Bancos de Dados Geográficos

#07 – Métodos de Acesso Multidimensionais

Dr. Gilberto Ribeiro de Queiroz <gribeiro@dpi.inpe.br>

Dr. Eymar Lopes <eymar@dpi.inpe.br>

Classificação dos Métodos de Acesso

- Métodos de acesso tradicionais, convencionais ou unidimensionais:
  - BST, B-Tree e Hash.

- Métodos de acesso multidimensionais ou espaciais:
  - *k*-d tree, Grid, Quadtree e R-tree.

k-d tree

Bentley (1975)

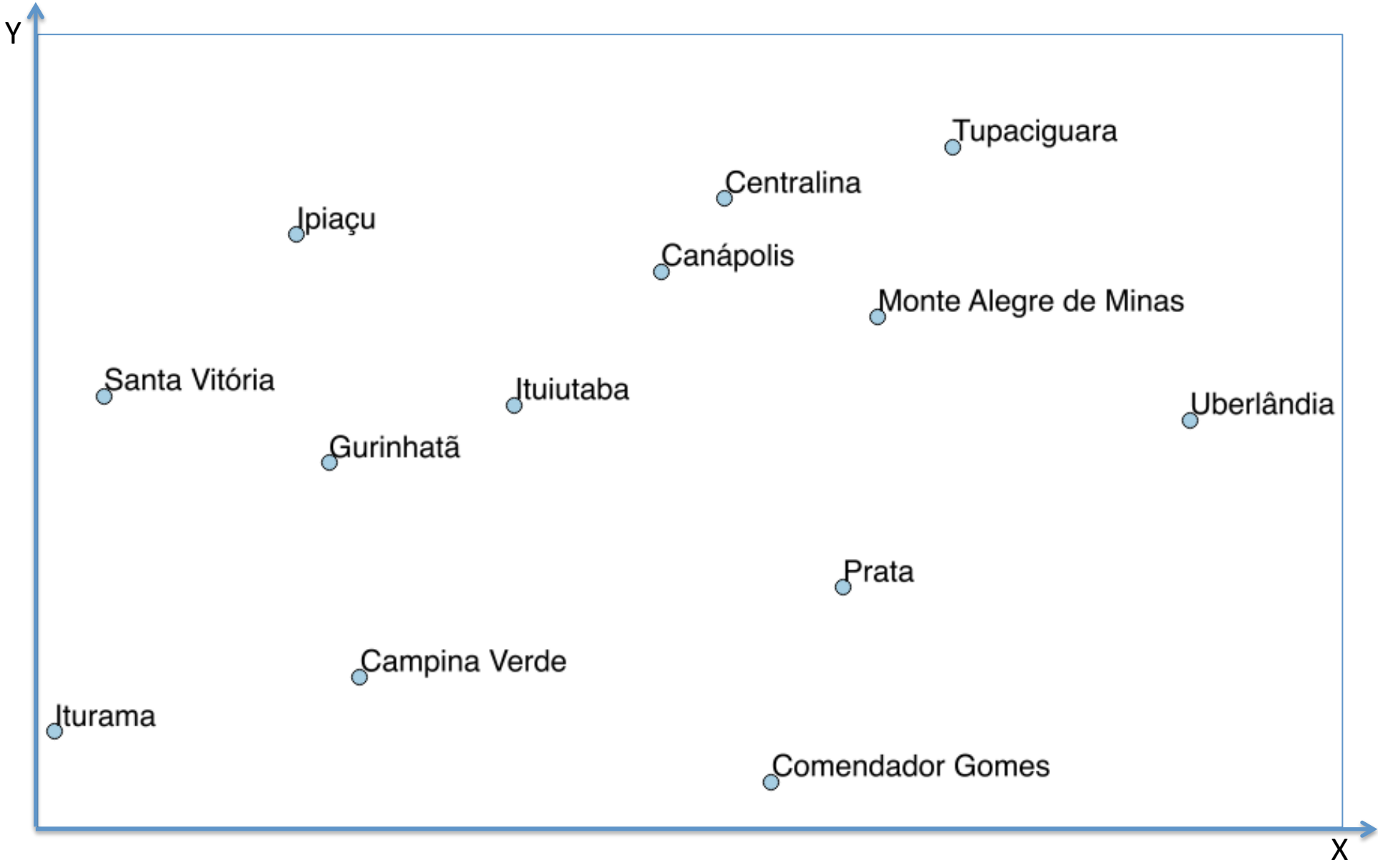
O que é uma *k*-d tree?

- Árvore binária de pesquisa multidimensional.
- *k*: dimensionalidade do espaço de busca.
- Complexidades:
  - Construção: $O(n \log_2 n)$
  - Pesquisa exata: $O(\log_2 n)$
  - Consultas Intervalo: $O(k \times N^{1-\frac{1}{k}})$
  - 1-vizinho mais próximo: $O(\log_2 n)$
  - Inserção 1-elemento: $O(\log_2 n)$
  - Remoção de 1-elemento: $O(\log_2 n)$

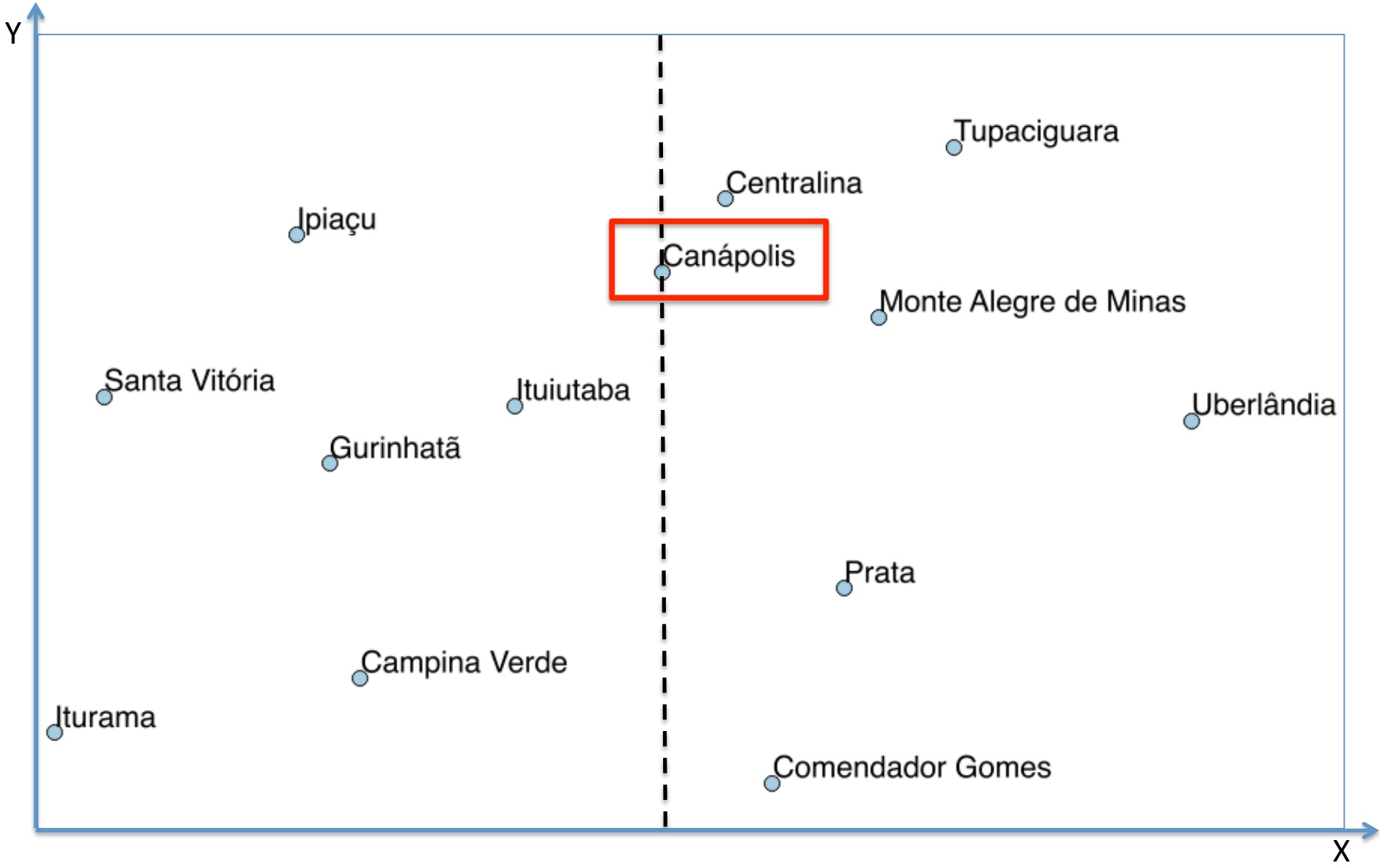
Se usado um algoritmo O(n) para encontrar a mediana em cada nível da construção

Construindo uma *k*-d tree

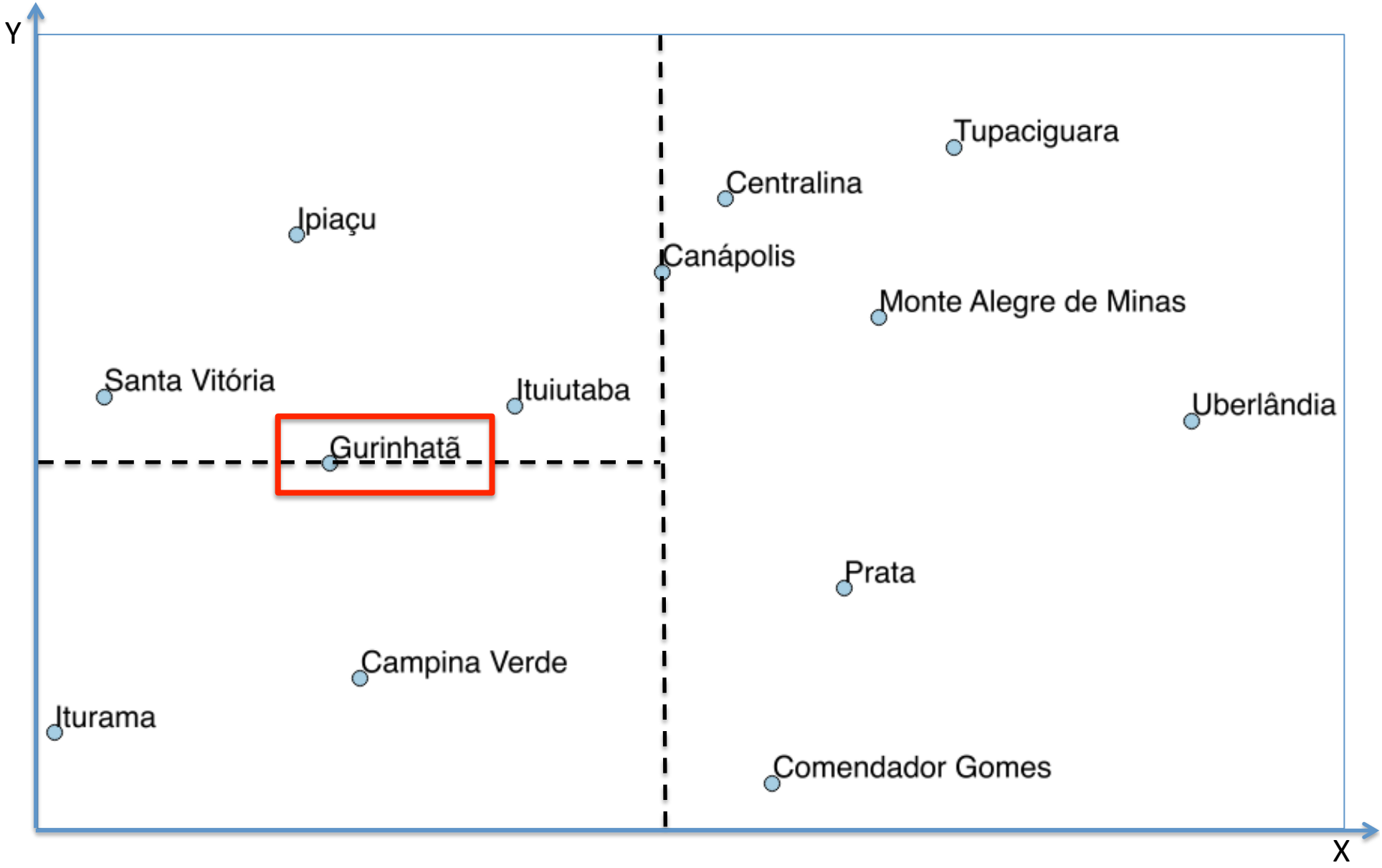
Construindo uma *k*-d tree



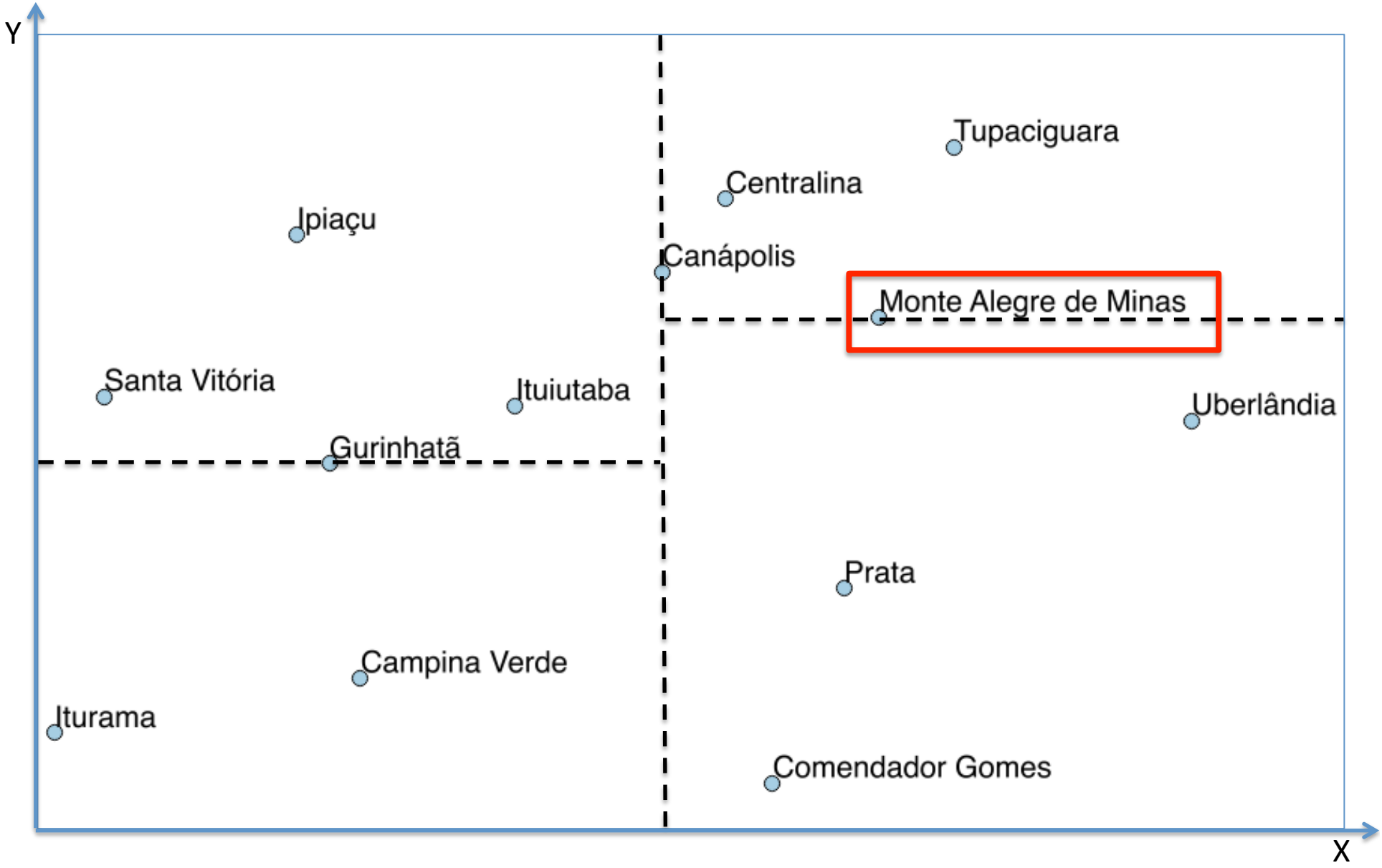
Construindo uma *k*-d tree



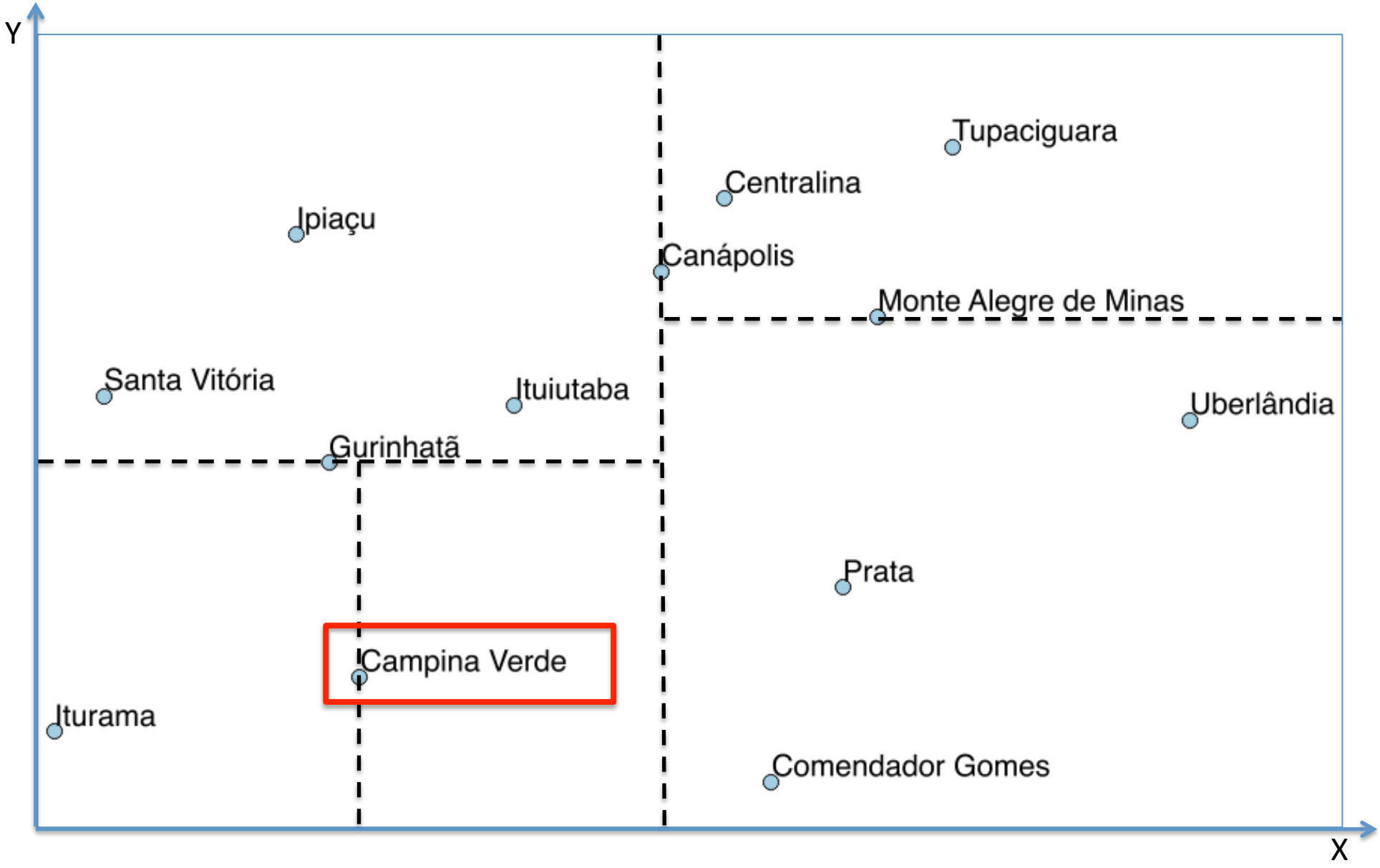
Construindo uma *k*-d tree



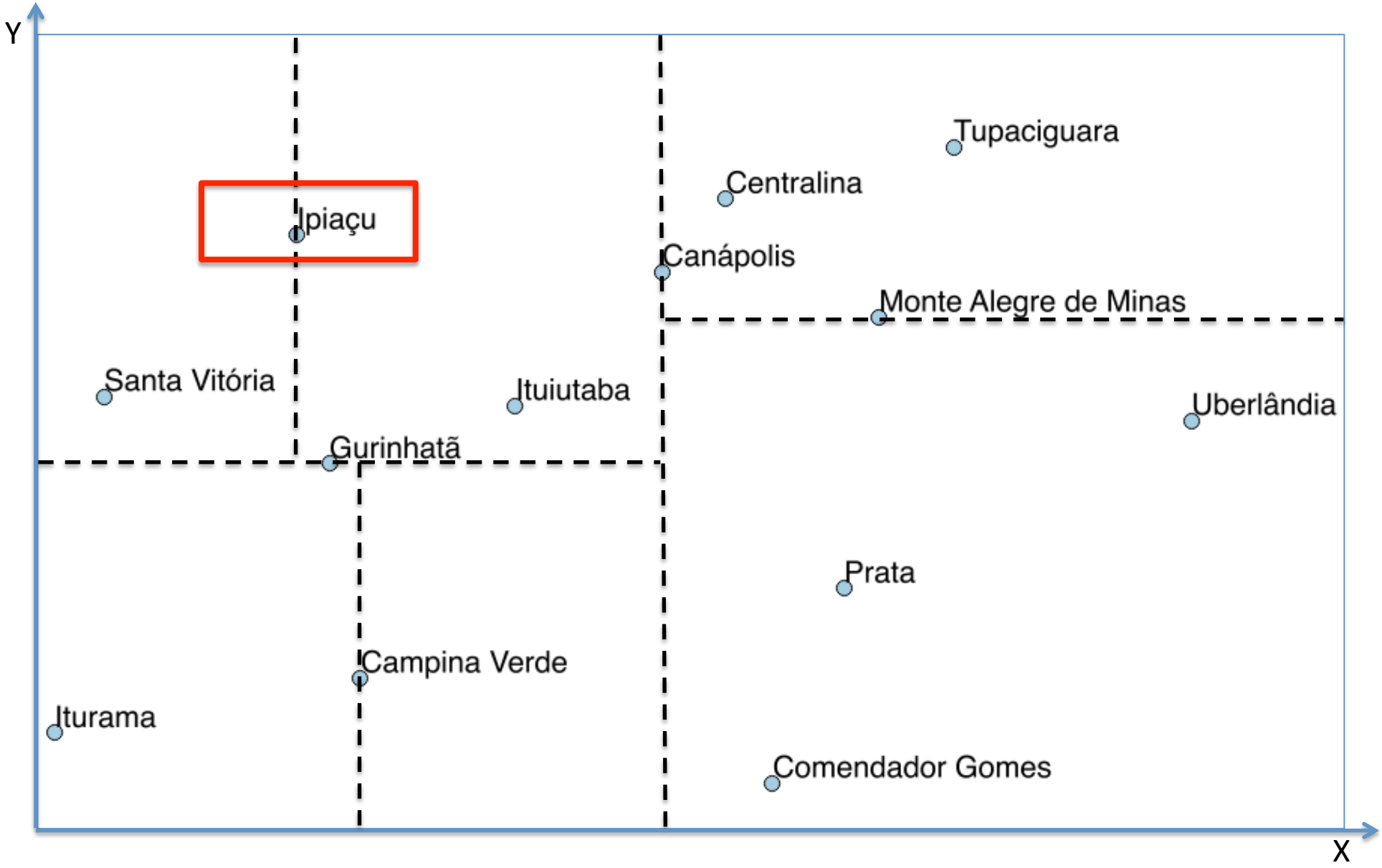
Construindo uma *k*-d tree



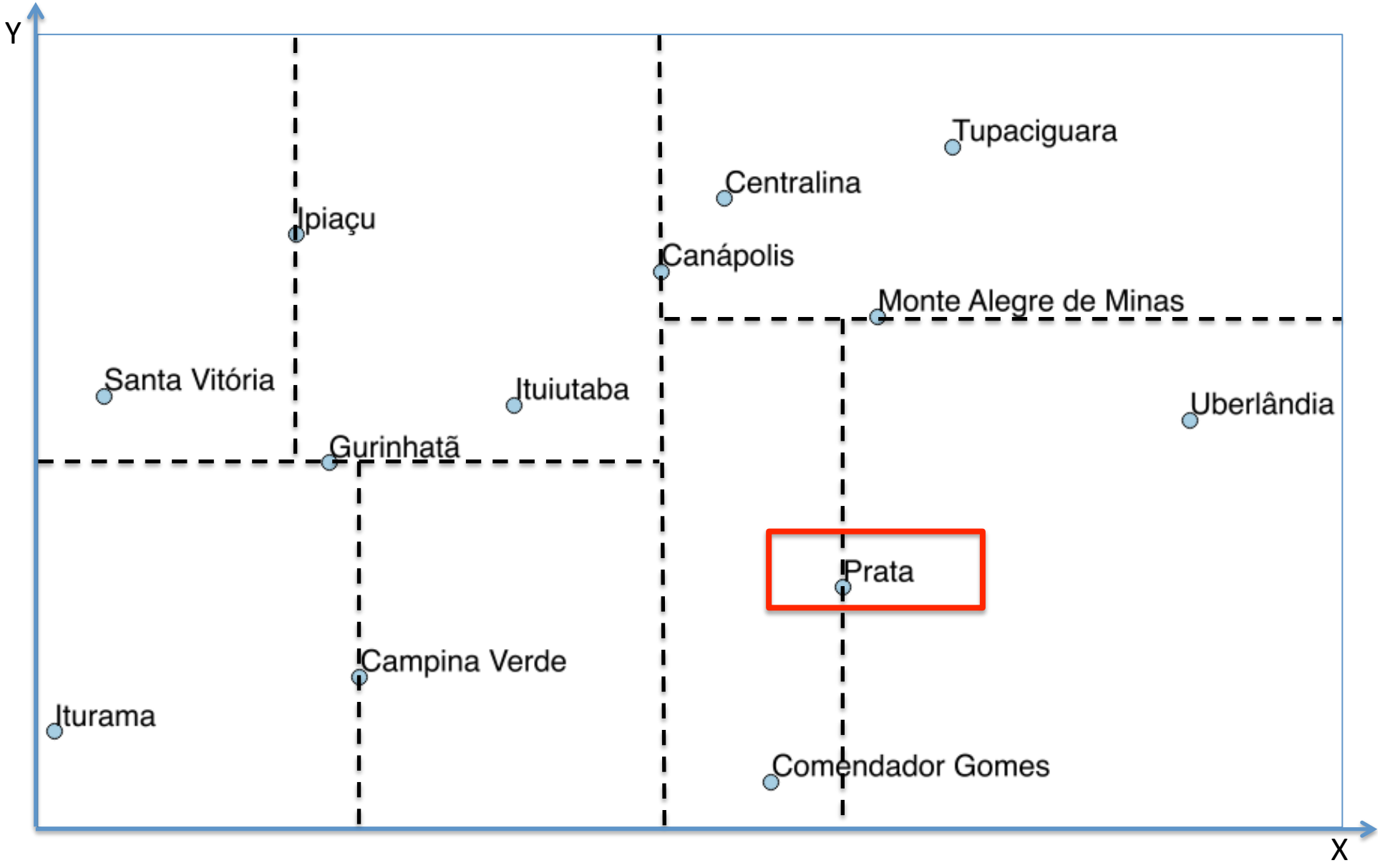
Construindo uma *k*-d tree



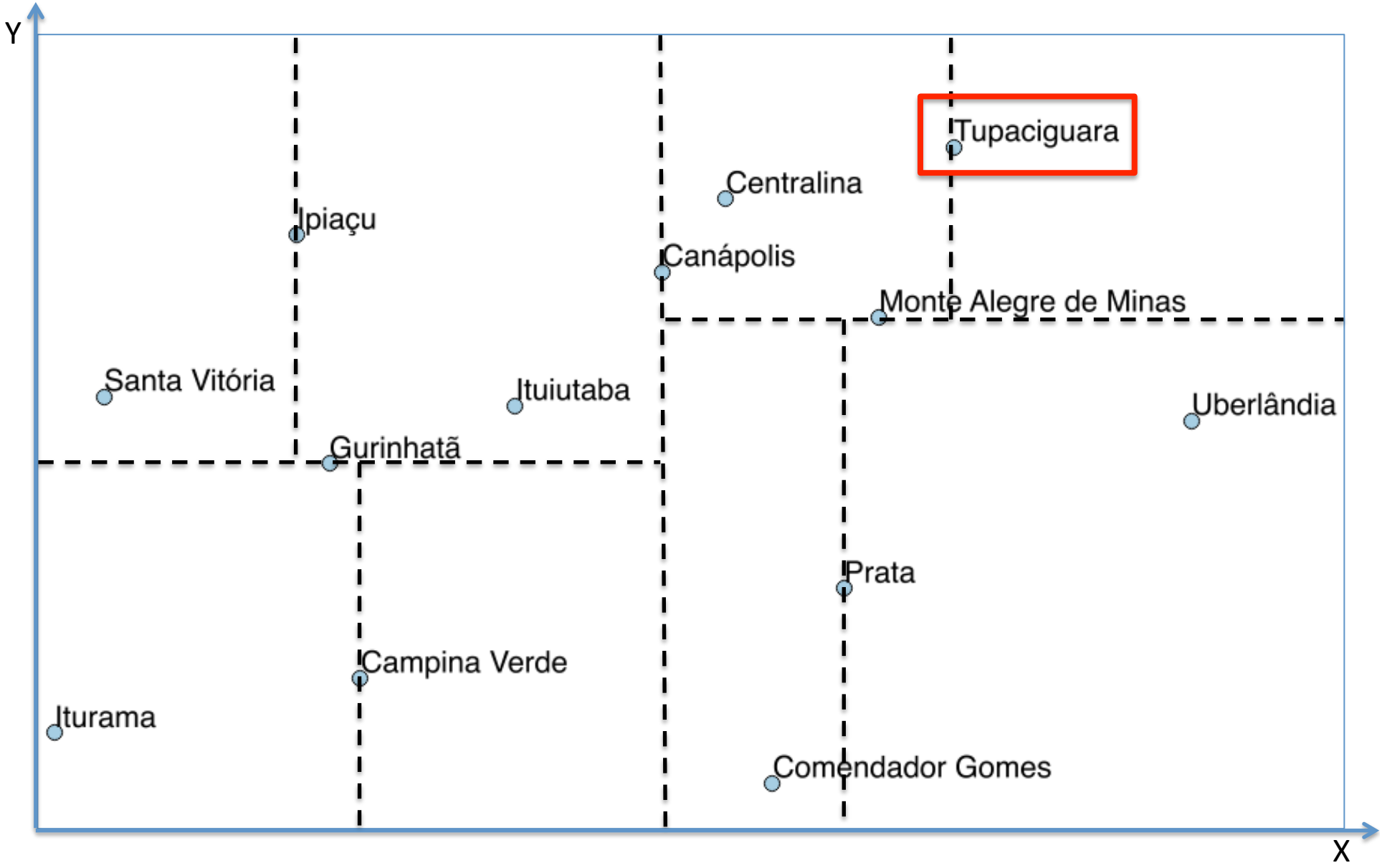
Construindo uma *k*-d tree



Construindo uma *k*-d tree



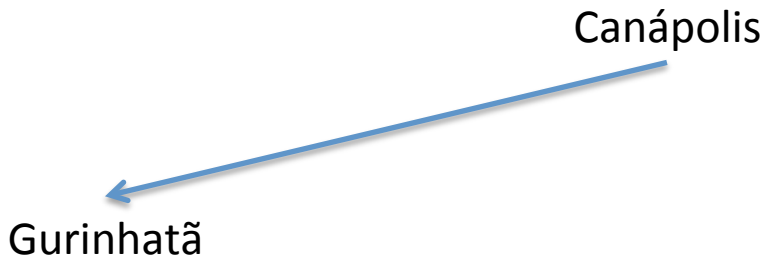
Construindo uma *k*-d tree



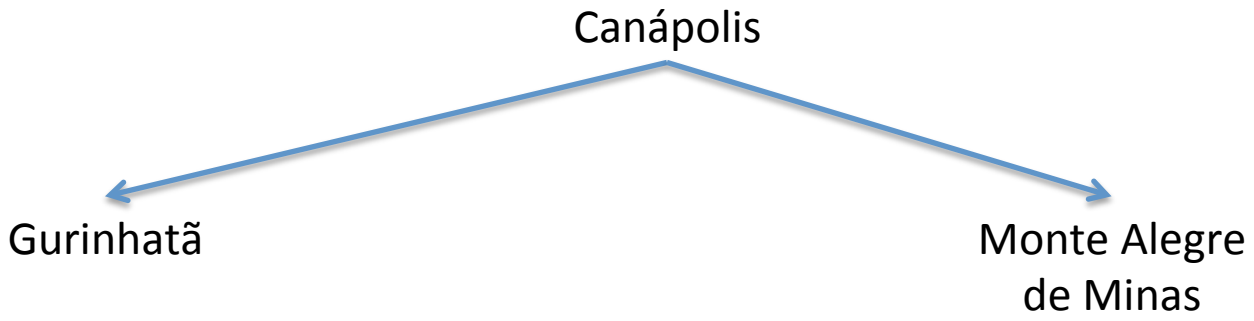
Construindo uma *k*-d tree

Canápolis

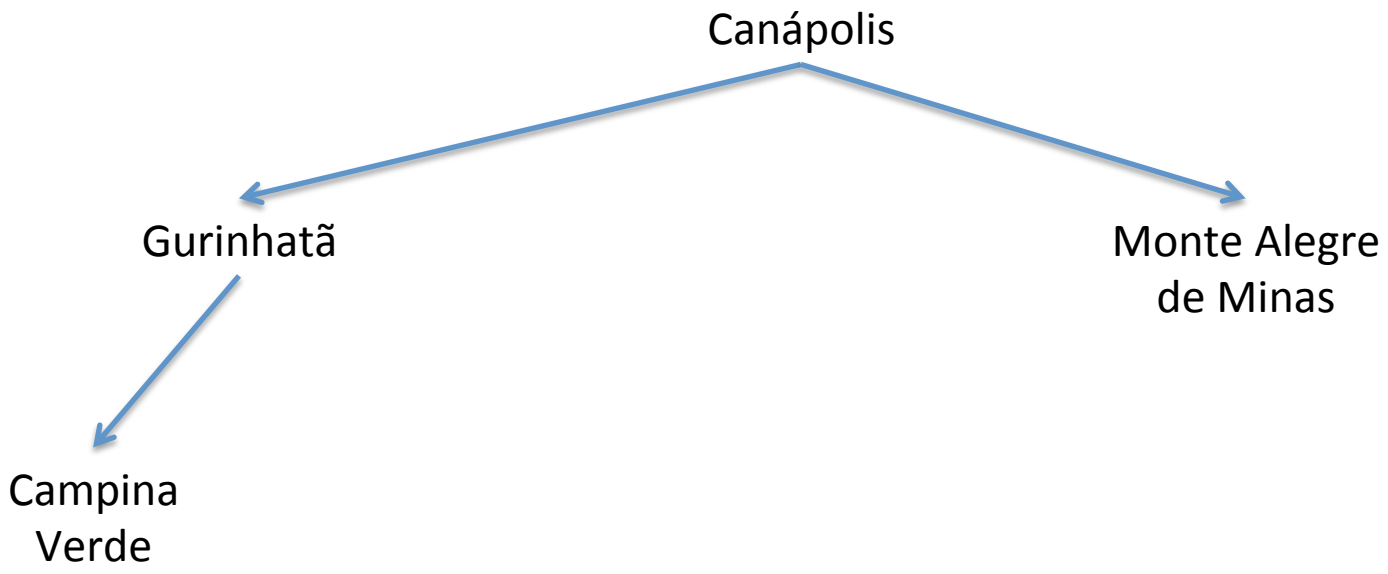
Construindo uma *k*-d tree



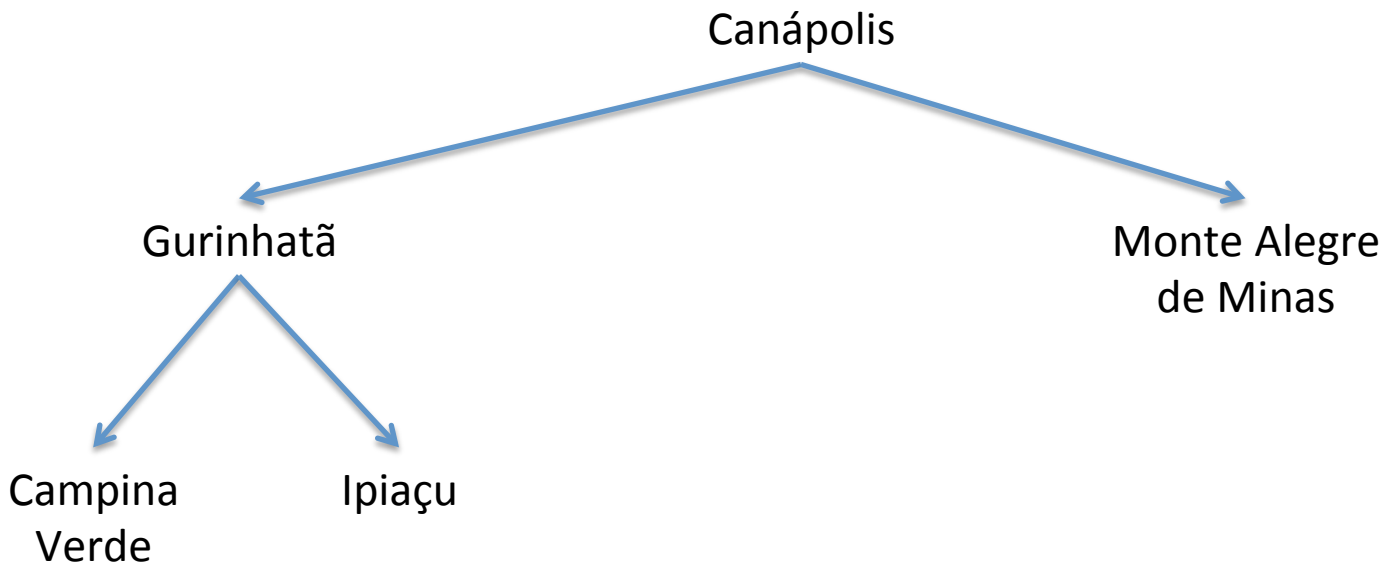
Construindo uma *k*-d tree



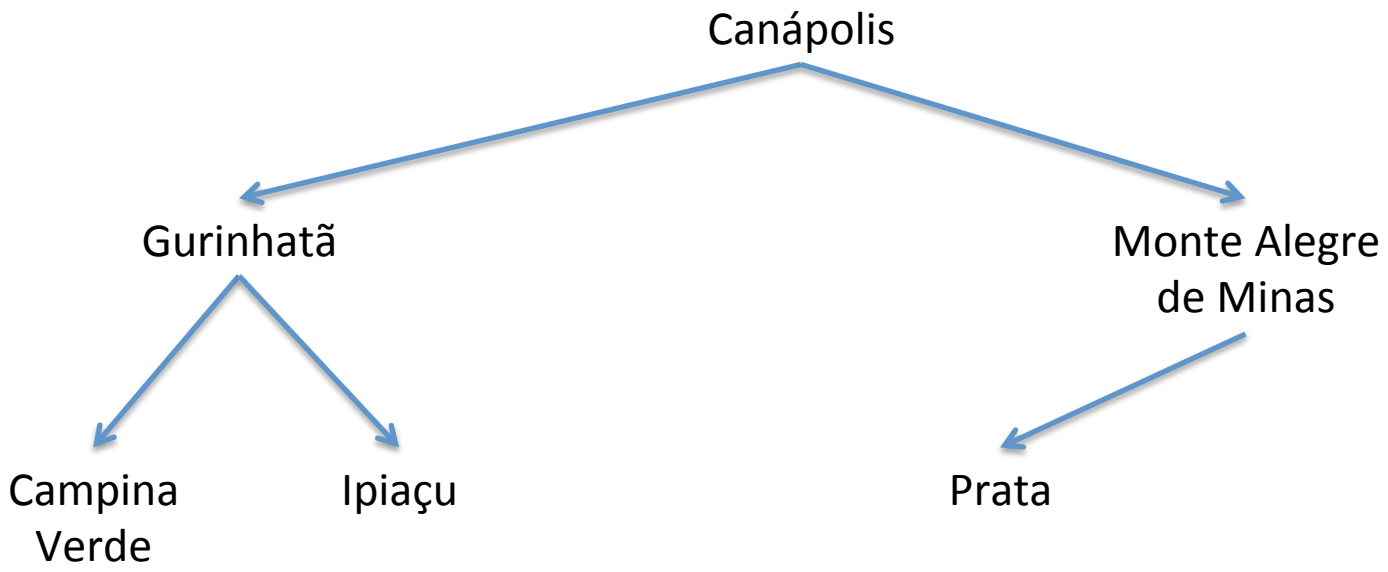
Construindo uma *k*-d tree



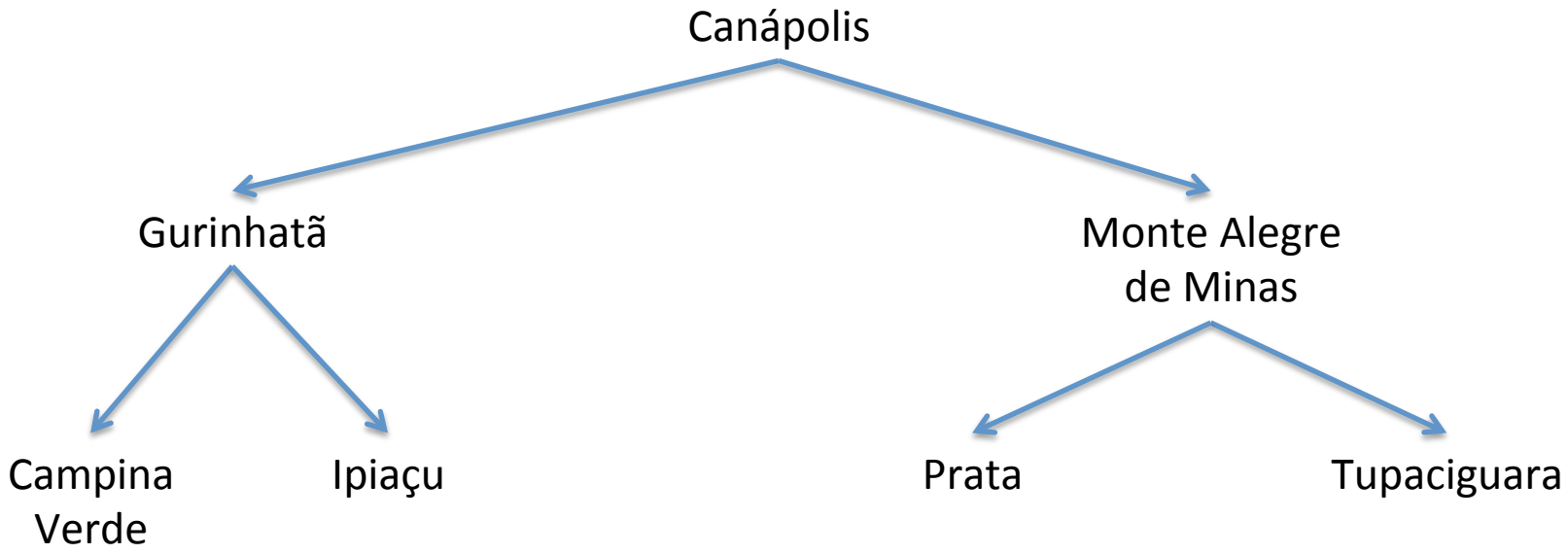
Construindo uma *k*-d tree



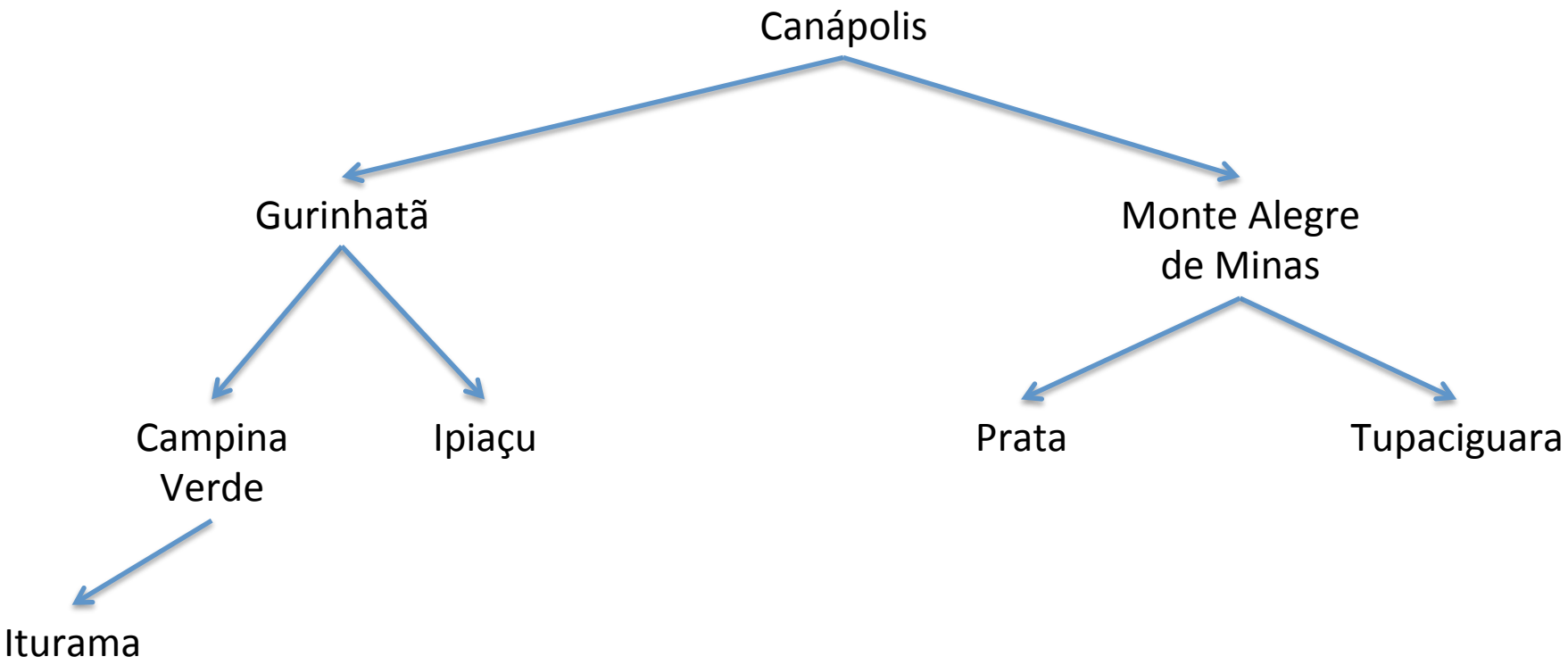
Construindo uma *k*-d tree



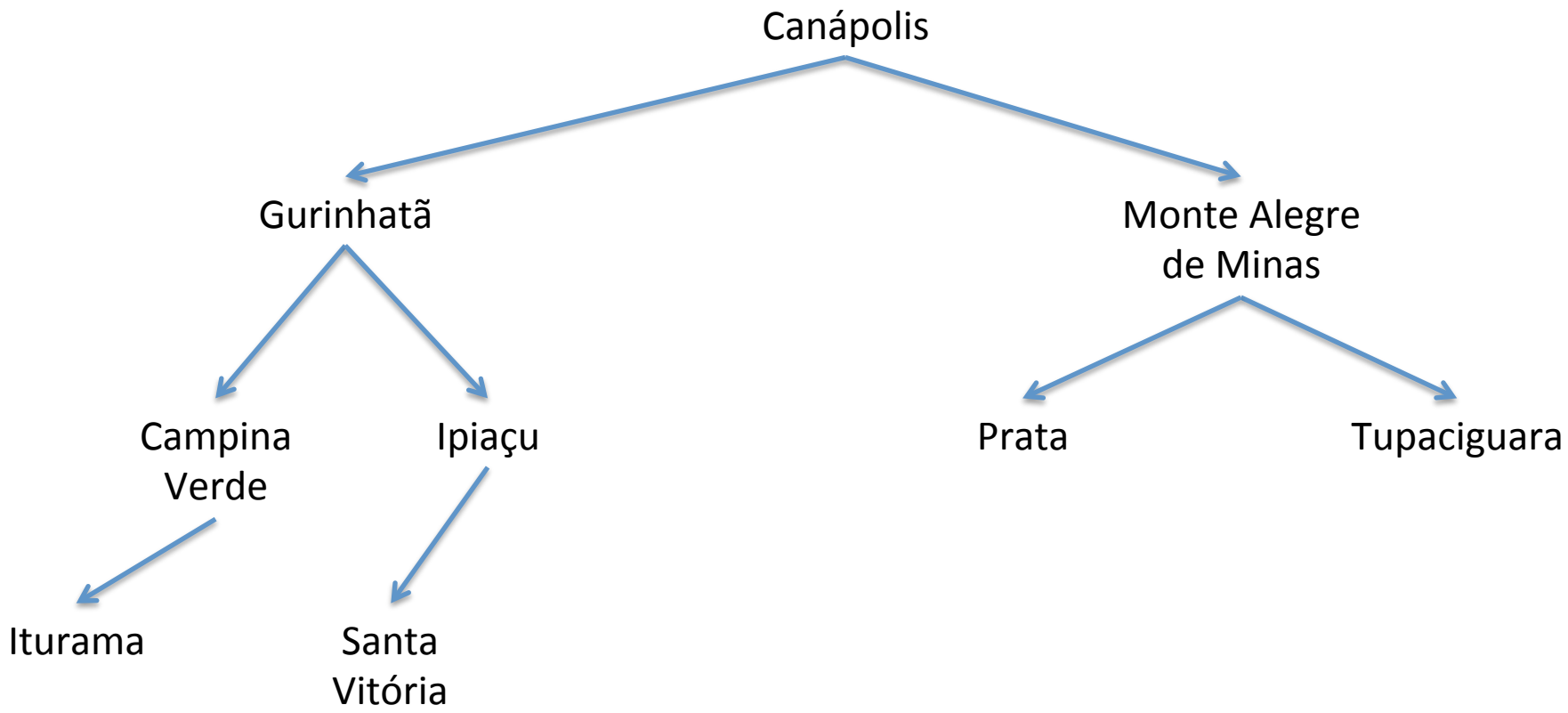
Construindo uma *k*-d tree



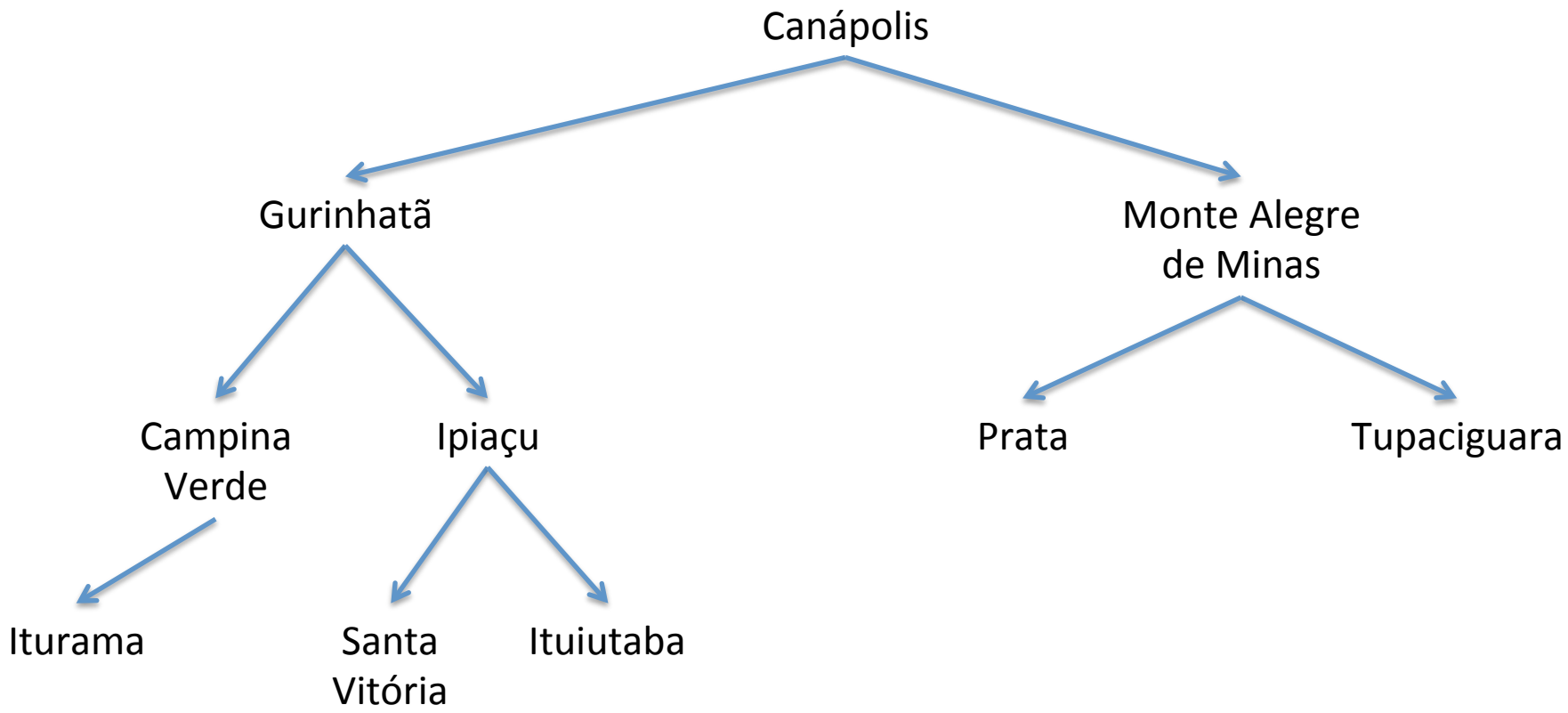
Construindo uma *k*-d tree



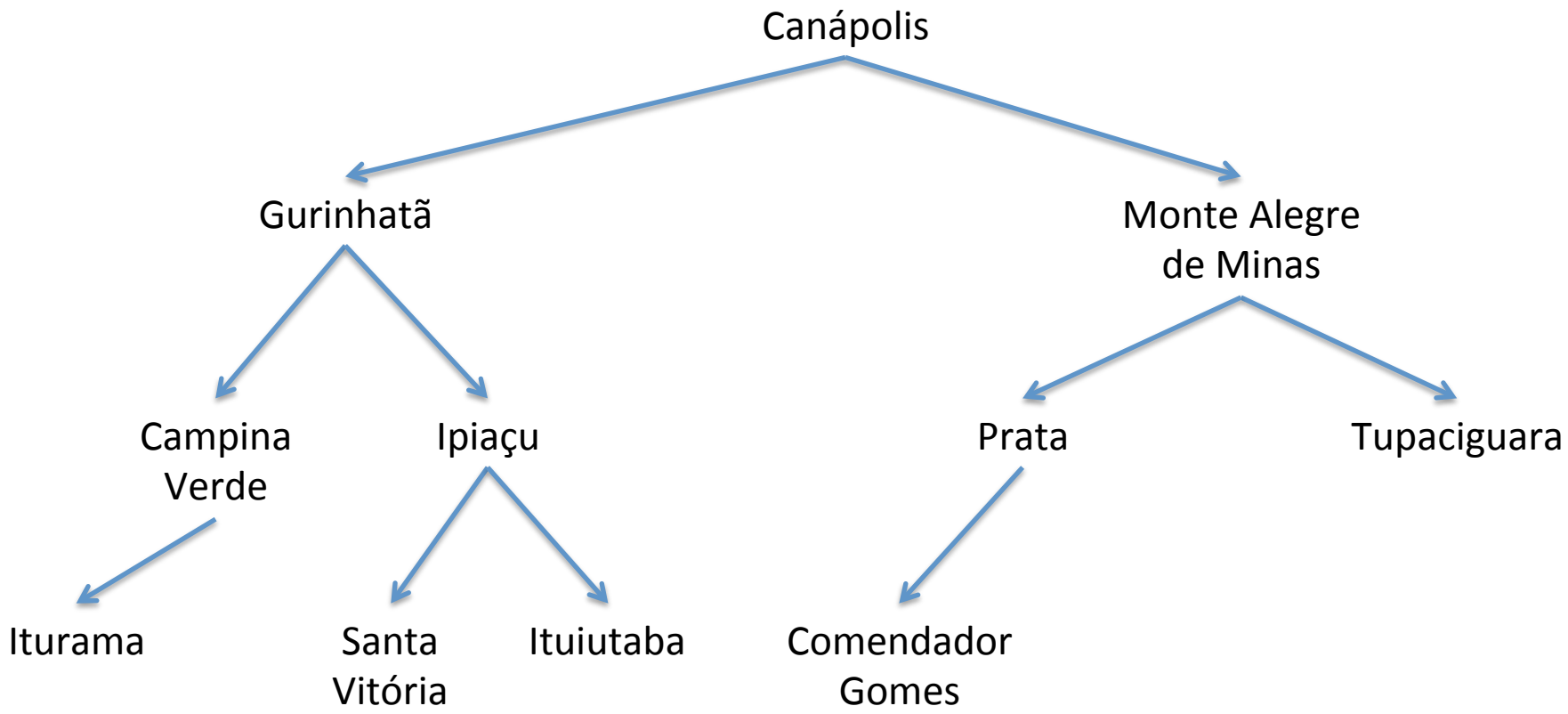
Construindo uma *k*-d tree



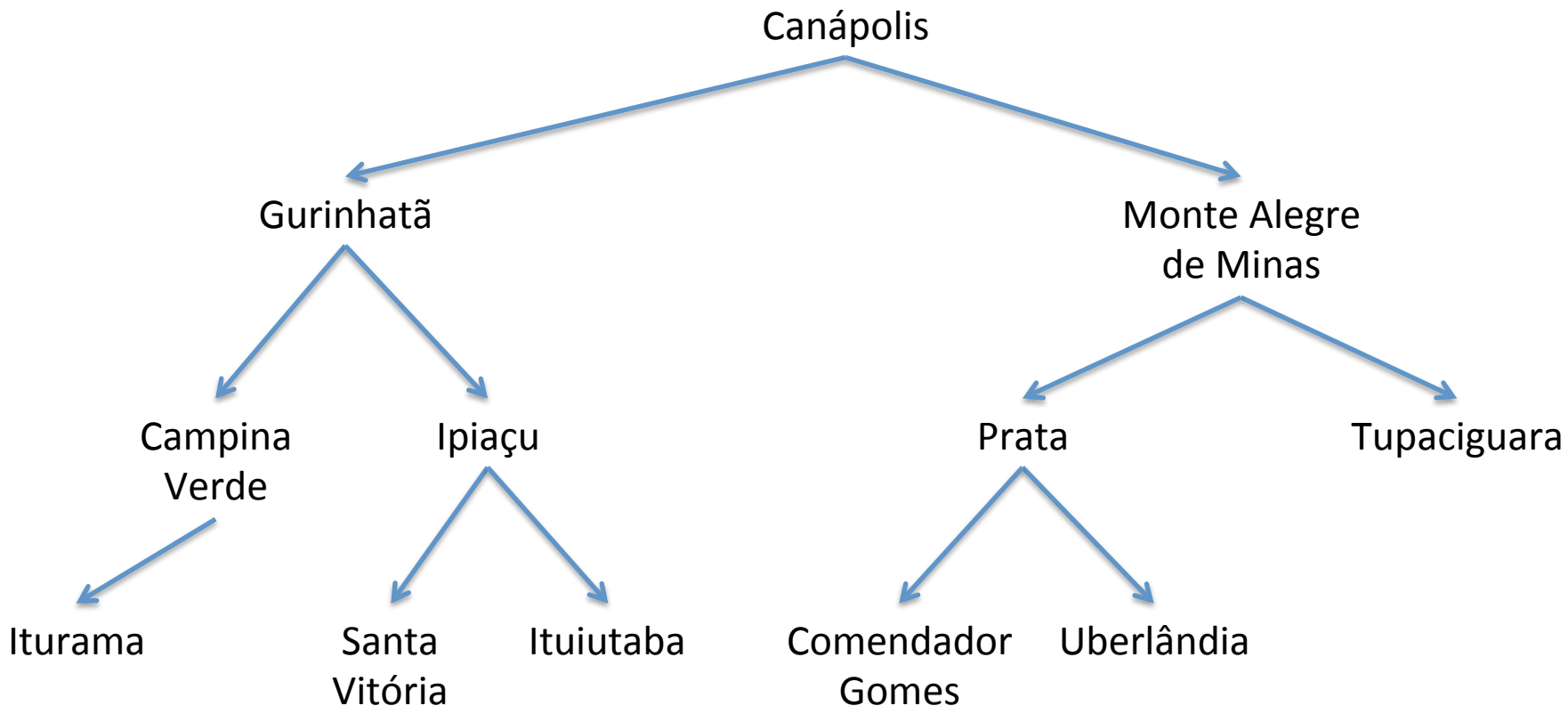
Construindo uma *k*-d tree



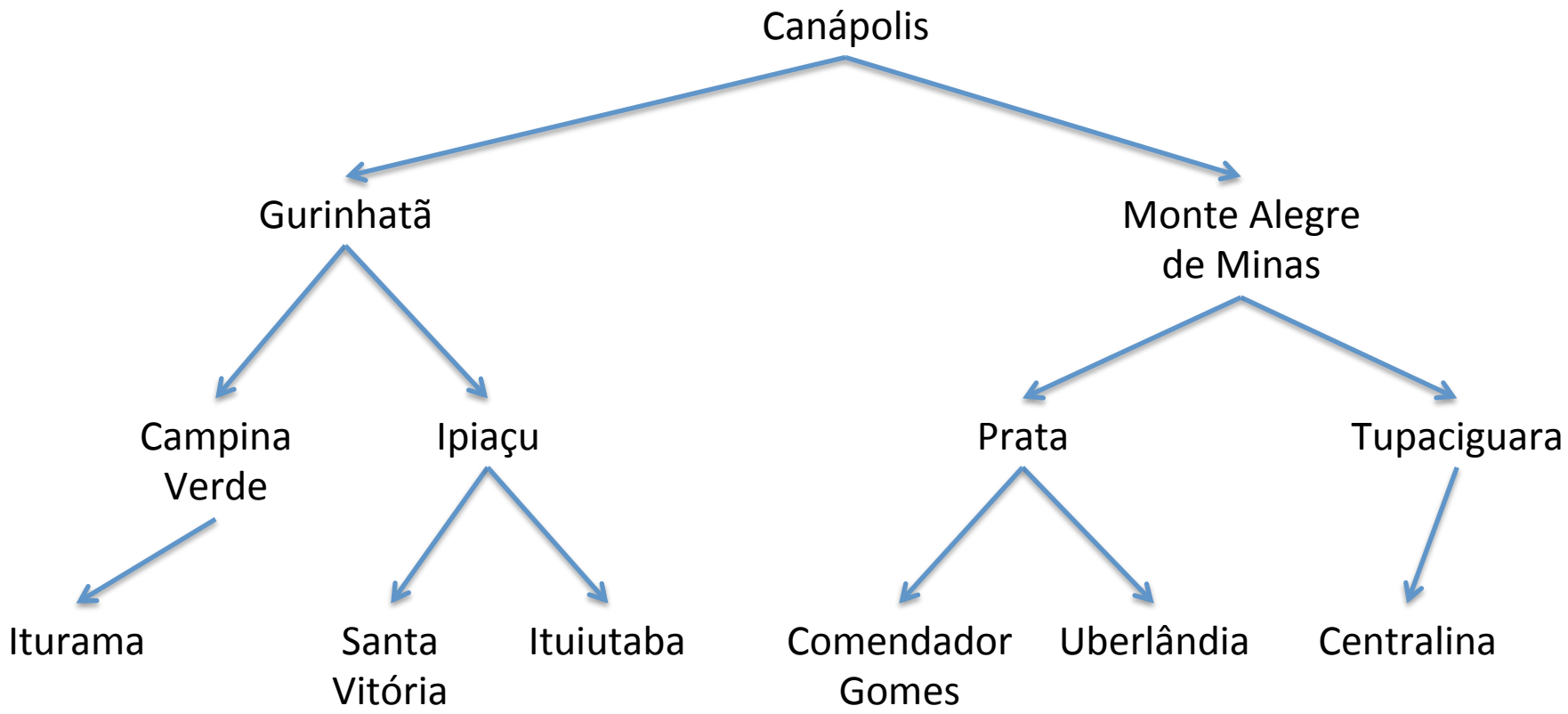
Construindo uma *k*-d tree



Construindo uma *k*-d tree



Construindo uma *k*-d tree



k-d tree: Considerações

- Aplicações em memória RAM:
  - Funcionalidade de Snap: localizar rapidamente um ponto em uma coleção.
  - Cálculo de vizinhança: interpoladores.

- A kd-tree é uma estrutura de dados que não é adequada a índices mantidos em memória secundária.

- Diversas variantes (balanceadas):
  - *k*-d B-tree (Robinson, 1981)
  - hB-tree (Lomet e Salzberg, 1990)

Aproximações Geométricas

Geometric Approximations

Aproximações mais Comuns

MBR

RMBR

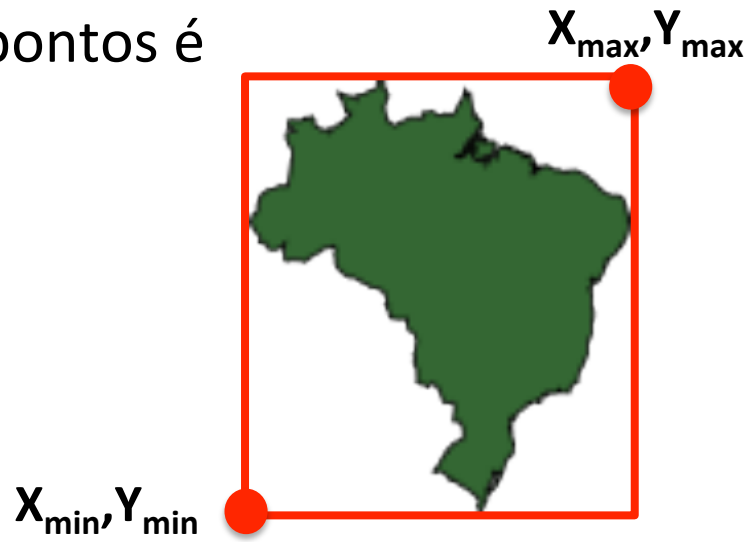
Convex Hull

MBC

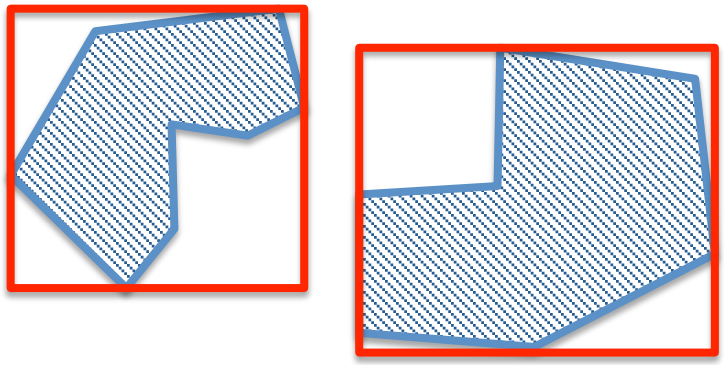
MBE

Minimum Bounding Rectangle (MBR)

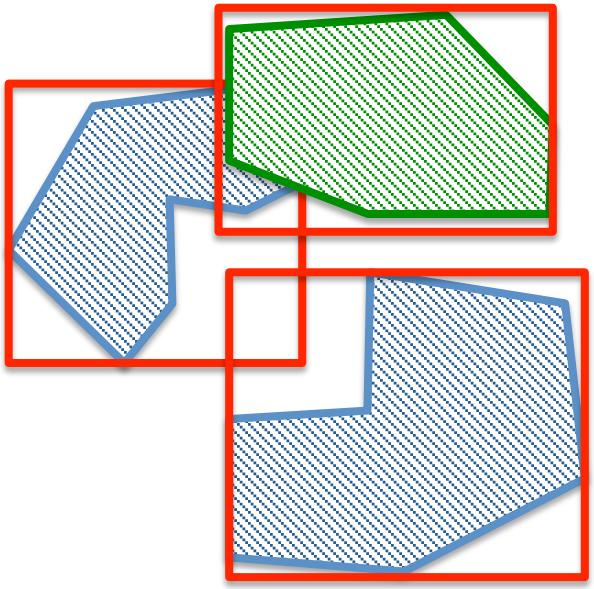
- O retângulo envolvente mínimo (REM) é uma das aproximações mais utilizadas em SIGs:
  - AKA: Minimum Bounding Box (MBB) ou Bounding Box (BBOX);
  - Aproximação para objetos com extensão (linhas e polígonos);
  - No espaço 2D apenas um par de pontos é necessário para representá-lo;
  - Filtro rápido para testes de relacionamento espacial.



Propriedades dos MBRs



Caso 1) Os retângulos não se interceptam: logo as geometrias ***não*** podem intersectar.



Case 2) Os retângulos se interceptam: logo as geometrias ***podem*** se intersectar.

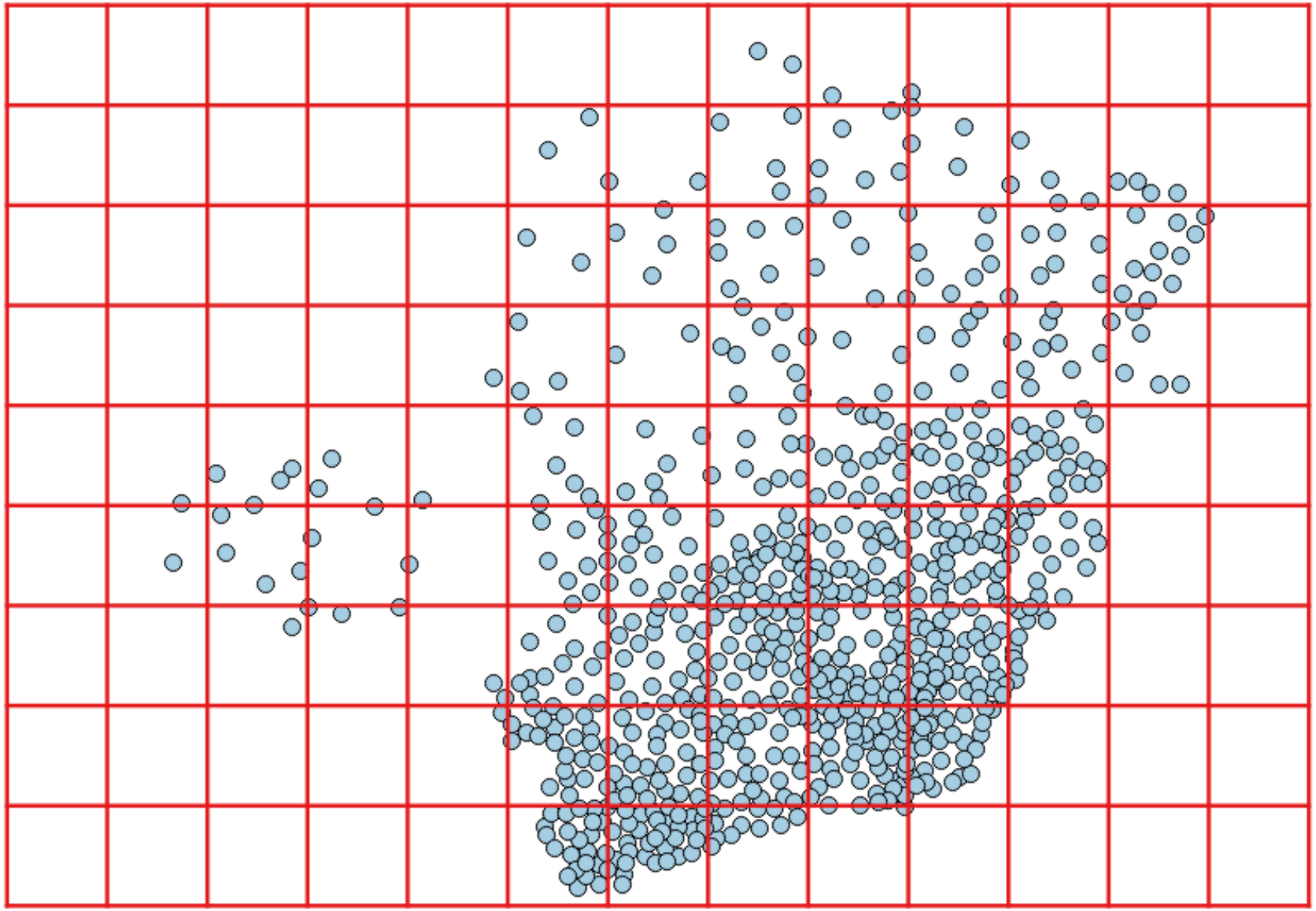
MBR: Considerações

- A grande maioria dos índices espaciais são projetados sobre o MBR.
- Seria muito complexo as chaves dos índices serem formadas/representadas pelas geometrias exatas de linhas e polígonos.
- Além disso, os índices seriam extremamente custosos e proibitivos, com as geometrias exatas.
- Portanto, todos os SGBD com suporte a tipos espaciais utilizam bastante as propriedades dos REMs das geometrias.

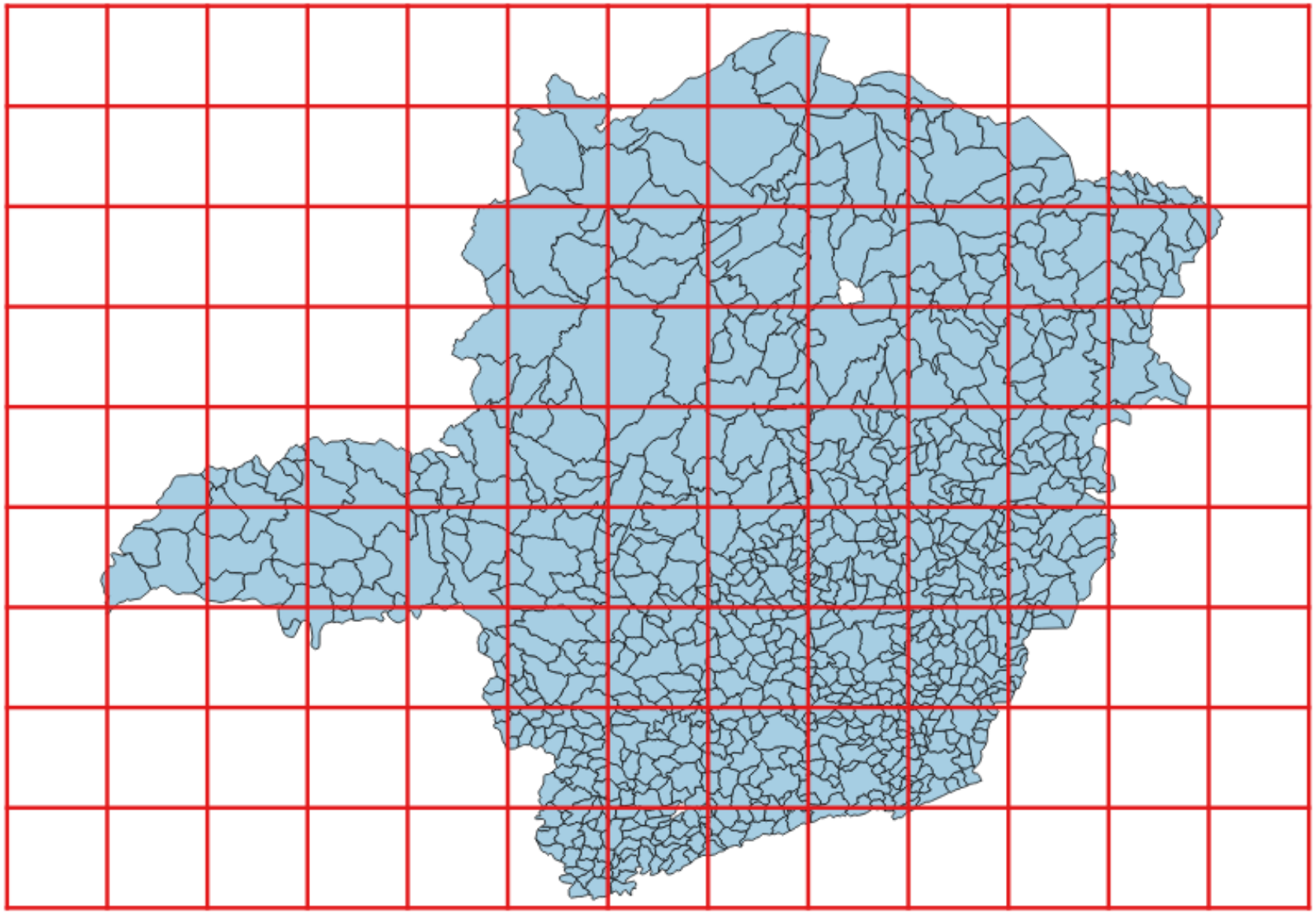
Grades Fixas

Nievergelt et al. (1984)

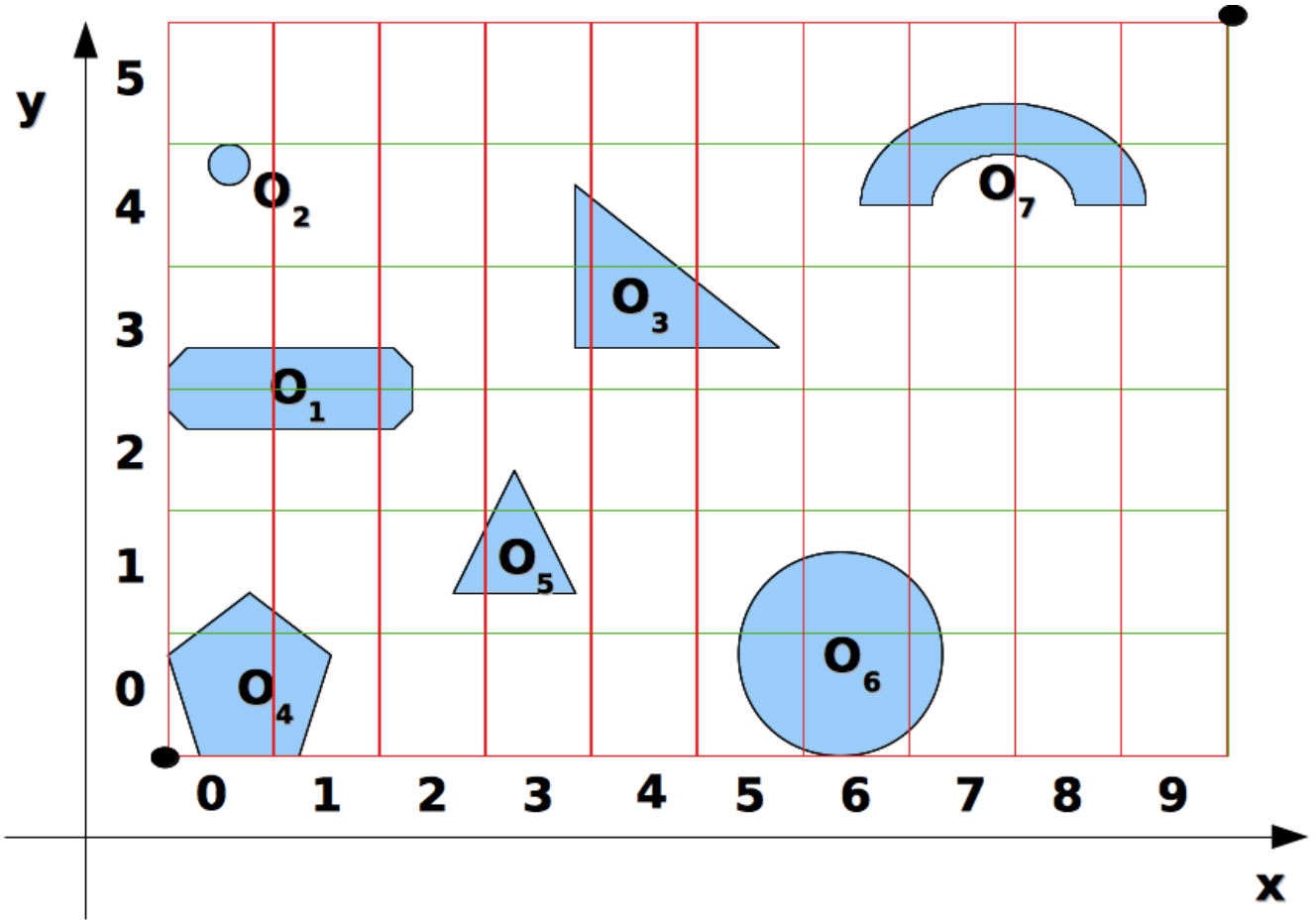
Grade Fixa: Pontos



Grade Fixa: Polígonos



Grade Fixa: Pontos



Célula	row-id / OID
0	O_4
1	O_4
6	O_4
7	O_4
2	O_1
3	O_1
8	O_1
9	O_1
14	O_1
15	O_1
…	…

Space-filling Curves: Hilbert

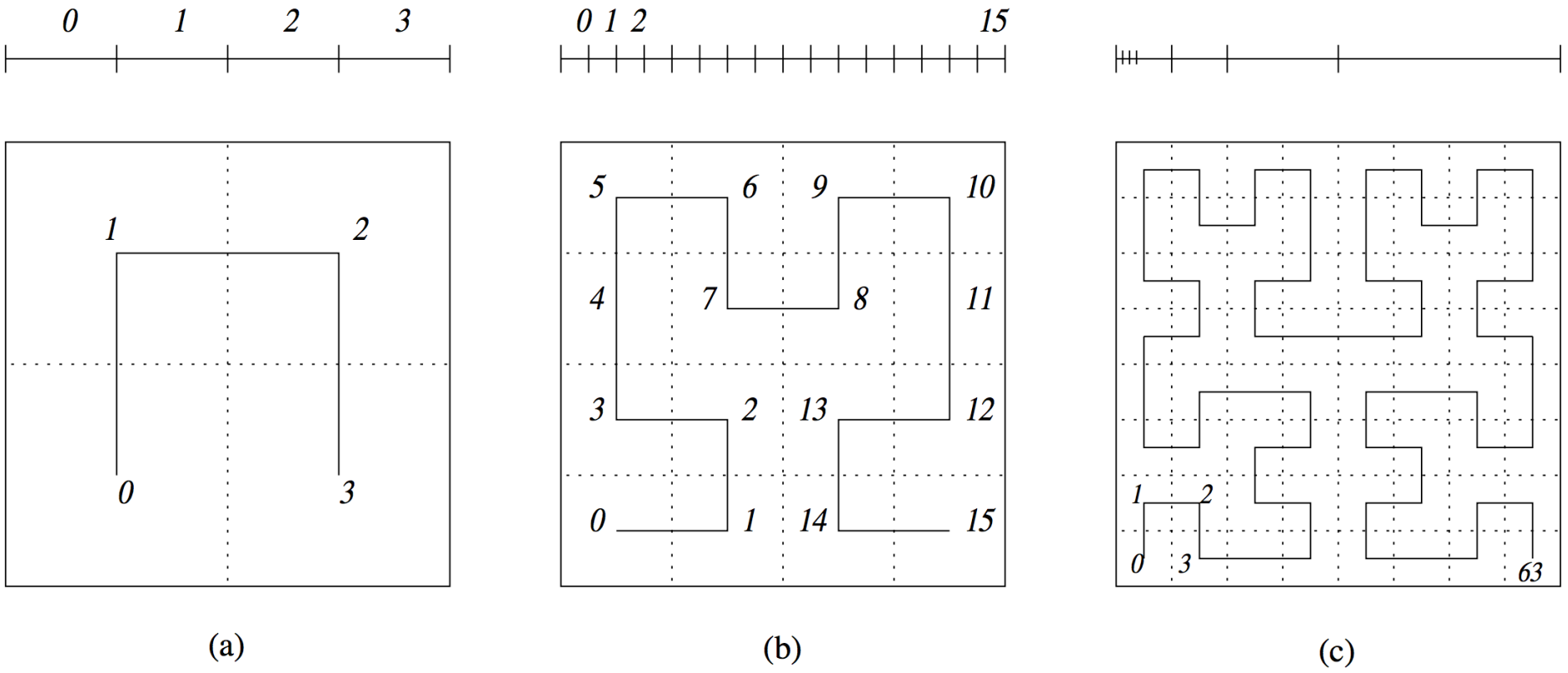
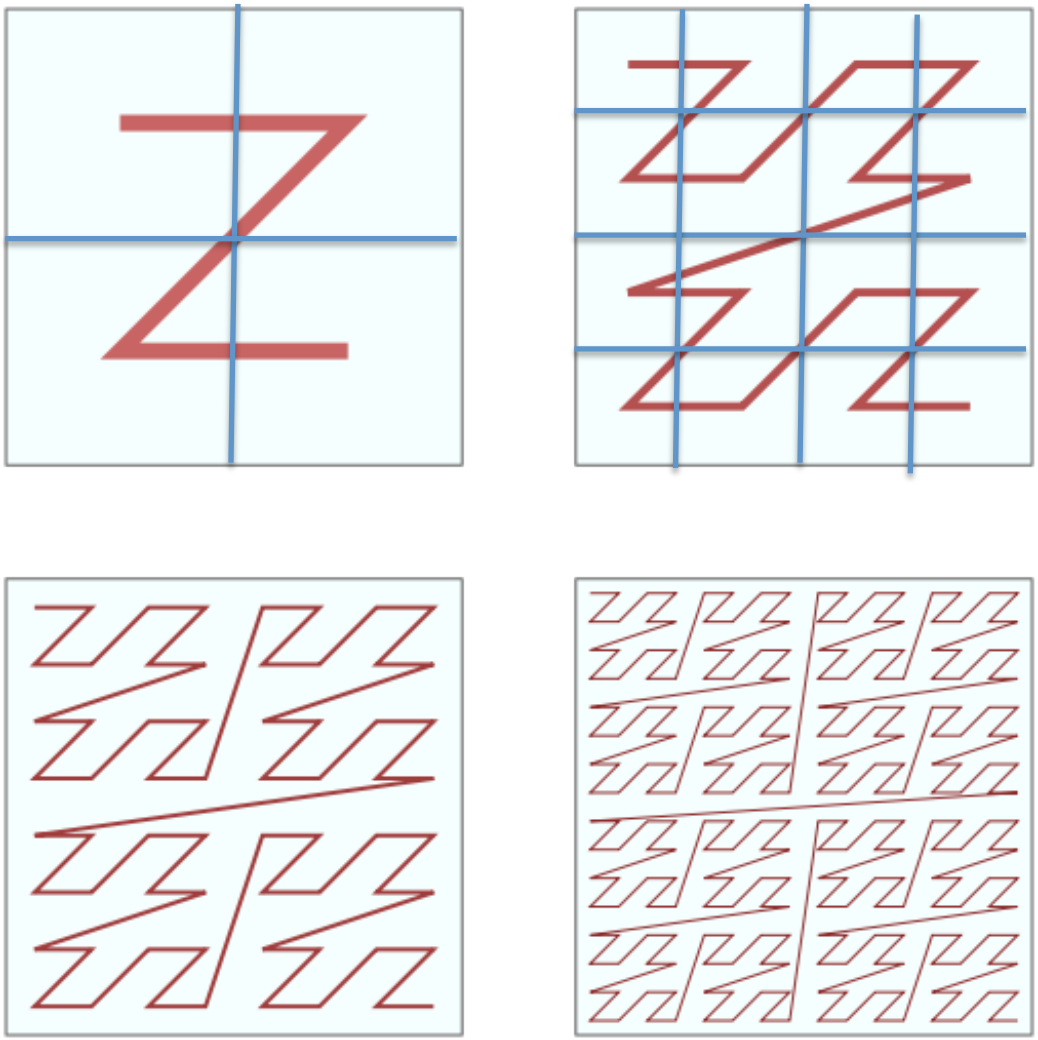


Fig. 1. Approximations of the Hilbert curve in 2 dimensions

Fonte: Lawder (????)

Space-filling Curves: z-Order



Fonte: Adaptado de Wikipedia

Grade Fixa: Considerações

- Problema:
  - Determinar a resolução adequada para a grade.
- Vantagem:
  - Pode ser implementada sobre a B-tree do próprio SGBD.
- IBM DB2 Spatial Extender:
  - Grades multiníveis (3 níveis)

Quadtrees

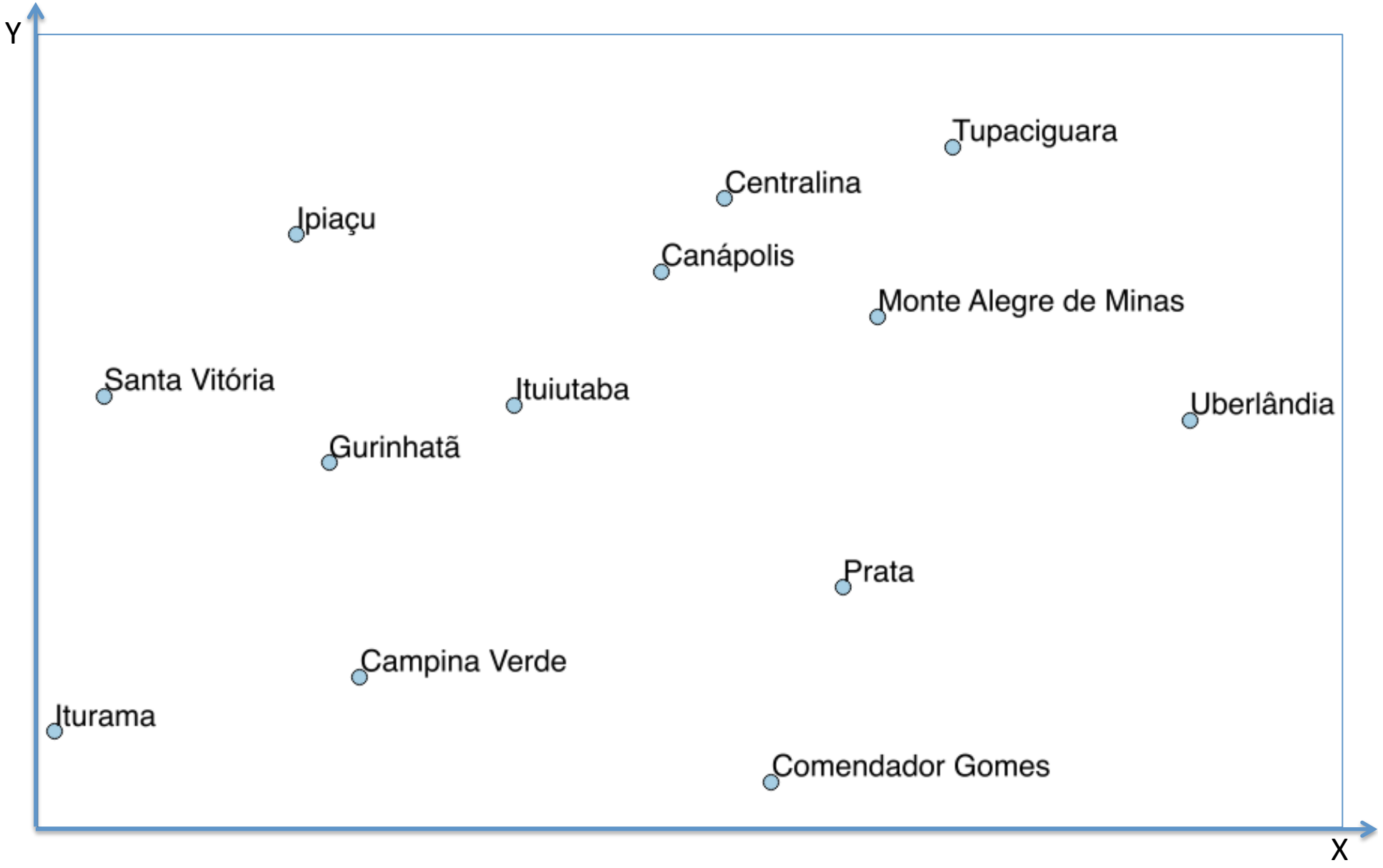
Finkel e Bentley (1974)

Samet (1990)

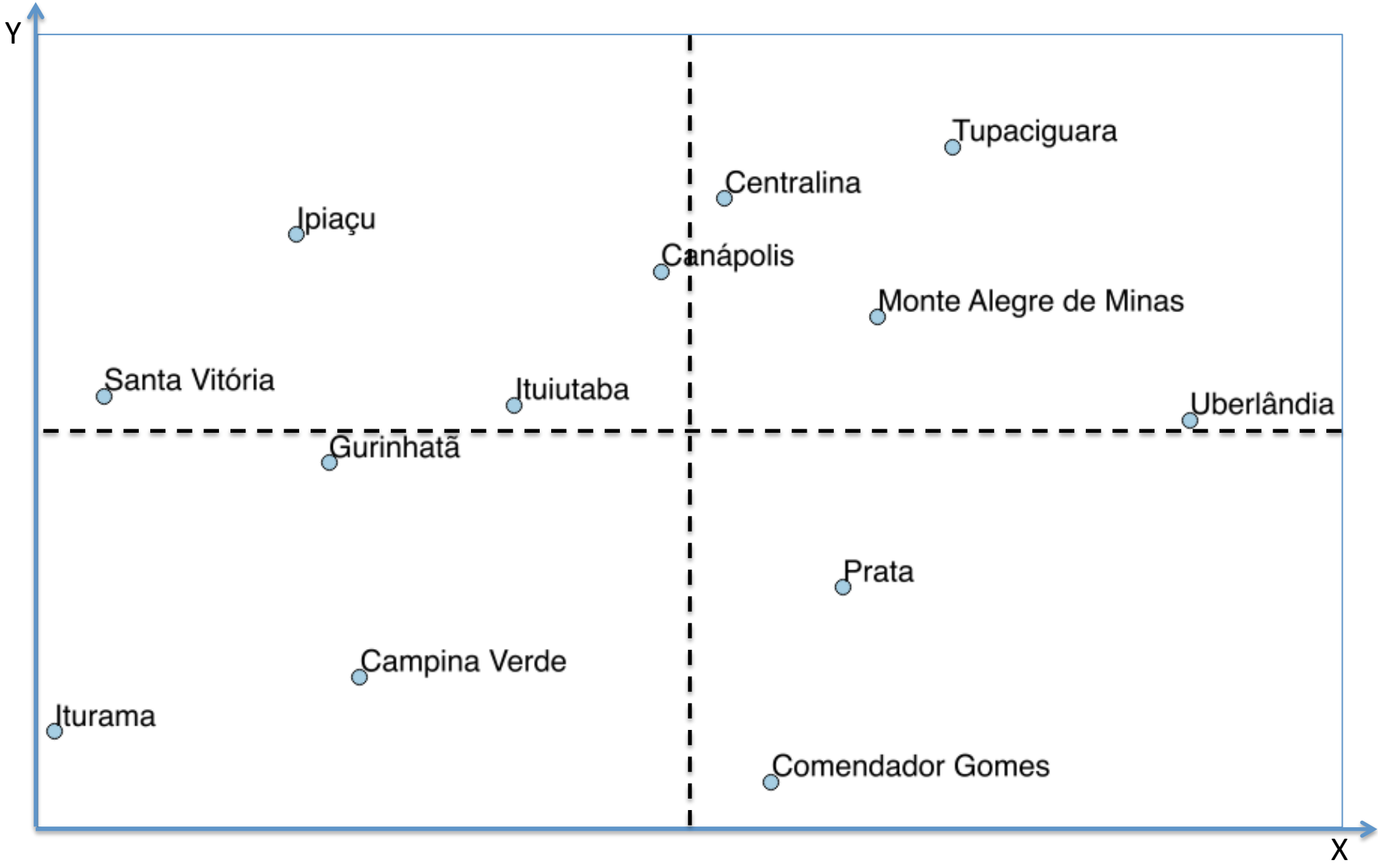
Quadtree


Tupaciguara
Centralina
Ipiaçu
Canápolis
Monte Alegre de Minas
Santa Vitória
Ituiutaba
Uberlândia
Gurinhatã
Prata
Campina Verde
Iturama
Comendador Gomes

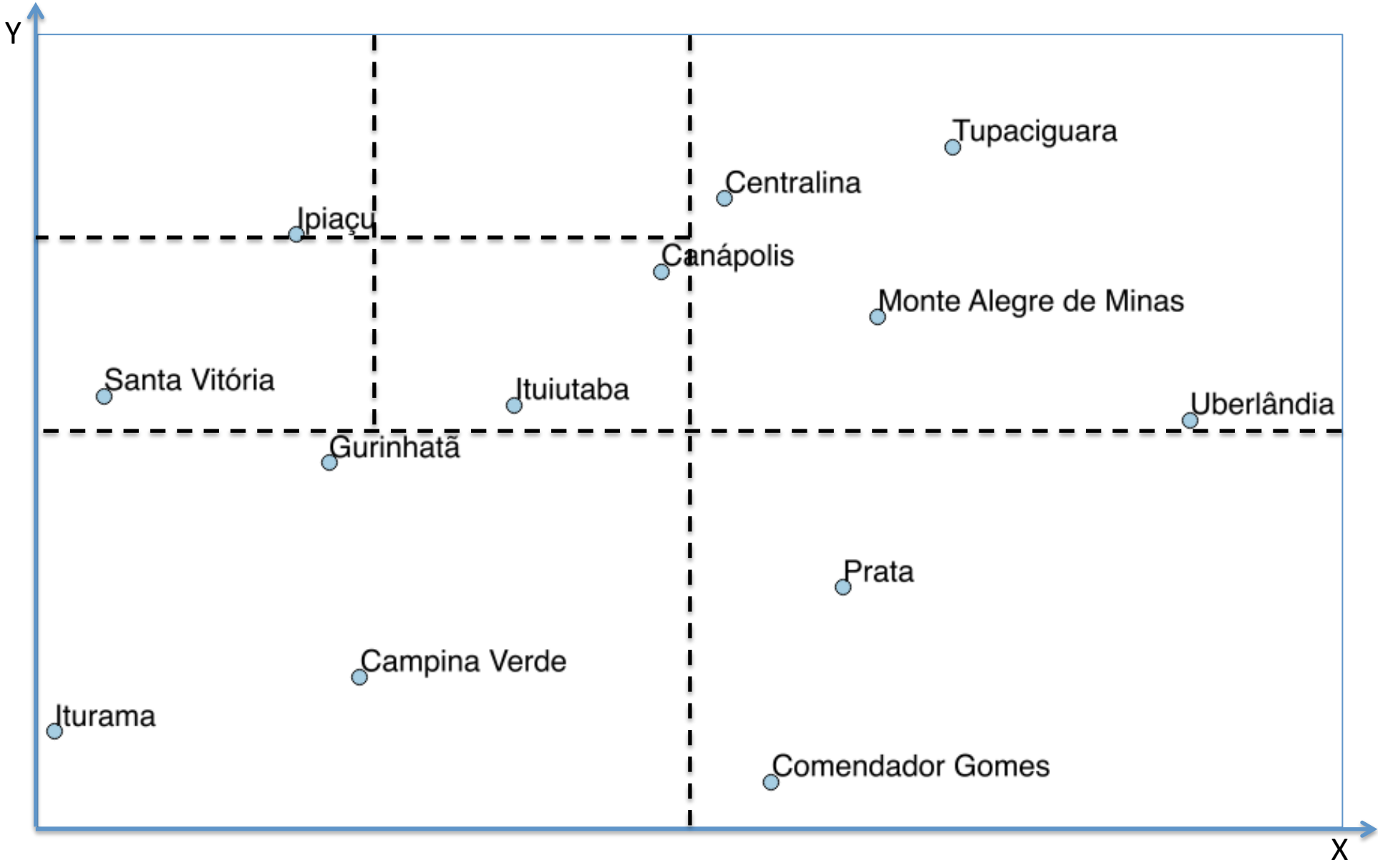
Quadtree


Y
Tupaciguara
Centralina
Ipiaçu
Canápolis
Monte Alegre de Minas
Santa Vitória
Ituiutaba
Uberlândia
Gurinhatã
Prata
Campina Verde
Iturama
Comendador Gomes
X

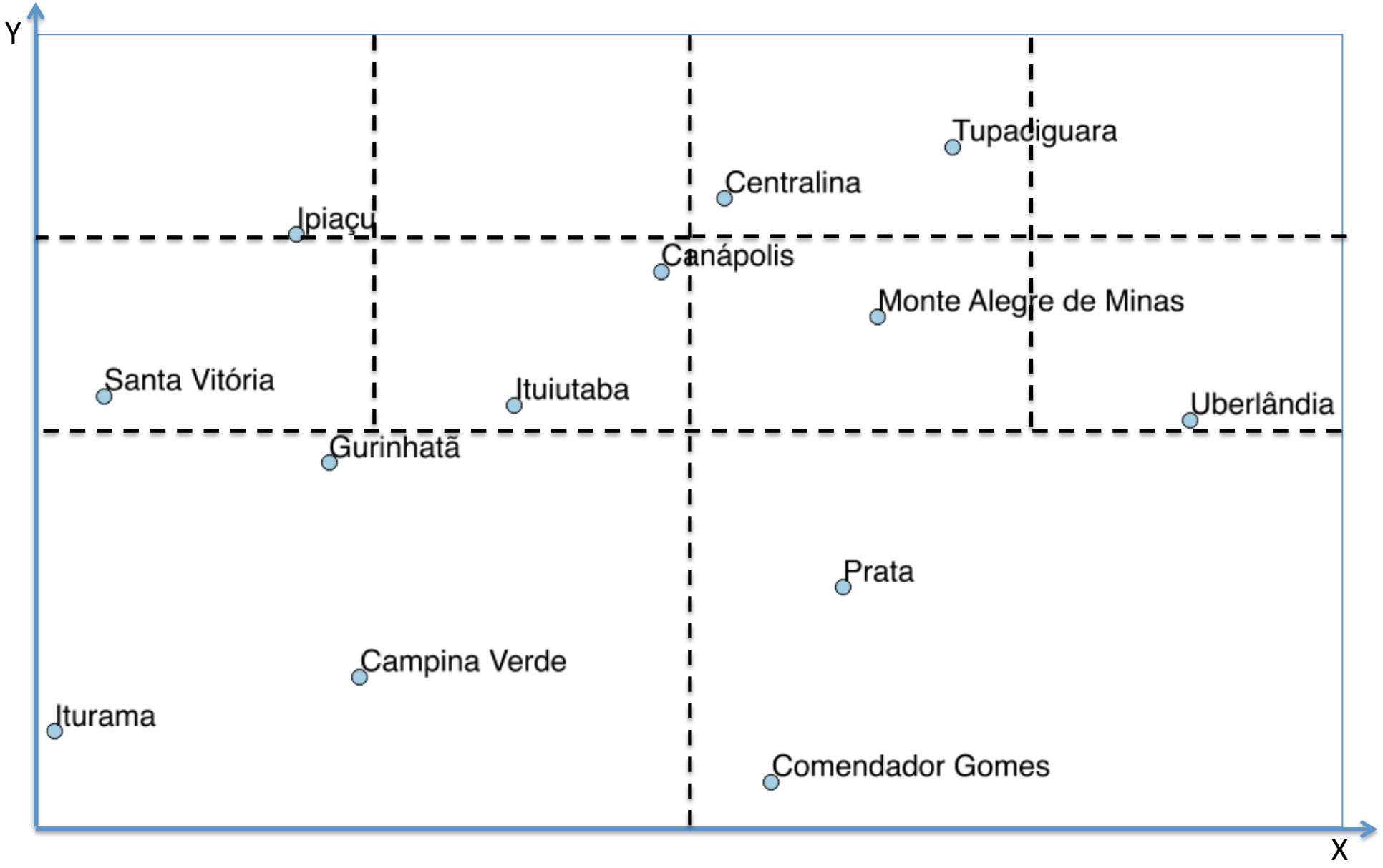
Quadtree


Y
X
Tupaciguara
Centralina
Ipiaçu
Canápolis
Monte Alegre de Minas
Santa Vitória
Ituiutaba
Uberlândia
Gurinhatã
Prata
Campina Verde
Iturama
Comendador Gomes

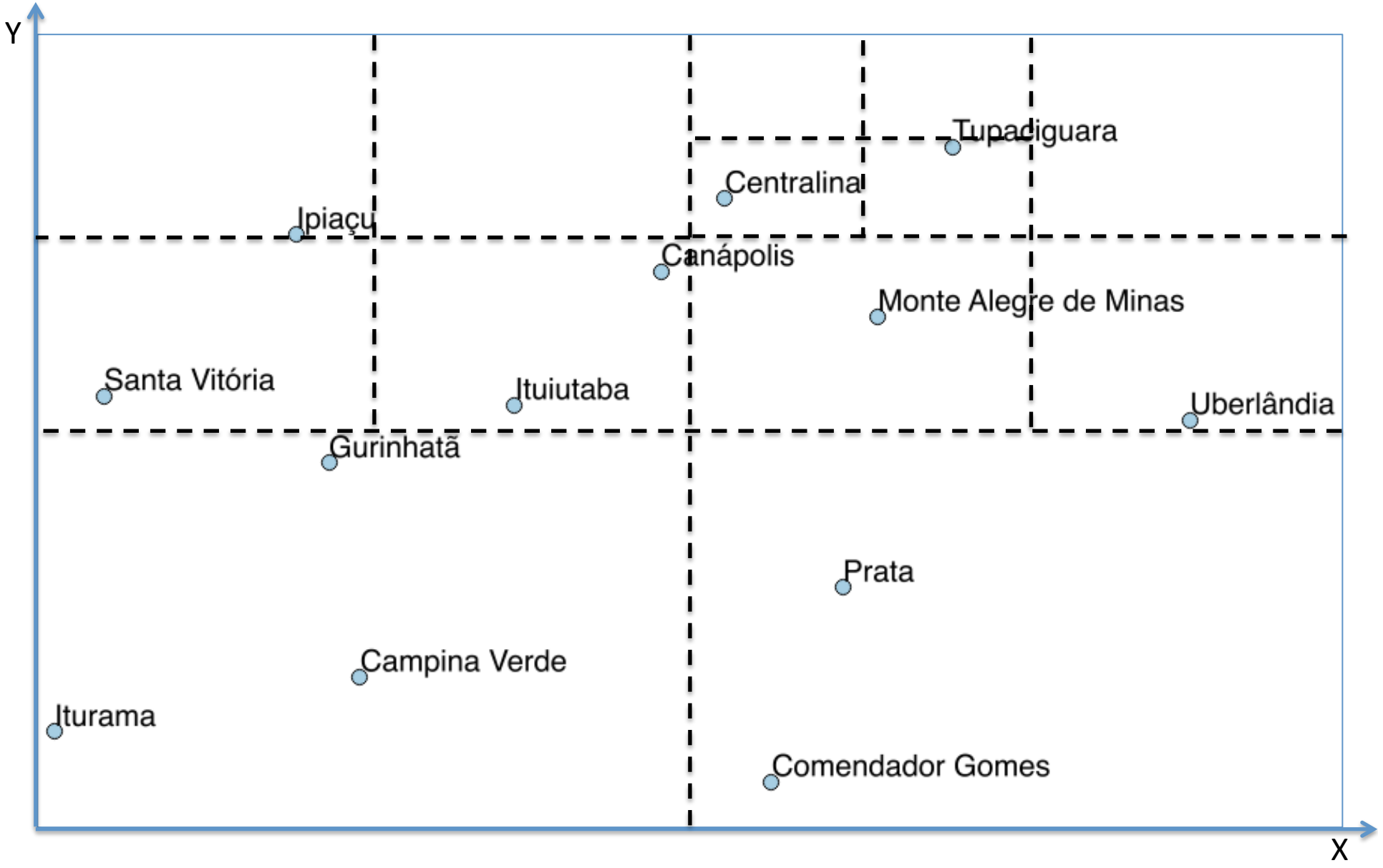
Quadtree


Y
Tupaciguara
Centralina
Ipiaçu
Canápolis
Monte Alegre de Minas
Santa Vitória
Ituiutaba
Uberlândia
Gurinhatã
Prata
Campina Verde
Iturama
Comendador Gomes
X

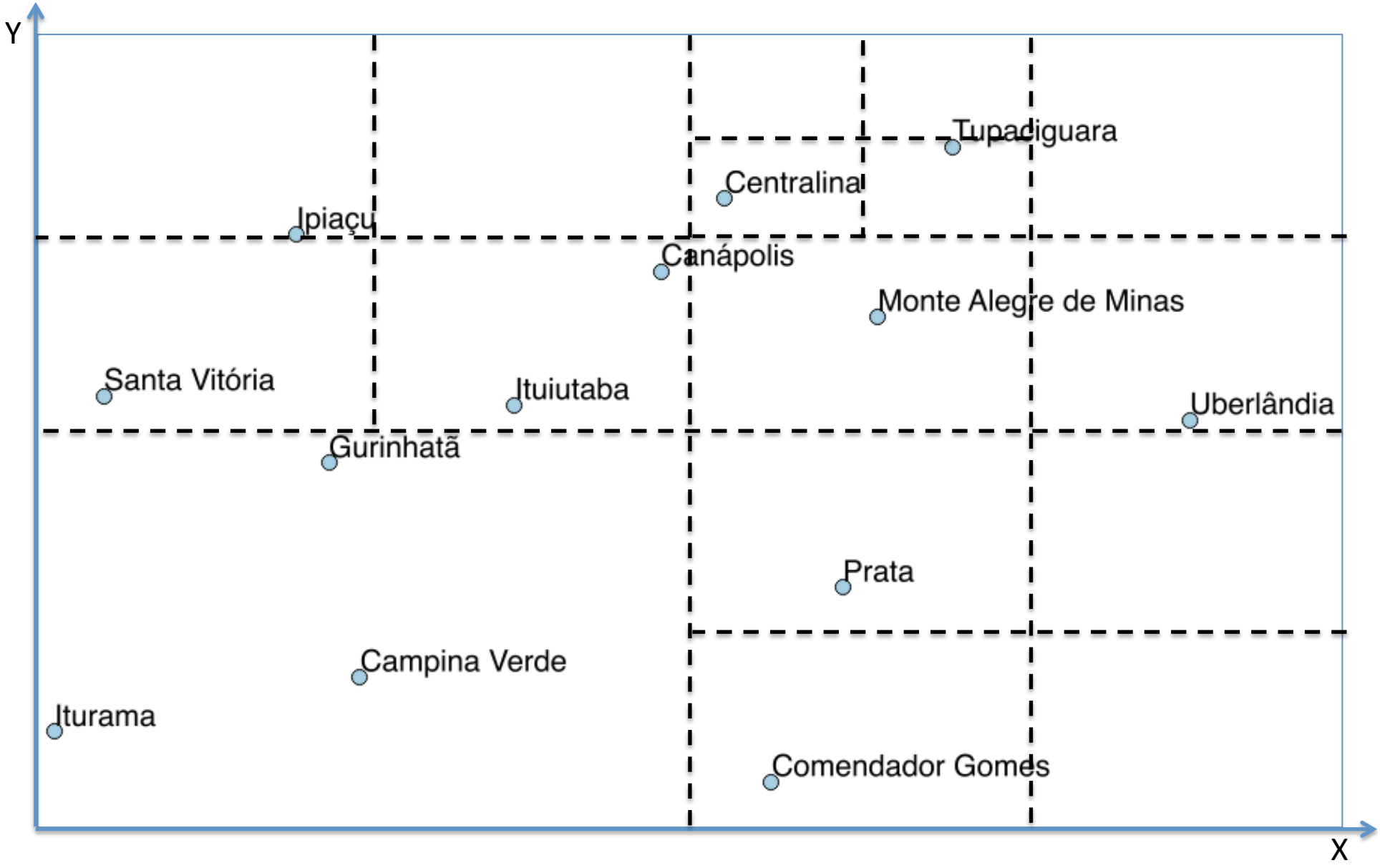
Quadtree


Y
Tupaciguara
Centralina
Ipiaçu
Canápolis
Monte Alegre de Minas
Santa Vitória
Ituiutaba
Uberlândia
Gurinhatã
Prata
Campina Verde
Iturama
Comendador Gomes
X

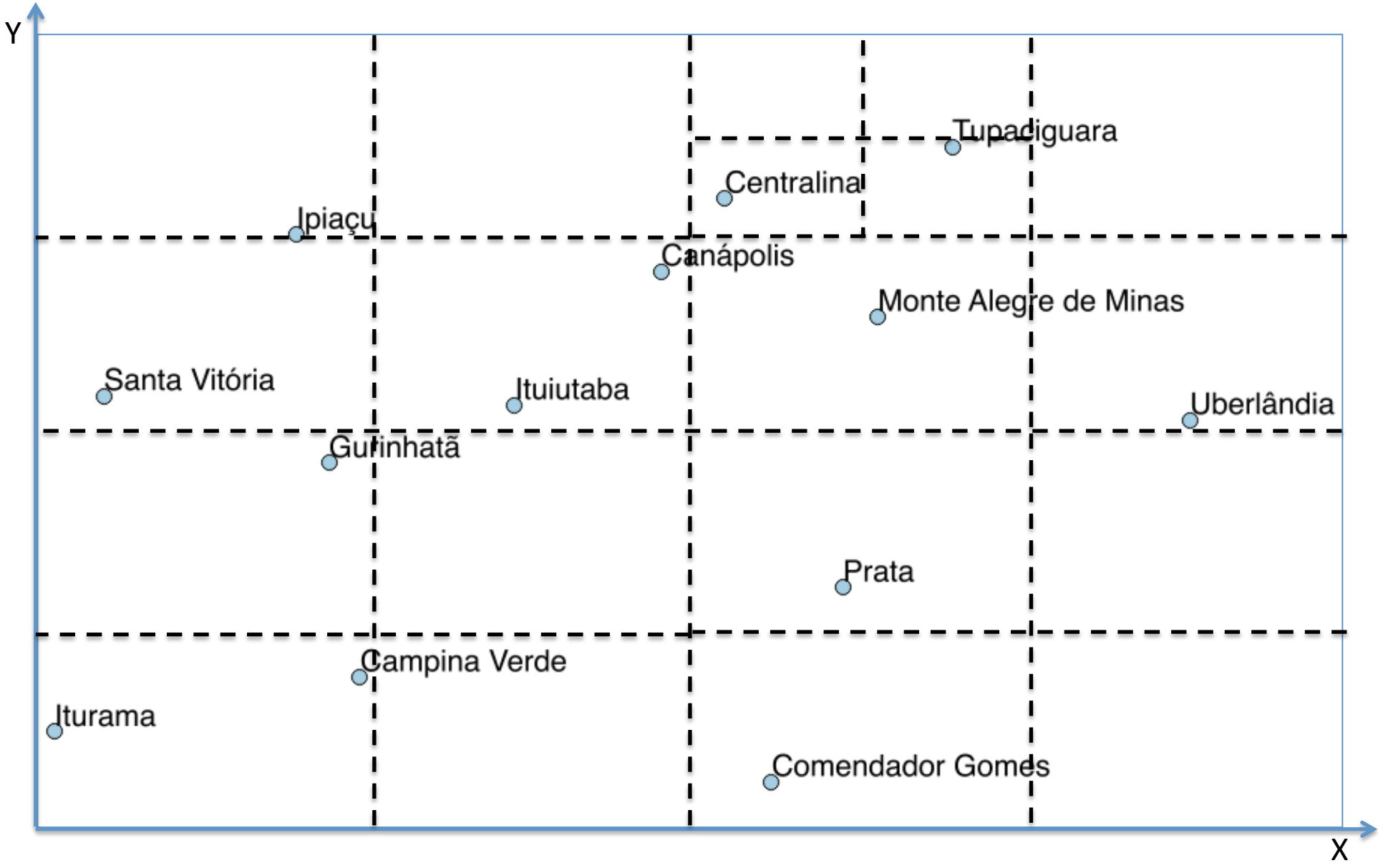
Quadtree


Y
Tupaciguara
Centralina
Ipiaçu
Canápolis
Monte Alegre de Minas
Santa Vitória
Ituiutaba
Uberlândia
Gurinhatã
Prata
Campina Verde
Iturama
Comendador Gomes
X

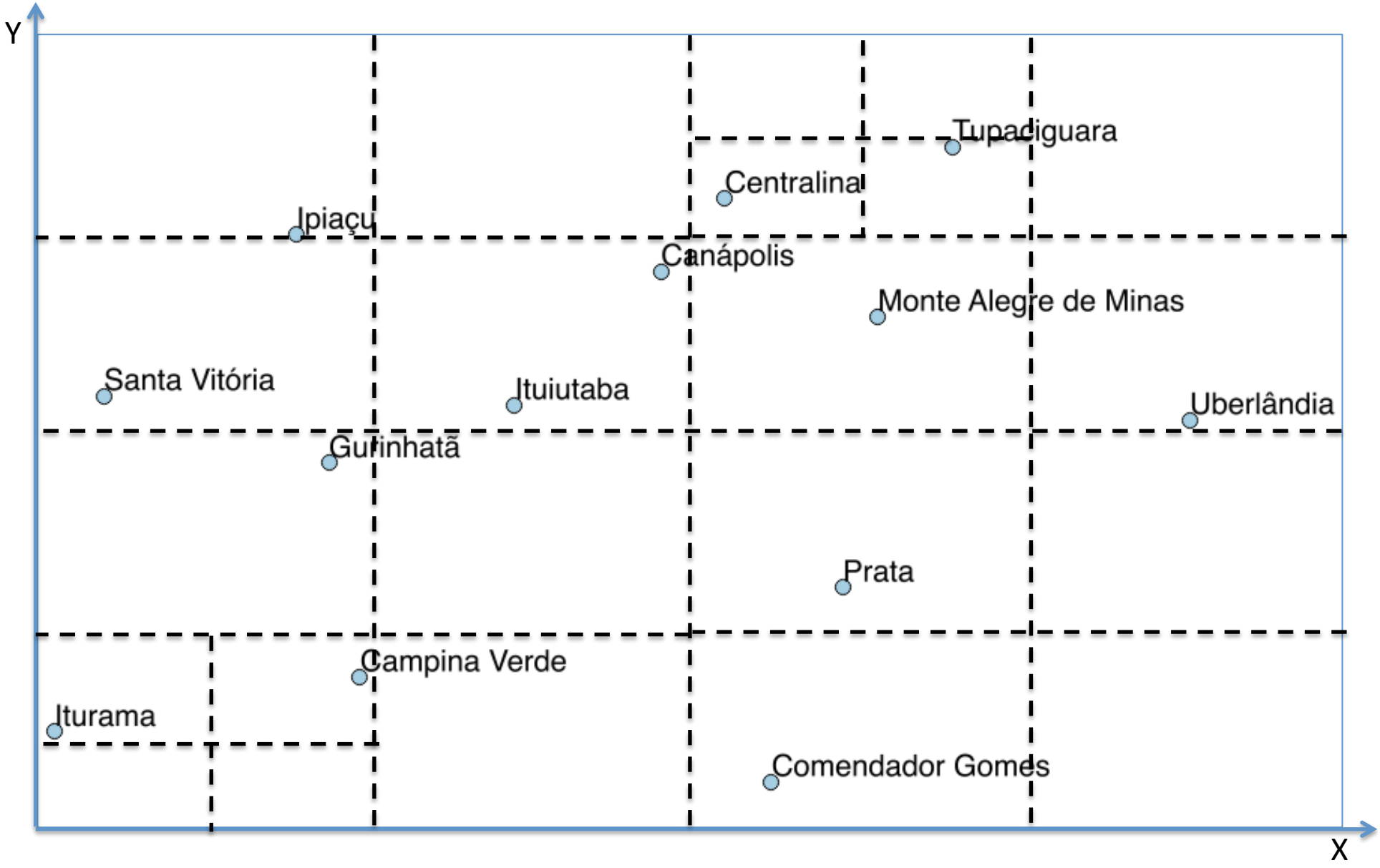
Quadtree


Y
Tupaciguara
Centralina
Ipiaçu
Canápolis
Monte Alegre de Minas
Santa Vitória
Ituiutaba
Uberlândia
Gurinhatã
Prata
Campina Verde
Iturama
Comendador Gomes
X

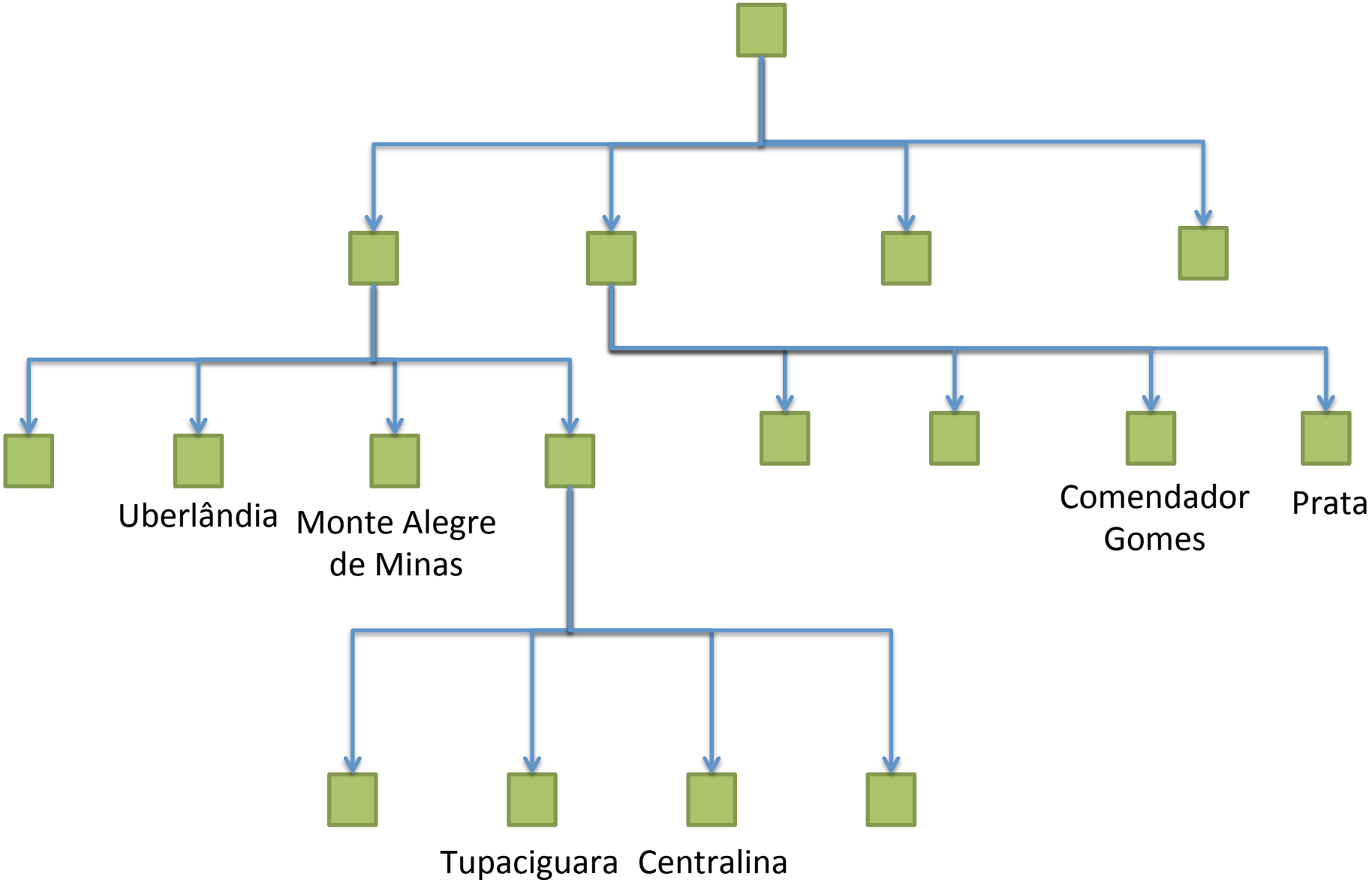
Quadtree


Y
Tupaciguara
Centralina
Ipiaçu
Canápolis
Monte Alegre de Minas
Santa Vitória
Ituiutaba
Uberlândia
Gurinhatã
Prata
Campina Verde
Iturama
Comendador Gomes
X

Quadtree


Y
Tupaciguara
Centralina
Ipiaçu
Canápolis
Monte Alegre de Minas
Santa Vitória
Ituiutaba
Uberlândia
Gurinhatã
Prata
Campina Verde
Iturama
Comendador Gomes
X

Construindo uma Quadtree



Quadtree: Considerações

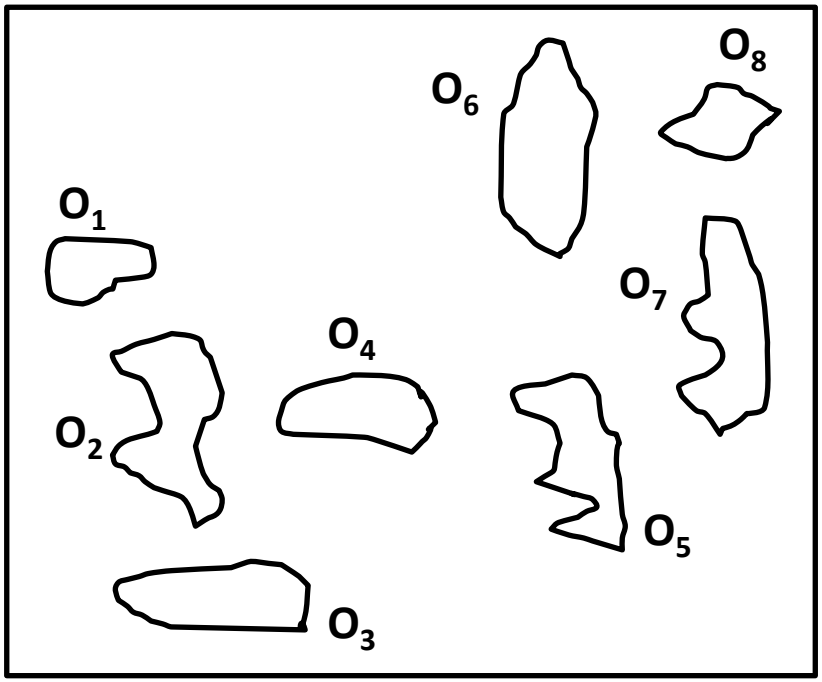
- Existe uma familha de índices espaciais denominados de Quadtree.
- Tendência de se ajustar melhor à distribuição dos dados.
- Também pode ser implementada sobre a B-tree do próprio SGBD.
- Oracle Spatial versões 8 e 9 forneciam este tipo de índice.

R-trees

Guttman (1984)

R-tree: Visão Geral

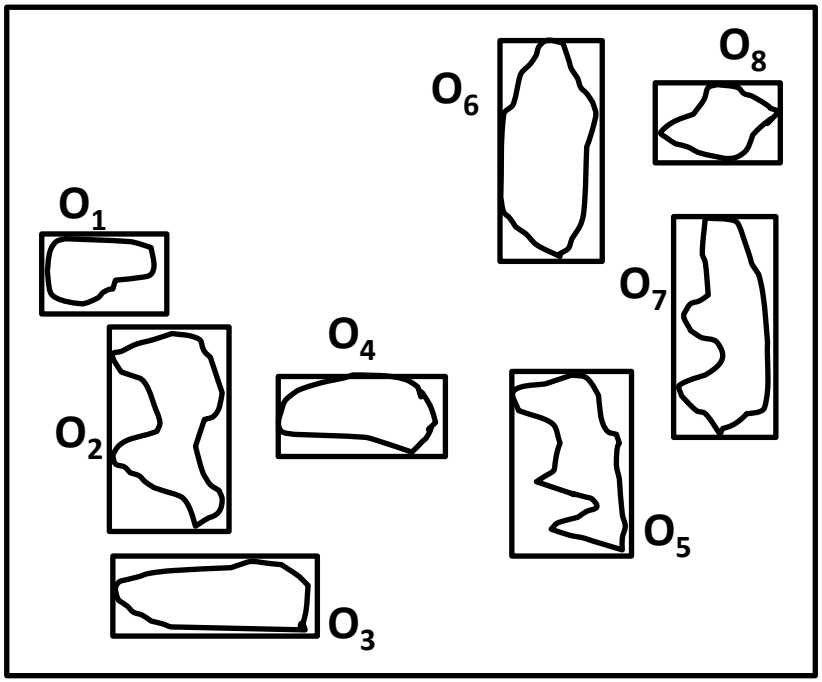
- Dado um conjunto de objetos no espaço R^k ($k > 1$), uma R-tree organiza o espaço subjacente em uma hierarquia de intervalos k-dimensionais (possivelmente com sobreposições):



8 Objetos no R^2

R-tree: Visão Geral

- Dado um conjunto de objetos no espaço R^k ($k > 1$), uma R-tree organiza o espaço subjacente em uma hierarquia de intervalos k-dimensionais (possivelmente com sobreposições):

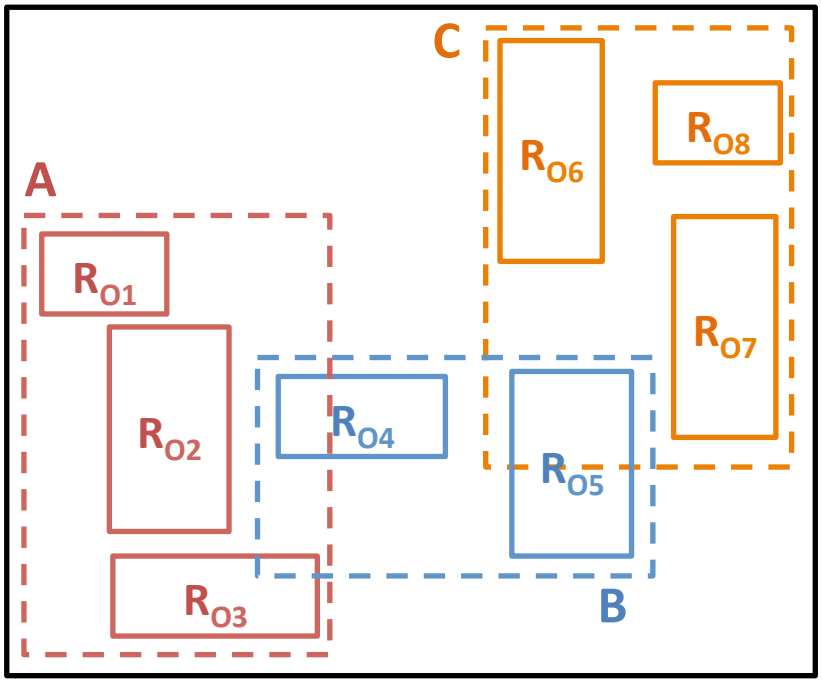


8 Objetos no R^2

MBR do objeto como aproximação

R-tree: Visão Geral

- Dado um conjunto de objetos no espaço R^k ($k > 1$), uma R-tree organiza o espaço subjacente em uma hierarquia de intervalos k-dimensionais (possivelmente com sobreposições):



8 Objetos no R^2

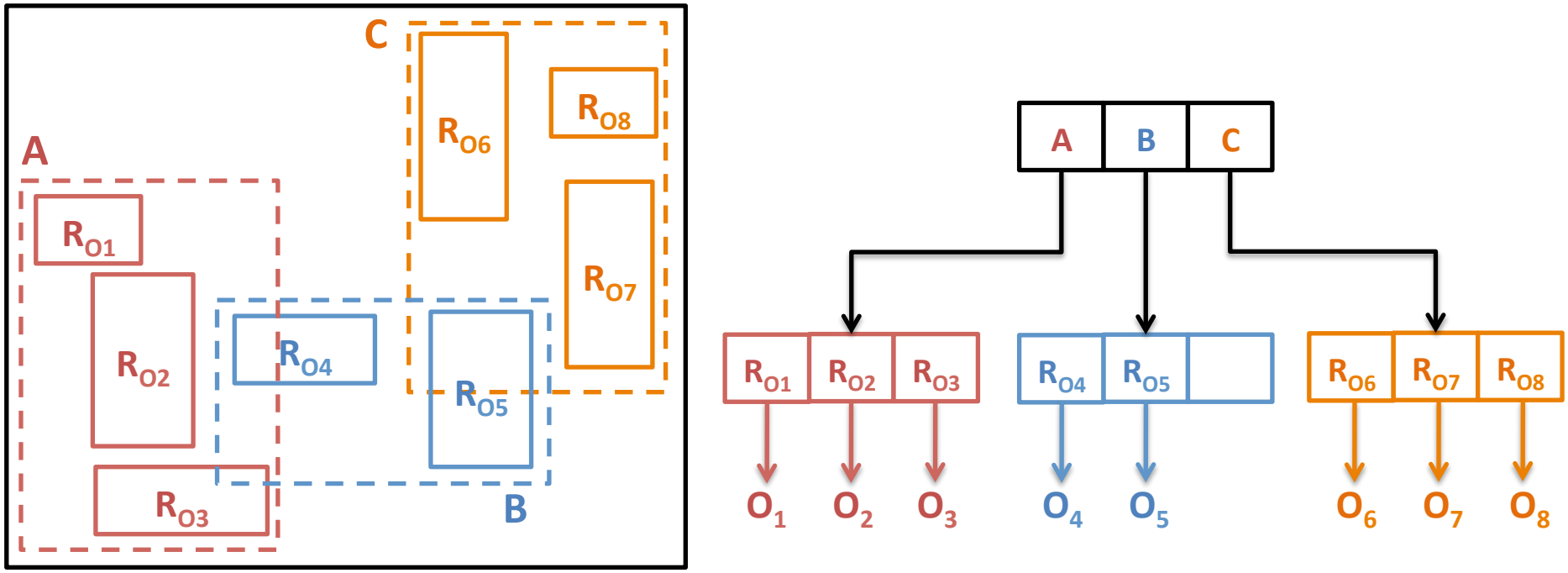
MBR do objeto como aproximação

Agrupar retângulos em uma hierarquia

R-tree: Visão Geral

- Intervalos são organizados em uma estrutura de árvore:
  - Leaf nodes: $\left[(I, \textit{tuple-id})\right]$
  - Child nodes: $\left[(I, \textit{child-ptr})\right]$

$$I = (I_0, I_1, \ldots, I_{k-1}) \quad \text{onde} \quad I_i = [a, b]$$



Uma R-tree deve satisfazer as seguintes propriedades

- O número máximo de entradas em um nó: M
- O número mínimo de entradas em um nó: $m \geq \frac{M}{2}$
- Todo nó contém entre *m* e *M* entradas válidas, a menos do nó raiz.
- Se a árvore tiver mais do que um nível, a raiz terá ao menos dois descendentes.
- Para cada entrada da forma $(I, child\text{-}ptr)$, I é o menor intervalo a conter os intervalos do nó descendente.
- Para cada entrada da forma $(I, tuple\text{-}id)$, I é o menor intervalo contendo o objeto espacial.
- Todas as folhas encontram-se no mesmo nível.

R-tree: Propriedades

- A altura de uma R-tree indexando *N* objetos espaciais é: $\left(\log_m N\right) - 1$

- O número máximo de nós é dado por:

$$\left\lceil \frac{N}{m} \right\rceil + \left\lceil \frac{N}{m^2} \right\rceil + \ldots + 1$$

- O pior caso de utilização de espaço dos nós (exceto a raiz): $\frac{m}{M}$

R-tree: Pesquisa Intervalo

- Travessia top-down:

```
Description:
  n -> search node
  I -> search interval

 1.    search(n, I)
 2.      if(is_not_leaf(n))
 3.        for_all entry in n do
 4.          if(intersects(entry.I, I))
 5.            search(entry.child-ptr, I)
 6.      else /* it is a leaf node */
 7.        for_all entry in n do
 8.          if(intersects(entry.I, I))
 9.            emit_found_candidate(entry.tuple-id)
```

R-tree: Inserções e Remoções

Idéia: aplicar uma heuristica que minimize o número de caminhos percorridos para aumentar a performance das consultas

R-tree: Operações Inserção

- Observações:
  - Quando o número de entradas válidas em um nó *n* ultrapassar *M*, u novo nó *n'* (sibling) deve ser criado e preenchido com parte das entradas do nó *n*.
  - Se o ancestral do nó *n* não tiver espaço para acomodar a entrada do novo nó *n'*, ele também terá que ser dividido (split).

R-tree: Algoritmo de Inserção

```
Description:
  I -> interval to be inserted
  n -> node

 1.    insert(I, n)
 2.      leaf_node <- choose_leaf(I, n)
 3.      if(has_room(leaf_node))
 4.        insert(I, leaf_node)
 5.        adjust_tree(leaf_node)
 6.      else /* leaf node doesn't have one more room */
 7.        leaf_node' <- split(leaf_node)
 8.        adjust_tree(leaf_node, leaf_node')
```

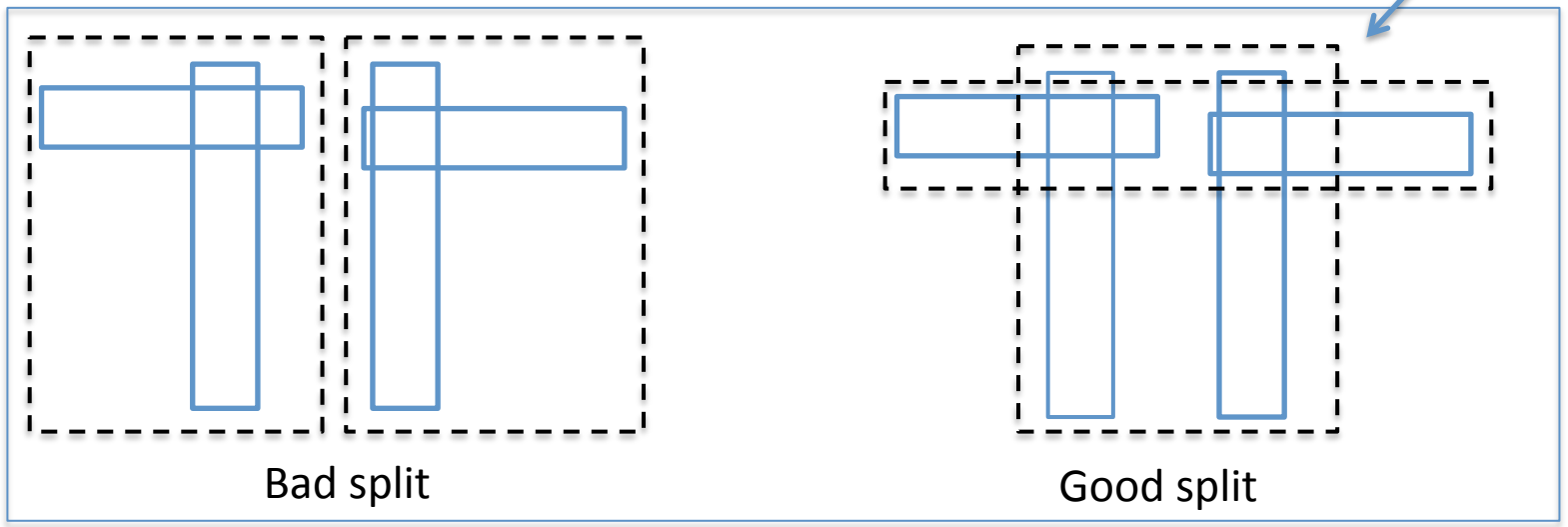
R-tree: Algoritmo de Inserção

- choose_leaf:
  - Escolhe o nó folha que requer a menor expansão de intervalo (ou MBR).

- adjust_tree:
  - Propaga mudanças para cima (upward).
  - As dvisões (splits) podem ser propagadas (upward).
  - Existem várias estratégia para fazer a divisão dos nós (splitting nodes).

R-tree: Node Splitting

- Guttman (1984) propos 3 stratégias para o particionamento de *M + 1* entradas em dois grupos:
  - Exhaustive algorithm:
    - Evaluate all the 2^{M-1} possible grouping choices => Computational Complexity: $O(2^M)$
  - Quadratic algorithm: Computational Complexity: $O(M^2)$
  - Linear algorithm: Computational Complexity: O(M)



Source: Guttman (1984)

R-tree: Operações de Remoção

- Quando o número de entradas de um nó for menor do que *m*, *m - 1* entradas deverão se movidas para outros nós irmãos (siblings).

R-tree: Algoritmo de Remoção

```
Description:
  I -> interval of the object to be removed from the index
  n -> node

 1.    delete(I, n)
 2.      leaf_node <- find_leaf(I, n)
 3.      remove_entry(I, leaf_node)
 4.      condense_tree(leaf_node)

 /* has the root node n only one child? */
 5.      if(num_children(root) == 1)
 6.        n <- first_child(n).child-ptr
```

R-tree: Algoritmo Remoção

- condense_tree:
  - Eliminará o nó se ele tiver poucas entradas e irá realizar a realocação dessas entradas (Ex: chamando *insert* para cada um deles).
  - Propaga a eliminação do nó para cima (upward), atualizando os intervalos no caminho até a raiz.

R*-trees

(Beckmann et al., 1990)

R*-tree

- Otimização baseada na combinação da área, perímetro e sobreposição de cada retângulo envolvente:
  - Não considera a minimização apenas da área.

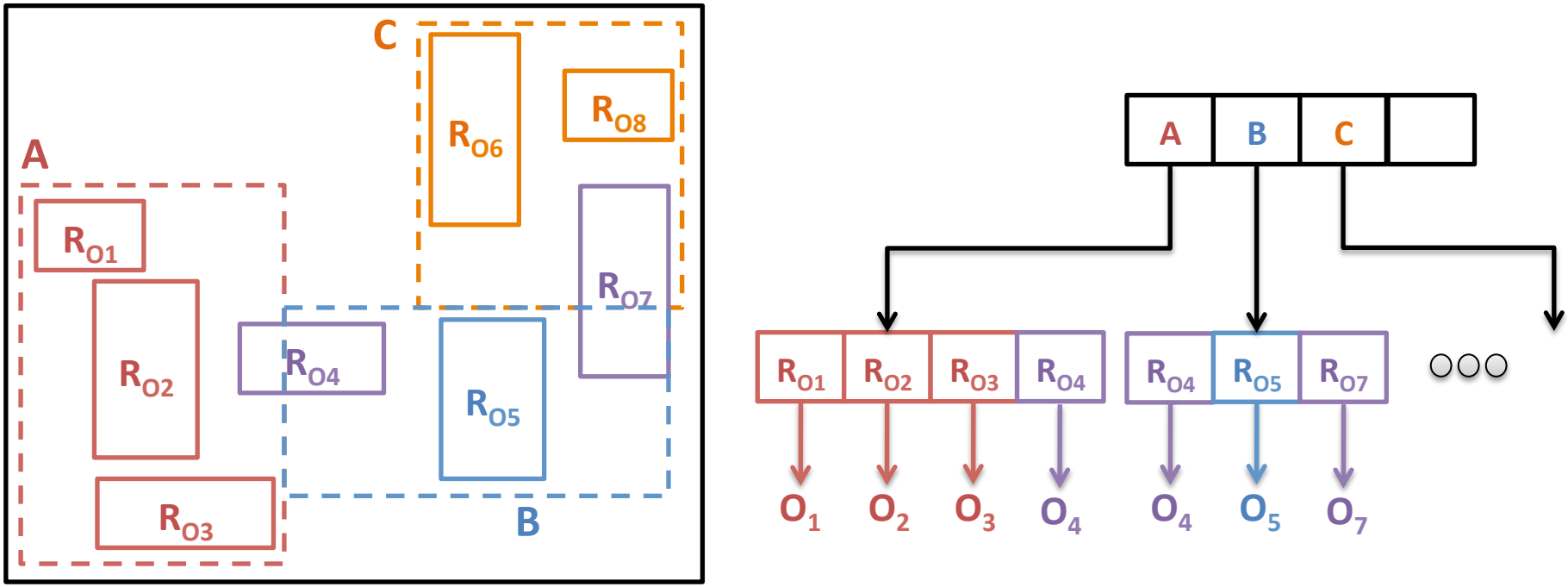
- Possui melhor performance do que as versões originais (linear e quadrática) de Guttman.

R^+-trees

(Sellis et al., 1987)

R^+-trees

- Variante da R-tree de Guttman que adota intervalos sem sobreposição:
  - Tentativa de melhorar operações de busca;
  - Drawback: necessita mais espaço.



Packed R-trees

(Roussopoulos and Leifker, 1985)

(Kamel and Faloutsos, 1993)

Packed R-trees

- Métodos para construção bottom-up de uma R-tree.
- Pack:
  - Tenta maximizar o fator de preenchimento (fill factor) dos nós.
- Variações bem conhecidas:
  - Lower-x packed R-tree
  - Distance Sorted R-tree
  - Hilbert R-tree

R-trees: Considerações

- Vários SGBD's optaram po utilizar este tipo de índice (R^*-tree):
  - Oracle, PostgreSQL + PostGIS, SQLite + SpatiaLite, MySQL.

- A grande vantagem deste tipo de índice:
  - Dinâmico (auto-ajustável).

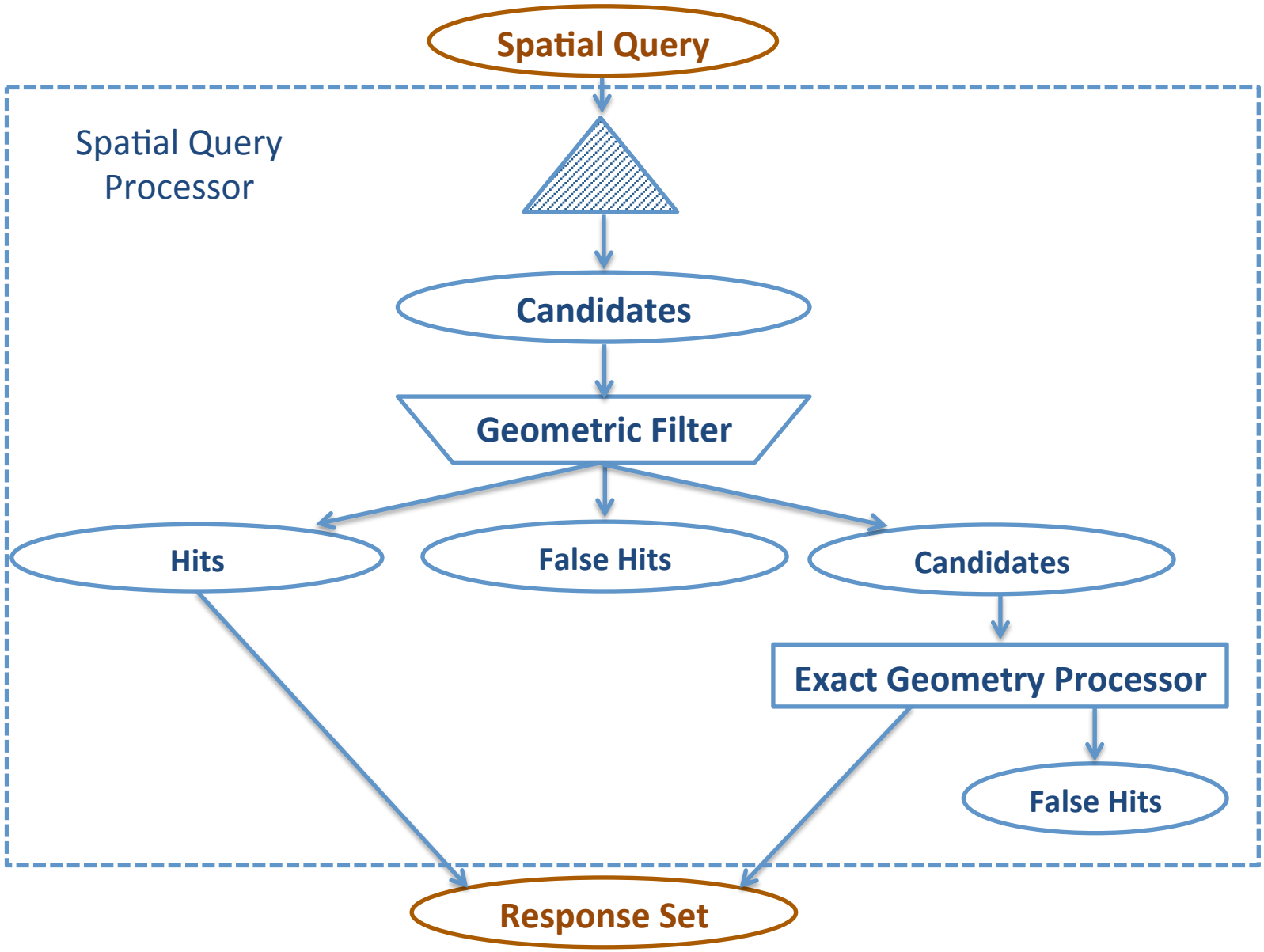
Processamento de Consultas Espaciais (Spatial Query Processing)

Seleção Espacial (Spatial Selections)

Junção Espacial (Spatial Joins)

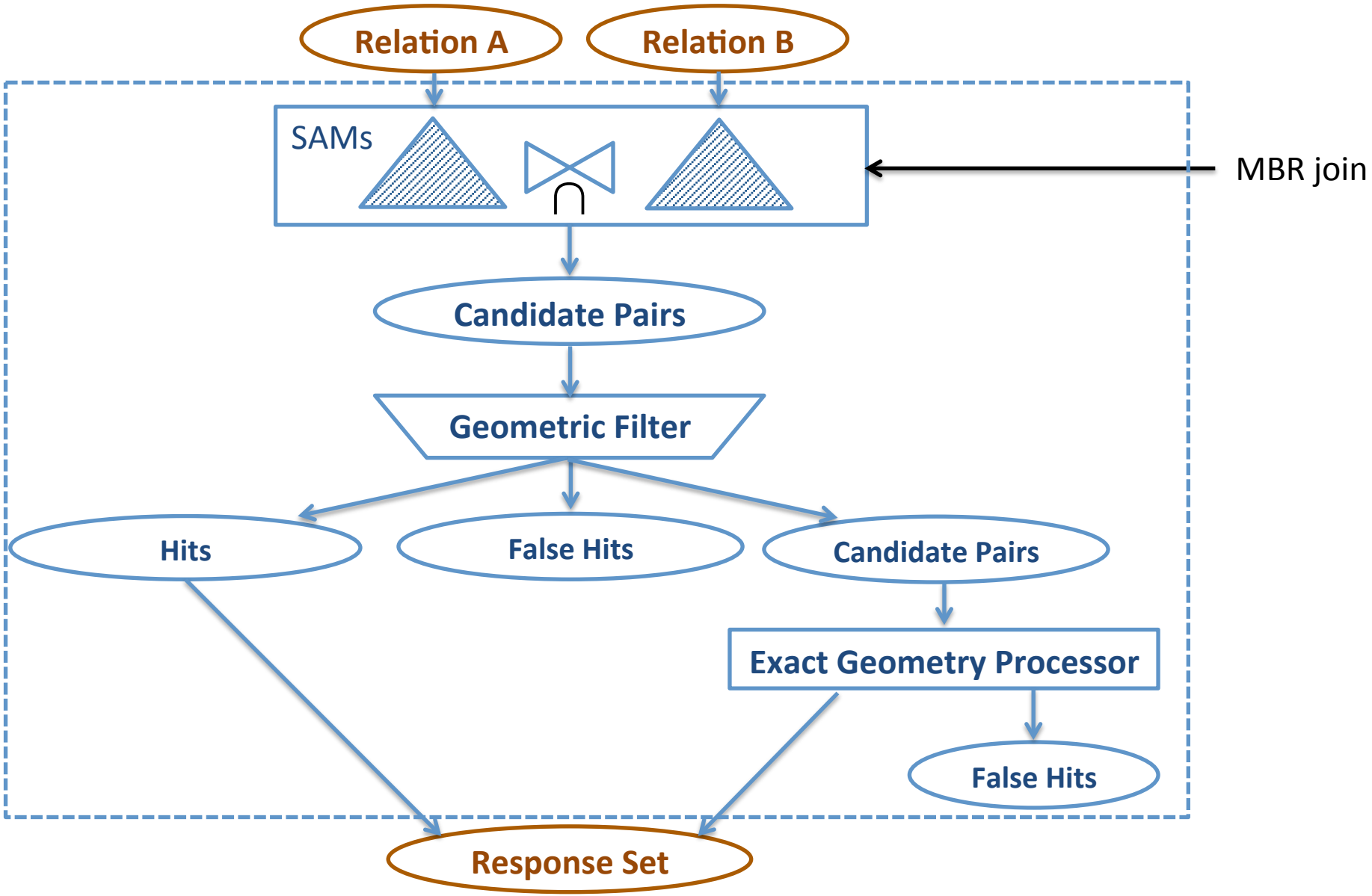
Consulta Espacial (Spatial Query)

(Kriegel et al., 1993)



Junção Espacial (Spatial Joins)

(Brinkhoff et al., 1994)



Indexação de Colunas Geométricas no PostGIS

Criação de Índices Espaciais

- O PostGIS implementa um R-tree sobre o GiST do PostgreSQL.

- Para cria um índice espacial em uma coluna geométrica podemos usar a seguinte sintaxe:

```
CREATE INDEX spidx_tabela_col
          ON tabela
  USING gist(coluna_geom);
```

- Ver a definição das tabelas importadas para o banco de dados.

Hands-on

- Usando o script plpgsql-pts-table-pg.sql, criar uma tabela de pontos chamada pts_pgis_1m com 1.000.000 pontos:

```
SELECT criar_tabela_pontos_pgis('pts_pgis_1m', 1000000);
```

- Criar um índice sobre a coluna gid:

```
CREATE INDEX pts_pgis_1m_pt_spidx ON pts_pgis_1m
  USING GIST (geom);
```

Hands-on

- Verificar a navegação no QGIS com e sem índice espacial na tabela pts_pgis_1m.
- Qual o espaço de armazenamento utilizado pela versão sem extensão espacial?
- Qual o espaço de armazenamento utilizado pela versão com extensão PostGIS?
- Qual o espaço de armazenamento de suas tabelas espaciais e dos índices espaciais?

Índices n-dimensionais no PostGIS

```
CREATE INDEX [indexname] ON [tablename]
  USING GIST ([geometryfield] gist_geometry_ops_nd);
```

Last Question…

How to process efficiently the nearest neighbor query?

Referências

Artigos

- BENTLEY, J. L. ***Multidimensional binary search trees used for associative searching***. *Communications of the ACM,* v. 18, n. 9, September 1975, pp. 509-517.

- ROBINSON, J. T. 1981. ***The K-D-B-tree: a search structure for large multidimensional dynamic indexes***. In Proceedings of the 1981 ACM SIGMOD international conference on Management of data (SIGMOD '81). ACM, New York, NY, USA, pp. 10-18.

- LOMET, D. B.; SALZBERG, B. ***The hB-tree: a multiattribute indexing method with good guaranteed performance***. ACM Trans. Database Syst. 15, 4, December 1990, pp. 625-658.

Artigos

- NIEVERGELT, J.; HINTERBERGER, H.; SEVCIK, K. C. ***The Grid File: An Adaptable, Symmetric Multikey File Structure***. ACM Trans. Database Syst. 9, 1, March 1984, pp. 38-71.

Artigos

- FINKEL, R. A.; BENTLEY, J. L. ***Quad trees: A data structure for retrieval on composite keys***. *Acta Informatica*, 4(1):1-9, 1974.

- SAMET, H., 1984. ***The Quadtree and Related Hierarchical Data Structures***. ACM Computing Surveys, v. 16, p. 187-260.

- LAWDER, J. K. ***Calculation of Mappings Between One and n-dimensional Values Using the Hilbert Space-filling Curve***.

Artigos

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

- Timos K. Sellis, Nick Roussopoulos, and Christos Faloutsos. 1987. ***The R+-Tree: A Dynamic Index for Multi-Dimensional Objects***. In *Proceedings of the 13th International Conference on Very Large Data Bases* (VLDB '87), Peter M. Stocker, William Kent, and Peter Hammersley (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 507-518.

- Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1990. ***The R*-tree: an efficient and robust access method for points and rectangles***. *SIGMOD Rec.* 19, 2 (May 1990), pp. 322-331.

Artigos

- Nick Roussopoulos and Daniel Leifker. 1985. ***Direct spatial search on pictorial databases using packed R-trees***. In *Proceedings of the 1985 ACM SIGMOD international conference on Management of data* (SIGMOD '85). ACM, New York, NY, USA, 17-31.

- Ibrahim Kamel and Christos Faloutsos. 1993. ***On packing R-trees***. In *Proceedings of the second international conference on Information and knowledge management* (CIKM '93), Bharat Bhargava, Tim Finin, and Yelena Yesha (Eds.). ACM, New York, NY, USA, 490-499.

Artigos

- Kriegel H.-P., Brinkhoff T., Schneider R. ***Efficient Spatial Query Processing in Geographic Database Systems***. IEEE Data Engineering Bulletin, Vol. 16, No. 3, 1993, pp. 10-15.

- Thomas Brinkhoff, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1994. ***Multi-step processing of spatial joins***. *SIGMOD Rec.* 23, 2 (May 1994), 197-208.

- Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. 1993. ***Efficient processing of spatial joins using R-trees***. *SIGMOD Rec.* 22, 2 (June 1993), 237-246.

Artigos

- GAEDE, V.; GÜNTHER, O. ***Multidimensional access methods***. ACM Computing Surveys, v. 30, p. 170-231, 1998.

- Joseph M. Hellerstein, Jeffrey F. Naughton, and Avi Pfeffer. 1995. ***Generalized Search Trees for Database Systems***. In Proceedings of the 21th International Conference on Very Large Data Bases (VLDB '95), Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 562-573.