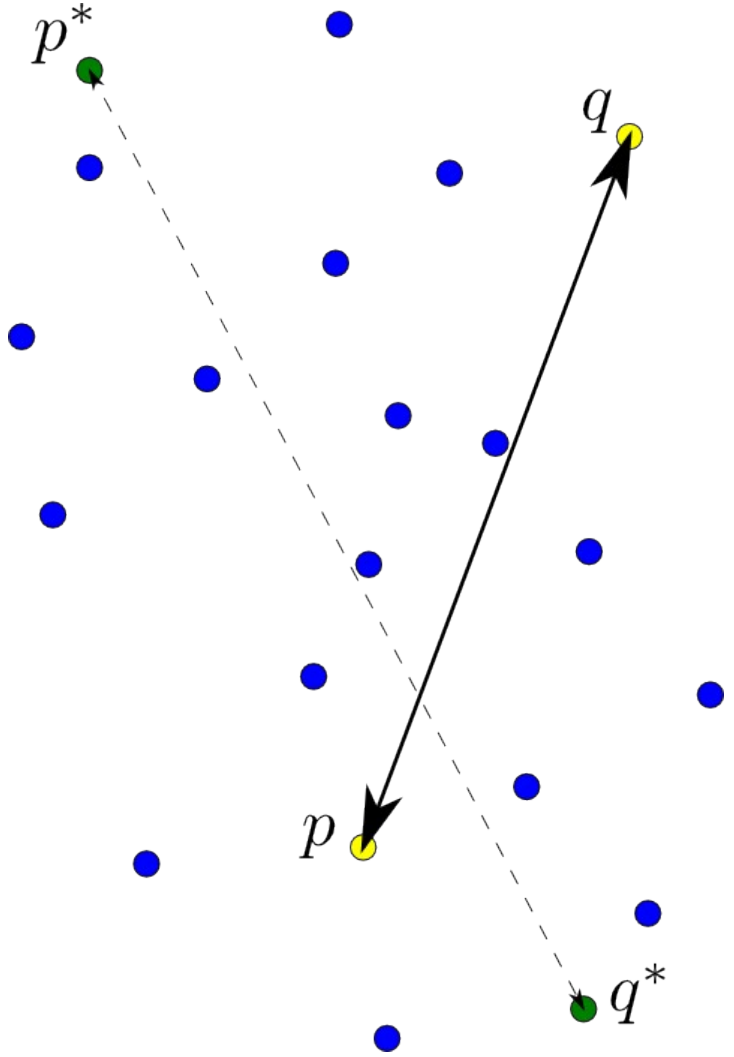
Aproximação Geométrica



Aula 1

Tópicos:

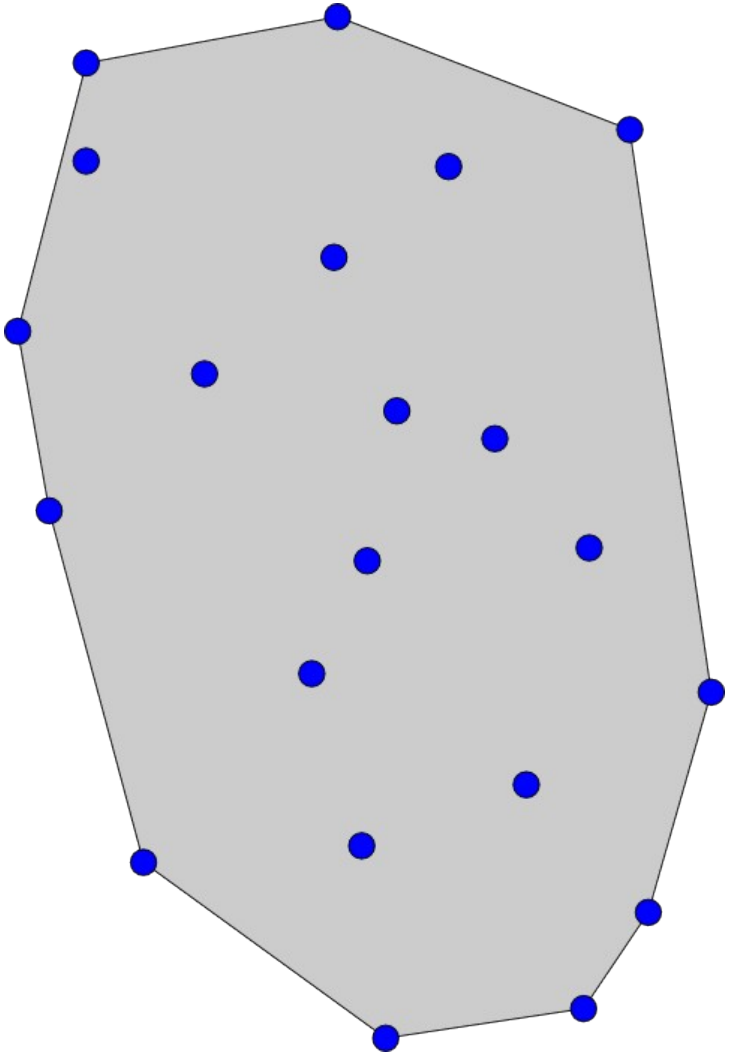
- Introdução
- Diâmetro

IMPA – Verão 2009

Guilherme D. da Fonseca

Geometria Computacional

- Estuda algoritmos e estruturas de dados para problemas geométricos.
- Espaços Euclidianos d-dimensionais, com ênfase em valores "pequenos" de d.
- Utiliza notação O e análise de pior caso para avaliar algoritmos e estruturas de dados.



Notação O

- Notação para desigualdade assintótica.
- Ignora constantes aditivas e multiplicativas.
- Considera n arbitrariamente grande.
- Também considera $1/\varepsilon$ arbitrariamente grande, mas *não* d.

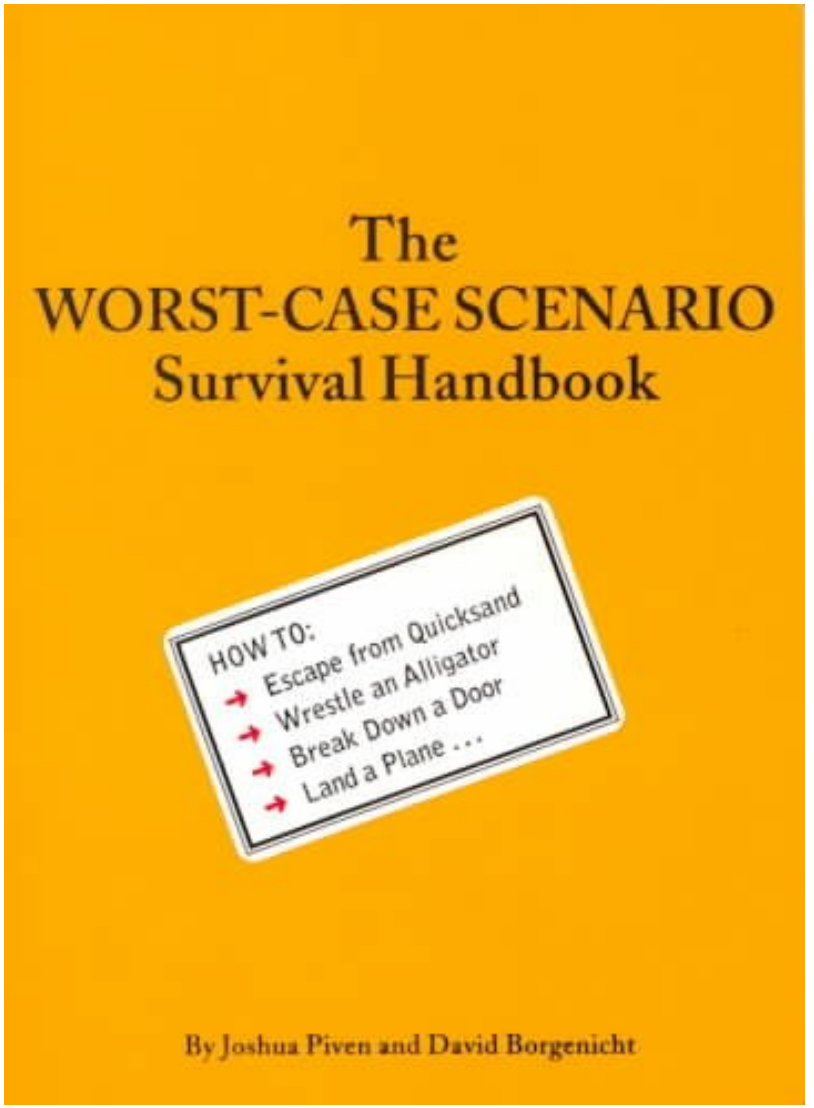
Dizemos que $f(n) = O(g(n))$ se existem constantes reais c e n_0 tal que, para todo $n > n_0$, vale que $f(n) \leq c\ g(n)$.

Dizemos que $f(n) = \Omega(g(n))$ se existem constantes reais c e n_0 tal que, para todo $n > n_0$, vale que $f(n) \geq c\ g(n)$.

Dizemos que $f(n) = \Theta(g(n))$ se $f(n) = \Omega(g(n))$ e $f(n) = O(g(n))$.

Complexidade de Pior Caso

- *Pior caso*: Considera o tempo *máximo* que o algoritmo leva para uma entrada de tamanho n.
- *Caso médio* (pouco usado): Considera o tempo *médio* que o algoritmo leva para uma entrada de tamanho n.
  - Precisa de uma distribuição de probabilidade para a entrada.



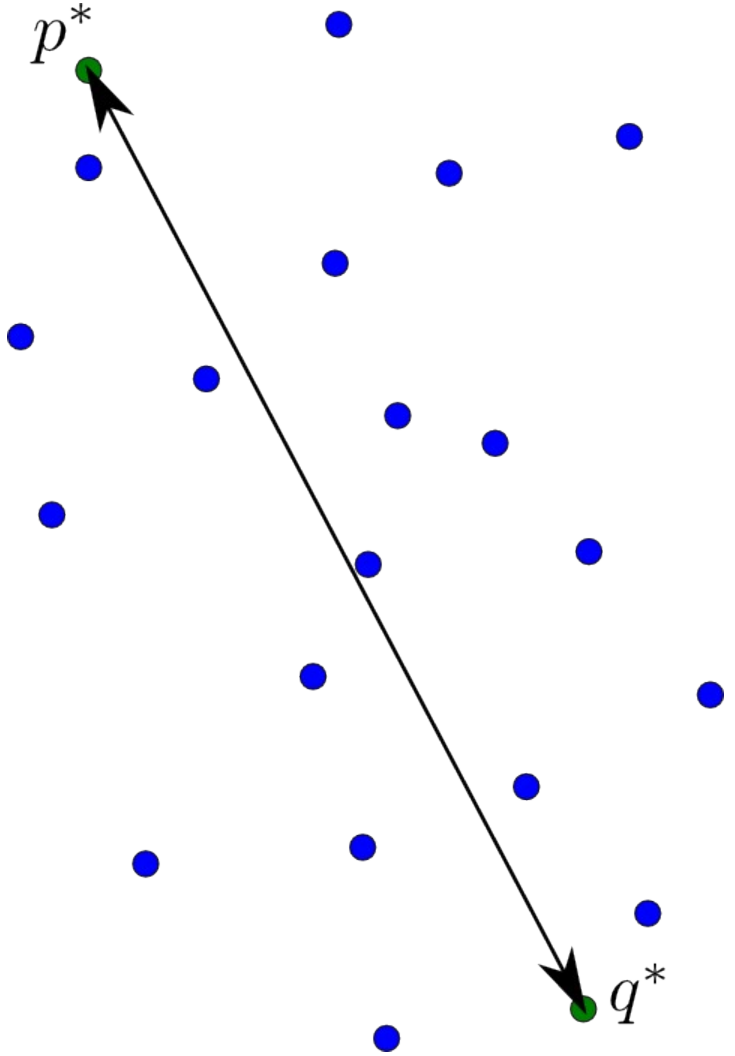
Algoritmos Randomizados

- Não confundir *algoritmos randomizados* com *caso médio*.
- Algoritmo randomizado: a complexidade corresponde a entrada que leva o *maior tempo esperado*.
- A esperança é função dos bits aleatórios usados pelo algoritmo para uma entrada fixada.

Por Que Aproximar?

- Porque é divertido!
- Porque é mais rápido.
  - Muitos algoritmos exatos tem complexidade tipo $O(n^d), O(n^{d\text{-}1}), O(n^{d/2})$ etc.
  - Algoritmos aproximativos são $O(n)$ e $O(n \log n)$.
- Porque tanto faz.
  - Entrada é aproximada (medição, representação).
  - Problema é aproximado.
  - Algoritmo é usado dentro de heurística.

Diâmetro



- Entrada: conjunto com n pontos em espaço d-dimensional.
- Saída: Pontos $p^*, q^* \in P$ que maximizam $||p^*q^*||$.
- Será alvo de nosso primeiro algoritmo aproximativo.
- Antes, alguns algoritmos exatos...

Calculando o Diâmetro 1

- Algoritmo 1 (força bruta):

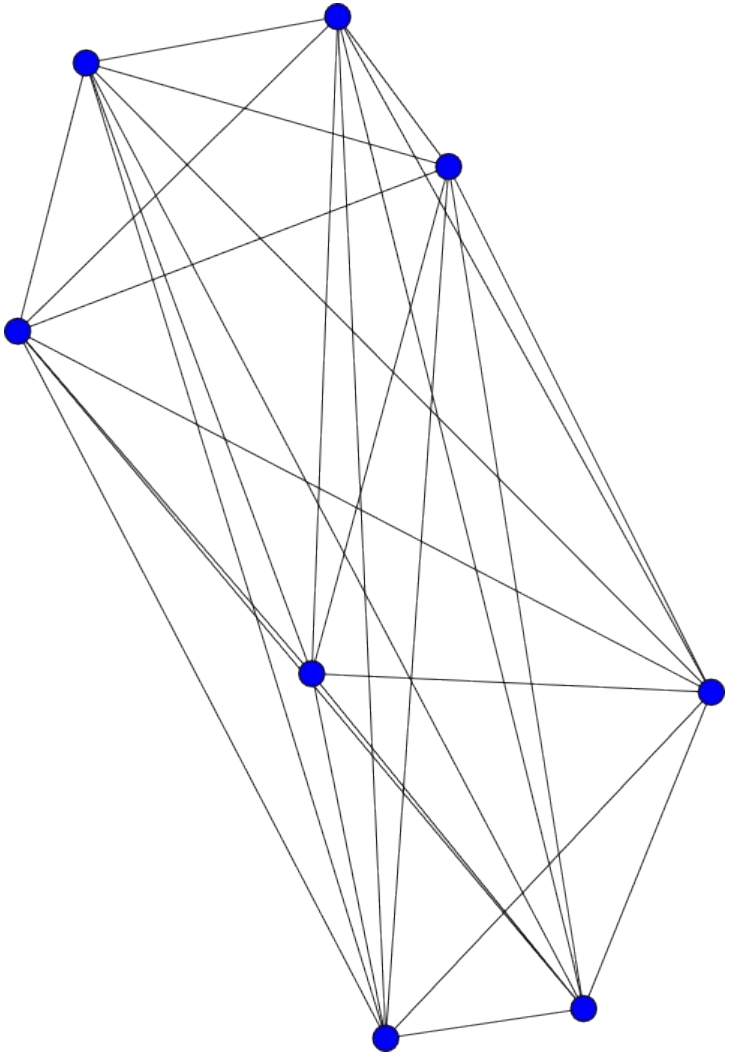
  diam = 0

  para cada par *p*,*q* ∈ *P:*

  diam = max(||*pq*||,*diam*)

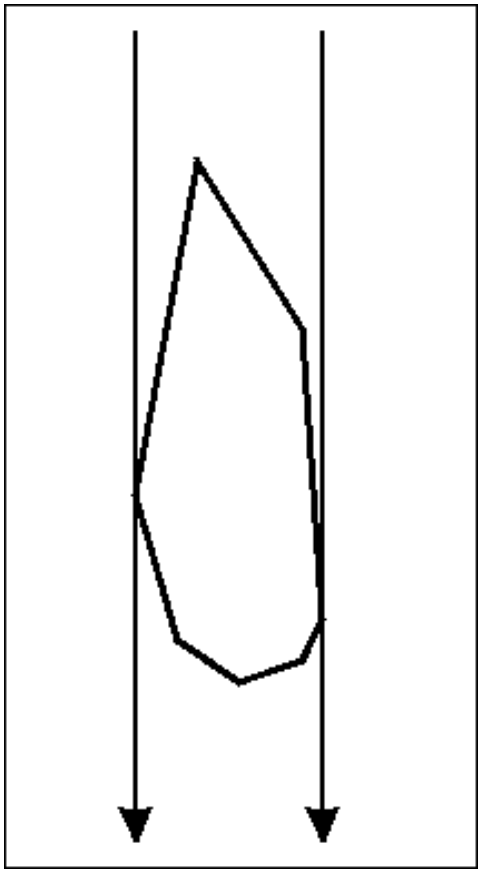
  retorne *diam*

- Lado bom: Funciona em dimensão arbitrária.
- Lado ruim: Leva tempo $O(n^2)$.



Calculando o Diâmetro 2

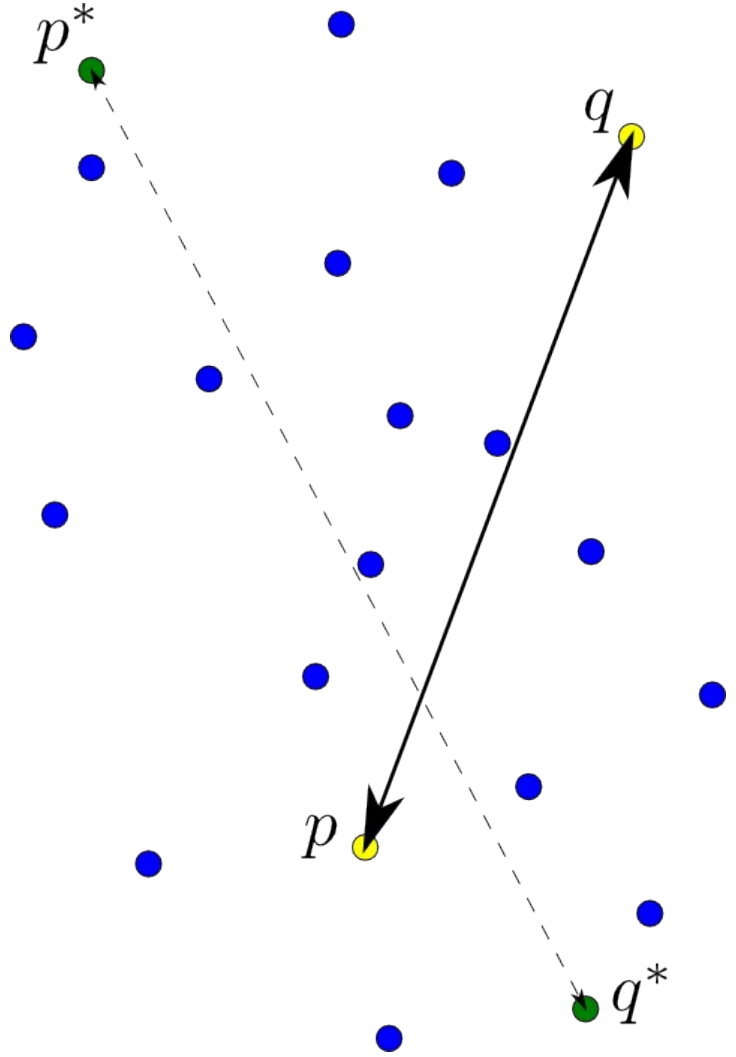
- Algoritmo 2: calcule primeiro o fecho convexo e depois...
- Lado bom: Sabemos calcular o fecho convexo em tempo $O(n \log n)$ para $d \leq 3$.
- Lado ruim: Calcular o fecho convexo leva tempo $\Omega(n^{\lfloor d/2 \rfloor})$.



http://cgm.cs.mcgill.ca/~godfried/research/calipers.html

Aproximação do Diâmetro

- Dizemos que um par de pontos $p,q \in P$ é uma k-aproximação do diâmetro se $k\ ||p^*q^*|| \leq ||pq||$.
- Claramente $||pq|| \leq ||p^*q^*||$.
- Como calcular uma 1/2-aproximação do diâmetro?



1/2-Aproximação do Diâmetro

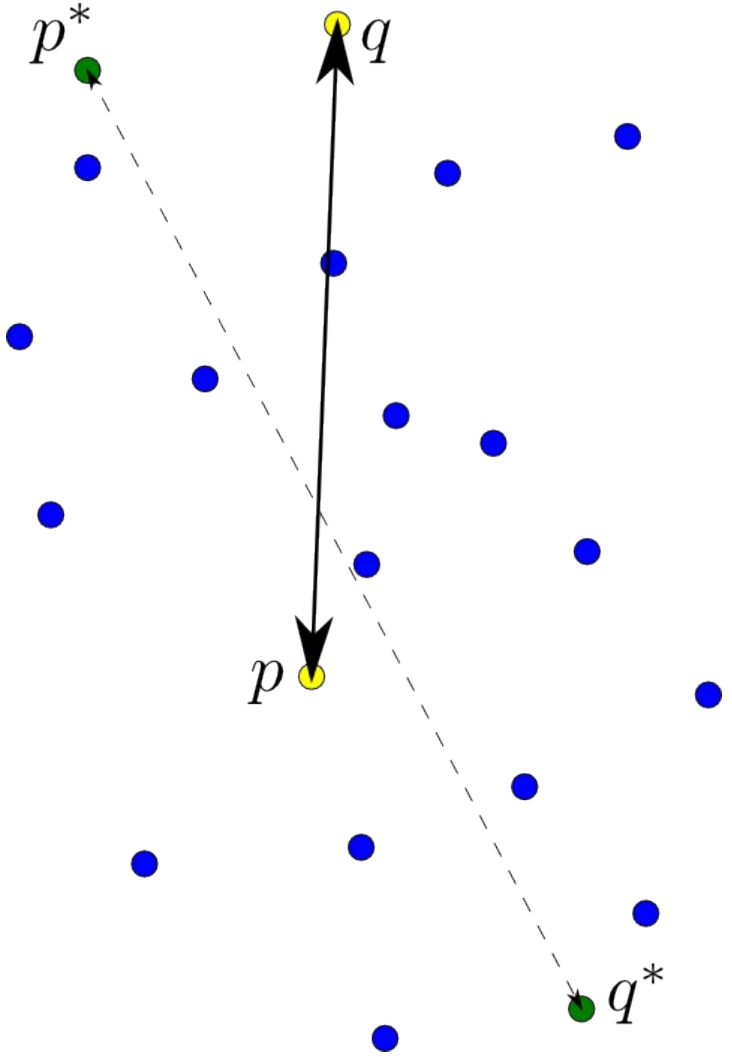
- Algoritmo:

  Escolhemos um ponto p arbitrário.

  Determinamos, em tempo $O(n)$, o ponto q mais distante de p.

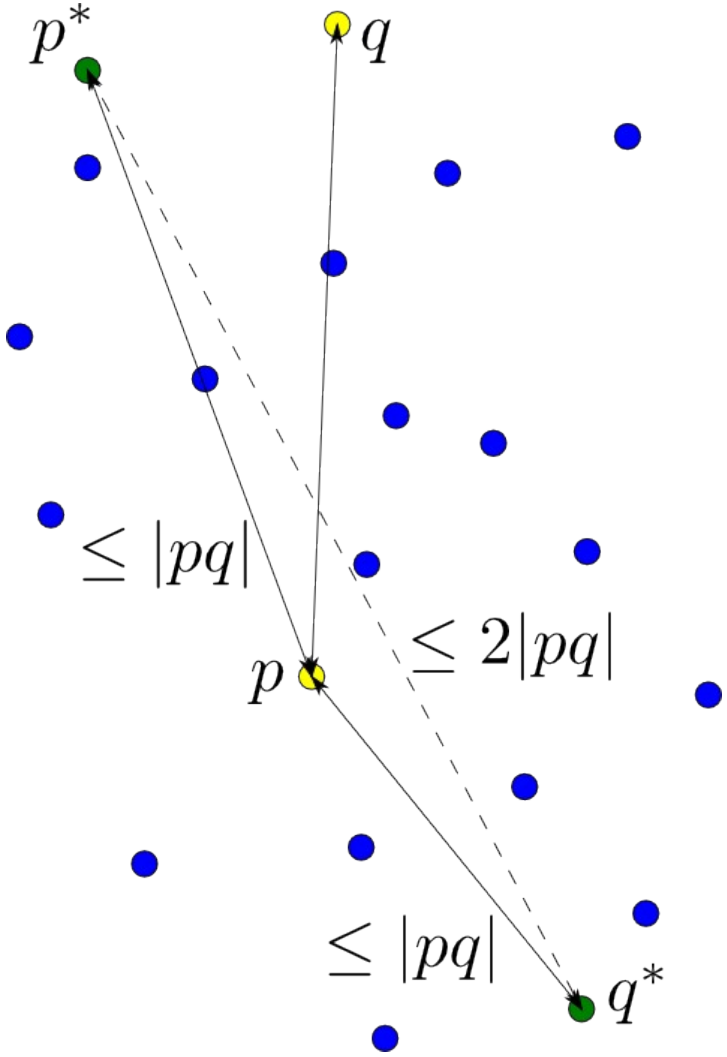
  Retornamos o par p,q.

- Precisamos provar que p,q é de fato uma 1/2-aproximação.



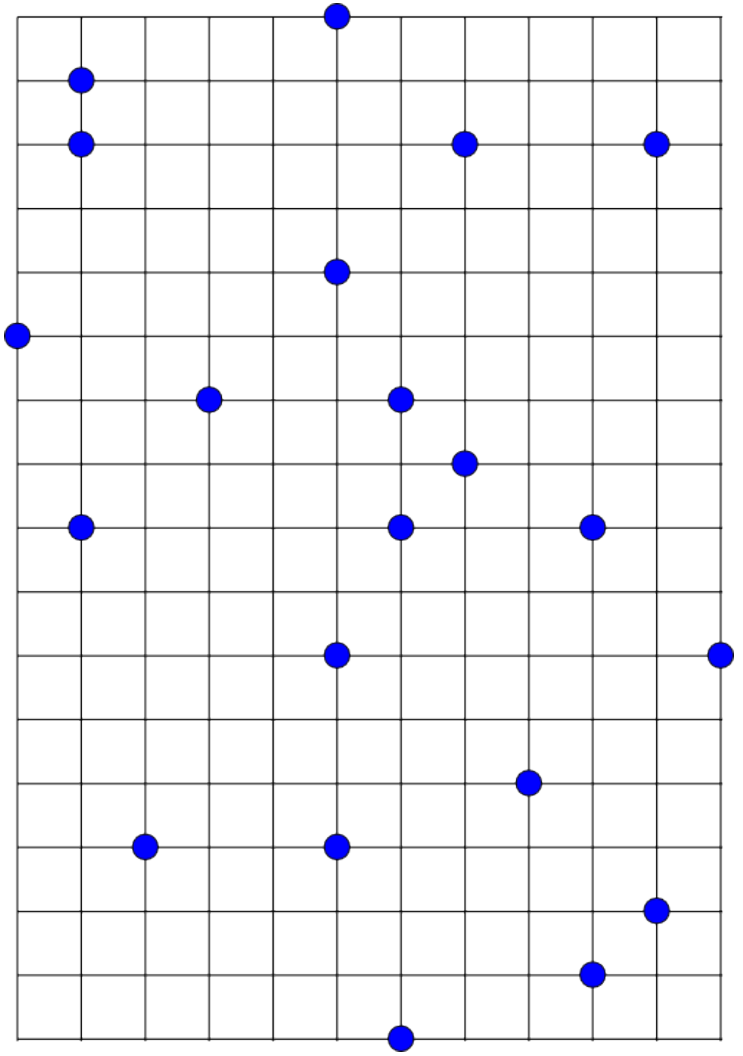
Prova de Corretude

- $||pp^*|| \leq ||pq||$, pois q é o ponto mais distante de p.
- $||pq^*|| \leq ||pq||$, pois q é o ponto mais distante de p.
- Usando desigualdade triangular, $||p^*q^*|| \leq 2||pq||$.

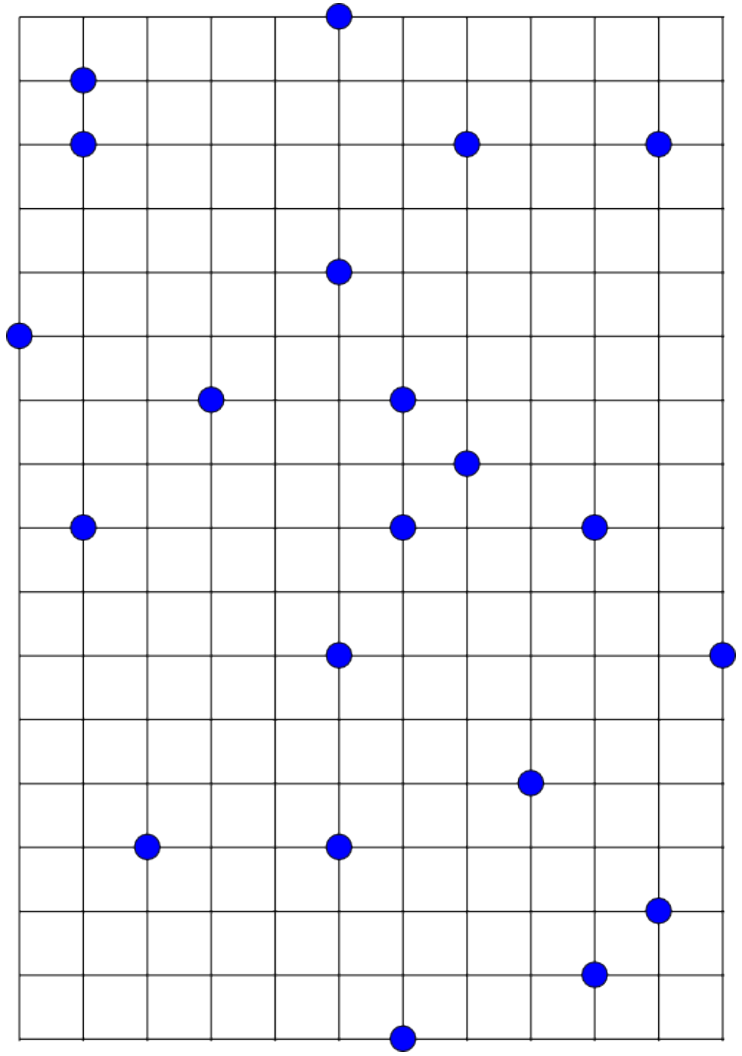


(1-ε)-Aproximação do Diâmetro

- Obtemos uma 1/2-aproximação, que chamamos de a.
- Criamos uma grade com células de diâmetro $a\varepsilon/2$.
- Aproximamos os pontos para vértices da grade, removendo duplicatas.
- Determinamos o diâmetro dos novos pontos usando força bruta.

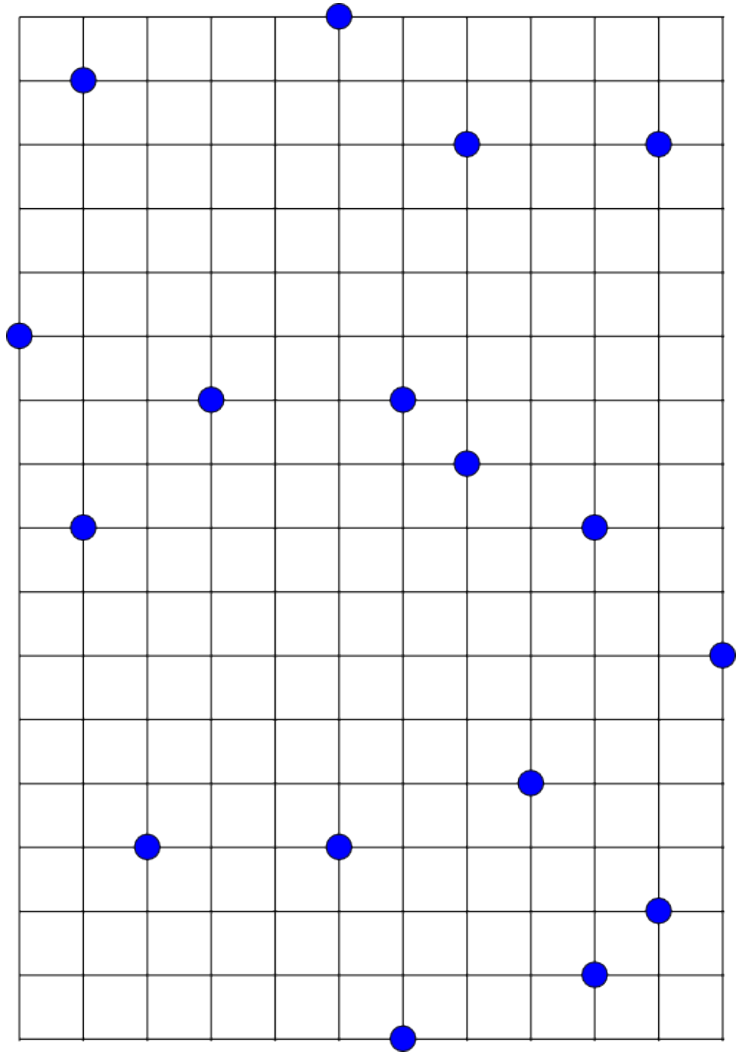


Análise



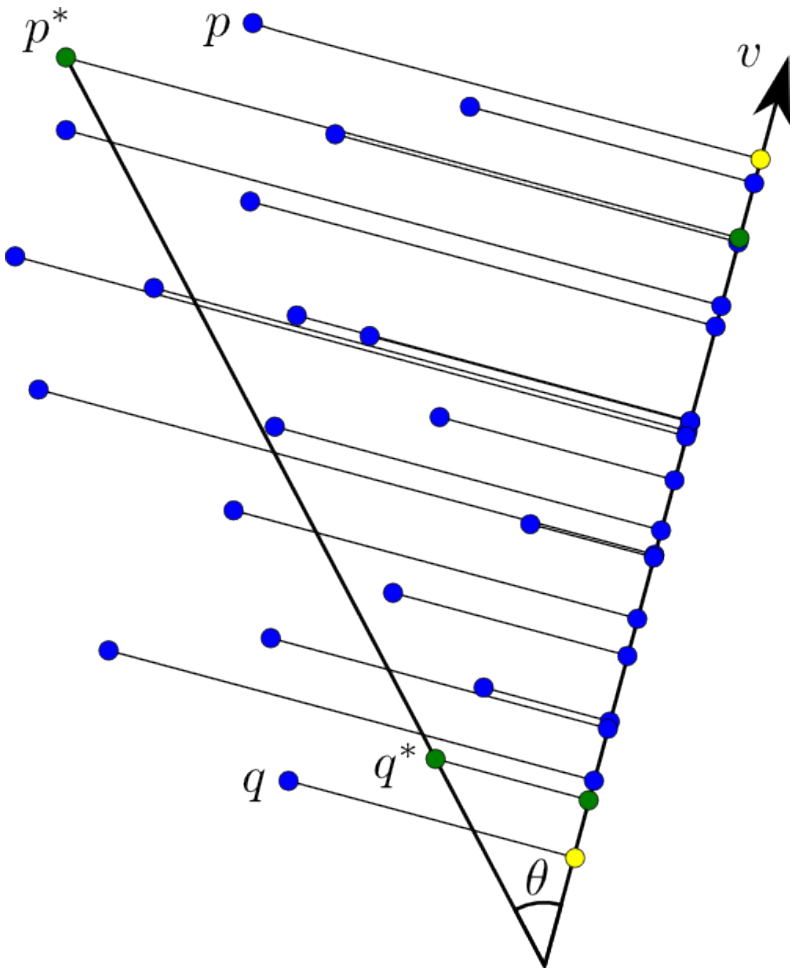
- Como a é uma aproximação de fator constante do diâmetro, a grade tem $O(1/\varepsilon^d)$ vértices.
- As duplicatas podem ser removidas em tempo $O(n)$ com hashing ou $O(n+1/\varepsilon^d)$.
- Como temos somente $O(1/\varepsilon^d)$ pontos, o algoritmo força bruta leva tempo $O(1/\varepsilon^{2d})$.
- Tempo total: $O(n+1/\varepsilon^{2d})$.

Uma Pequena Melhoria



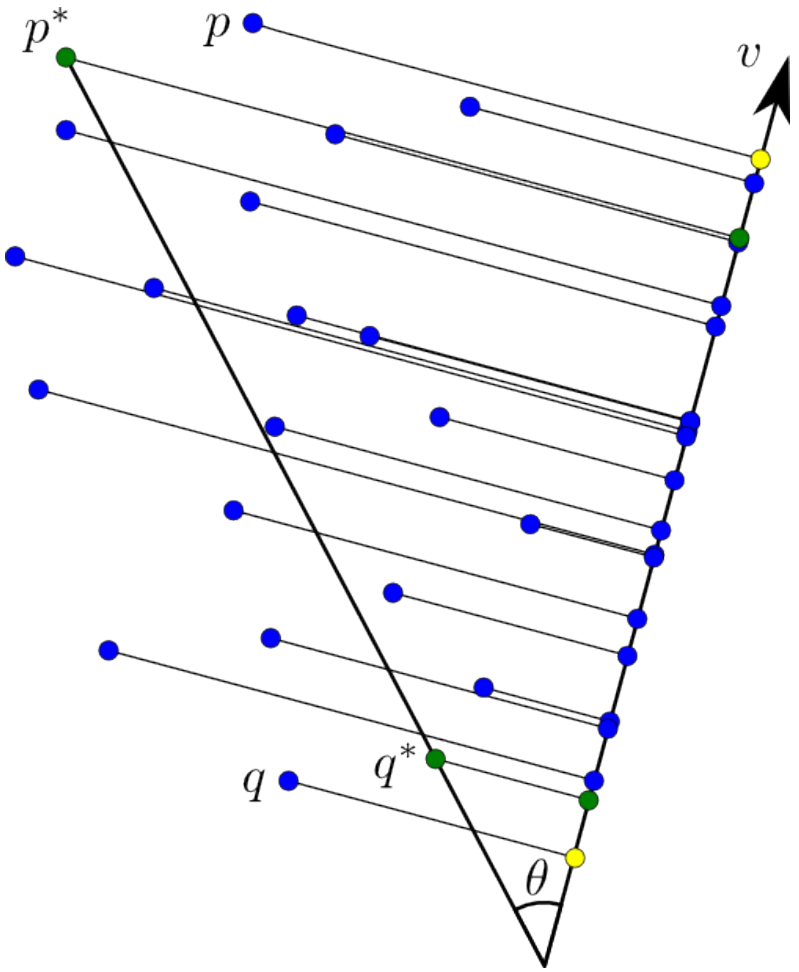
- Podemos manter somente os pontos mais altos e mais baixos em cada coluna da grade.
- Como temos apenas $O(1/\varepsilon^{d-1})$ colunas, a complexidade de tempo se reduz para $O(n+1/\varepsilon^{2(d-1)})$.
- Veremos mais dois algoritmos para o problema, sendo o terceiro uma combinação dos dois primeiros.

Outra Abordagem

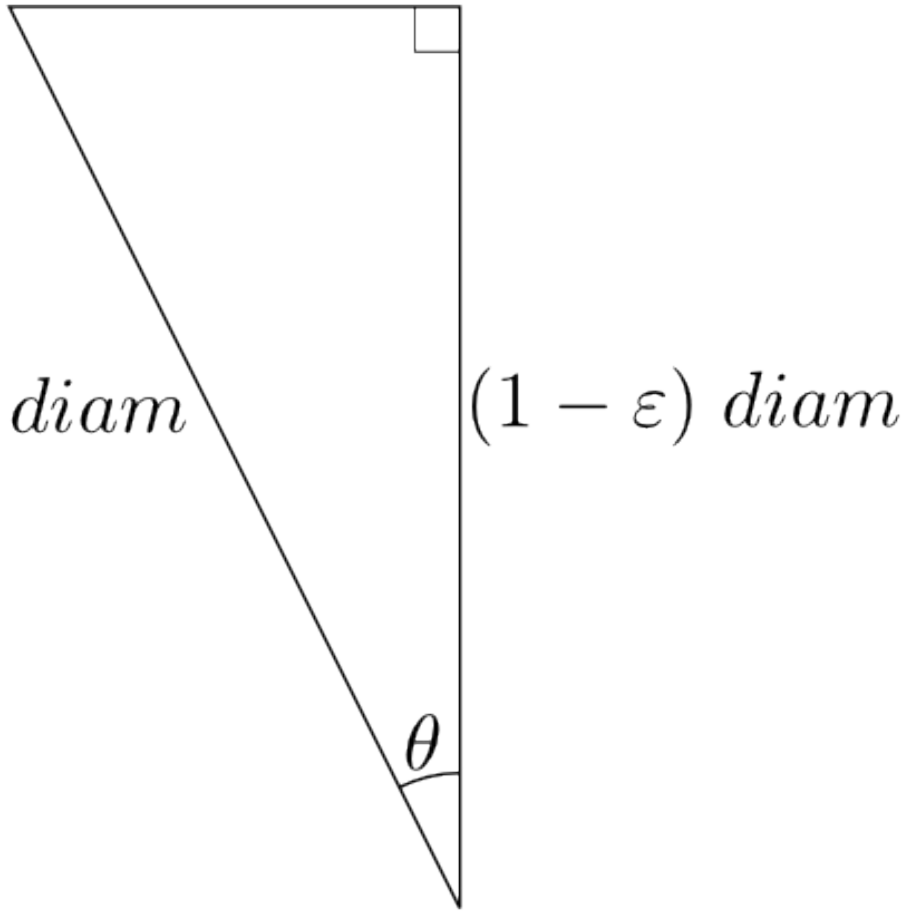


- Podemos tomar um vetor de direção arbitrário v e projetar todos os pontos em um espaço unidimensional.
- Então, determinamos os pontos extremos no espaço unidimensional.
- A qualidade de aproximação depende do ângulo θ entre v e a reta p^*q^*.
- Como é essa dependência?

Outra Abordagem

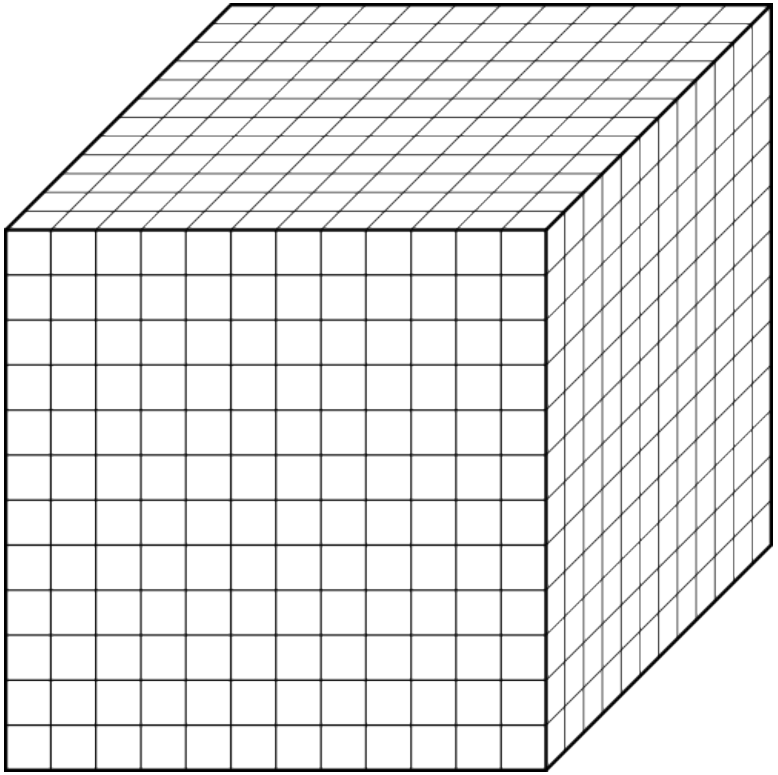


$$\sqrt{2\varepsilon - \varepsilon^2} > \sqrt{2\varepsilon}$$



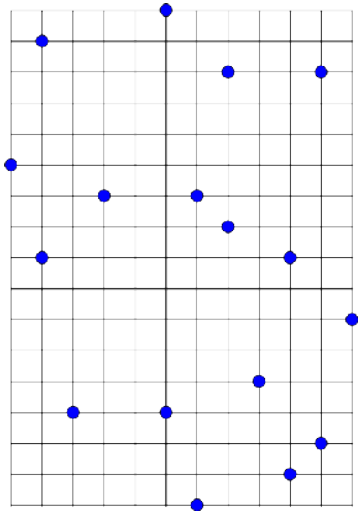
$$\theta = \arccos(1-\varepsilon) = \Theta(\sqrt{\varepsilon})$$

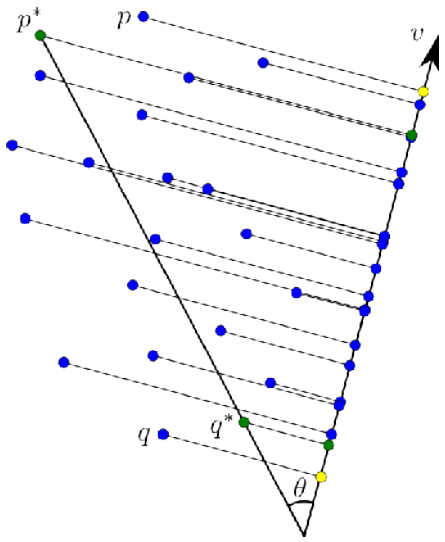
Aproximando Ângulos



- Aproximamos todas as direções possíveis com um erro de no máximo $\sqrt{\varepsilon}$ usando um total de $O(1/\varepsilon^{(d-1)/2})$ direções.
- Para cada direção, precisamos examinar n pontos, levando tempo total de $O(n/\varepsilon^{(d-1)/2})$.

Combinando os Algoritmos



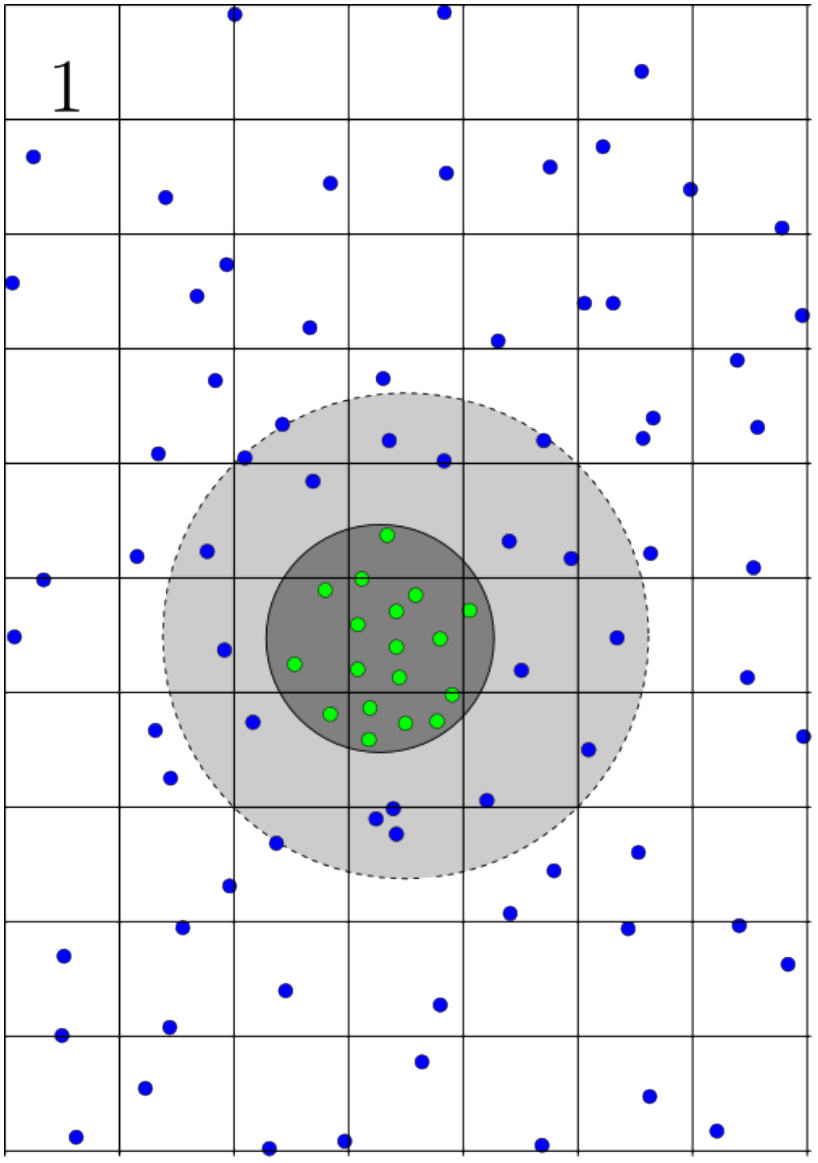


- Temos dois algoritmos aproximados, com complexidades: $O(n+1/\varepsilon^{2(d-1)})$ e $O(n/\varepsilon^{(d-1)/2})$.
- Podemos combinar os dois algoritmos:
  - Obtemos um conjunto de apenas $O(1/\varepsilon^{d-1})$ pontos.
  - Usamos então o segundo algoritmo, com $n=O(1/\varepsilon^{d-1})$.
- Tempo: $O(n+1/\varepsilon^{3(d-1)/2})$.

Referências

- Algoritmos apresentados aqui para aproximar o diâmetro:
  T. M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. Int. J. Comput. Geom. Appl., 12:67-85, 2002.
- Algoritmo mais eficiente conhecido ($O\left(n+1/\varepsilon^{d-3/2)}\right)$):
  T. M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. Computational Geometry: Theory and Applications, 35:20-35, 2006.
- Diâmetro exato em 3d (determinístico):
  Edgar A. Ramos. An Optimal Deterministic Algorithm for Computing the Diameter of a Three-Dimensional Point Set. Discrete & Computational Geometry 26(2): 233-244, 2001.

Aproximação Geométrica



Aula 2

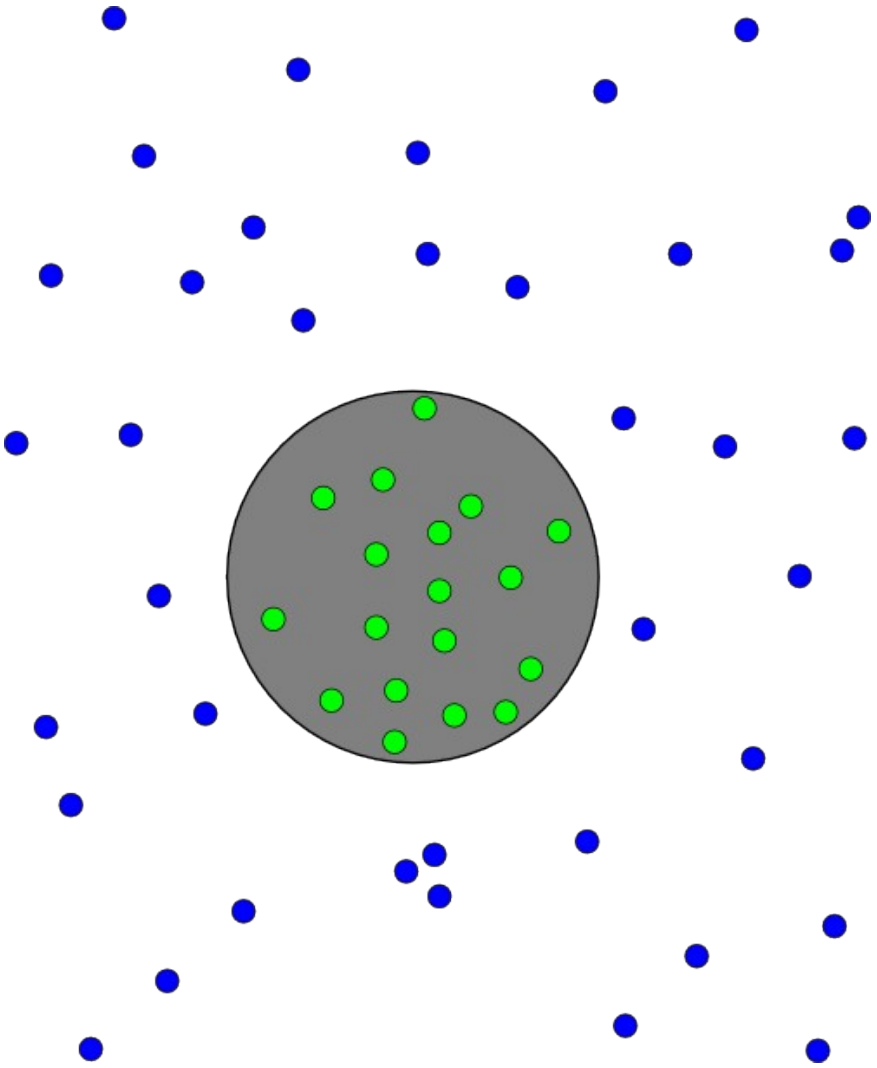
Tópicos:

- Disco unitário com mais pontos
- Menor disco com k pontos

IMPA – Verão 2009

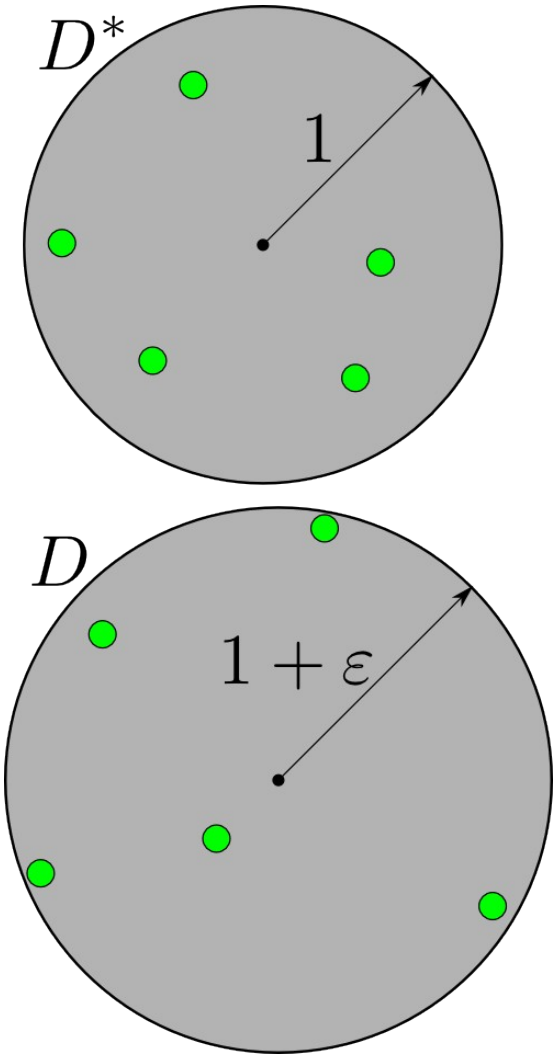
Guilherme D. da Fonseca

Disco Unitário com Mais Pontos



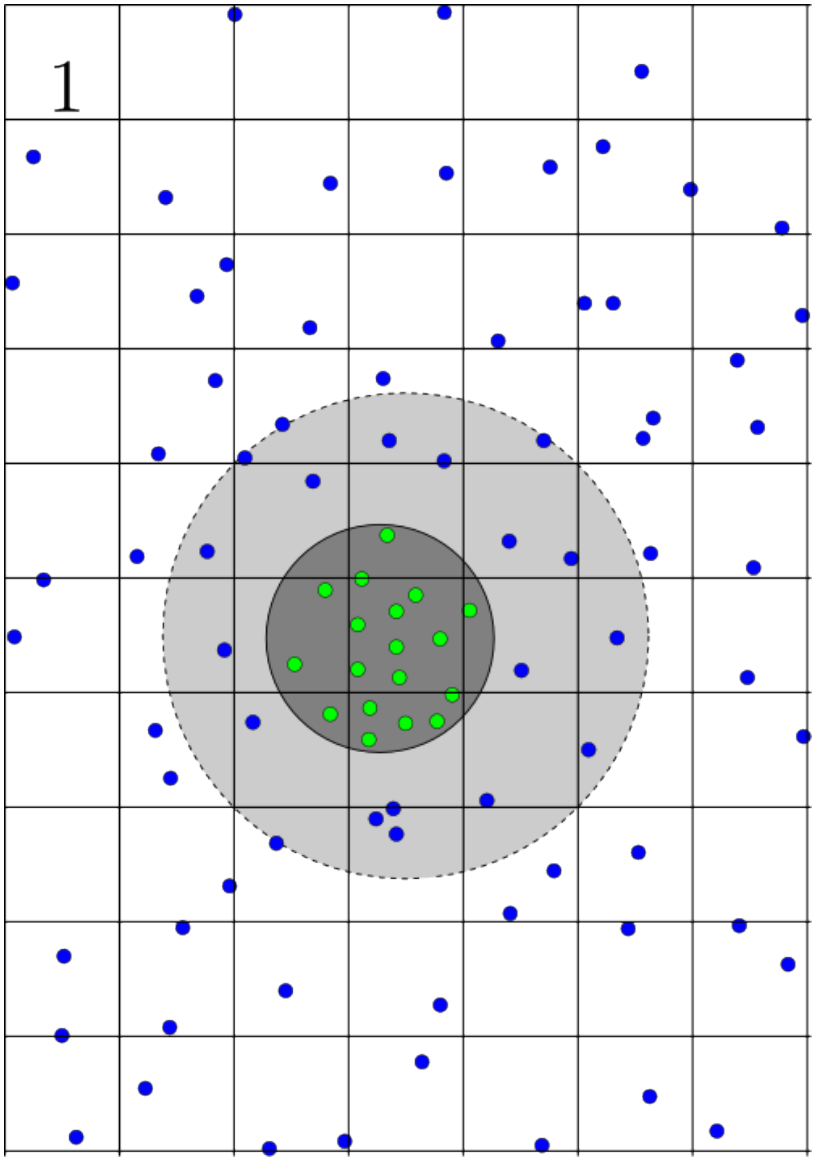
- Entrada: conjunto com n pontos em espaço d-dimensional.
- Saída: Disco unitário D que maximiza $|P \cap D|$.
- Algoritmos exatos conhecidos levam tempo $\Theta(n^2)$ no plano e $\Omega(n^d)$ no geral.
- Problema é 3SUM-difícil mesmo no plano.

Disco Aproximadamente Unitário com Mais Pontos



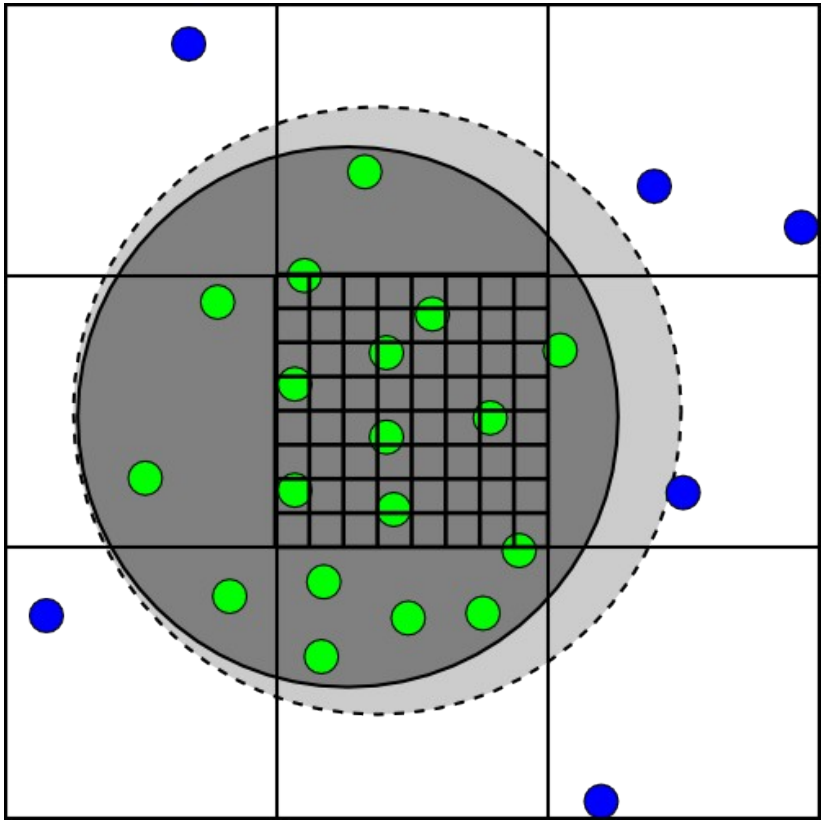
- Seja D^* um disco unitário com o número máximo de pontos.
- (1+ε)-aproximação: Obtemos um disco D de raio 1+ε contendo pelo menos tantos pontos quanto D^*.
- Começamos com uma aproximação de fator “constante”.

O(1)-aproximação



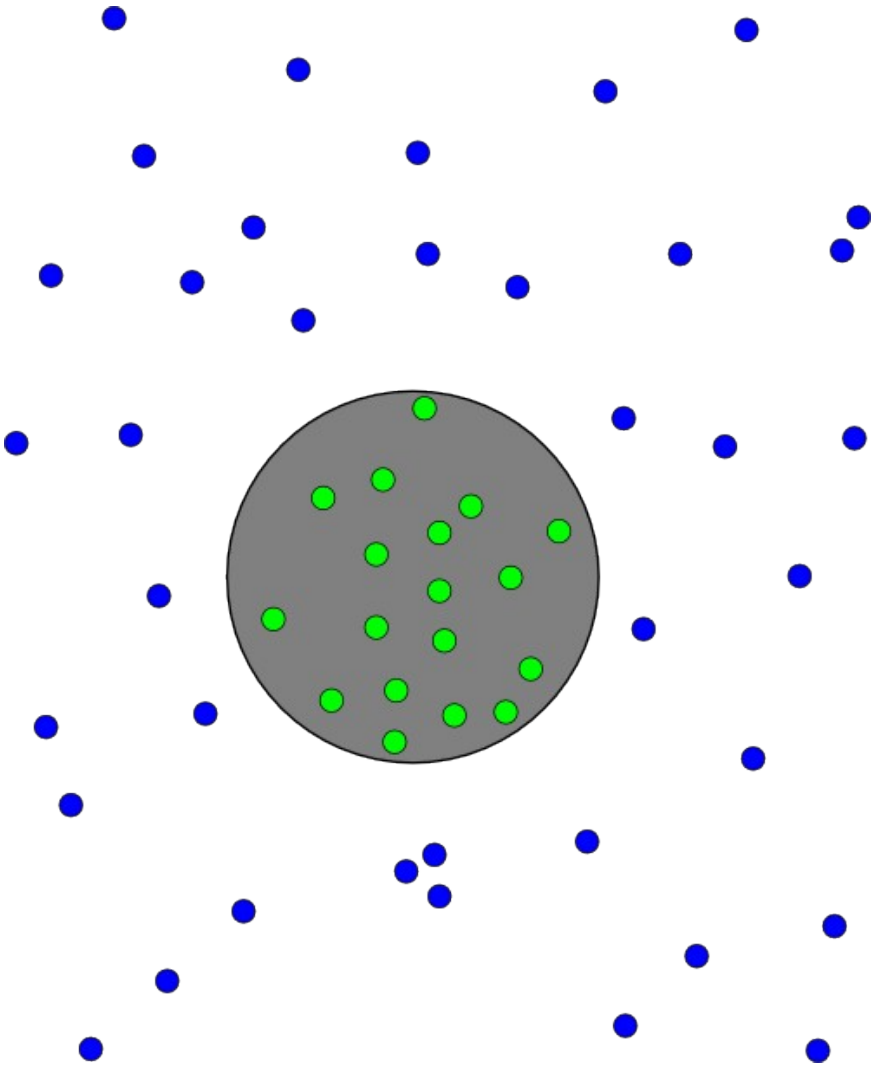
- Criamos uma grade com células de lado 1.
- Determinamos a célula com o maior número de pontos nas 3^d células adjacentes.
- Envolvemos a adjacência da célula com um disco de raio $\frac{3\sqrt{d}}{2}$.
- Usando hashing, leva tempo $O(n)$.

(1+ε)-aproximação



- Criamos uma subgrade com células de diâmetro ε no interior de cada célula.
- Para cada vértice v das subgrades, contamos quantos pontos estão no disco de raio 1+ε centrado em v.
- Retornamos o disco com mais ponto examinado.
- Leva tempo $O(n/\varepsilon^d)$. Como?

Menor Disco com k Pontos

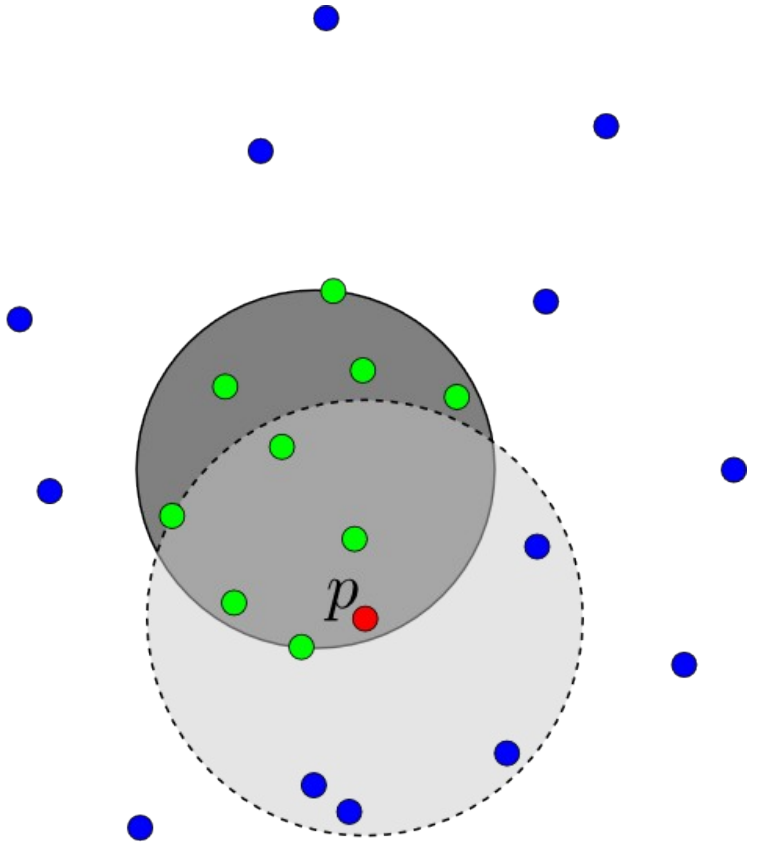


- Entrada: conjunto com n pontos e inteiro $k \leq n$.
- Saída: Menor disco D com $|P \cap D| = k$.
- Nesta versão, k é dado, mas o raio não!
- Algoritmos exatos conhecidos levam tempo $\Theta(n \log n + nk)$ no plano.
- Focamos em valores grandes de $k = \Theta(n)$.

Las Vegas x Monte Carlo

- Las Vegas: a resposta está sempre correta, mas o tempo de execução é um valor esperado.
- Monte Carlo: a resposta está correta com uma certa probabilidade.
- As probabilidades dependem somente dos bits aleatórios usados, e não da entrada.

Uma 2-Aproximação Monte Carlo

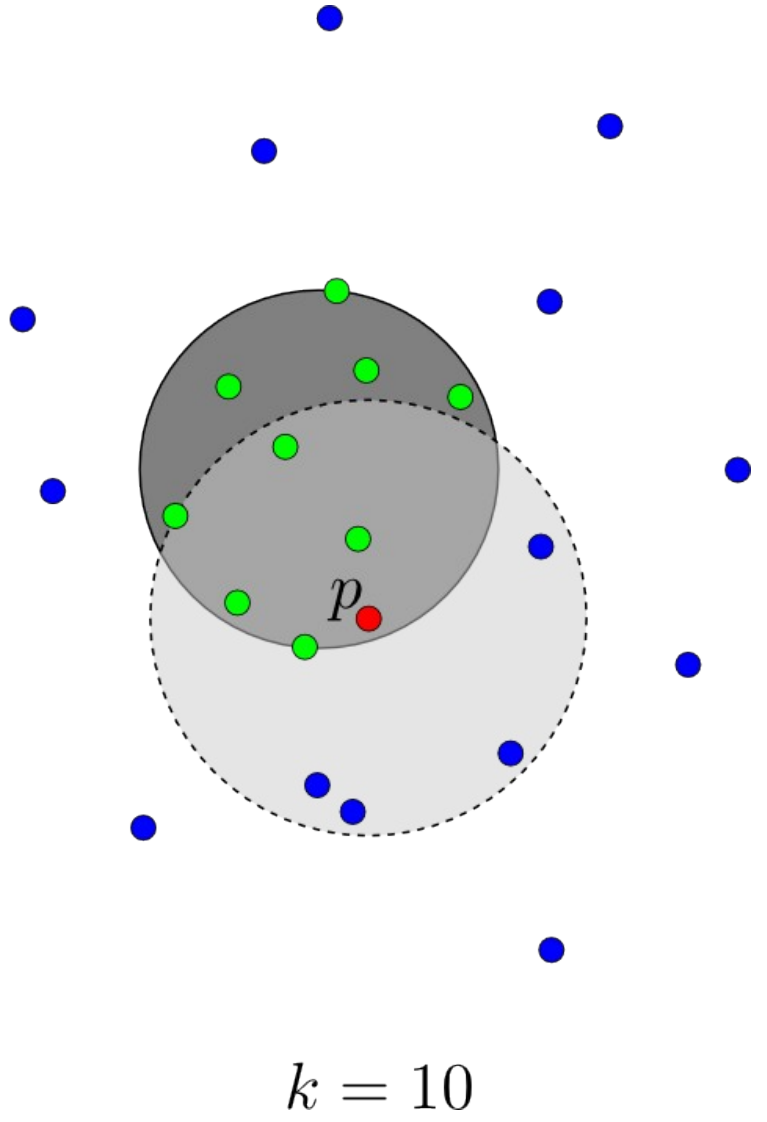


$k = 10$

- Pegamos um ponto aleatório $p \in P$.
- Calculamos a distância de p até cada ponto de P.
- Selecionamos a k-ésima menor distância (em tempo $O(n)$).
- Se o ponto p pertence ao disco ótimo, então temos uma 2-aproximação.
- Qual a probabilidade?

Análise

- Por definição, temos k pontos no disco ótimo.
- Se escolhemos um ponto dentre n, a probabilidade do ponto escolhido estar no disco ótimo é k/n.
- Para obtermos uma 2-aproximação com probabilidade constante repetimos o procedimento $O(n/k)$ vezes.
- Tempo: $O(n^2/k)$.

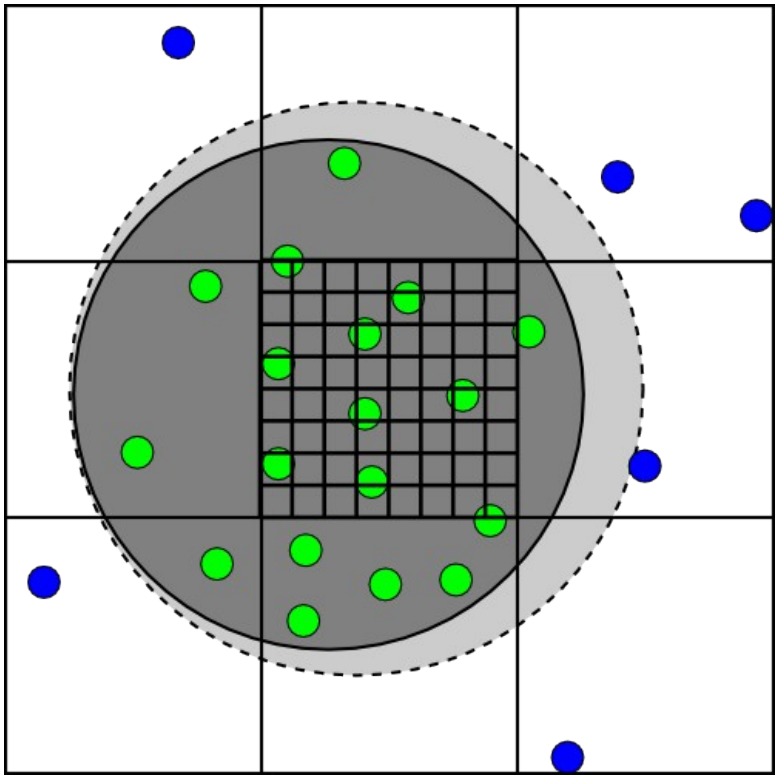


De Monte Carlo para Las Vegas

- Usamos o algoritmo de disco unitário com mais pontos (DU) para verificar a solução Monte Carlo (MC).
- MC: menor disco com k pontos tem raio $r/2 \leq r^* \leq r$.
- Usamos DU para obter um disco de raio $r/3$ que contém tantos pontos quanto qualquer disco de raio $r/4$. Seja m o número de pontos neste disco.
- Se $m \geq k$, a resposta do Monte Carlo está errada.
- Se $m < k$, então a resposta é uma 4-aproximação!
- O valor esperado do número de repetições até obter a resposta certa é $O(n/k)$, levando tempo $O(n^2/k)$.

(1+ε)-aproximação

- Obtemos uma 4-aproximação a do raio.
- Criamos grade com lado a.
- Criamos subgrade com diâmetro $a\varepsilon/4$ no interior de cada célula.
- Para cada vértice v das subgrades, determinamos o raio do disco com k pontos centrado em v.
- Retornamos o menor disco encontrado.



- Tempo: $O(n^2/k + n/\varepsilon^d)$.

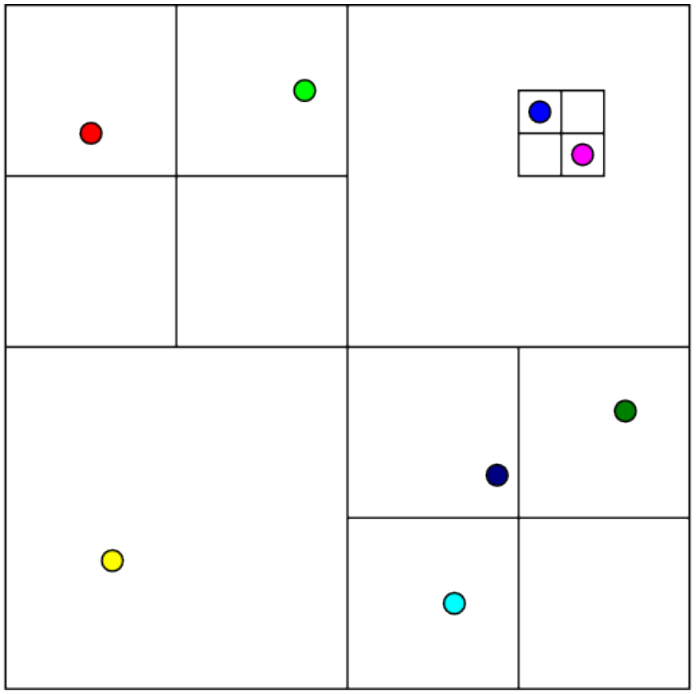
Referências (Disco Unitário com Mais Pontos)

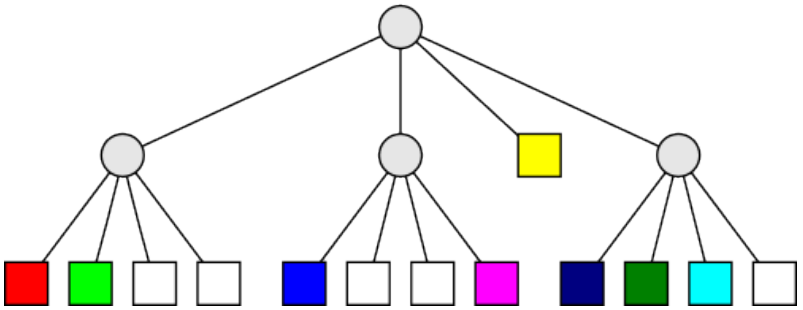
- Algoritmo exato $O(n^2)$ no plano:
  B. Chazelle and D.T. Lee. On a circle placement problem. Computing, 36(1-2):1–16, 1986.
- Algoritmo descrito aqui baseado em:
  S. Funke, T. Malamatos, and R. Ray. Finding Planar Regions in a Terrain, in Practice and with a Guarantee. Int. J. Comput. Geom. Appl., 15(4):379-401, 2005.
- Algoritmo aproximado mais eficiente conhecido ($O\left(n/\varepsilon^{d-1}\right)$):
  C. M. H. de Figueiredo and G. D. da Fonseca. Enclosing Weighted Points with an Almost-Unit Ball. Information Processing Letters, 109(21-22):1216-1221, 2009.

Referências (Menor Disco com k Pontos)

- Algoritmos exatos e aproximados:
  S. Har-Peled and S. Mazumdar. Fast algorithms for computing the smallest k-enclosing circle. Algorithmica, 41(3):147–157, 2005.

Aproximação Geométrica





Aula 3

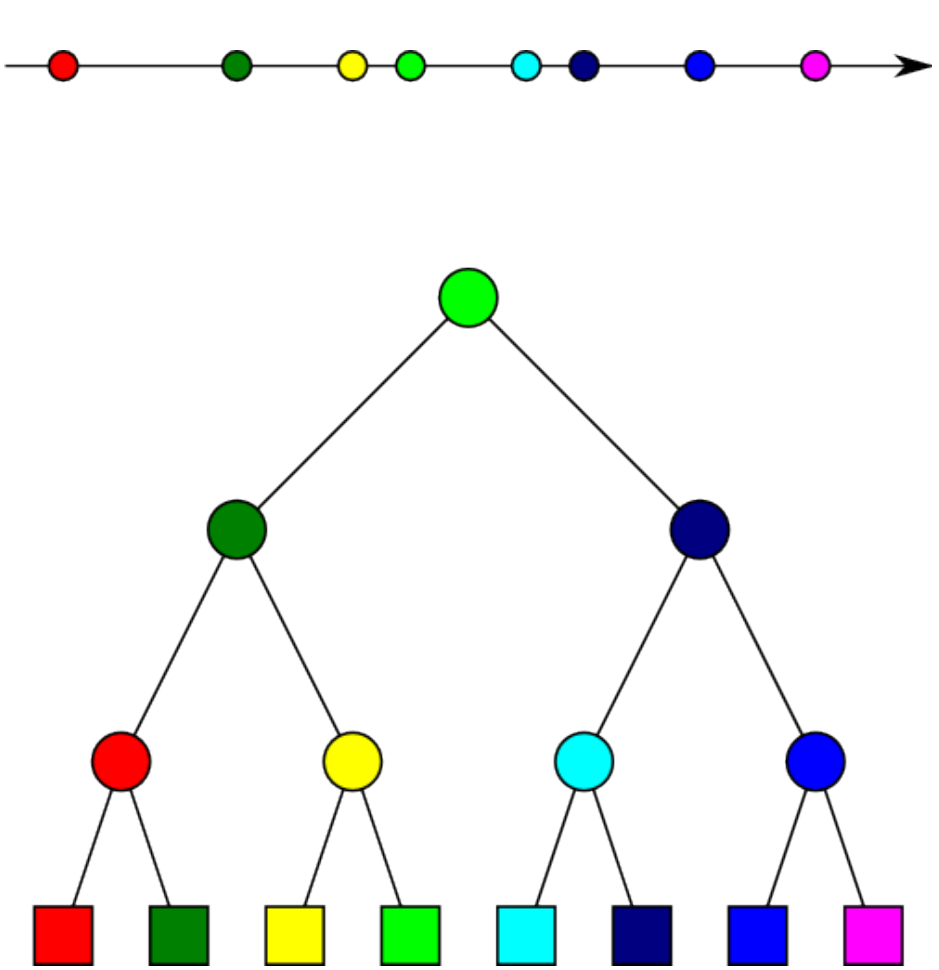
Tópico:

- Quadtrees

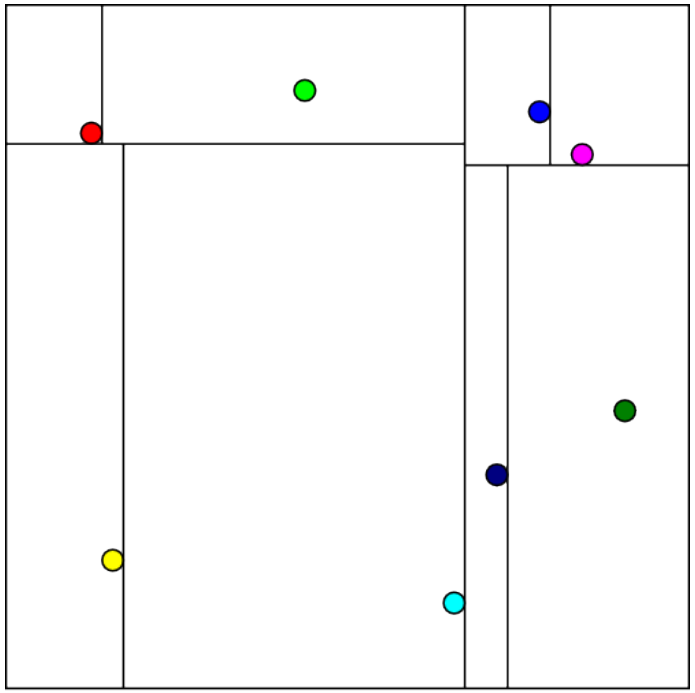
IMPA – Verão 2009

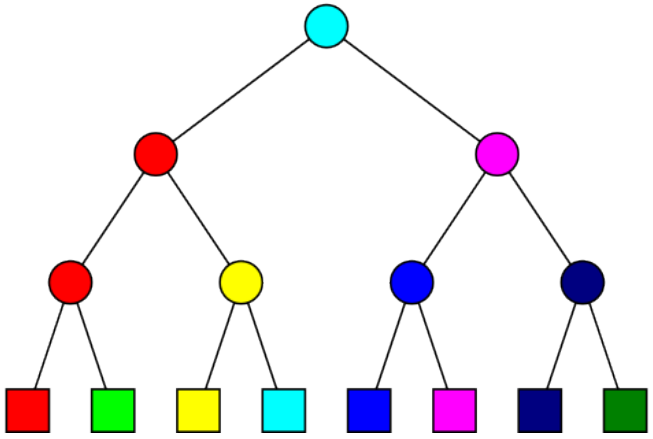
Guilherme D. da Fonseca

Subdivisões Hierárquicas



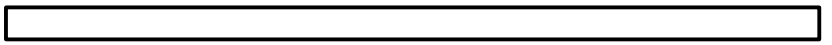
- Subdivisões hierárquicas aparecem nas mais diversas áreas da computação.
- O exemplo mais simples são árvores binárias de busca, fortemente baseadas na ordenação total dos elementos.
- Como generalizar para espaços com $d>1$?
- Várias maneiras...



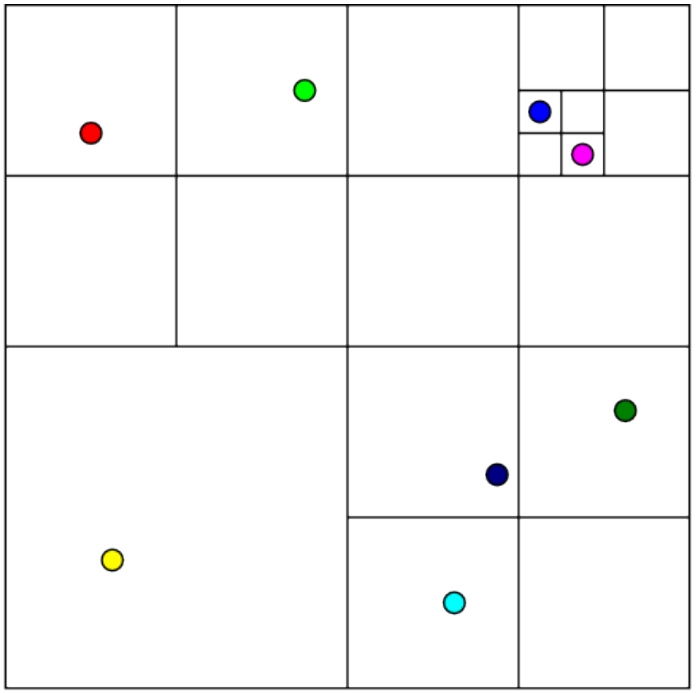


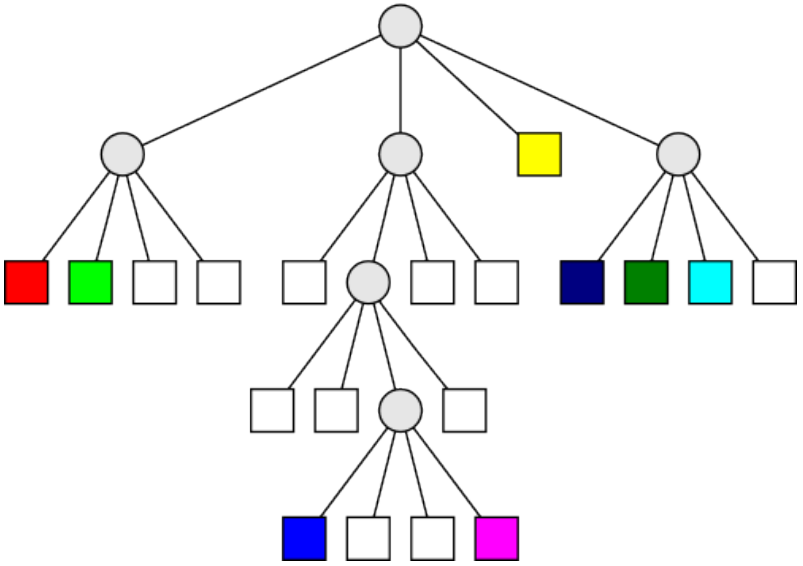
kd-tree

- Pontos fortes:
  - Tamanho $O(n)$.
  - Altura $O(\log n)$.
  - Subdivide muito bem os pontos.
- Pontos fracos:
  - Células podem ser muito compridas e finas:



  - Subdivide mal o espaço.

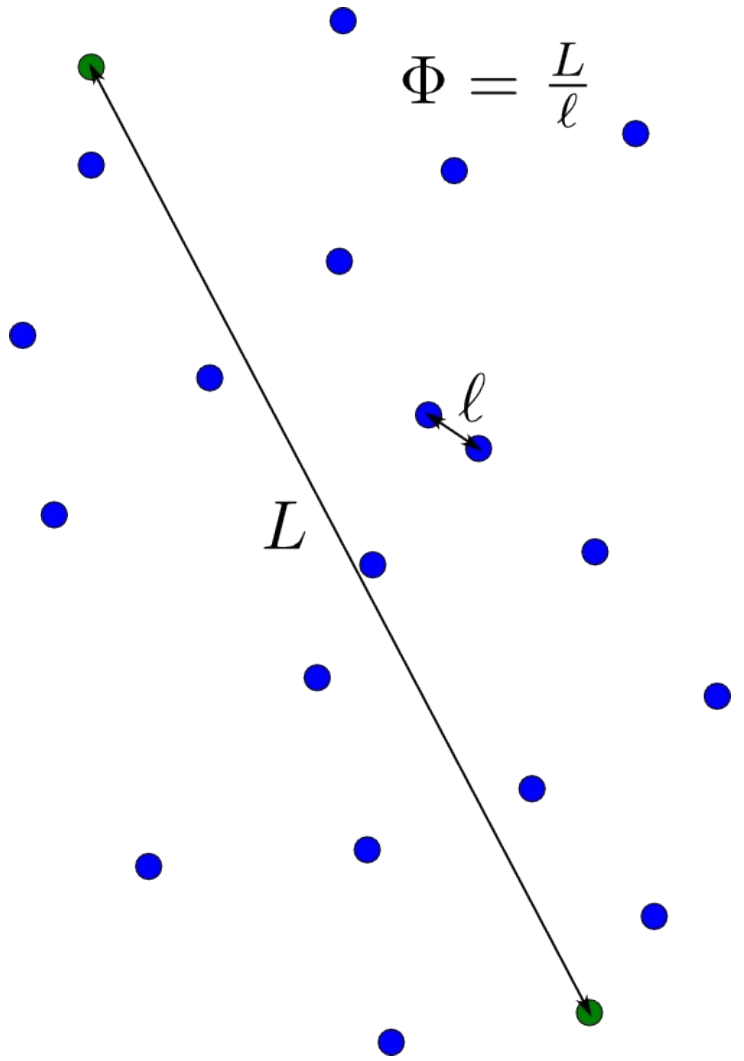




Quadtree

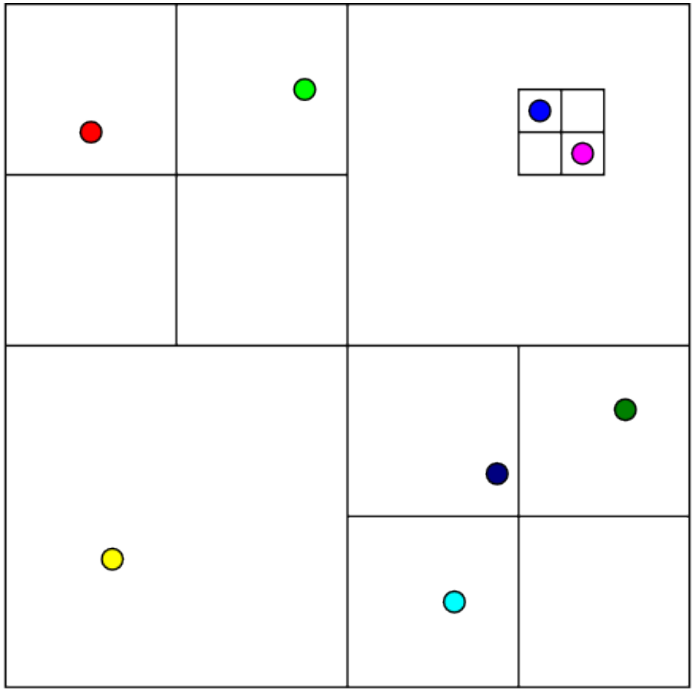
- Pontos fortes:
  - Subdivide muito bem o espaço.
  - Células são cubos.
  - Lemas de empacotamento.
- Pontos fracos:
  - Tamanho ilimitado em n.
  - Altura ilimitada em n.
- Chamamos as células potenciais de caixas.

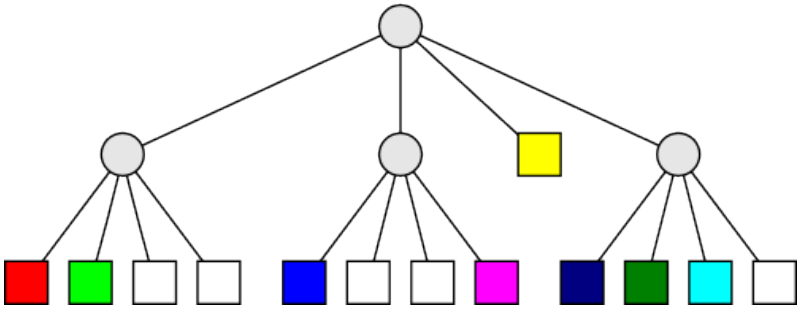
Espalhamento



- O espalhamento Φ de um conjunto de pontos é a razão entre a maior e a menor distância dentro os pontos.
- A quadtree é eficiente para espalhamento Φ pequeno.
- Tamanho: $O(n \log \Phi)$.
- Altura: $O(\log \Phi)$.
- Construção: $O(n \log \Phi)$.
- E se Φ é grande?

Quadtree Compactada

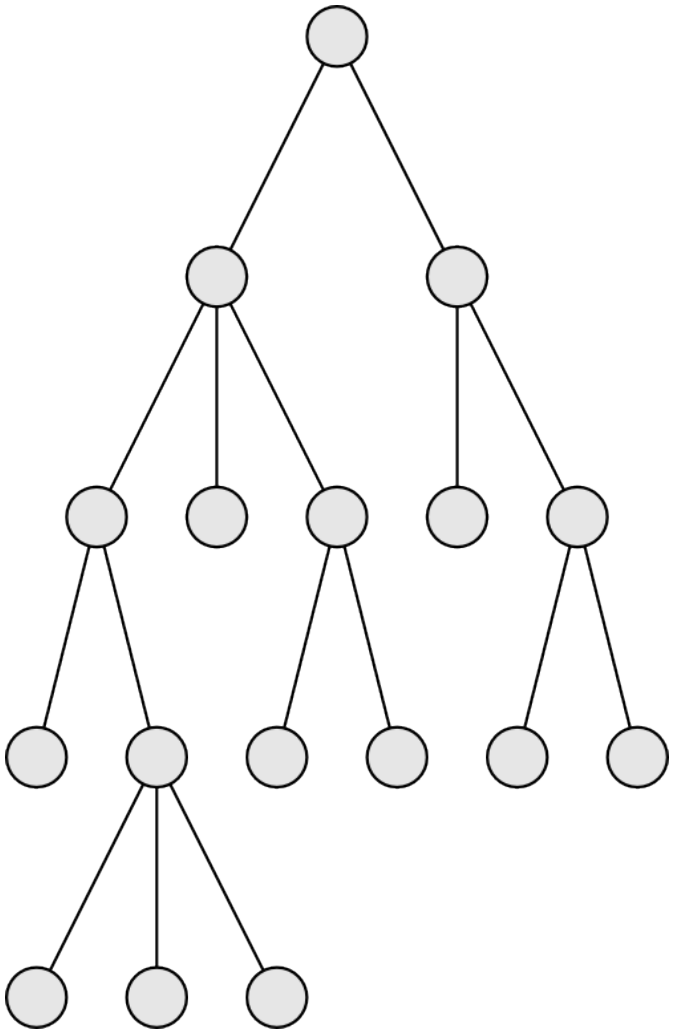




- Pontos fortes:
  - Subdivide muito bem o espaço.
  - Células são cubos.
  - Lemas de empacotamento.
  - Tamanho $O(n)$.
- Ponto fraco:
  - Altura $O(n)$.

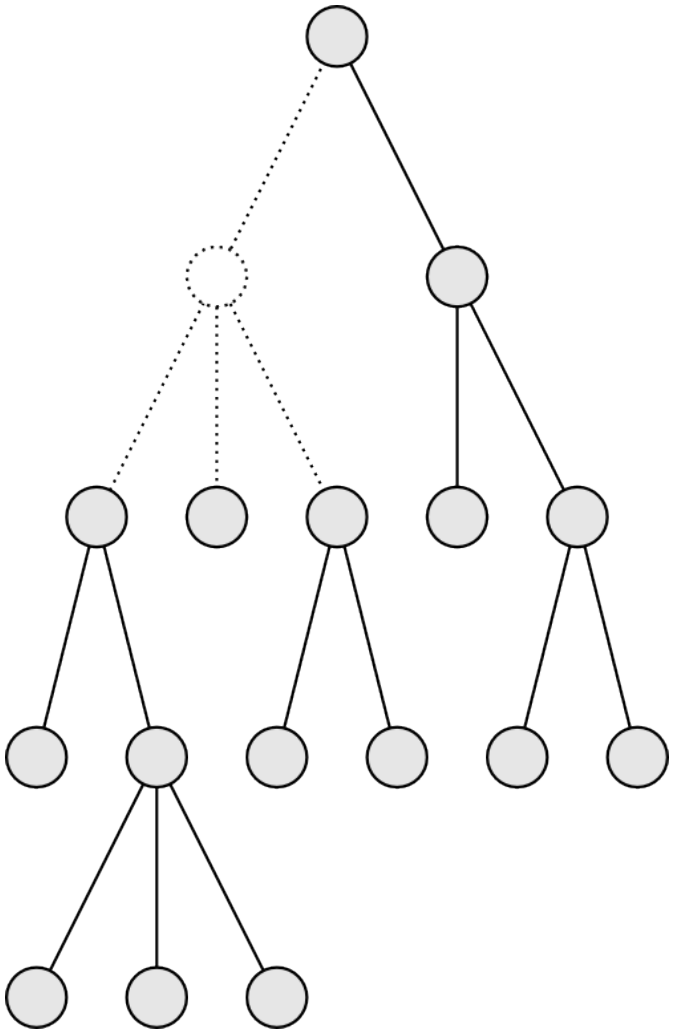
Índice da Quadtree Compactada

- Um separador é um vértice que, quando removido, decompõe a árvore em árvores com no máximo metade dos vértices.
- Toda árvore possui um separador que pode ser encontrado em tempo $O(n)$.
- Chamamos a hierarquia de separadores de índice.



Índice da Quadtree Compactada

- Um separador é um vértice que, quando removido, decompõe a árvore em árvores com no máximo metade dos vértices.
- Toda árvore possui um separador que pode ser encontrado em tempo $O(n)$.
- Chamamos a hierarquia de separadores de índice.

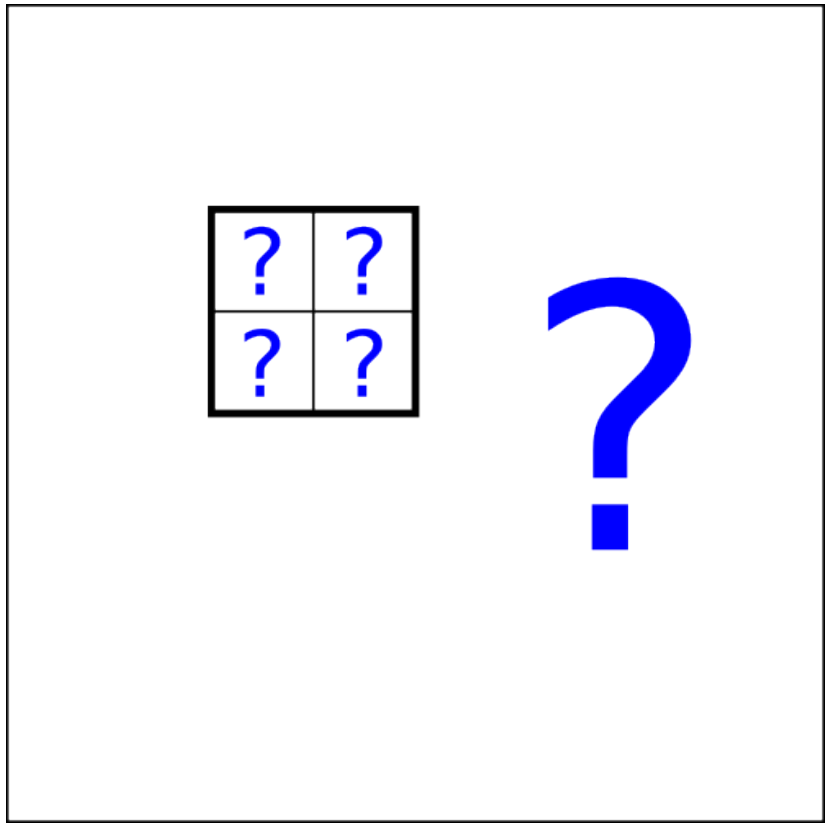


Índice da Quadtree Compactada

- Um separador é um vértice que, quando removido, decompõe a árvore em árvores com no máximo metade dos vértices.
- Toda árvore possui um separador que pode ser encontrado em tempo $O(n)$.
- Chamamos a hierarquia de separadores de índice.

- Pontos fortes:
  - Subdivide muito bem o espaço.
  - Células são cubos.
  - Lemas de empacotamento.
  - Tamanho $O(n)$.
  - Altura (do índice) $O(\log n)$.

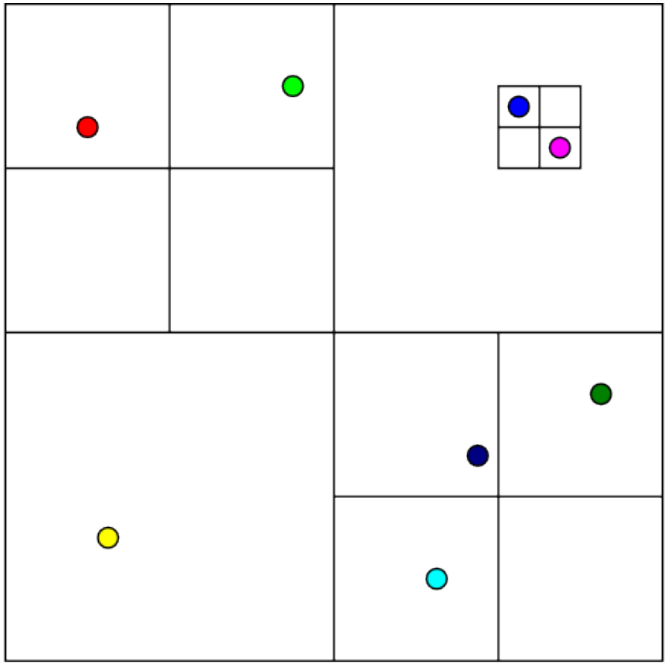
Usando o Índice

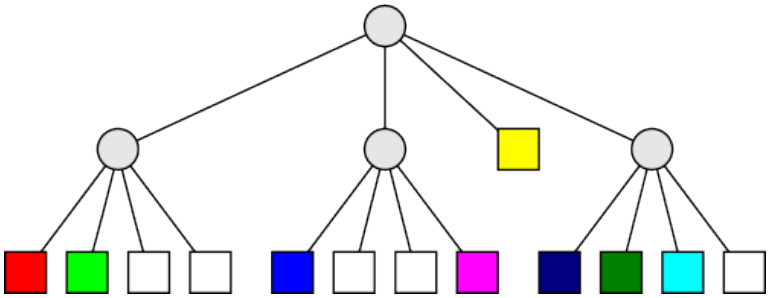


- Um vértice do índice contém 2^d+1 ponteiros para as subdivisões da célula e seu exterior.
- Uma consulta de célula consiste em localizar a célula que contém o mesmo conjunto de pontos que uma dada caixa.
- Pode ser respondida em tempo $O(\log n)$ seguindo os ponteiros apropriados.

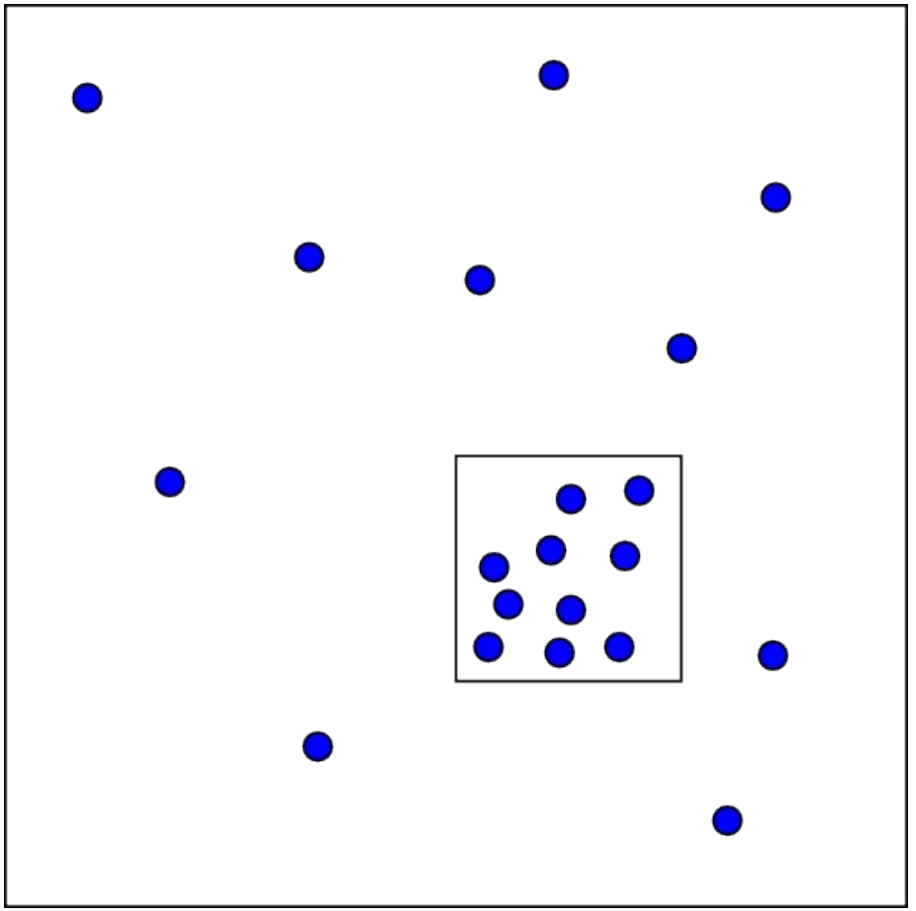
Construção Eficiente

- É fácil construir uma quadtree compactada em tempo $O(n^2)$.
- Porém, é possível construir em tempo $O(n \log n)$.
- Para isso, usamos o algoritmo de encontrar o menor disco com k pontos e a técnica de divisão e conquista.





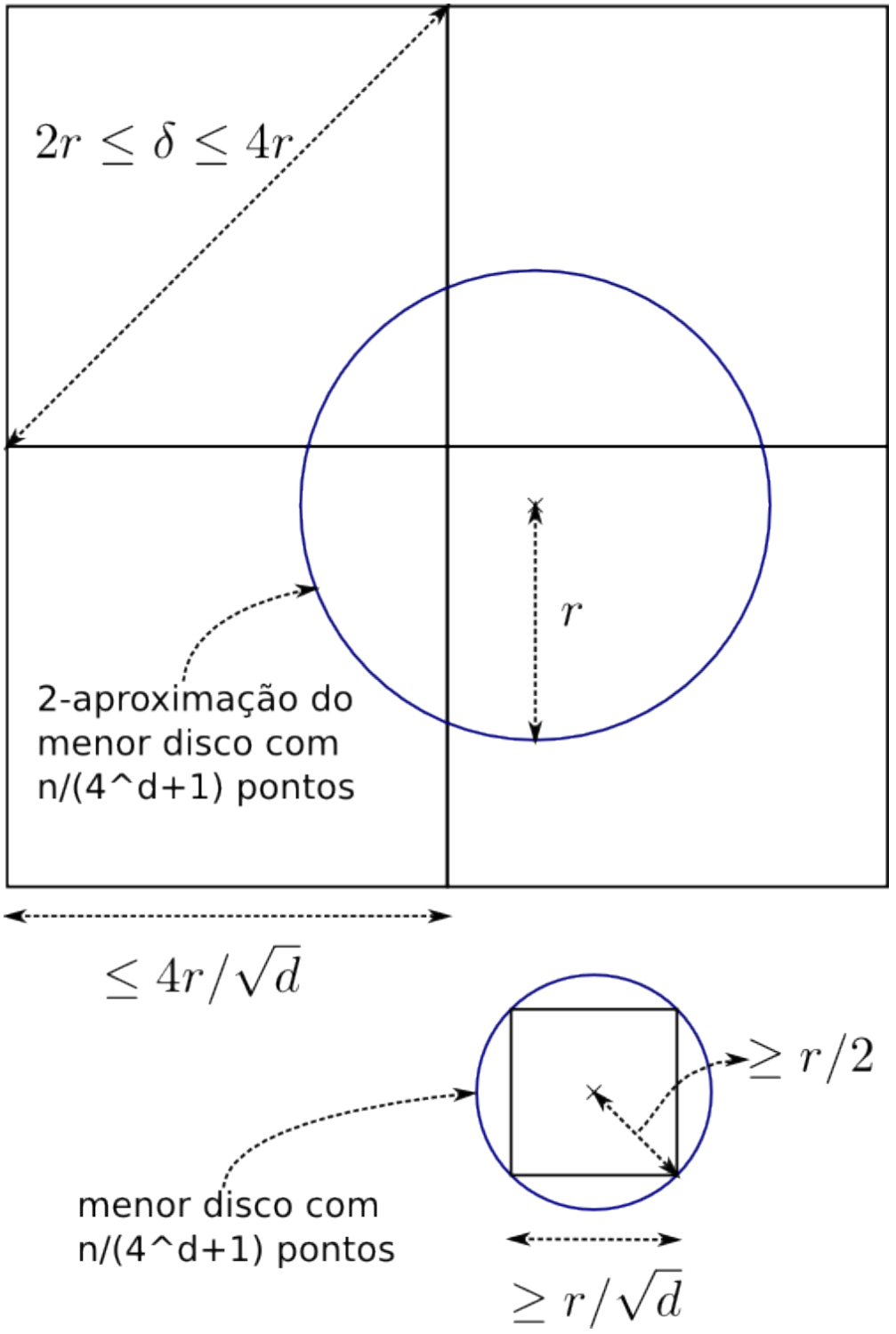
Divisão e Conquista



- Encontramos uma caixa quadtree com uma fração constante dos pontos.
- Então construímos recursivamente a quadtree dentro e fora desta caixa.
- Obtemos a recorrência

$$\begin{aligned} T(n) &= O(n) + 2\,T(n/c) \\ &= O(n \log n) \end{aligned}$$

- Como achar tal caixa?



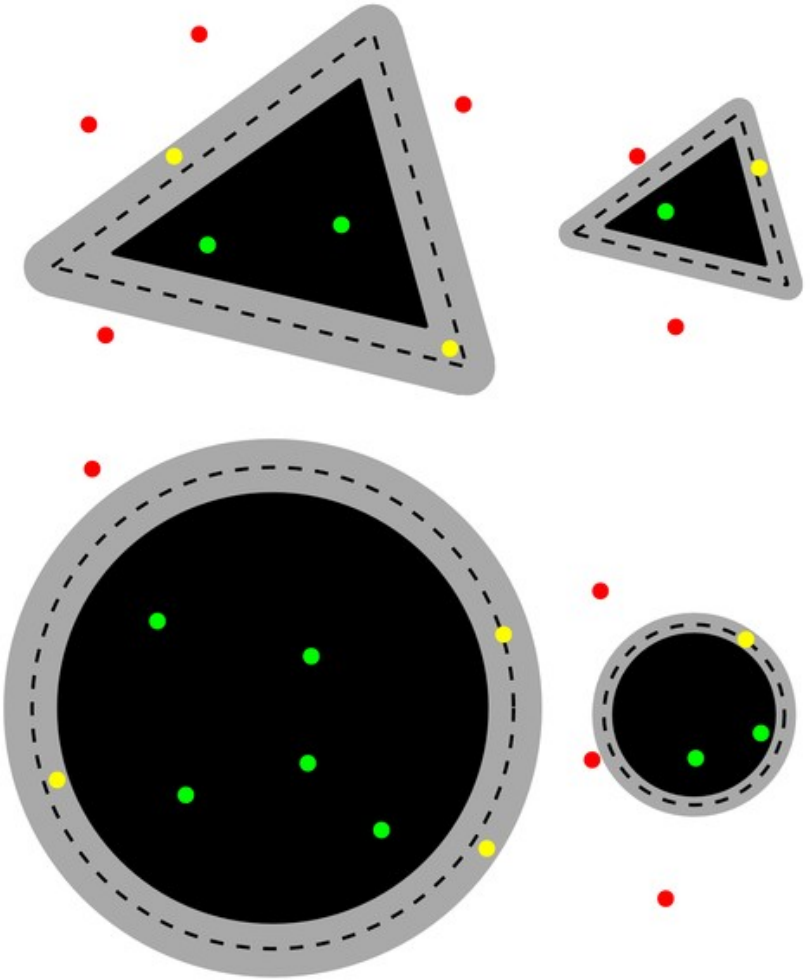
Achando a Caixa

- Obtemos uma 2-aproximação do menor disco com $n/(4^d+1)$ pontos.
- Achamos as 2^d caixas de diâmetro ~$2\mathrm{r}$ que contém o disco.
- A caixa com mais pontos obtida tem entre $n/(2^d(4^d+1))$ e $4^d n/(4^d+1)$ pontos (no plano $\mathrm{n}/68$ e $16n/17$).

Referências

- Notas de aula do Har-Peled:
  http://valis.cs.uiuc.edu/~sariel/teach/notes/aprx/
- Artigo que descreve e usa a quadtree compactada com índice:
  Sunil Arya, Guilherme D. da Fonseca, and David M. Mount.
  SIBGRAPI 2008.
  http://www.cos.ufrj.br/~fonseca/tradeoffs-sibgrapi.pdf
- Texto introdutório de geometria computacional que descreve a kd-tree e o básico de quadtree:
  Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. Computational Geometry: Algorithms and Applications, third edition, 2008.

Aproximação Geométrica



Aula 4

Tópicos:

- Lemas de Empacotamento
- Busca de Região

IMPA – Verão 2009

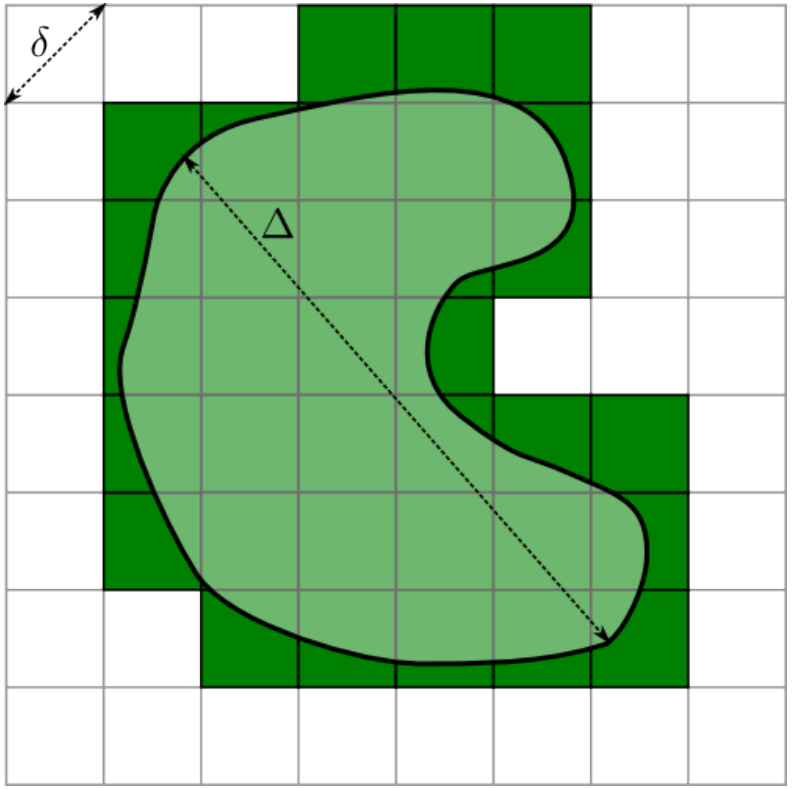
Guilherme D. da Fonseca

Lemas de Empacotamento

- Lema de empacotamento: define quantos objetos disjuntos podem interceptar uma determinada região.
- No nosso caso, os objetos são sempre caixas quadtree com diâmetro pelo menos δ.
- Veremos 2 lemas de empacotamento.

Lema 1: Se S é um conjunto de caixas quadtree disjuntas, cada uma com diâmetro mínimo δ, que interceptam uma região de diametro Δ, então

$$|S| = O\left(1 + \left(\frac{\Delta}{\delta}\right)^d\right).$$

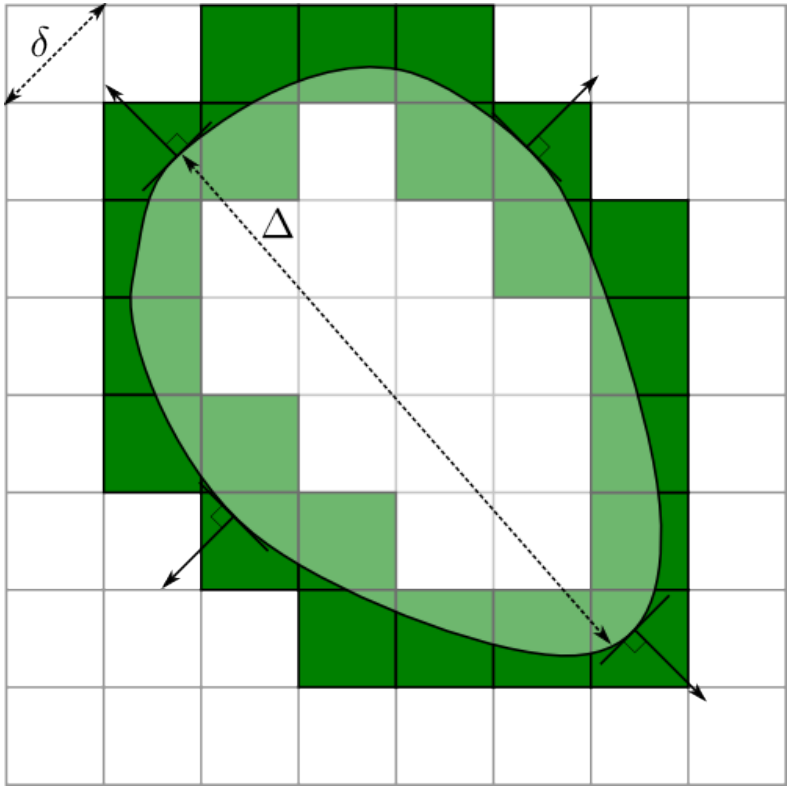


Prova:

- Basta considerarmos a grade, pois toda caixa quadtree de diâmetro mínimo δ contém alguma célula da grade.
- No máximo $1+\Delta/\delta$ caixas em cada direção.

Lema 2: Se S é um conjunto de caixas quadtree disjuntas, cada uma com diâmetro mínimo δ, que interceptam a borda de uma região convexa de diametro Δ, então

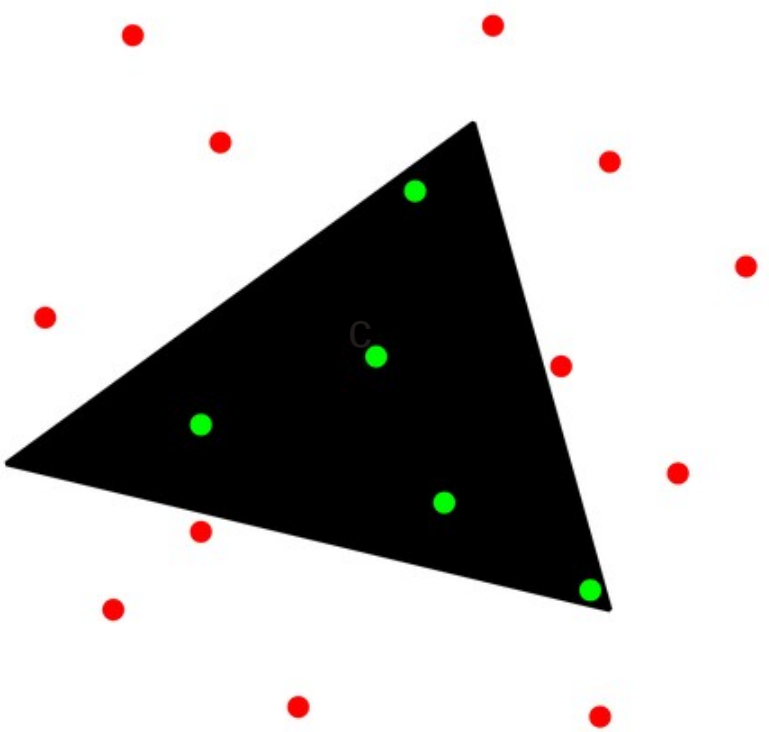
$$|S| = O\left(1 + \left(\frac{\Delta}{\delta}\right)^{d-1}\right).$$



Prova:

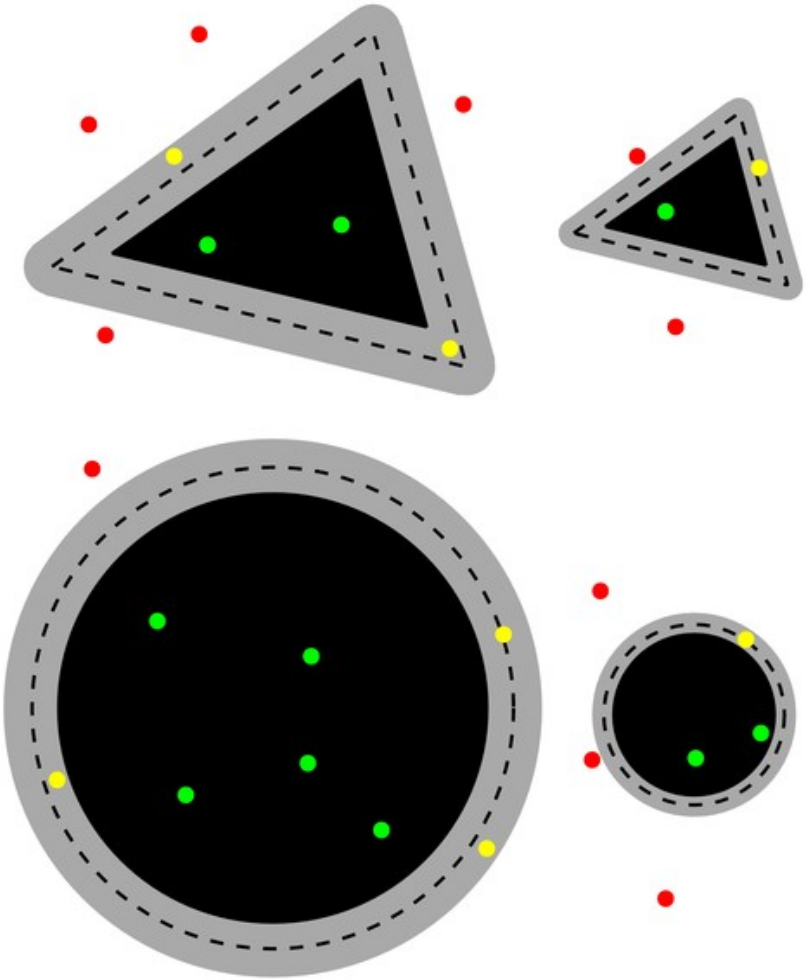
- Particionar a borda usando a face do cubo de direções apontada pela normal.
- Cada partição intercepta no máximo 2 células na direção normal a face do cubo correspondente.

Busca de Região



- Deseja-se preprocessar um conjunto de pontos para poder contar quantos pontos estão em uma região de consulta.
- Soluções exatas ineficientes para muitos tipos de região. Por exemplo, simplexos:
- Tamanho: $O(n)$.
- Consulta: $O(n^{1-1/d})$.

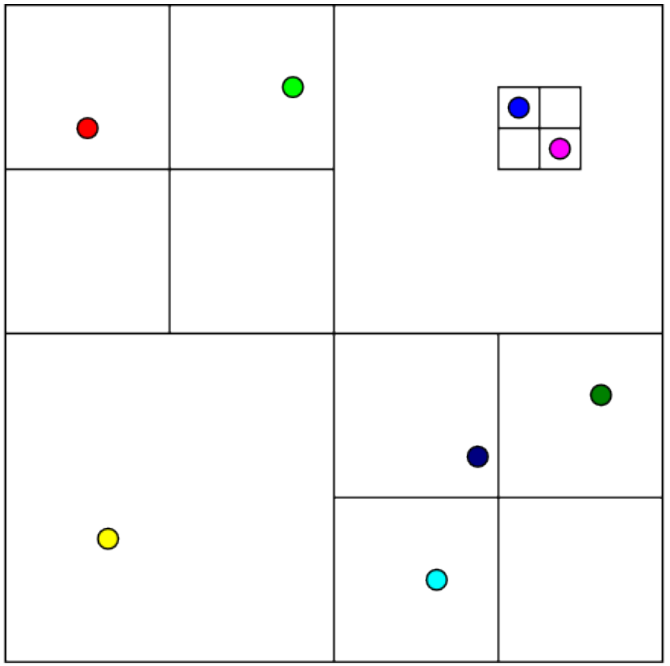
Busca de Região Aproximada

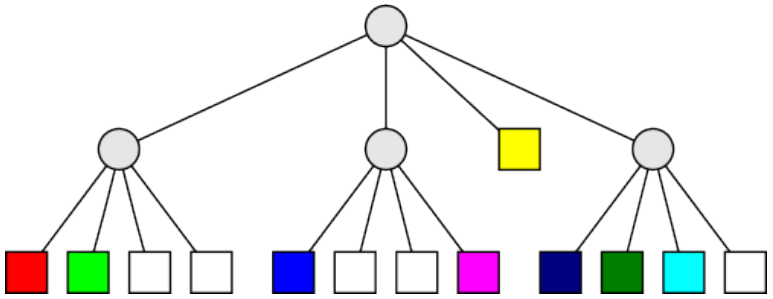


- Seja Δ o diâmetro da região de consulta.
- Pontos a uma distância $\varepsilon\Delta$ da borda da região podem ser contados ou não.
- Hipótese do custo unitário:

Dada uma caixa quadtree Q, é possível determinar em tempo $O(1)$ se $R \cap Q = \emptyset$, $Q \subseteq R$ ou nenhum dos dois.

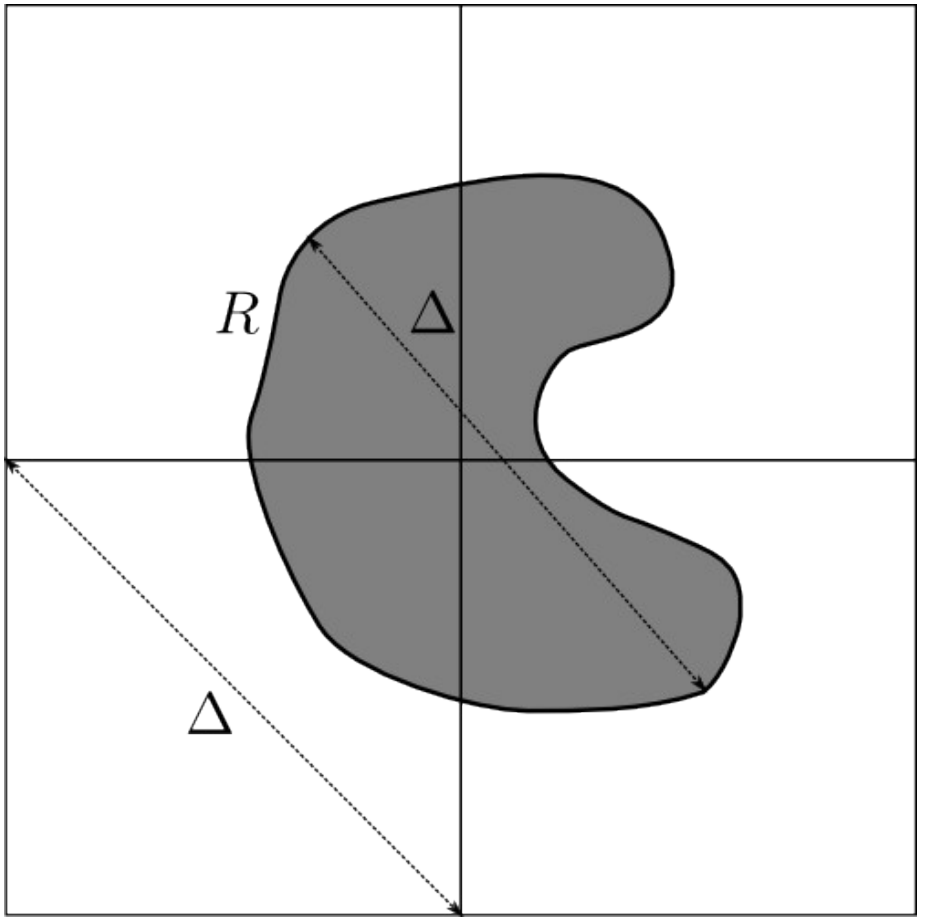
Regiões Arbitrárias





- Consideramos uma região de consulta R qualquer satisfazendo a hipótese do custo unitário.
- A estrutura de dados é uma quadtree compactada com índice, portanto:
- Tamanho: $O(n)$.
- Construção: $O(n \log n)$.
- Cada vértice armazena o número de pontos na célula.

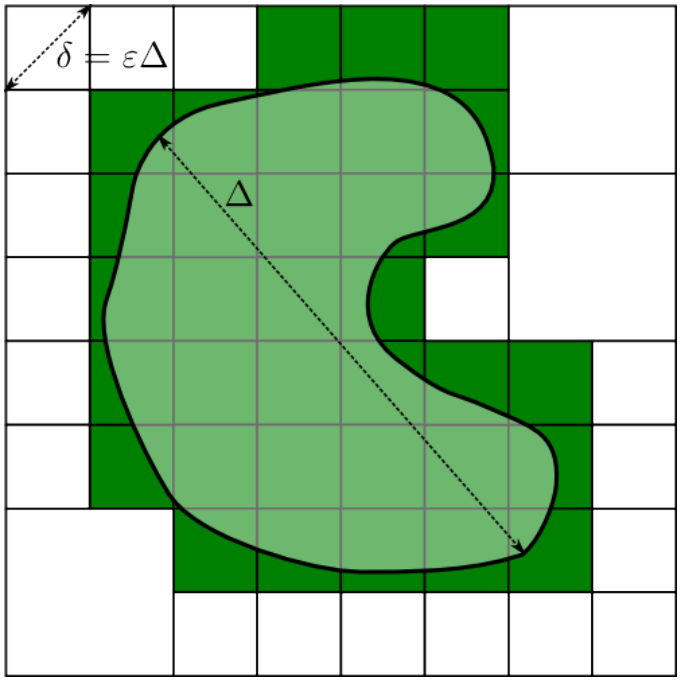
Consulta



- Usando busca de célula, localizamos em tempo $O(\log n)$ as 2^d caixas quadtree de diâmetro aproximadamente Δ que contém R.
- Respondemos a consulta separadamente para cada caixa Q destas caixas e somamos as respostas.

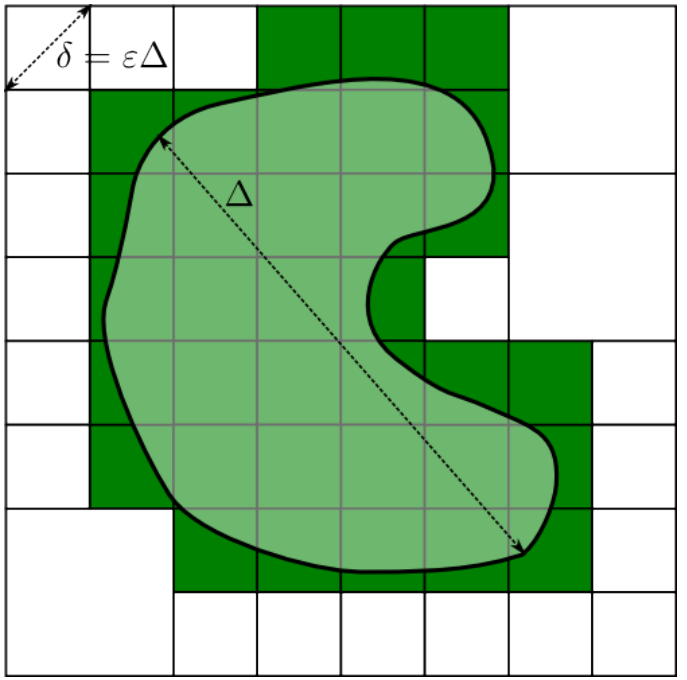
Consulta

- Se $R \cap Q = \emptyset$ ou Q está vazia, então retorne 0.
- Se o diâmetro de Q é menor que $\varepsilon\Delta$, então retorne o número de pontos em Q.
- Caso contrário, faça chamadas recursivas para as subdivisões de Q.



Consulta

- Se $R \cap Q = \emptyset$ ou Q está vazia, então retorne 0.
- Se o diâmetro de Q é menor que $\varepsilon\Delta$, então retorne o número de pontos em Q.
- Caso contrário, faça chamadas recursivas para as subdivisões de Q.
- Complexidade de tempo desta fase:

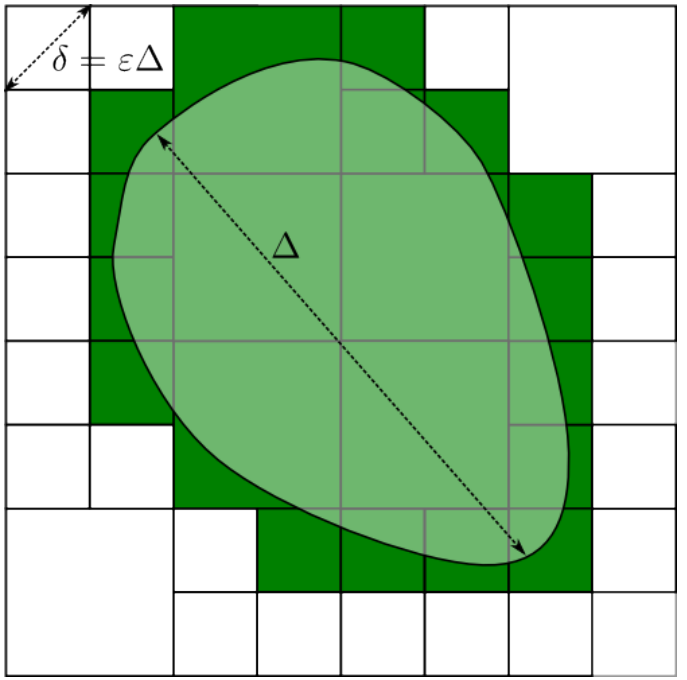


$$\sum_{i=0}^{\log(\Delta/\delta)} \left(\frac{\Delta}{2^i\delta}\right)^d = O\left(\left(\frac{1}{\varepsilon}\right)^d\right)$$

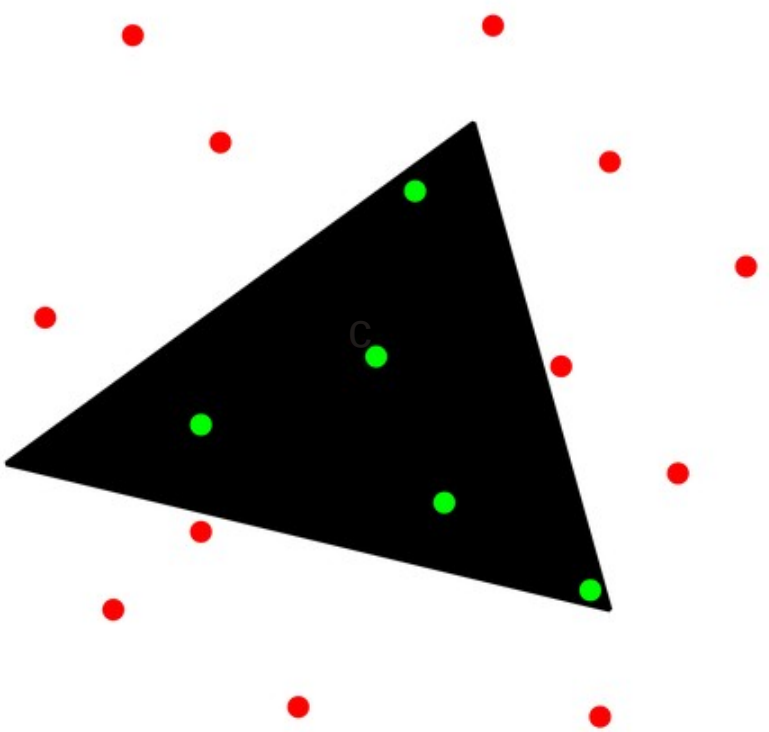
- Tempo total da consulta: $O\left(\log n + 1/\varepsilon^d\right)$

Regiões Convexas

- Se $R \cap Q = \emptyset$ ou Q está vazia, então retorne 0.
- ➔ Se $Q \subseteq R$, então retorne o número de pontos em Q.
- Se o diâmetro de Q é menor que $\varepsilon\Delta$, então retorne o número de pontos em Q.
- Caso contrário, faça chamadas recursivas para as subdivisões de Q.
- Tempo: $O\left(\log n + 1/\varepsilon^{d-1}\right)$

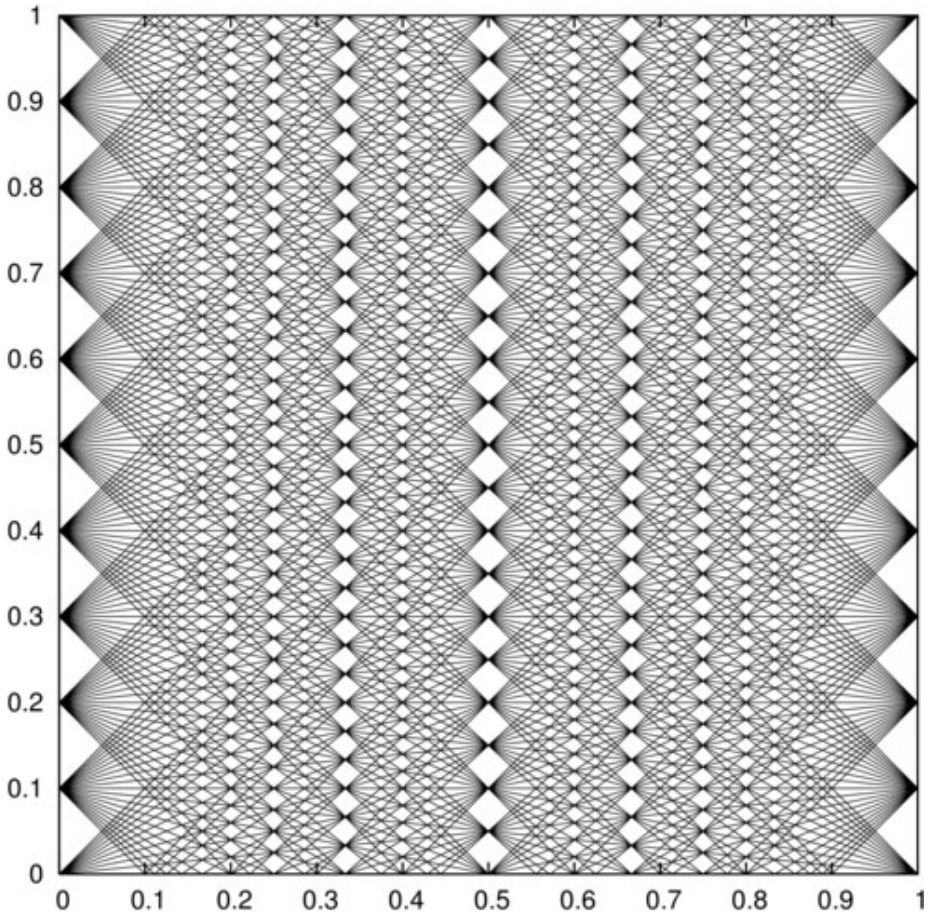


Resumo Até Agora



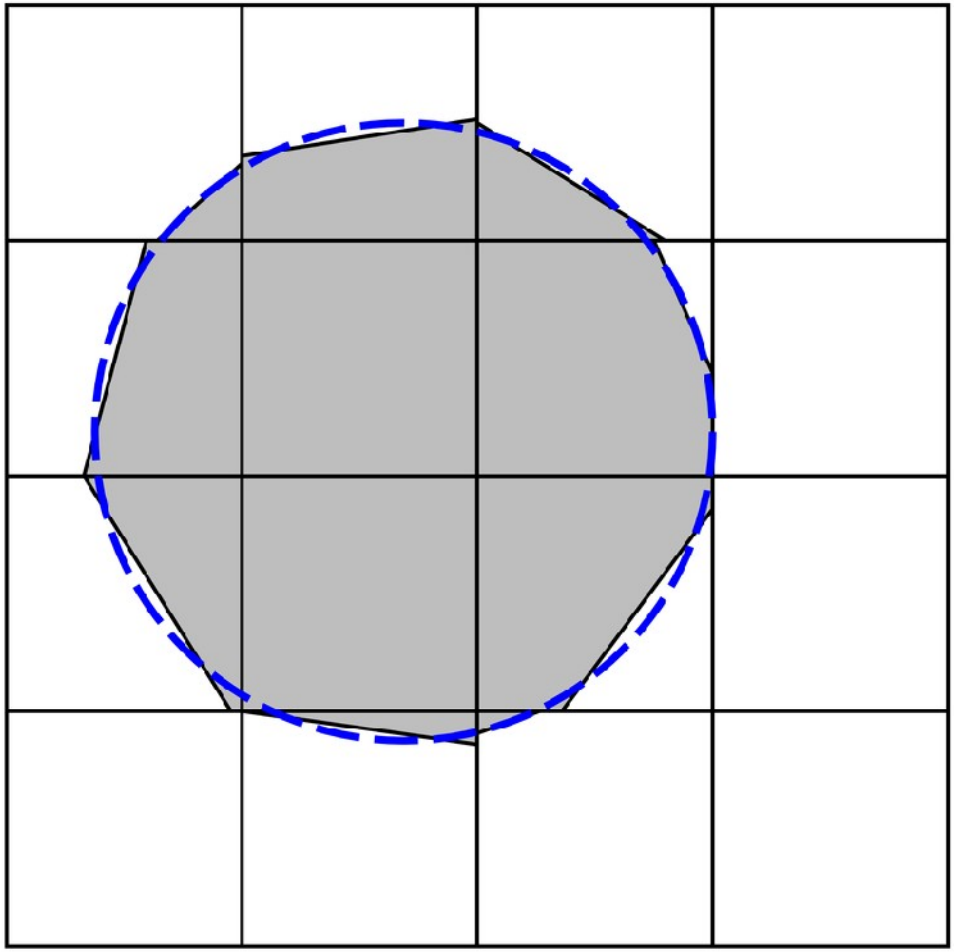
- Vimos estruturas com:
- Tamanho: $O(n)$.
- Construção: $O(n \log n)$.
- Consulta geral: $O(\log n + 1/\varepsilon^d)$.
- Consulta convexa: $O(\log n + 1/\varepsilon^{d-1})$.
- Veremos como acelerar a consulta para esferas usando mais espaço.

Estrutura de Semicaixas

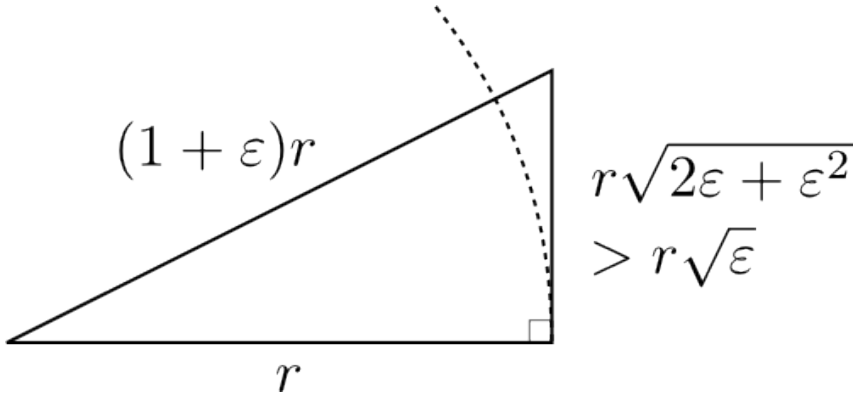


- Uma semicaixa é a insterseção de uma caixa com um semi-espaço.
- Podemos aproximar qualquer semicaixa da caixa unitária com erro ε e $O(1/\varepsilon)$ semicaixas.

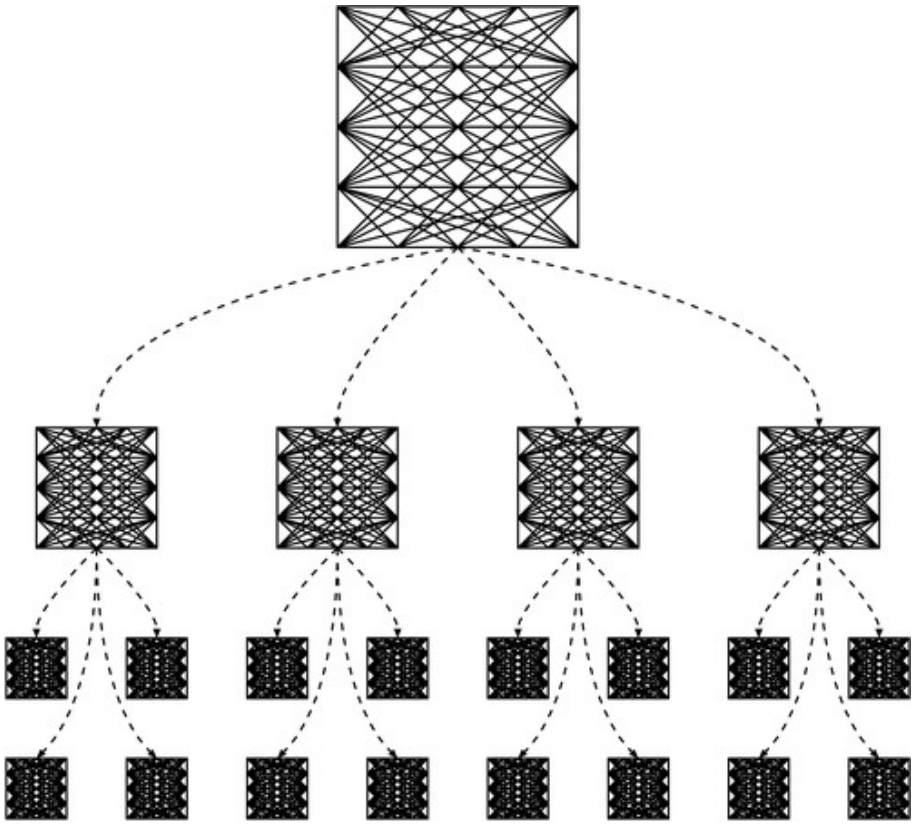
Semicaixas



- Podemos aproximar uma esfera com $O\left(1/\varepsilon^{(d-1)/2}\right)$ semicaixas.



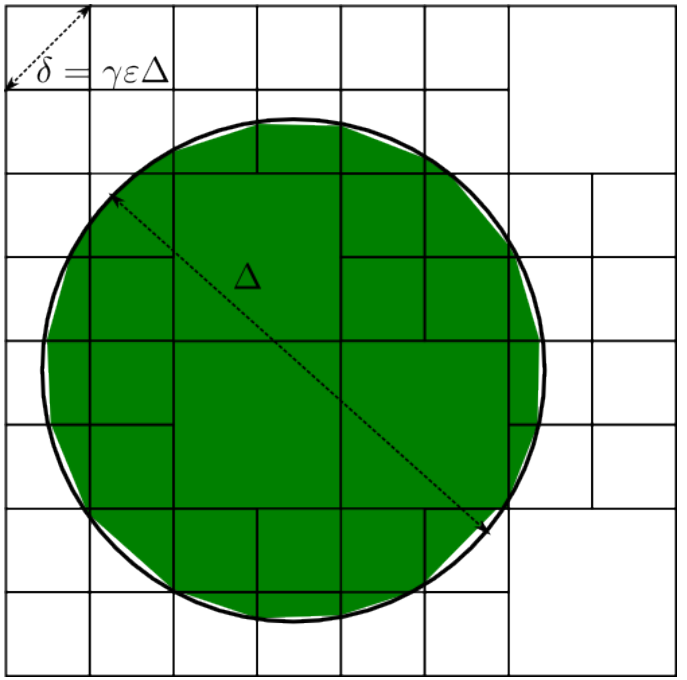
Quadtree de Semicaixas



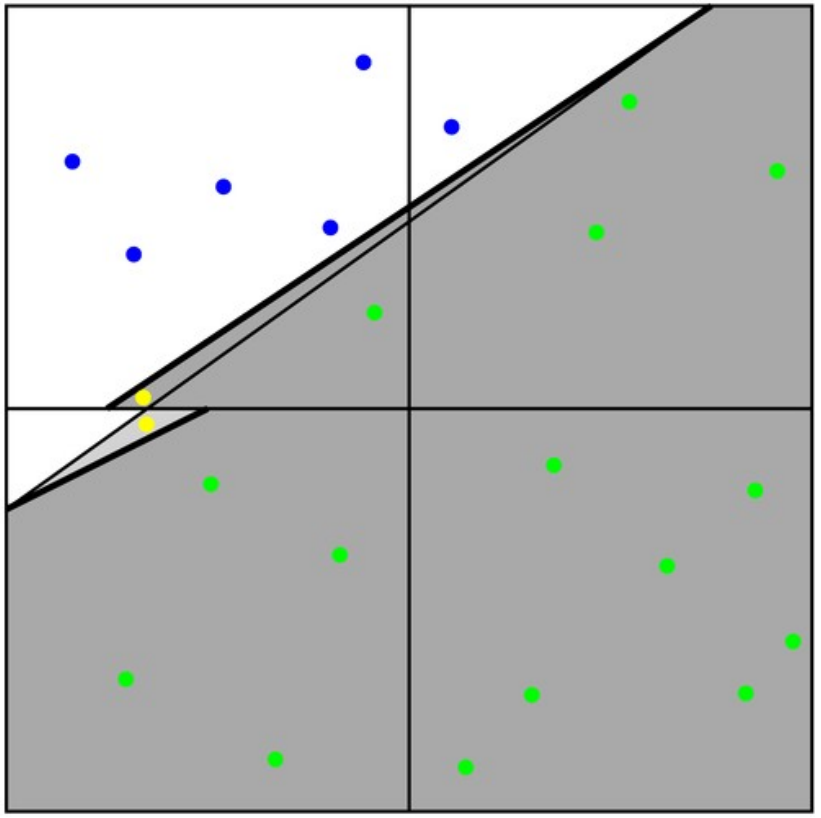
- Seja γ entre 1 e $1/\sqrt{\varepsilon}$ um parâmetro para controlar a relação espaço/tempo.
- Para cada célula com diâmetro δ da quadtree compactada, criamos uma estrutura de semicaixas com erro δ/γ.
- Tamanho: $O(n\gamma^d)$.

Consulta de Esferas

- Se $R \cap Q = \emptyset$ ou Q está vazia, então retorne 0.
- Se $Q \subseteq R$, então retorne o número de pontos em Q.
- Se o diâmetro de Q é menor que $\gamma\varepsilon\Delta$, então aproxime a borda com uma semicaixa.
- Caso contrário, faça chamadas recursivas para as subdivisões de Q.
- Tempo: $O\left(\log n + 1/(\gamma\varepsilon)^{d-1}\right)$



Construção



- Construção ingênua da quadtree de semicaixas leva tempo $O(n^2 \gamma^d)$.
- Porém, podemos começar a construção das folhas e construir cada semicaixa para os nós internos com 2^d consultas nos filhos.
- Construção: $O(n \gamma^d + n \log n)$.

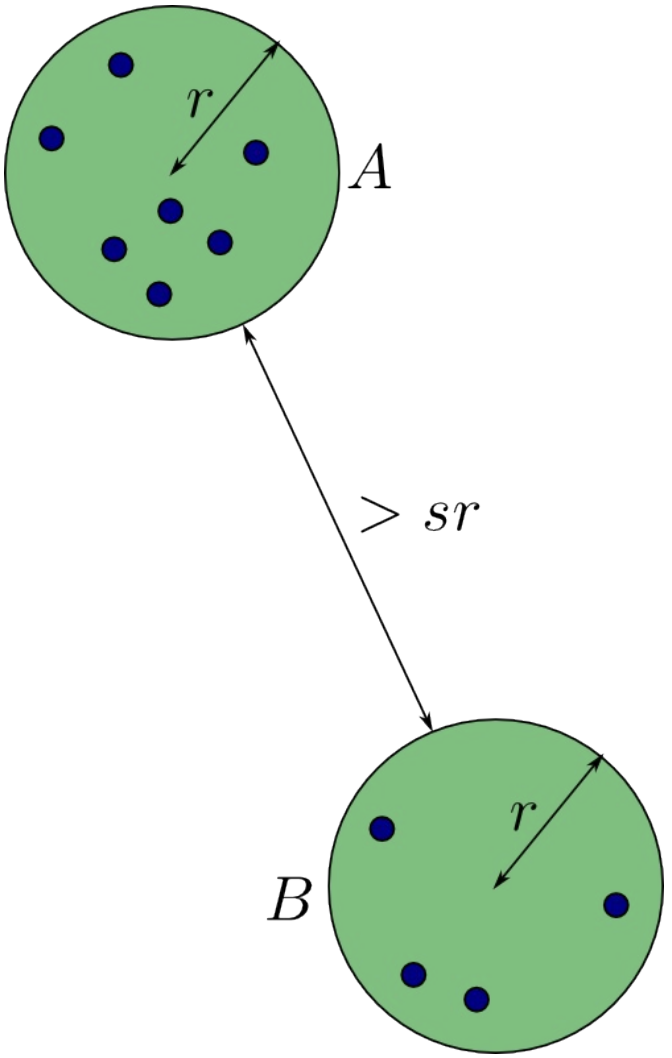
Resumo das Complexidades

- Vimos estruturas com:
- Tamanho: $O(n)$.
- Construção: $O(n \log n)$.
- Consulta geral: $O(\log n + 1/\varepsilon^d)$.
- Consulta convexa: $O(\log n + 1/\varepsilon^{d-1})$.
- Para γ entre 1 e $1/\sqrt{\varepsilon}$:
- Tamanho: $O(n\gamma^d)$.
- Consula esférica: $O(\log n + 1 \,/\, (\gamma\,\varepsilon)^{d-1})$.
- Construção: $O(n\,\gamma^d + n \log n)$.

Referências

- Sunil Arya and David M. Mount. Approximate range searching. Comput. Geom. Theory Appl., 17(3-4):135-152, 2000.
- Sunil Arya, Guilherme D. da Fonseca, and David M. Mount. SIBGRAPI 2008. http://www.cos.ufrj.br/~fonseca/tradeoffs-sibgrapi.pdf

Aproximação Geométrica



Aula 5

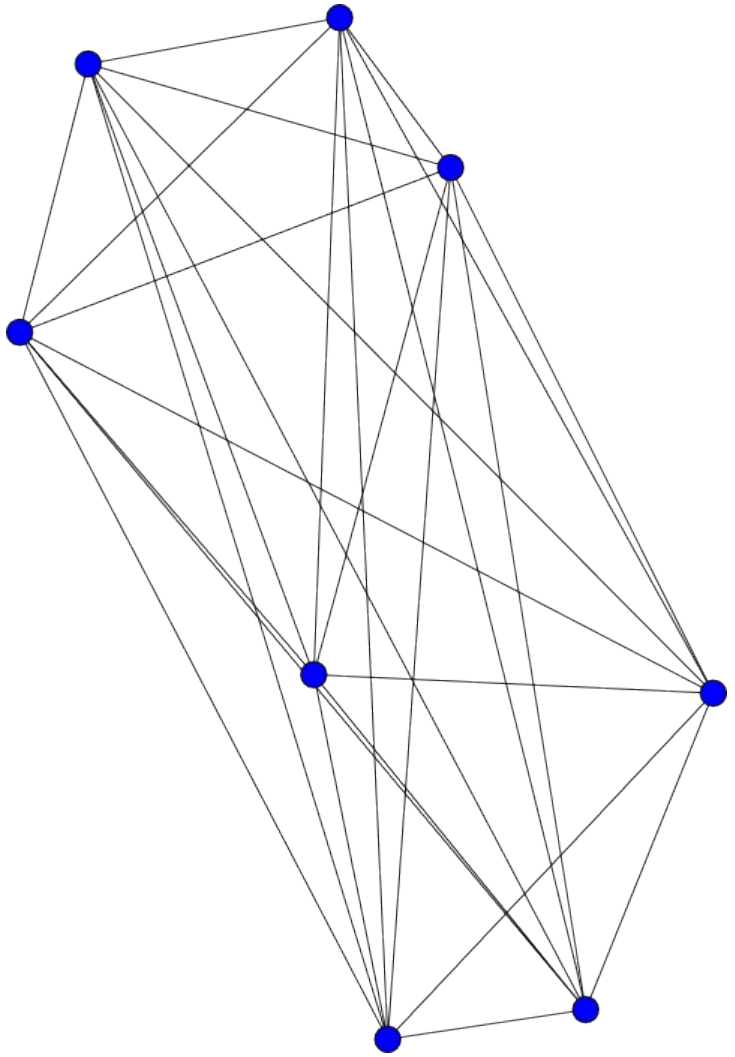
Tópicos:

- Decomposição em pares bem separados
- Par mais próximo
- Diâmetro

IMPA – Verão 2009

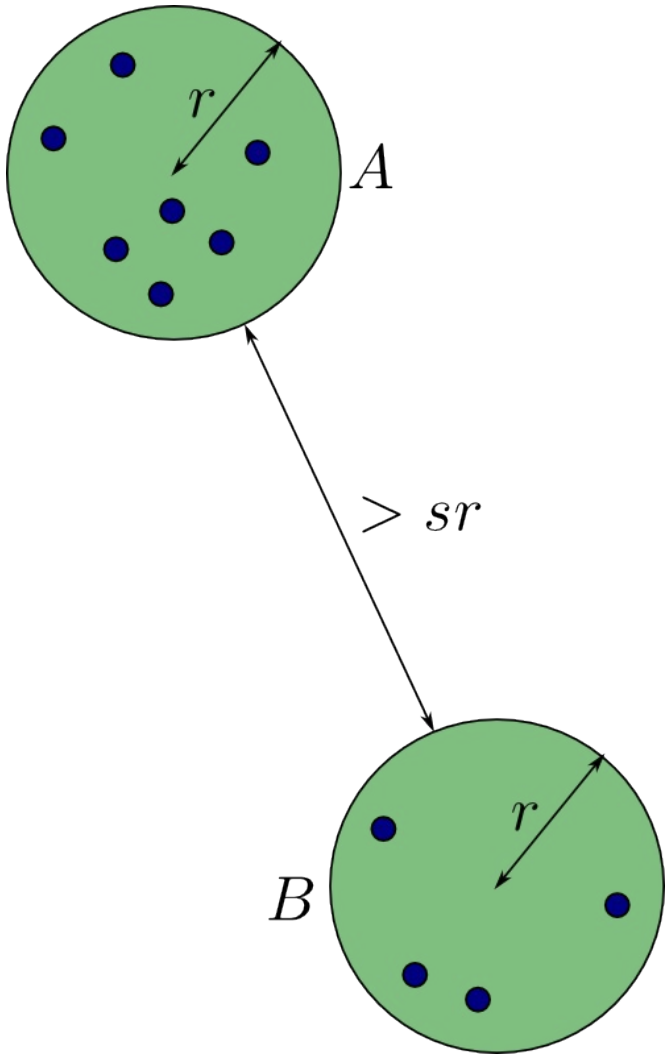
Guilherme D. da Fonseca

Distâncias Entre Pares



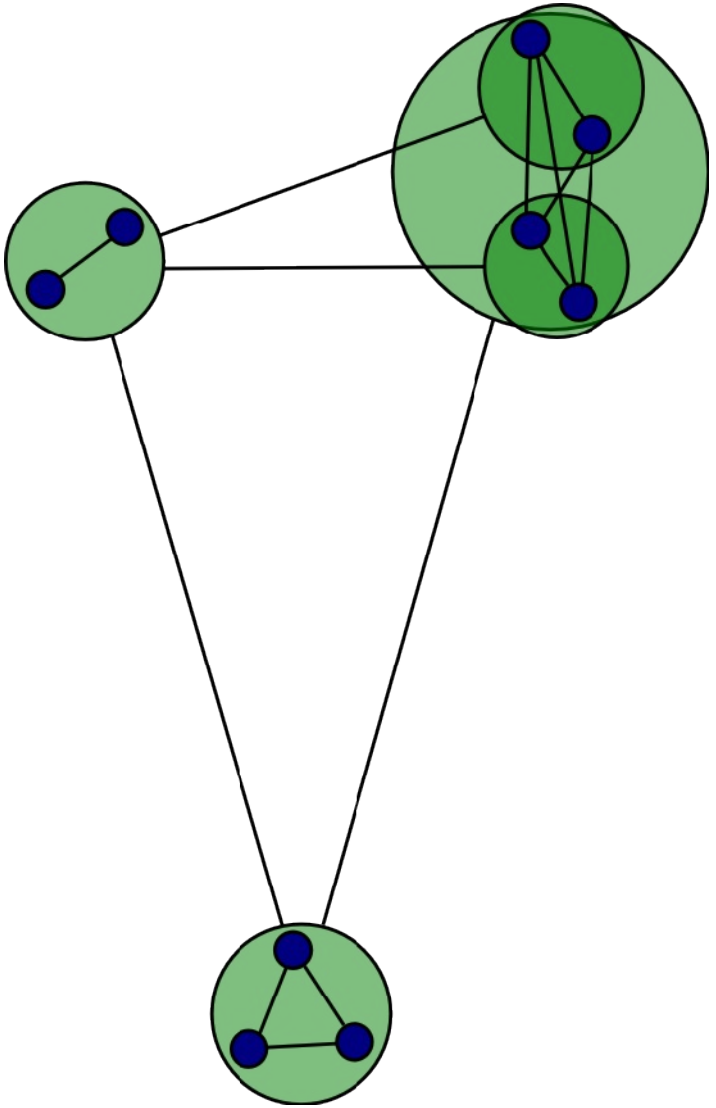
- Um conjunto de n pontos define $\Theta(n^2)$ distâncias.
- Em várias aplicações, todas as distâncias devem ser consideradas: simulação de sistemas gravitacionais, projeto de redes de interconexão...
- Como podemos aproximar as distâncias usando apenas $O(n)$ delas?

Par Bem Separado



- Dizemos que dois conjuntos de pontos A e B são um par bem separado com fator de separação $s \geq 1$ se:
  - O conjunto A e respectivamente B estão contidos em uma esfera de raio r.
  - A distância entre A e B é maior que sr.
- Se $a \in A, b \in B$, diz-se que o par A,B separa a e b.

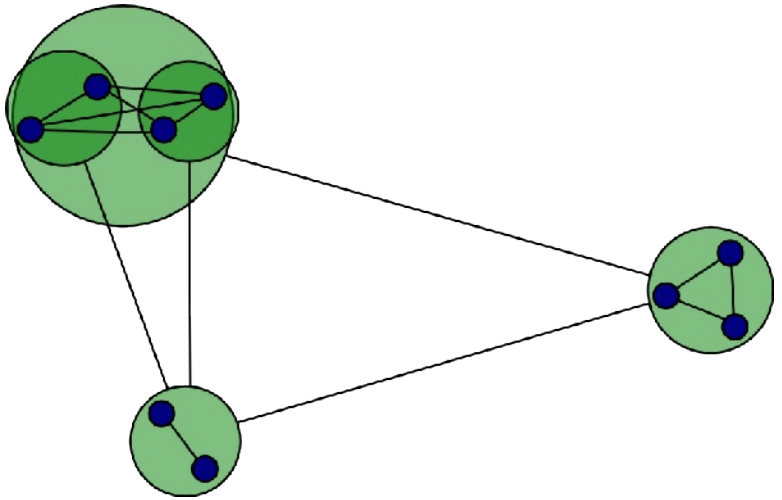
Decomposição em Pares Bem Separados



- Um conjunto de pares bem separados é uma decomposição em pares bem separados (WSPD) se todo par de pontos é separado por exatamente um par bem separado.
- Como construir uma WSPD?
- Qual o tamanho de uma WSPD?

Construção da WSPD

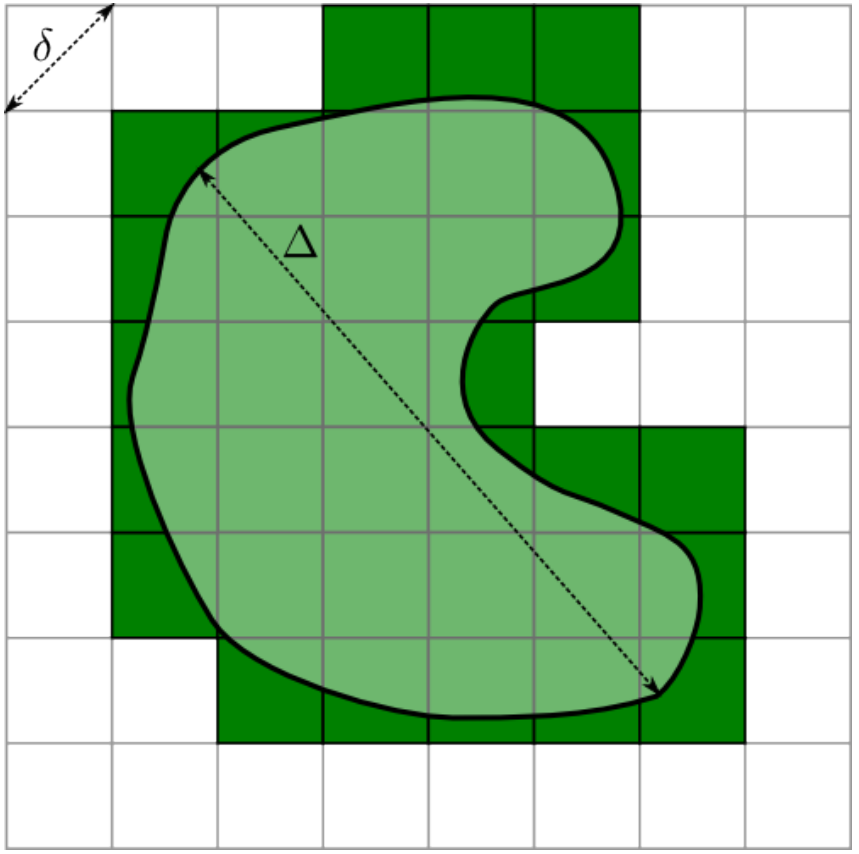
- Primeiro construímos uma quadtree compactada.
- Chamos então a rotina separar(raiz,raiz).



separar(A,B):

- Se A e B são bem separados, então imprima A,B e retorne.
- Seja A a maior caixa dentre A e B.
- Chame separar(Q,B) para toda subdivisão Q da caixa A.

Lema de Empacotamento



Lema: Se S é um conjunto de caixas quadtree disjuntas, cada uma com diâmetro mínimo δ, que interceptam uma região de diametro Δ, então

$$|S| = O\left(1 + \left(\frac{\Delta}{\delta}\right)^d\right).$$

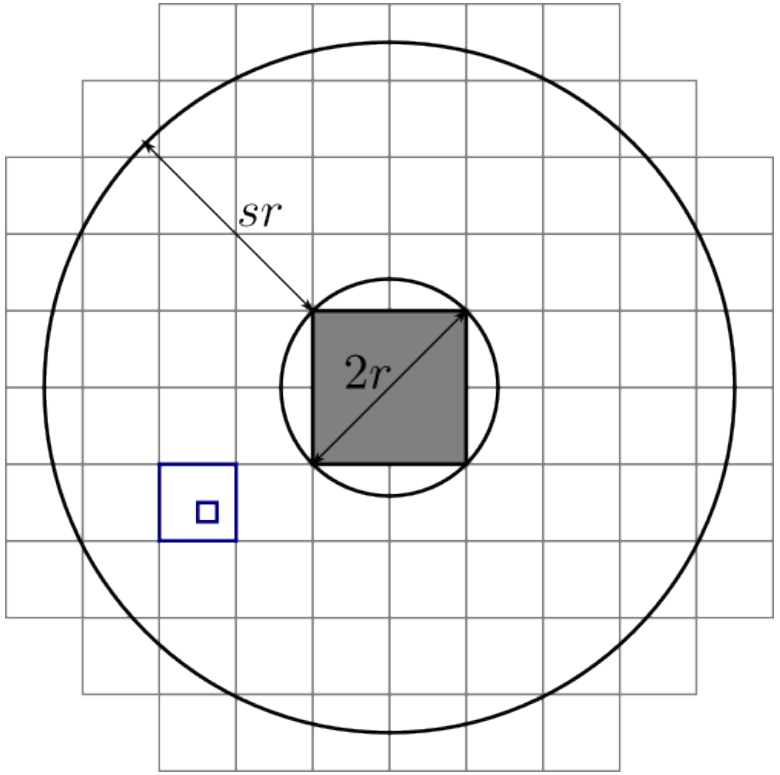
Tamanho da WSPD

separar(A,B):

- Se A e B são bem separados, então imprima A,B e retorne.
- Seja A a maior caixa dentre A e B.
- Chame separar(Q,B) para toda subdivisão Q da caixa A.

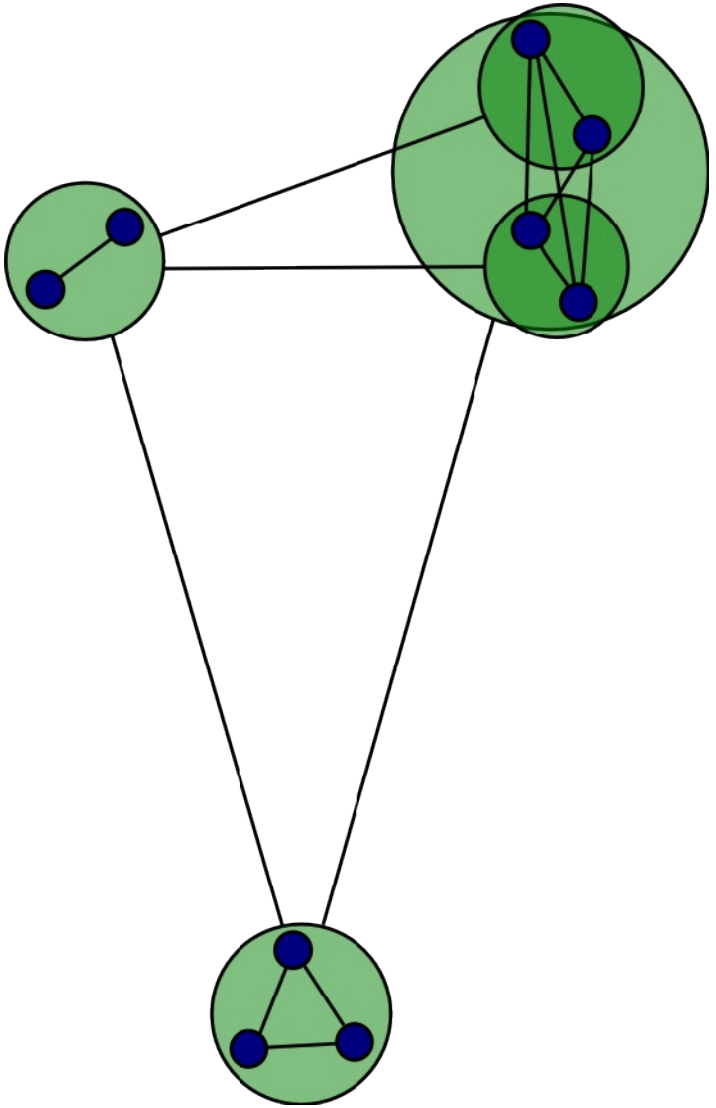
- A caixa menor tem pelo menos metade do tamanho da maior.
- No caso de caixas vindas de compactação, considere a maior caixa com o mesmo conjunto de pontos.
- Pelo lema, o número de chamadas recursivas envolvendo uma dada caixa e outras menores ou iguais é $O(s^d)$.

Tamanho da WSPD



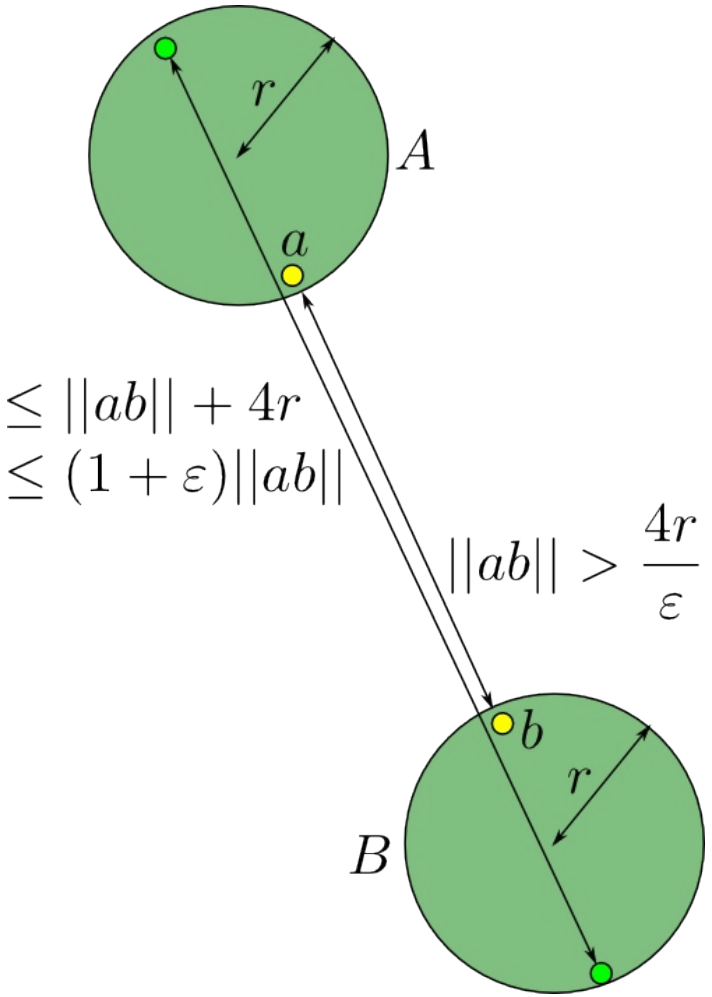
- A caixa menor tem pelo menos metade do tamanho da maior.
- No caso de caixas vindas de compactação, considere a maior caixa com o mesmo conjunto de pontos.
- Pelo lema, o número de chamadas recursivas envolvendo uma dada caixa e outras menores ou iguais é $O(s^d)$.

Resumo da WSPD



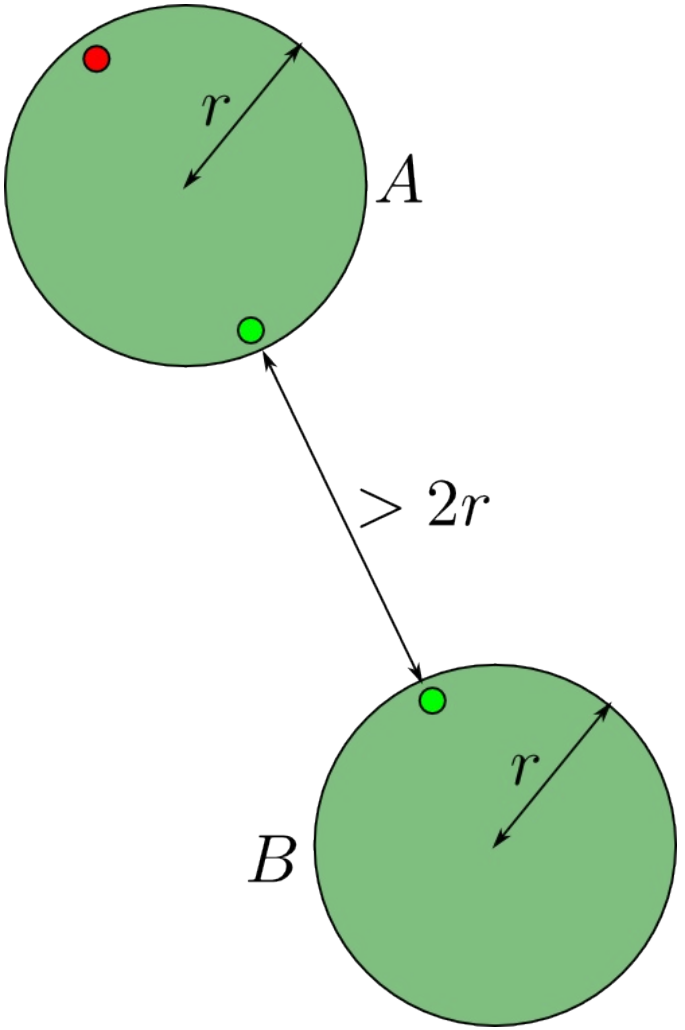
- Tamanho: $O(ns^d)$.
- Construção: $O(n \log n + ns^d)$.
- Chamamos de representante de um conjunto A um elemento arbitrário deste conjunto.
- Há várias definições semelhantes de WSPD com s diferente por um fator constante.

Diâmetro



- Podemos determinar uma $(1+\varepsilon)$-aproximação do diâmetro usando uma WSPD com $s=4/\varepsilon$.
- Examinamos a distância entre os representantes dos pares bem separados e retornamos a maior delas.
- Tempo: $O(n \log n + n/\varepsilon^d)$.

Par Mais Próximo

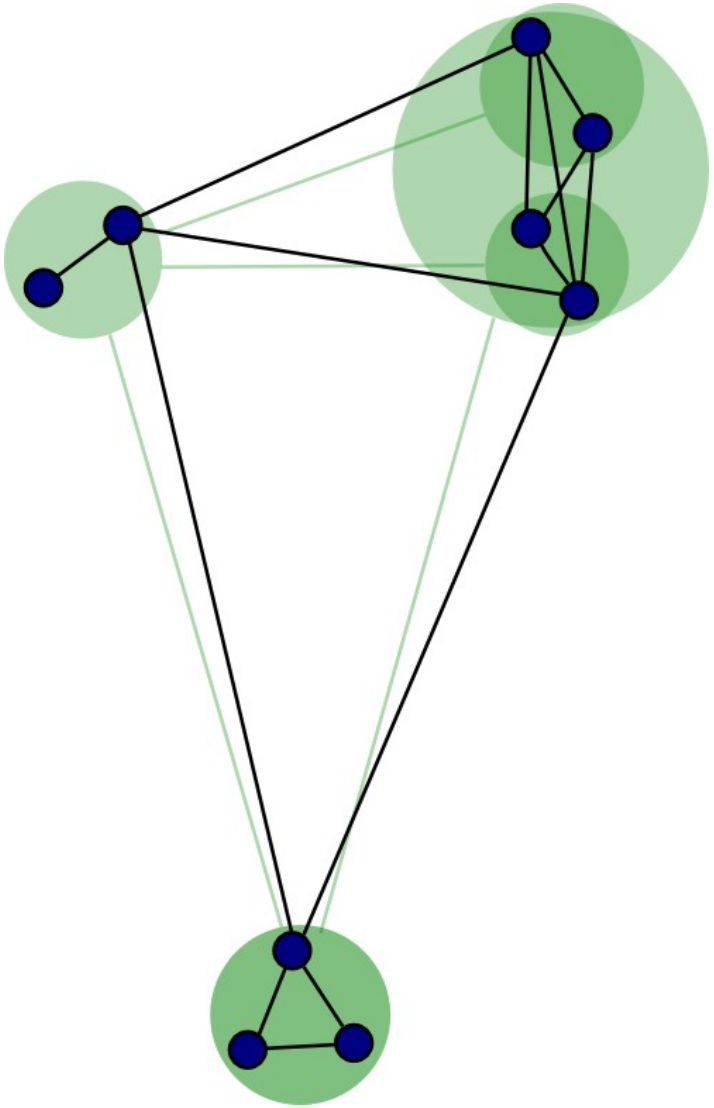


- O par mais próximo é o par de pontos p,q distintos que minimiza ||pq||.
- O par mais próximo é separado por um par de conjuntos unitários na WSPD com $s \geq 2$.
- Portanto, dada a WSPD, podemos determinar exatamente o par mais próximo em tempo $O(n)$.

Referências

- A decomposição em pares bem separados foi introduzida em: Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. J. ACM, 42(1):67-90, 1995.
- A construção usando a quadtree compactada é baseada nas notas do Sariel Har-Peled
  http://valis.cs.uiuc.edu/~sariel/teach/notes/aprx/
  e do David Mount
  http://www.cs.umd.edu/class/spring2007/cmsc754/Lects/754lects.pdf
- Uma construção em tempo linear para conjuntos de pontos com espalhamento polinomial está em:
  Timothy Chan. Well-separated pair decomposition in linear time?. Information Processing Letters, 107:138-141, 2008

Aproximação Geométrica



Aula 6

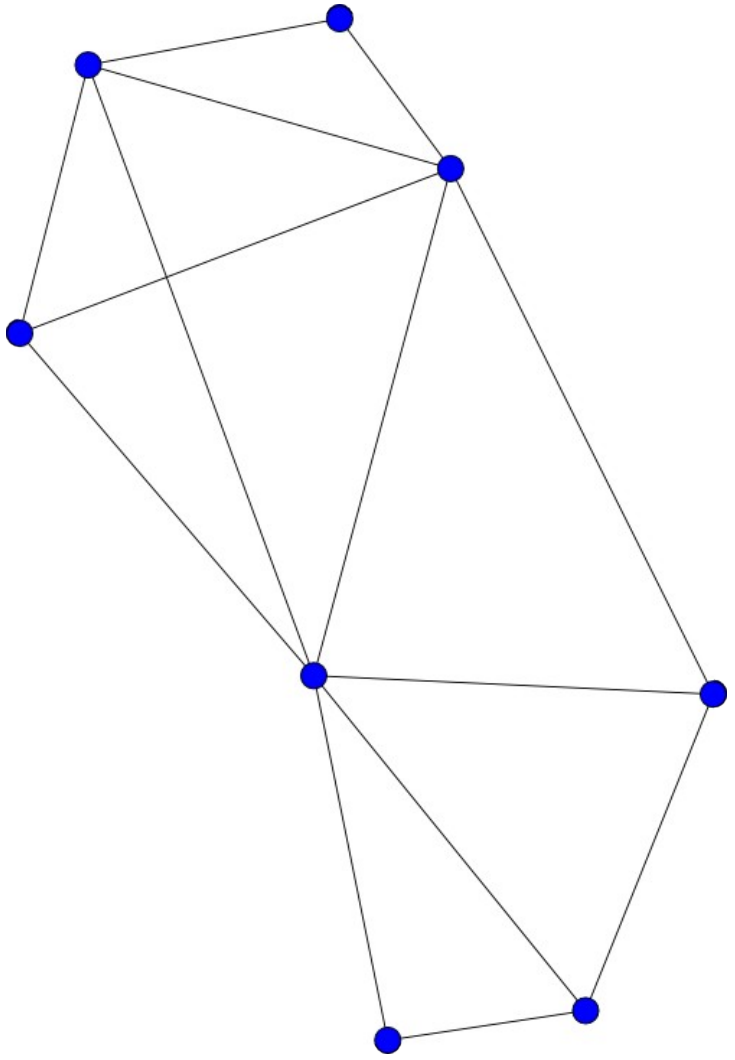
Tópicos:

- Spanners
- Árvore geradora mínima

IMPA – Verão 2009

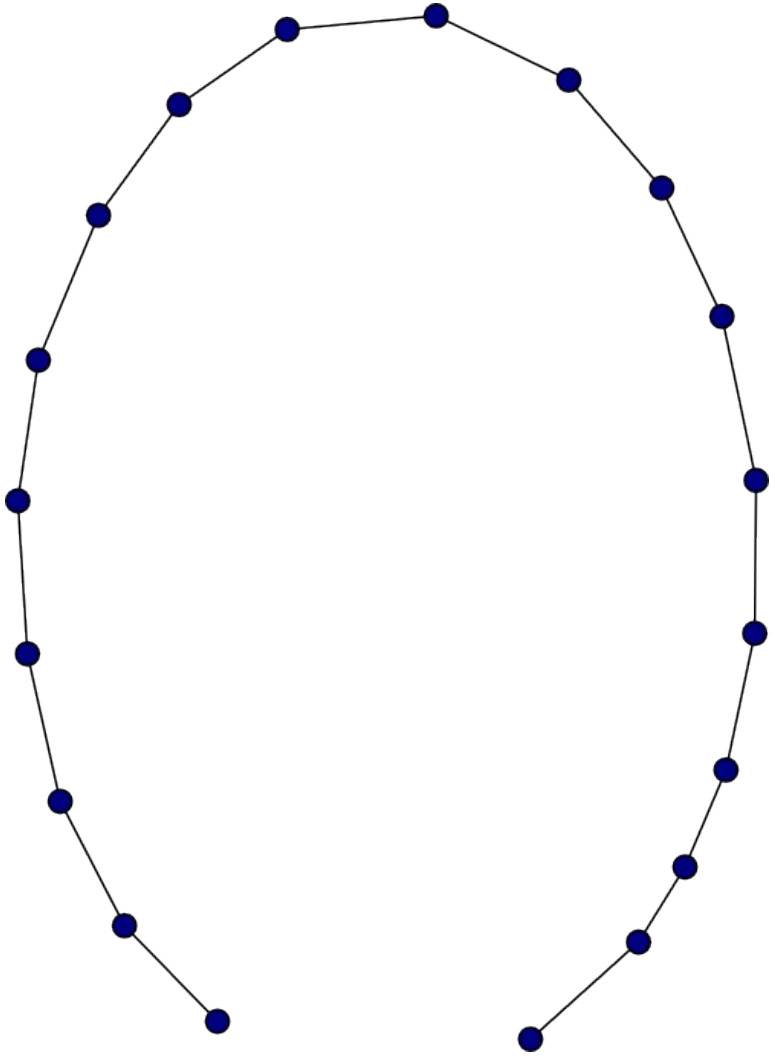
Guilherme D. da Fonseca

Spanner



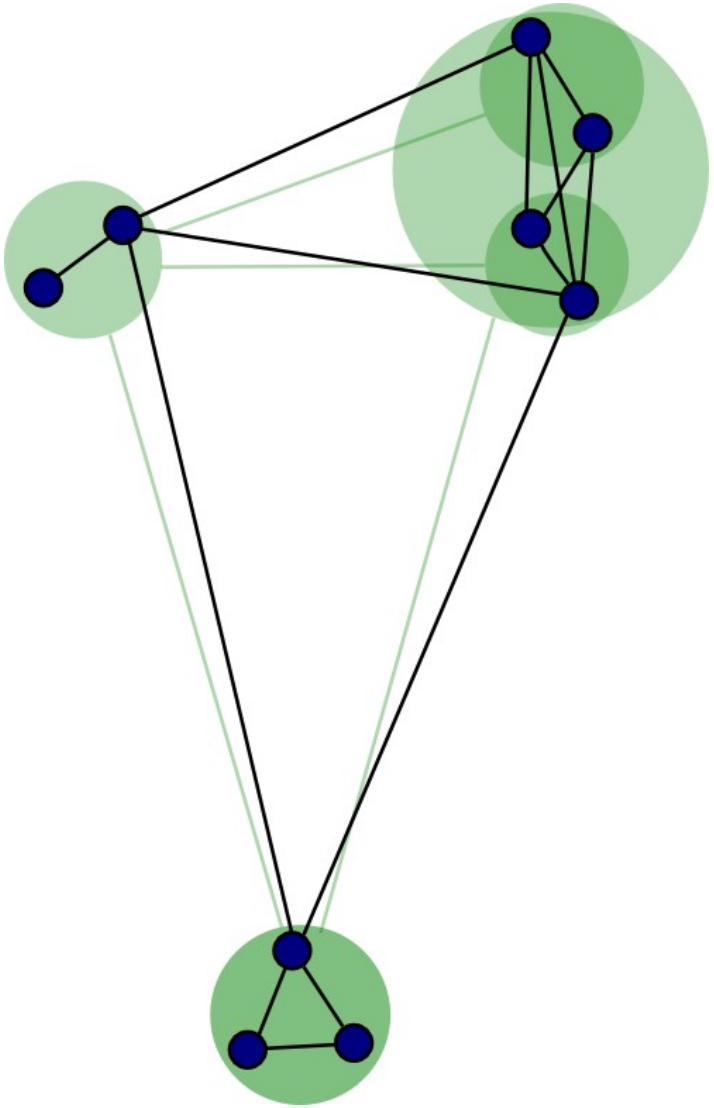
- Seja $||pq||$ a distância Euclidiana entre dois pontos e $||pq||_G$ a distância entre dois vértices em um grafo G.com os pontos como vértices.
- Um grafo G é um spanner com distorção t se para todo p,q vale que:
  $$||pq|| \leq ||pq||_G \leq t||pq||$$
- Queremos um spanner com poucas arestas.

Observações Sobre Spanners



- O grafo completo é um spanner com distorção 1, porém tem $O(n^2)$ arestas.
- A árvore geradora mínima não tem distorção limitada, portanto não é um spanner.
- A triangulação de Delaunay é um spanner plano. A distorção é desconhecida, mas está entre 1.5932 e 1.998.

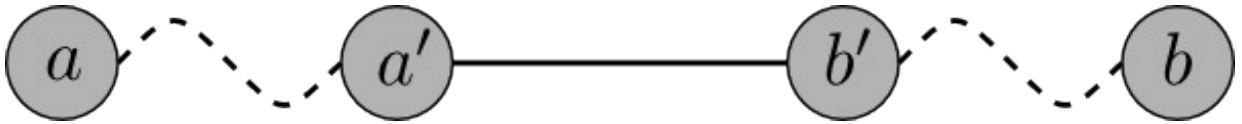
Construção



- Primeiro construímos uma decomposição em pares bem separados.
- Para cada par de representantes da WSPD, criamos uma aresta no spanner.
- Precisamos relacionar a distorção t com o fator de separação s.

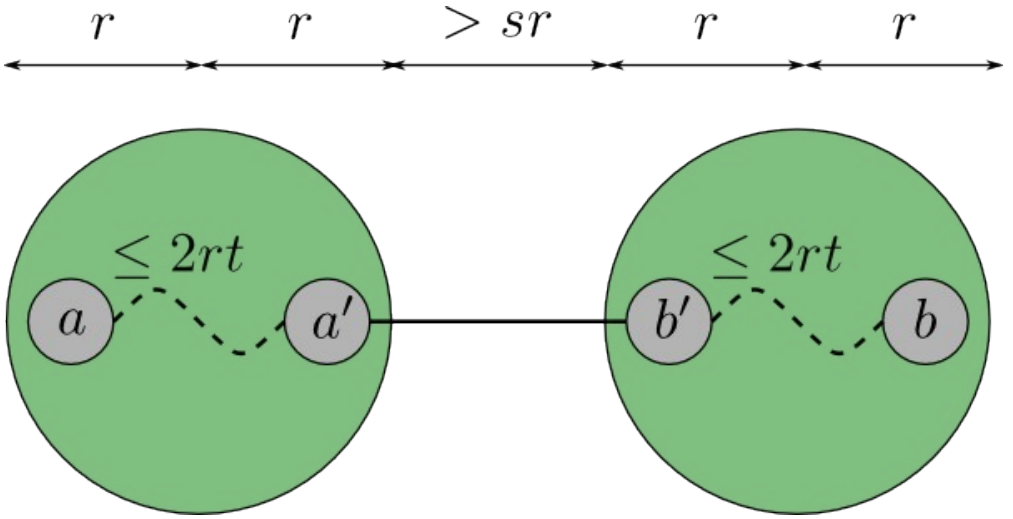
Prova

- A prova usa um caminho específico, que não é necessariamente o mais curto.
- Usamos indução no número de arestas do caminho.
- Seja a' o representante de a no par que separa a de b.
- O caminho de a até b é definido recursivamente como o caminho de a até a', seguido da aresta a'b' e do caminho de b' até b.
- O caso base de caminhos com uma aresta é trivial.



Queremos mostrar:

$$||ab||_G \leq t||ab||$$



Por indução temos:

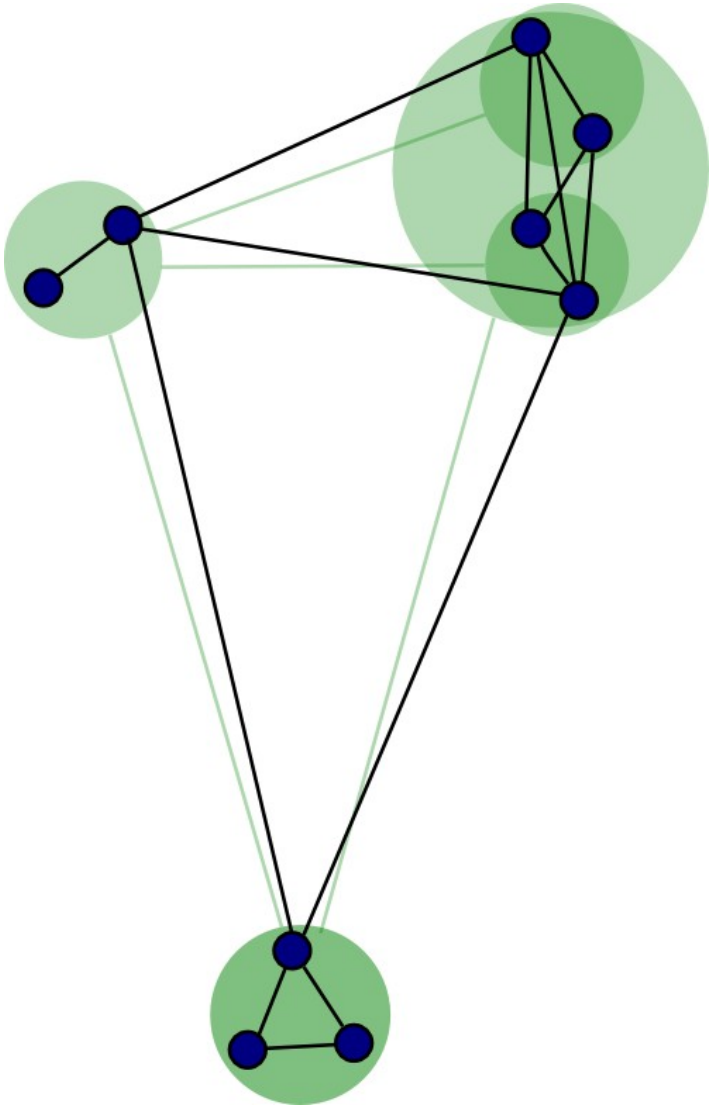
$$||ab||_G \leq 4rt + 4r + ||ab|| = 4r(t+1) + ||ab||$$

Usando que $r < \dfrac{||ab||}{s}$ temos:

$$||ab||_G \leq \left(4\frac{t+1}{s} + 1\right)||ab||$$

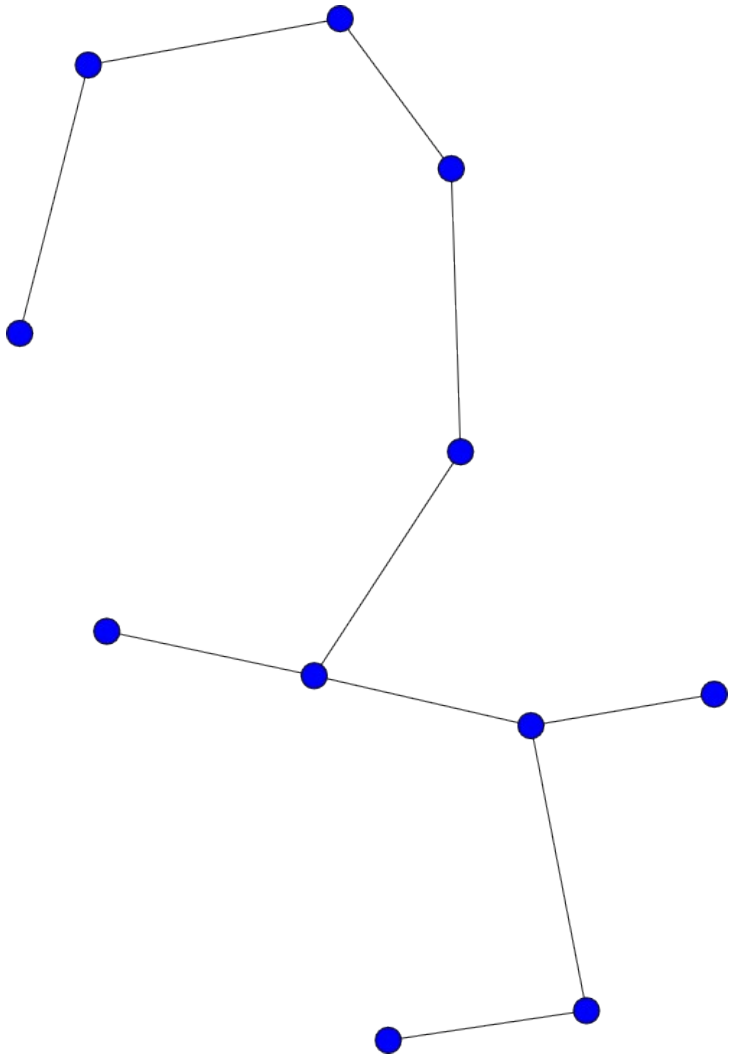
Fazendo $s = 4\dfrac{t+1}{t-1}$ concluímos a prova.

Complexidades



- Usamos:
$$s = 4\frac{t+1}{t-1}$$
- Fazendo $t = 1+\varepsilon$, temos um spanner com distorção $1+\varepsilon$.
- Tamanho: $O(n/\varepsilon^d)$.
- Construção: $O(n \log n + n/\varepsilon^d)$.

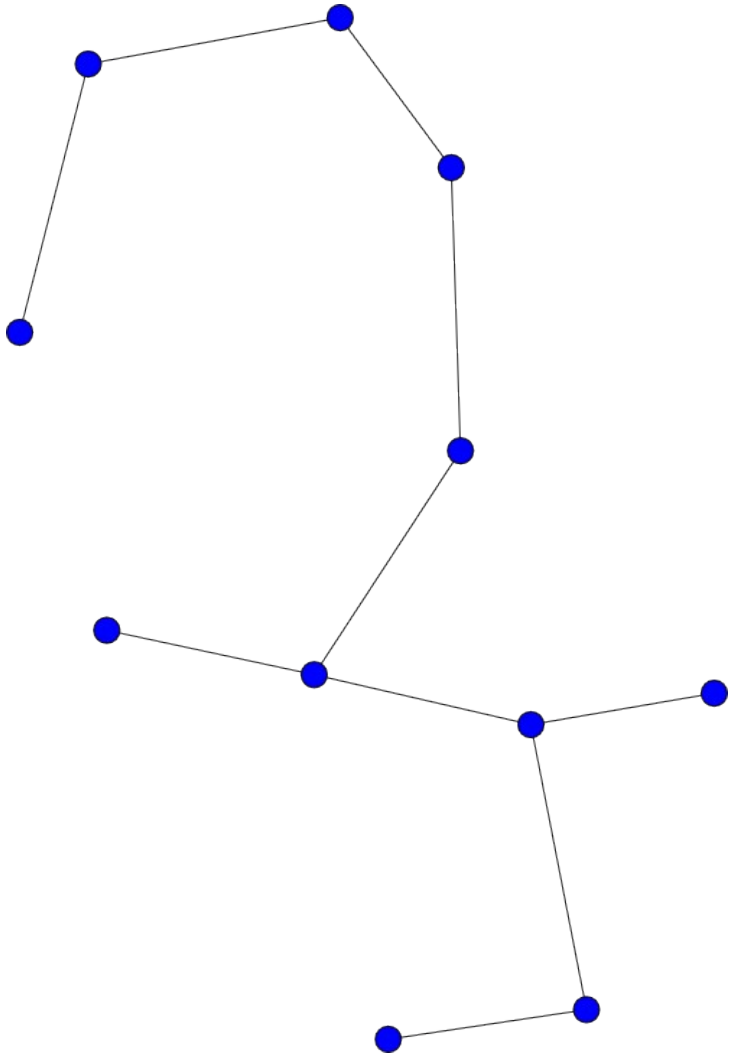
Árvore Geradora Mínima



- A árvore geradora mínima é o grafo conexo acíclico com os pontos como vértices que minimiza a soma dos comprimentos das arestas.
- No plano, pode ser construída em tempo $O(n \log n)$ usando a triangulação de Delaunay.
- Para dimensões maiores que 2, não se conhece algoritmo quase linear.

Árvore Geradora Mínima

- Em um grafo com n vértices e m arestas, a árvore geradora mínima pode ser construída em tempo $O(n \log n + m)$.
- Queremos achar uma árvore cuja soma dos comprimentos seja no máximo $1+\varepsilon$ vezes maior que o ótimo.
- Basta construir um spanner com $t=1+\varepsilon$ e achar a árvore geradora mínima no spanner.



Prova

- Considere o seguinte subgrafo do spanner: para cada aresta uv da árvore geradora mínima, as arestas do caminho mais curto de u para v pertencem ao subgrafo.
- Este subgrafo é conexo e tem peso no máximo igual a $1+\varepsilon$ vezes o peso da árvore geradora mínima.
- A árvore geradora mínima do spanner tem peso no máximo igual ao peso deste subgrafo.

- Tempo: $O(n \log n + n/\varepsilon^d)$.

Referências

- Livro somente sobre spanners geométricos:
  Giri Narasimhan and Michiel Smid. Geometric Spanner Networks. Cambridge University Press, New York, NY, USA, 2007.
- Notas do Sariel Har-Peled
  http://valis.cs.uiuc.edu/~sariel/teach/notes/aprx/
- Notas do David Mount
  http://www.cs.umd.edu/class/spring2007/cmsc754/Lects/754lects.pdf