo que:
  - se a reta passa por um vértice, a interseção deve ser considerada apenas se for o vértice com maior ordenada do segmento, e ignorada caso contrário;
  - se a reta passa por um segmento do contorno do polígono, nenhuma interseção deve ser considerada;
  - se o ponto *Q* pertence a um segmento do contorno (exceto pontos extremos), considerar como uma interseção.

Ponto em polígono

- Caso especial: polígono com buracos ou ilhas (*região*)



(a) (b)

Ponto em região

- Ponto contido em apenas um polígono
  - neste caso, o polígono só poderá ser uma ilha, e portanto o ponto está dentro da região
  - caso o polígono seja um buraco, existe erro topológico.
- Ponto contido em mais de um polígono:
  - se o número de ilhas for igual ao número de buracos, o ponto está fora da região
  - se o número de ilhas for maior que o número de buracos, o ponto está dentro da região
  - o caso de se ter o número de buracos superior ao número de ilhas indica um erro topológico.

Interseção, união e diferença de polígonos

- Base de todo o processamento de *polygon overlay*
- Fundamental para análise espacial
- Em alguns SIG, a operação de *polygon overlay* é executada sobre uma estrutura topológica
  - Mesmo assim, no caso de existir a necessidade de geração de buffers, o processamento tem que ser feito geometricamente
- Problemas análogos em computação gráfica
  - recorte de polígonos para visualização 3D
  - *ray-tracing*

Interseção, união e diferença de polígonos

- Teste preliminar: retângulos envolventes mínimos
- Se não for possível que P e Q tenham interseção, então, automaticamente temos

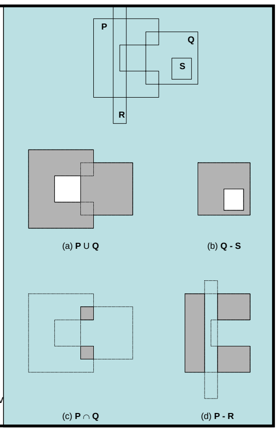
$$P \cup Q = \{P, Q\}$$

$$P \cap Q = \varnothing$$

$$P - Q = P$$

$$Q - P = Q$$

Interseção, união e diferença de polígonos



União, interseção e diferença de polígonos

- Algoritmo Margalitt-Knott
- Permite buracos e ilhas nos dois polígonos
- Orientação inicial do polígono *Q*:

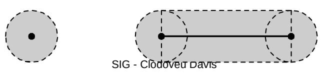
Polígonos		Operações			
P	***Q***	$P \cap Q$	$P \cup Q$	$P - Q$	$Q - P$
ilha	**ilha**	*manter*	*manter*	*inverter*	*inverter*
ilha	**buraco**	*inverter*	*inverter*	*manter*	*manter*
buraco	**ilha**	*inverter*	*inverter*	*manter*	*manter*
buraco	**buraco**	*manter*	*manter*	*inverter*	*inverter*

União, interseção e diferença de polígonos

- Passo 1: normalizar a orientação (tabela anterior)
- Passo 2: classificar os vértices de um polígono em relação ao outro, e vice-versa: dentro, fora, na fronteira
- Passo 3: encontrar as interseções entre arestas dos dois polígonos; formar lista de fragmentos de arestas
- Passo 4: classificar os fragmentos de arestas (interior, exterior, fronteira)
- Passo 5: selecionar e organizar as arestas
- Passo 6: construir os polígonos de resultado

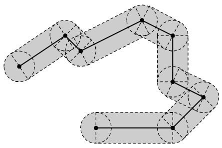
Criação de *buffers*

- *Buffer*: lugar geométrico dos pontos do plano mais próximos que uma determinada distância *d* de um elemento geométrico dado
  - ponto: círculo com centro no ponto e raio *d*
  - segmento de reta: retângulo envolvente limitado e dois semi-círculos
  - linhas e polígonos: união de buffers elementares ao redor de pontos e segmentos de reta

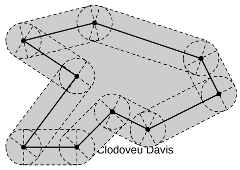


Criação de *buffers*

Ao redor de linha

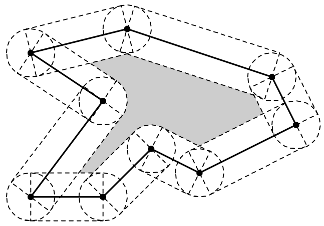


Ao redor de polígono
(*o interior é incluido por definição*)



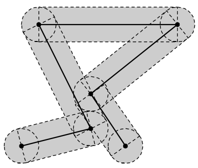
Criação de *buffers*

- Buffer "negativo"

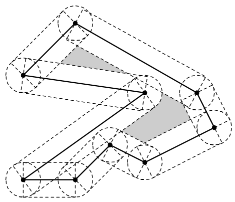


Criação de *buffers*

- Buffers podem produzir polígonos com buracos e/ou descontínuos



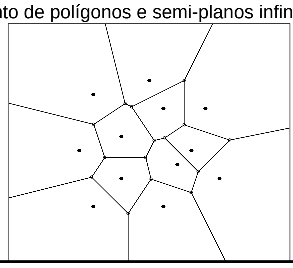
(a)



(b)

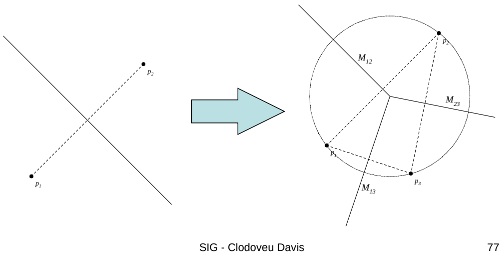
Voronoi e Delaunay

- Diagrama de Voronoi
  - traçado a partir de um conjunto de pontos (*locais*)
  - conjunto de polígonos e semi-planos infinitos



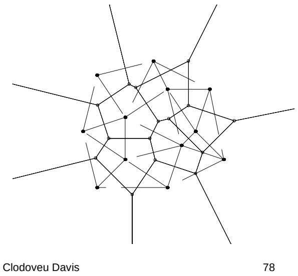
Voronoi

- Construção



Triangulação de Delaunay

- Obtido unindo com segmentos de reta os locais que compartilham uma fronteira
- Uma das alternativas de triangulação preferidas, especialmente para modelagem digital de terrenos

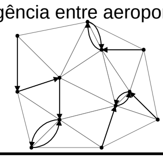


Voronoi e Delaunay: propriedades

- A fronteira de *D* é o fecho convexo do conjunto de locais
- O interior de cada triângulo de *D* não contém nenhum local
- Um polígono de *V* é ilimitado se e somente se ele pertence ao fecho convexo do conjunto dos locais
- Existe uma aresta de *D* entre um ponto P_i e seu vizinho mais próximo P_j
- *D* maximiza o menor ângulo interno dos triângulos, tornando a aparência da triangulação mais regular

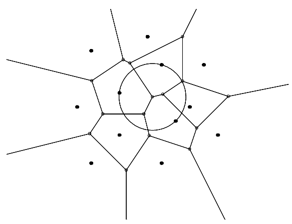
Voronoi e Delaunay: aplicações

- Ponto mais próximo
  - verificar em que polígono de V o ponto cai
  - exemplo: indicar o ponto de ônibus mais próximo de um dado endereço
- Vizinhos mais próximos
  - encontrar o local mais próximo de um local dado
  - exemplo: roteamento de emergência entre aeroportos (grafo de proximidade)



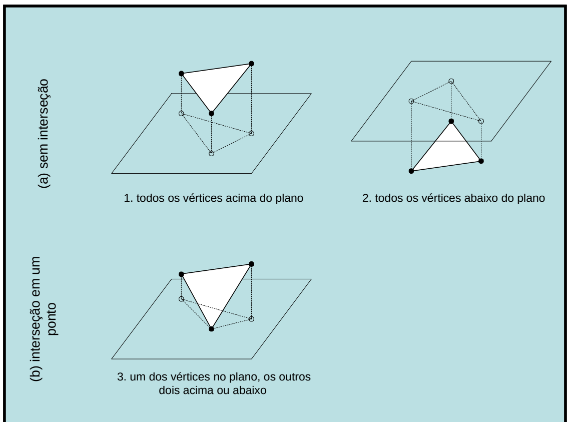
Voronoi e Delaunay: aplicações

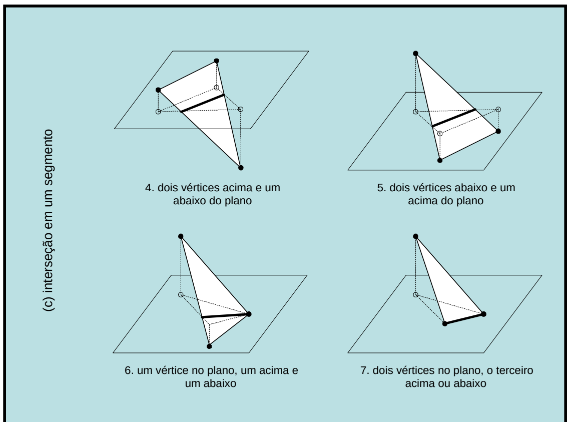
- Maior círculo vazio
  - encontrar o maior círculo contido no fecho convexo dos locais e que não contenha nenhum local
  - exemplo: localização de ponto comercial



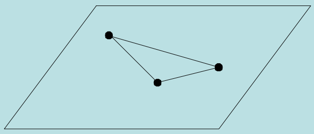
Voronoi e Delaunay: aplicações

- Modelagem digital de terrenos: traçado de isolinhas
- Comparar cada triângulo com o plano em que a isolinha ocorre
- Obter as interseções e fundí-las
  - nenhuma interseção
  - segundo um ponto (vértice)
  - segundo um segmento
    - diagonal
    - bissetriz
    - aresta
  - inserção total





(d) inserção total



8. todos os vértices no plano

Caminho mínimo em redes

- Caminho de menor custo entre dois nós dados, considerando a soma dos custos associados aos arcos percorridos
- Solução: algoritmo de Dijkstra (1959)
  - limite: os custos associados aos arcos não podem ser negativos
  - expansão de busca "cega", estilo borrão de tinta
  - encontra sempre a solução ótima
  - pode ser usado para identificar, em uma só passada, caminhos entre um nó e todos os demais

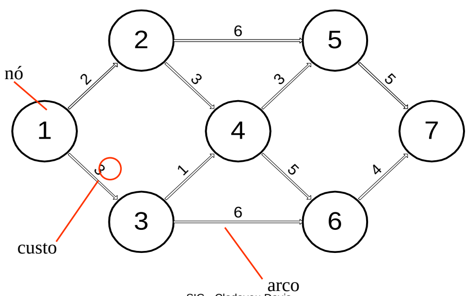
Aplicações de Redes

- Rede viária
  - Nós = cruzamentos
  - Arcos = trechos de via
  - Custos = tempo ou distância percorrida
- Redes de computadores
  - Nós = equipamentos
  - Arcos = conexões
  - Custos = 1 / largura de banda

Aplicações de Redes

- Em SIG:
  - Os nós são em geral elementos simbólicos (cruzamentos, equipamentos, conexões)
  - Os arcos são em geral objetos lineares (trechos de via, trechos de cabeamento, tubulações)
  - Os custos são quaisquer grandezas (distâncias, tempo, perdas, ganhos, despesas) que se acumulem linearmente ao longo do percurso

Exemplo



Variações

- único destino
  - menor caminho de todos os nós até um nó de destino
- única origem
  - menor caminho entre um nó de origem e todos os demais
- origem-destino
  - menor caminho entre dois nós
- todos os pares

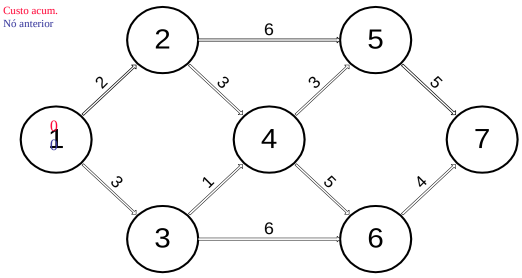
Algoritmo de Dijkstra (1959)

- Aperfeiçoado desde sua introdução pelo uso de estruturas de dados mais avançadas
- Método mais empregado na prática
- Apenas admite custos positivos
- Resolve problemas de única origem
- Pode ser interrompido ao alcançar o destino desejado

Algoritmo de Dijkstra

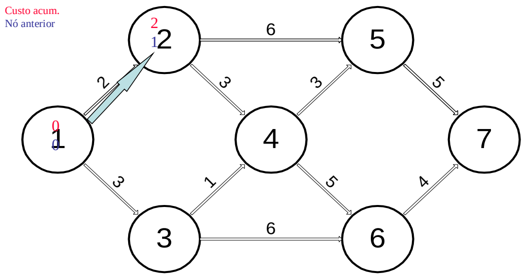
- Dado um grafo *G = (N, A)*, encontrar o caminho mais curto de um vértice de início *o* até um vértice de destino *d* pertencente a *N*
- Estratégia *greedy* ("guloso")
  - a cada iteração é escolhido o vértice mais próximo dos já processados.
- Os nós são divididos em três grupos:
  - já visitados (conjunto *V*)
  - candidatos ou de fronteira (conjunto *F*)
  - nunca visitados ou "desconhecidos" (conjunto *D*).

Caminho mínimo em redes



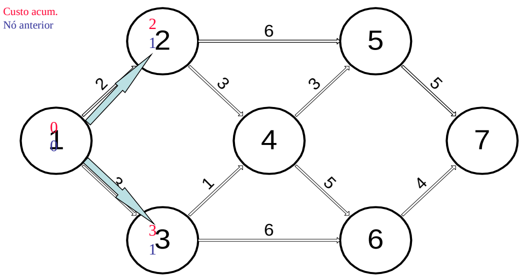
Fronteira: 2 - 3

Caminho mínimo em redes



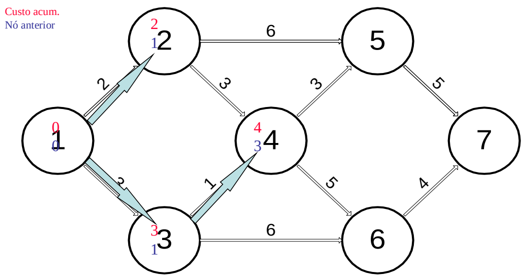
Fronteira: 3 - 4 - 5

Caminho mínimo em redes



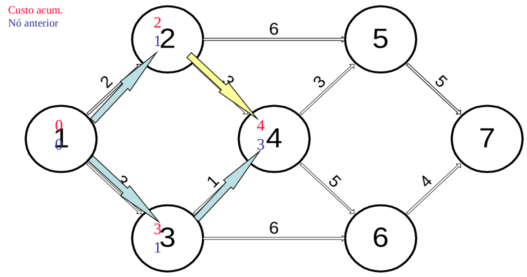
Fronteira: 4 - 5 - 6

Caminho mínimo em redes



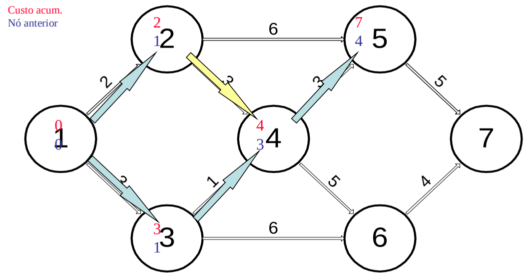
Fronteira: 4 - 5 - 6

Caminho mínimo em redes



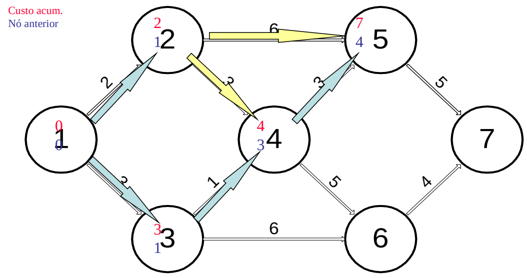
Fronteira: 5 - 6

Caminho mínimo em redes



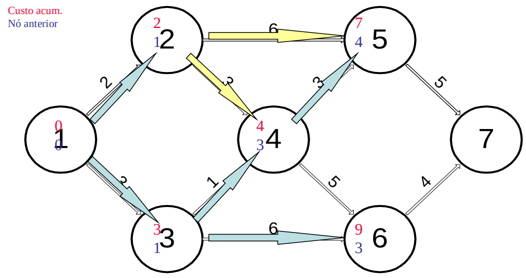
Fronteira: 5 - 6 - 7

Caminho mínimo em redes



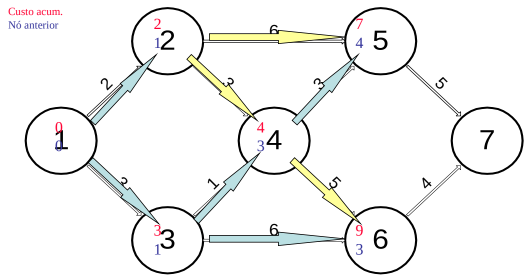
Fronteira: 6 - 7

Caminho mínimo em redes



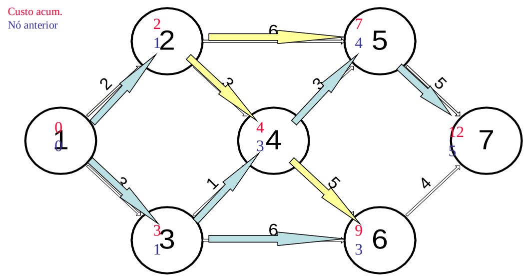
Fronteira: 6 - 7

Caminho mínimo em redes



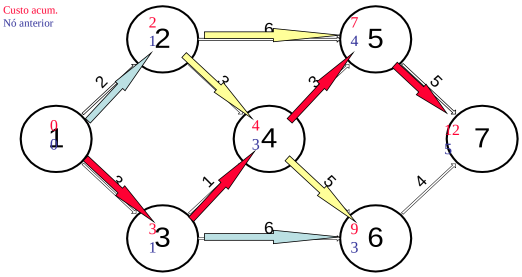
Fronteira: 7

Caminho mínimo em redes



Fronteira: 7

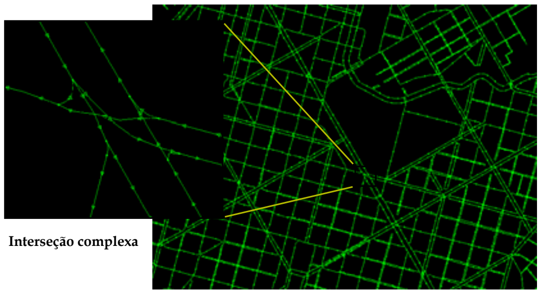
Caminho mínimo em redes



Caminho mínimo: 1-3-4-5-7 (custo 12)

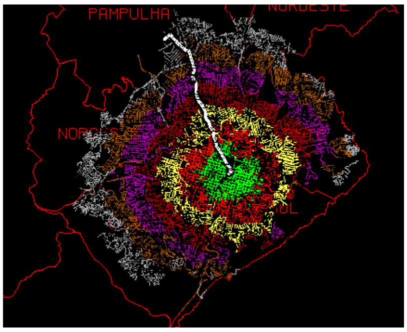
Aplicação em Belo Horizonte

Malha Viária: 35.500 nós e 89.000 arcos



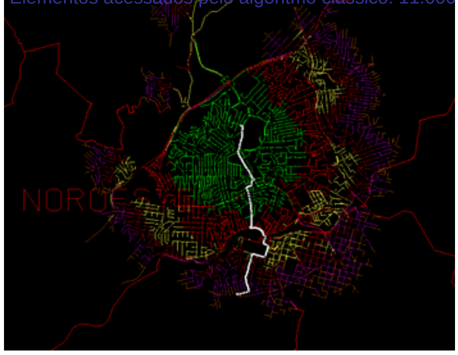
Exemplo 1

Elementos acessados pelo algoritmo clássico: 35.000



Exemplo 2

Elementos acessados pelo algoritmo clássico: 11.000



Comparação de Tempos

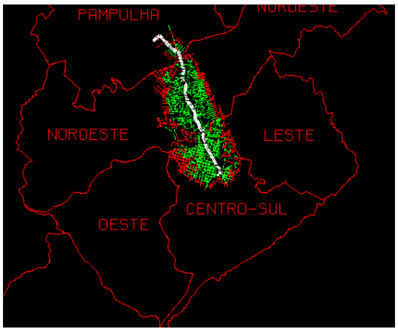
	APIC	Dijkstra Clássico	Fator de Aceleração
Exemplo 1	1580 s	5,8s	270 vezes
Exemplo 2	421 s	1,35s	310 vezes

Aperfeiçoamentos

Heurística de Hart-Nillson-Raphael:

- A escolha do próximo nó privilegia os nós *geograficamente* mais próximos do nó de destino
- Modifica o padrão de pesquisa: de “borrão de tinta” para um padrão direcional
- Reduz significativamente o número de objetos visitados
- Só se aplica a redes com distribuição geográfica

Exemplo 1
Elementos acessados usando a heurística: 7.000



Exemplo 2:
Elementos acessados pela heurística: 2.500

Comparação

	APIC	Dijkstra	Heurística
Exemplo 1	1580 s	5,8s / 270x	1,3s / 1200x
Obj. Visit. 1	N.D.	35.088	7.234 (20,6%)
Exemplo 2	421 s	1,35s / 310x	0,2s / 2100x
Obj. Visit. 2	N.D.	11.062	2.540 (23,0%)

Resultados

- Programa até 400 vezes mais rápido, usando o algoritmo clássico de Dijkstra, em comparação com o tempo de processamento no APIC
- Redução de até 80% no número de objetos visitados com a utilização da heurística de Hart-Nilsson-Raphael
- Diversas aplicações passam a ser viáveis em tempo real, tais como:
  - roteamento de viaturas (emergência e serviços)
  - informações por telefone aos cidadãos e a visitantes
  - otimização de rotas no sistema de transporte coletivo

Leitura complementar

- Davis Jr., C. A. e Queiroz, G. R. *Algoritmos Geométricos e Relacionamentos Topológicos*. Capítulo 2 de Casanova, M. A., Câmara, G., Davis Jr., C. A., Vinhas, L., Queiroz, G. R. (editores) *Bancos de Dados Geográficos*. Ed. MundoGeo, 2005.