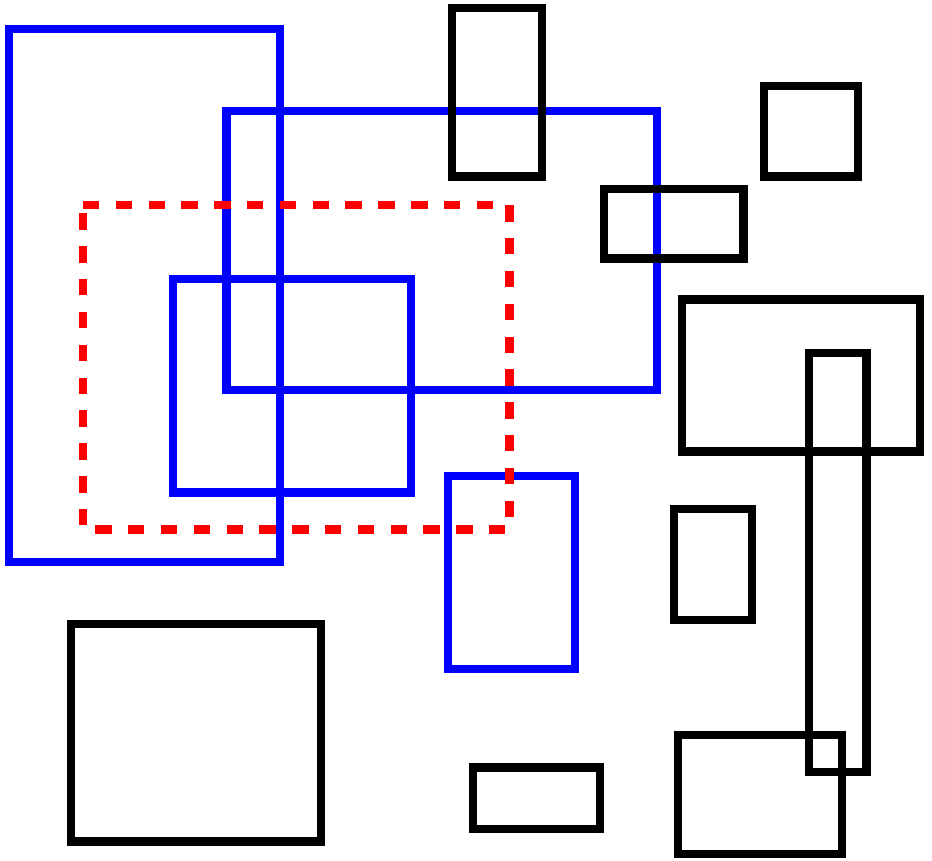
Priority R-Tree



Paper by:

Lars Arge, Mark de Berg, Herman J. Haverkort, Ke Yi

Presentation by:

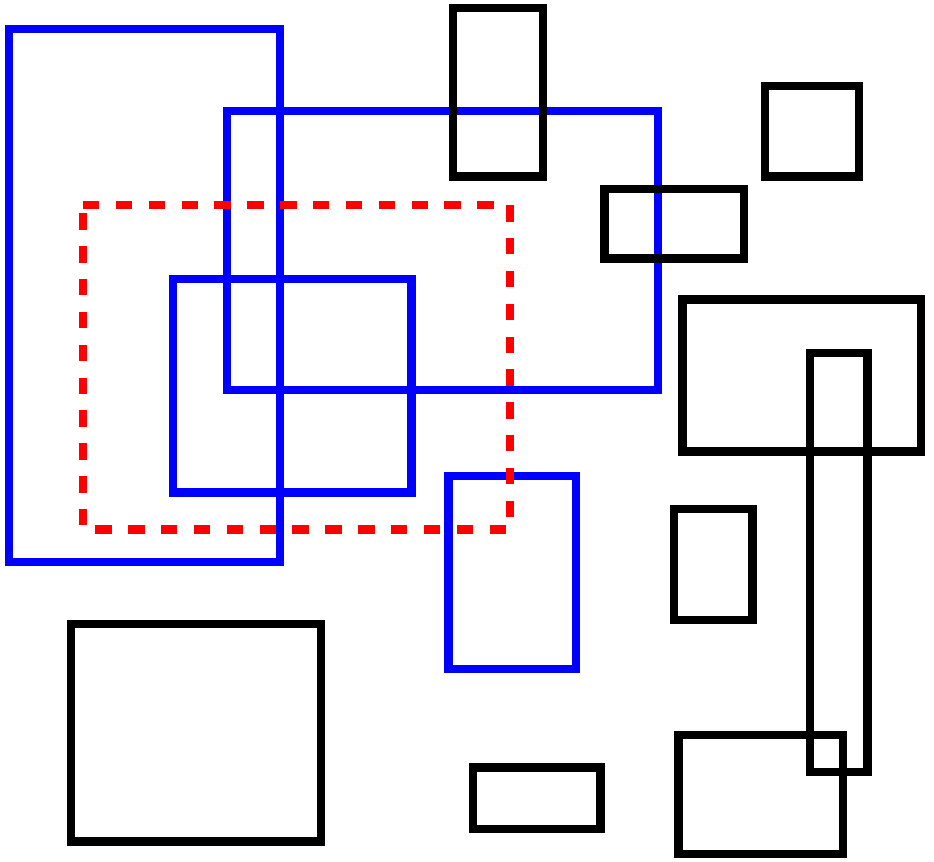
Guilherme Fonseca

Professor:

Hanan Samet

04/2005

Problem



- Data: Set of boxes *S*.
- Query: Given a box *q*, find all boxes in *S* that intersect *q*.
- Data is stored on the disk.
- Bounding boxes can be used to approximate complex shapes.

Goal

- Efficient in **practice** and **worst-case analysis**.

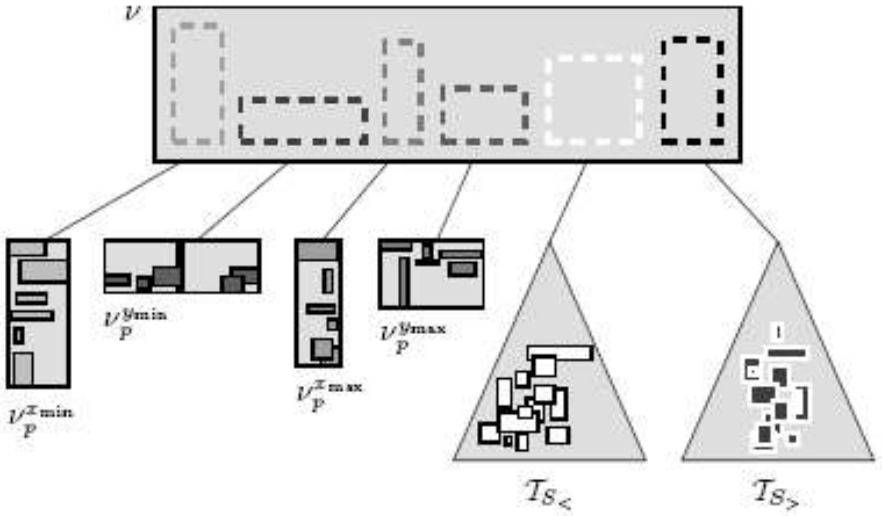
- Competitive with the best R-Tree variants on real-life data and nicely distributed data.
- Significantly outperform best R-Tree variants on extreme data.
- Optimal asymptotic worst-case number of disk accesses.
- The PR-tree is an R-tree, so all queries can be performed the same way.

Complexity

- N: Number of boxes stored.
- T: Number of boxes reported in the query.
- B: Size of disk block.
- d: Dimension of space.

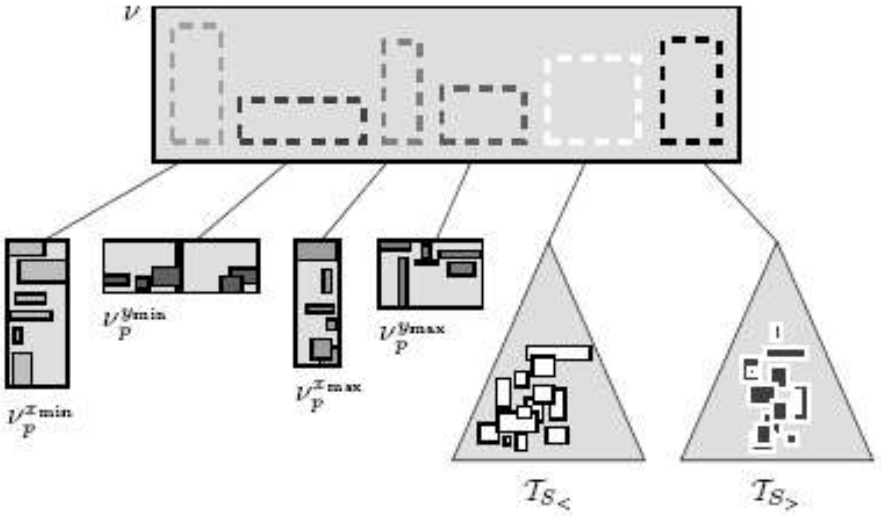
- Query I/Os: $O((N/B)^{1-1/d} + T/B)$

- Other structures may visit all leaves in the tree even when T=0!!!



Pseudo PR-tree

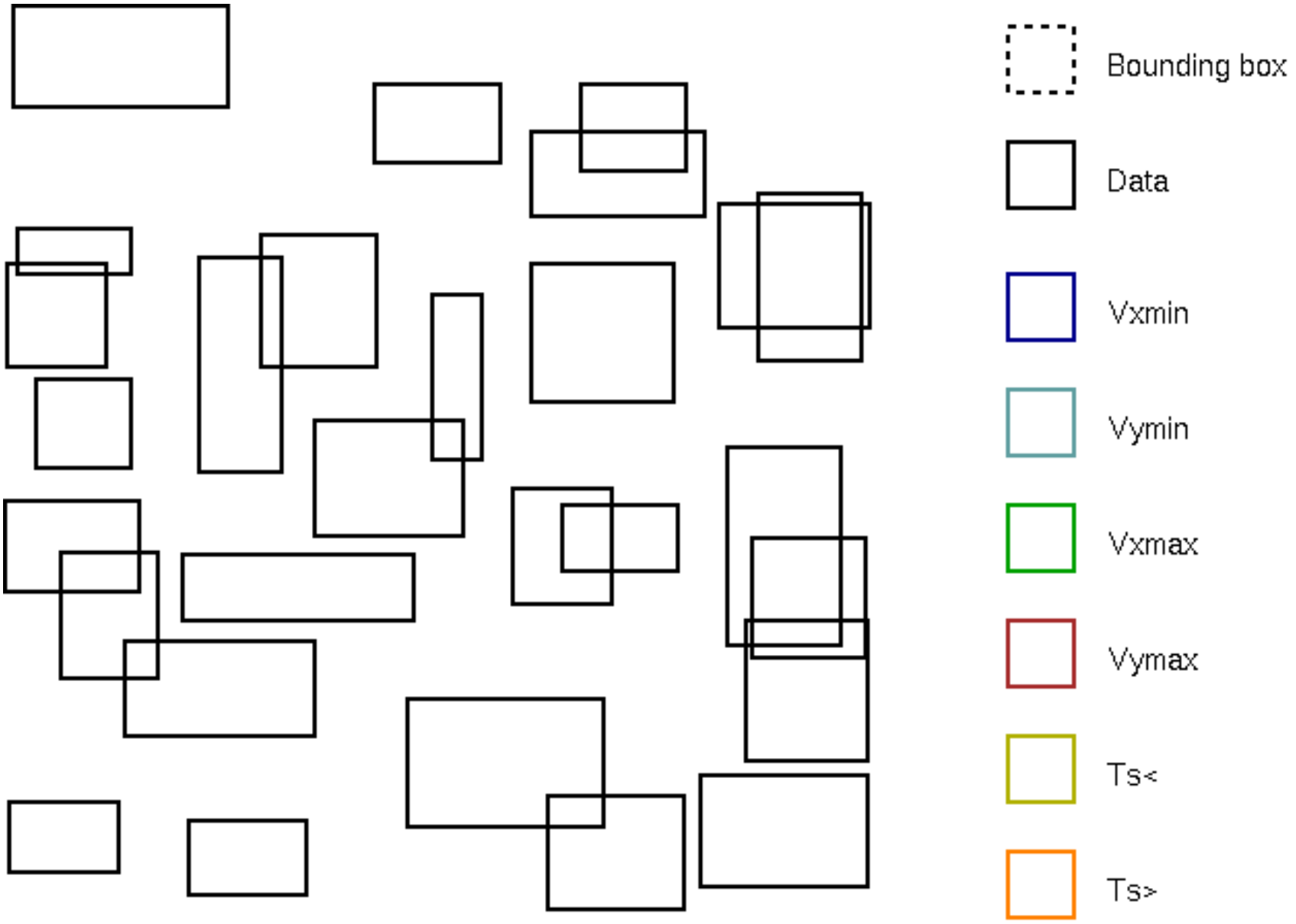
- *Vxmin*: *B* boxes with minimal *xmin* coordinate.
- *Vymin*: *B* **remaining** boxes with minimal *ymin* coordinate.
- *Vxmax*: ... maximal *xmax* coordinate.
- *Vymax*: ... maximal *ymax* coordinate.



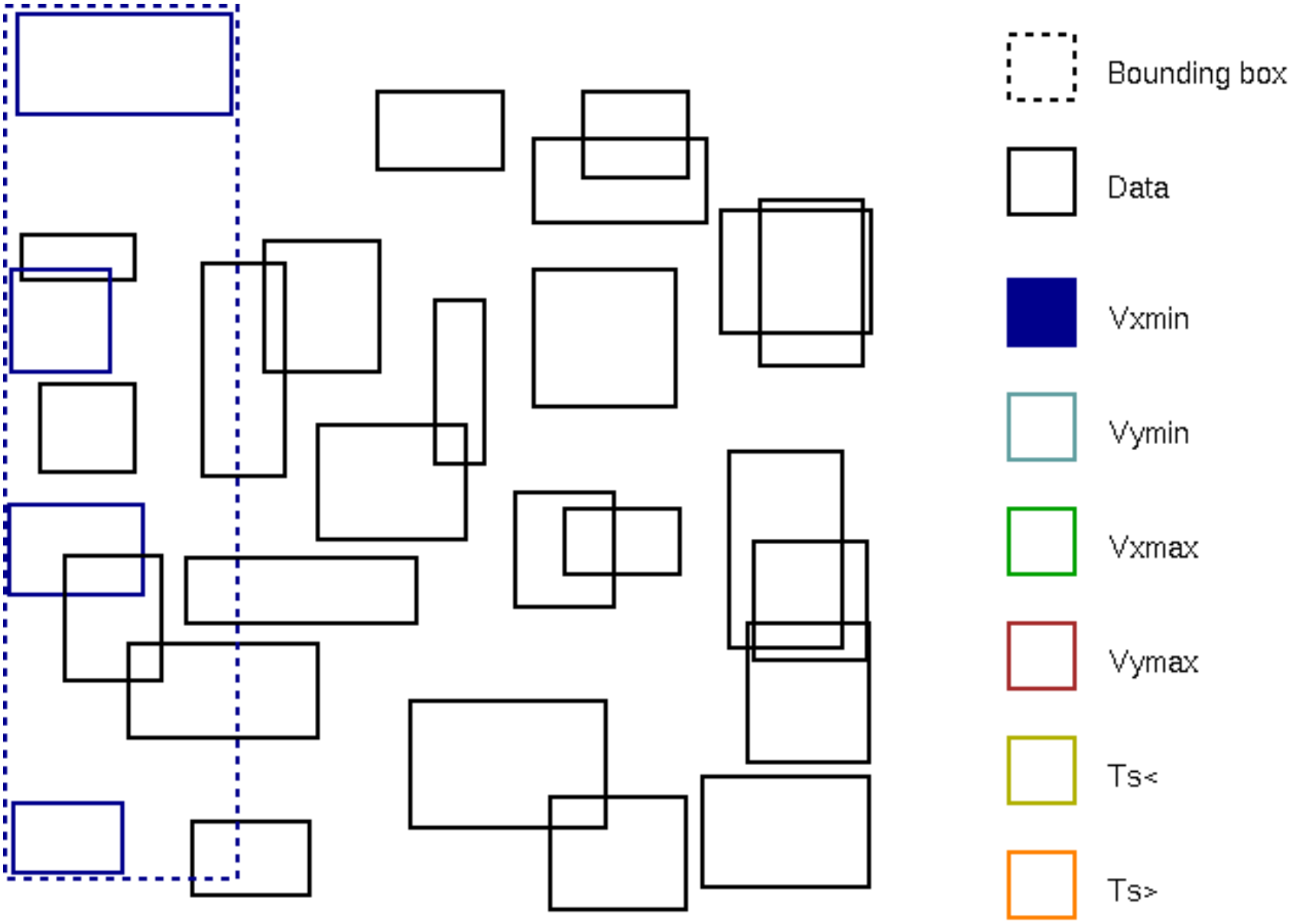
Pseudo PR-tree

- Remaining boxes are split in two according to *xmin*, *xmax*, *ymin*, *ymax*, in a **round-robin** fashion.
- The two subtrees are built recursively.
- The round-robin split is essential for the worst case analysis. It makes the structure behave like a kd-tree.
  (Remember a rectangle in d dimensions is a point in 2d dimensions.)

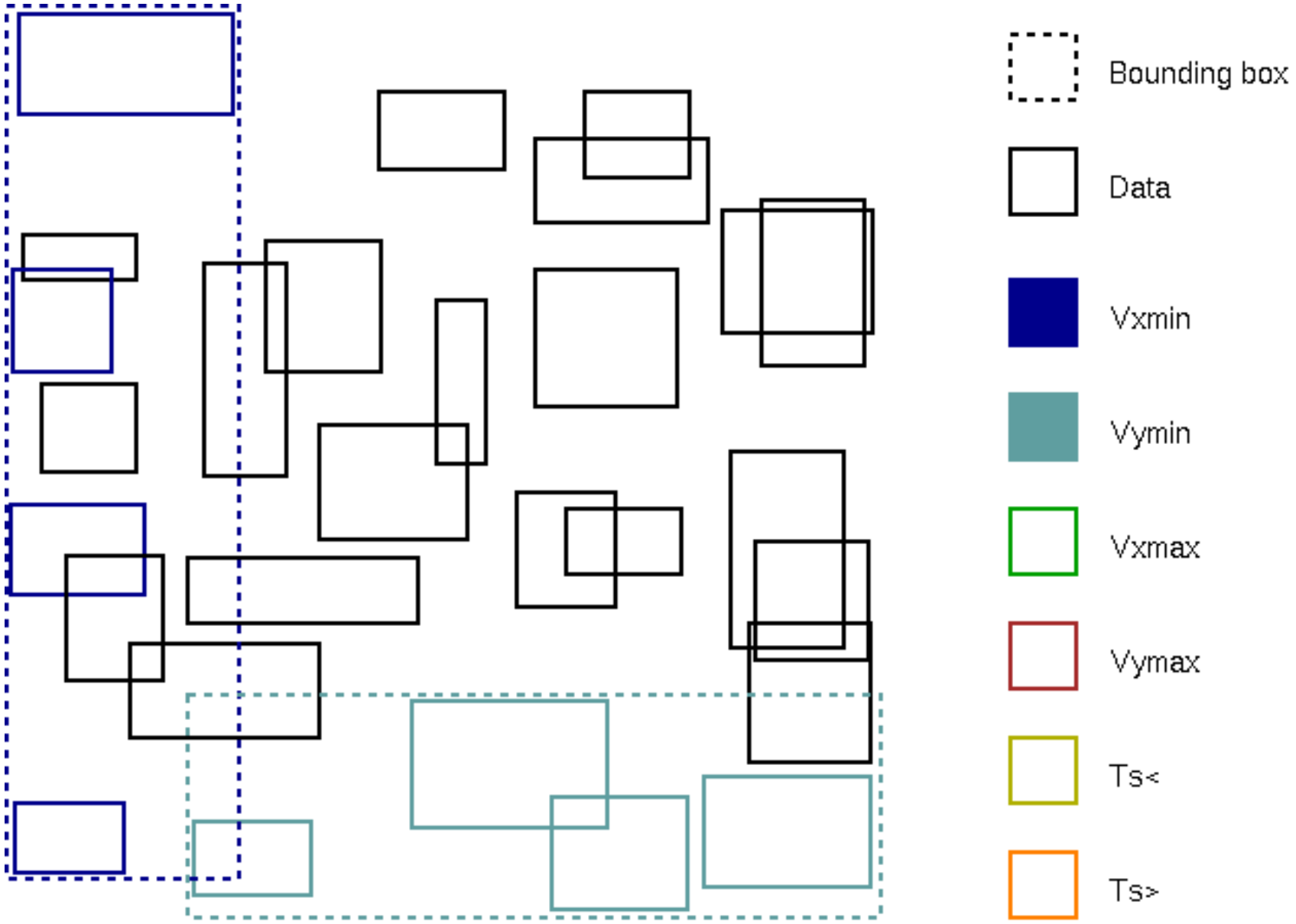
- Example with *B*=4



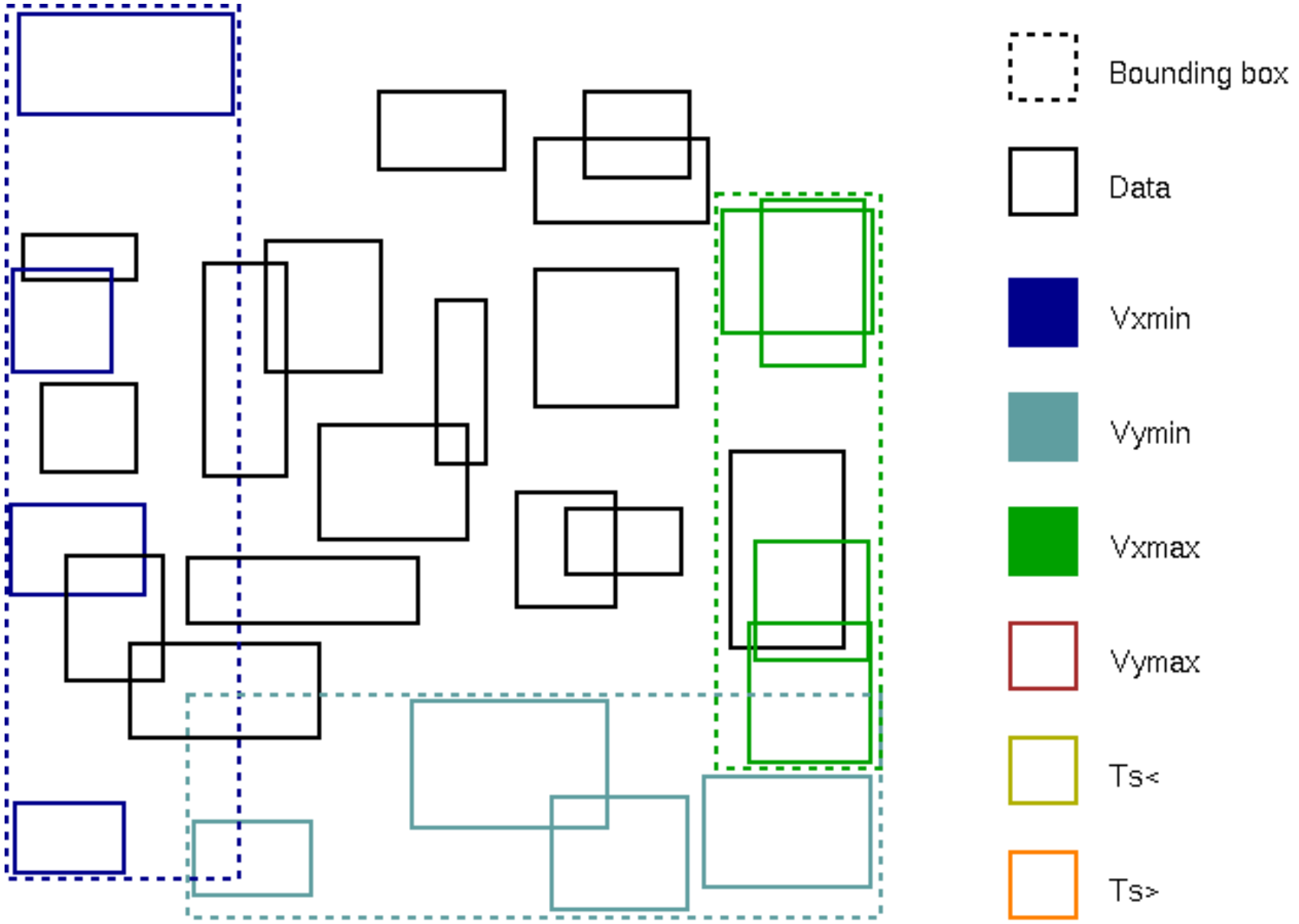
- *Vxmin*: 4 boxes with minimal *xmin* coordinate.



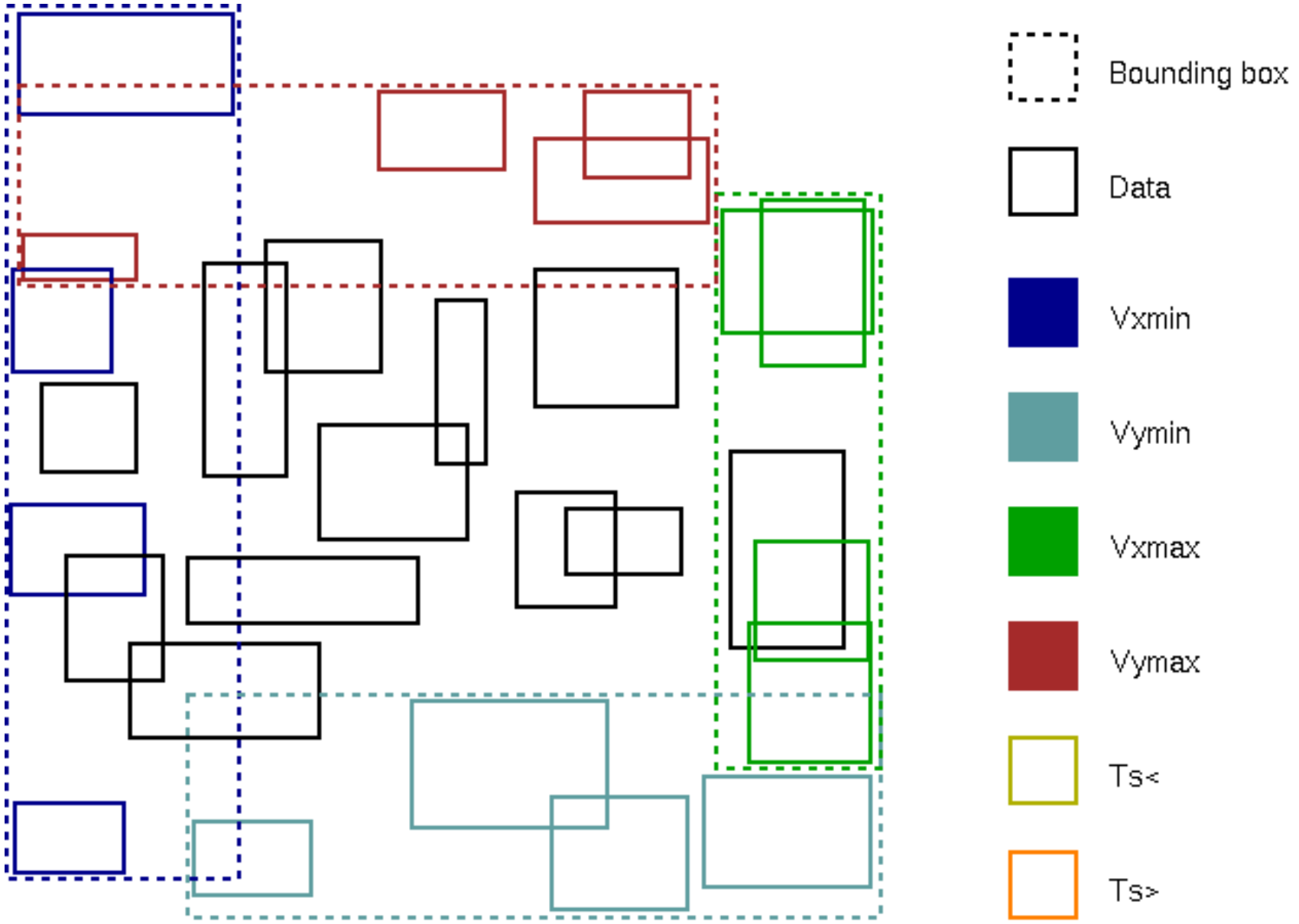
- *Vymin*: 4 remaining boxes with minimal y*min* coordinate.



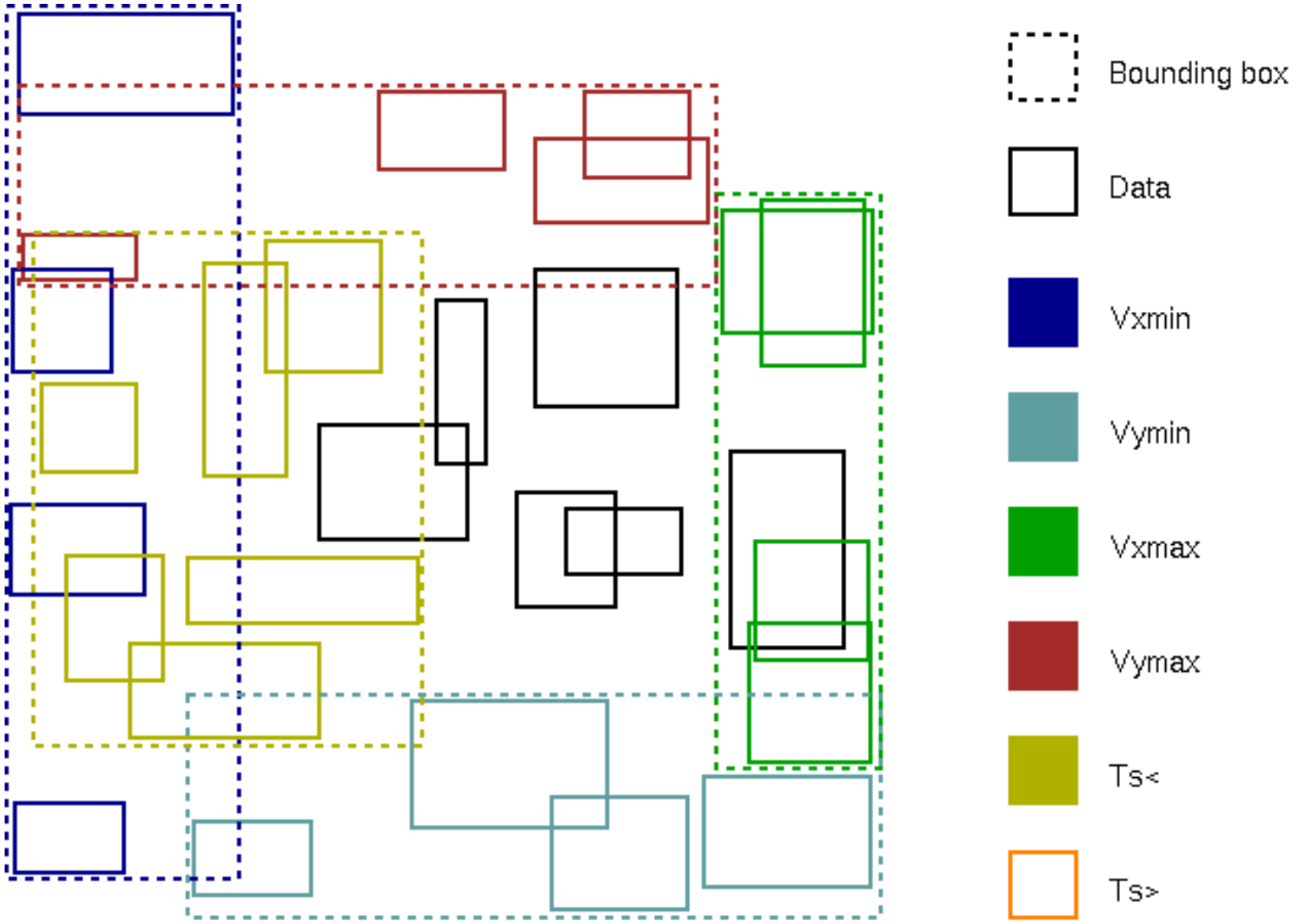
- *Vxmax*: 4 remaining boxes with maximal *xmax* coordinate.



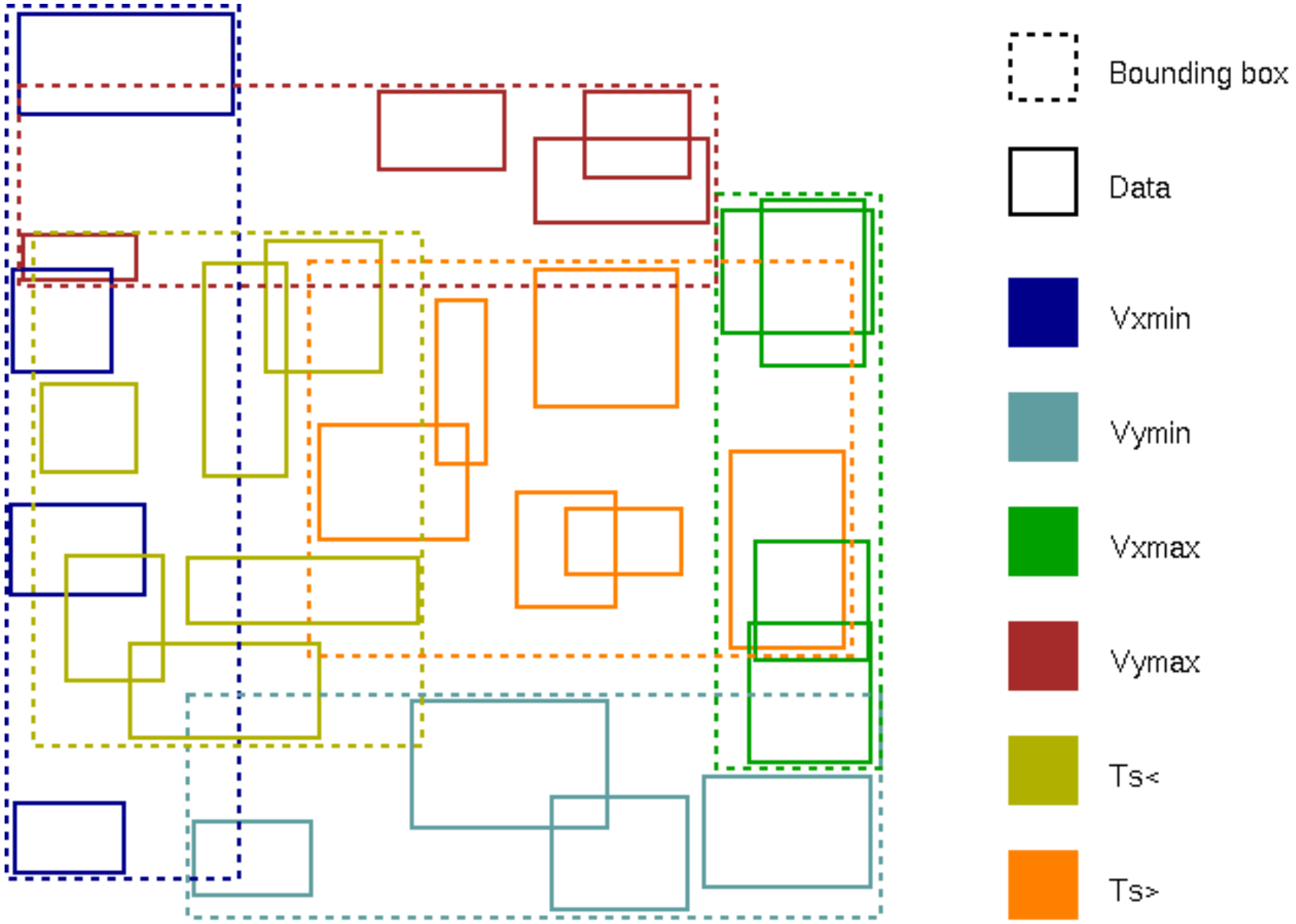
- *Vymax*: 4 remaining boxes with maximal *xmax* coordinate.



- *Ts*<: Half remaining boxes with minimal x*min* coordinate.

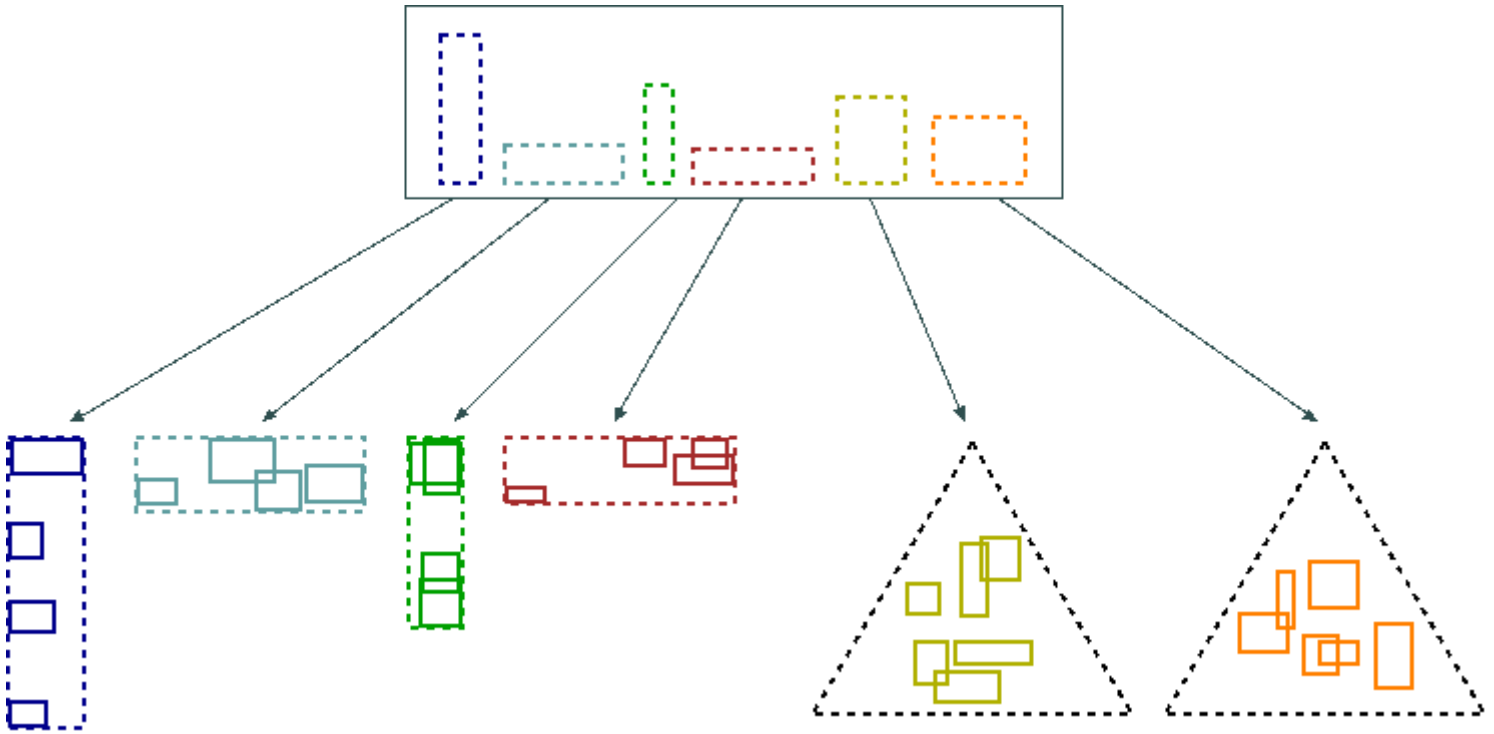


– *Ts*>: Remaining boxes.



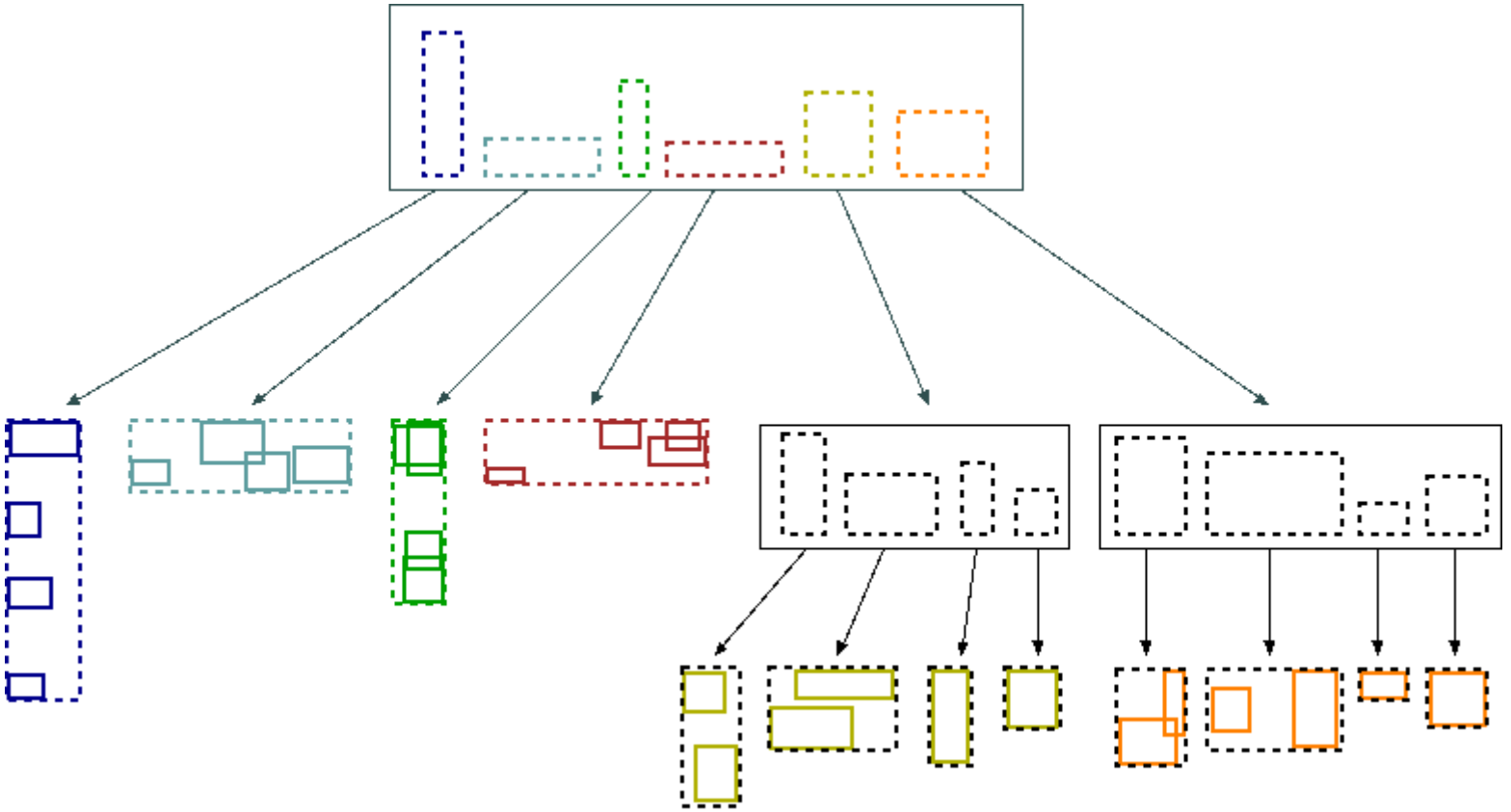
Pseudo PR-tree

- Pseudo PR-trees are built recursively for *Ts*< and *Ts*>.



Pseudo PR-tree

- Pseudo PR-trees are built recursively for *Ts*< and *Ts*>.



R-tree

- R-trees must have:
  - all leafs on the same level;
  - all internal nodes (except root) must have $\Theta(B)$ children;
  - all leaves must store $\Theta(B)$ boxes.
- The pseudo PR-tree will be used to construct the PR-tree, which is an R-tree.

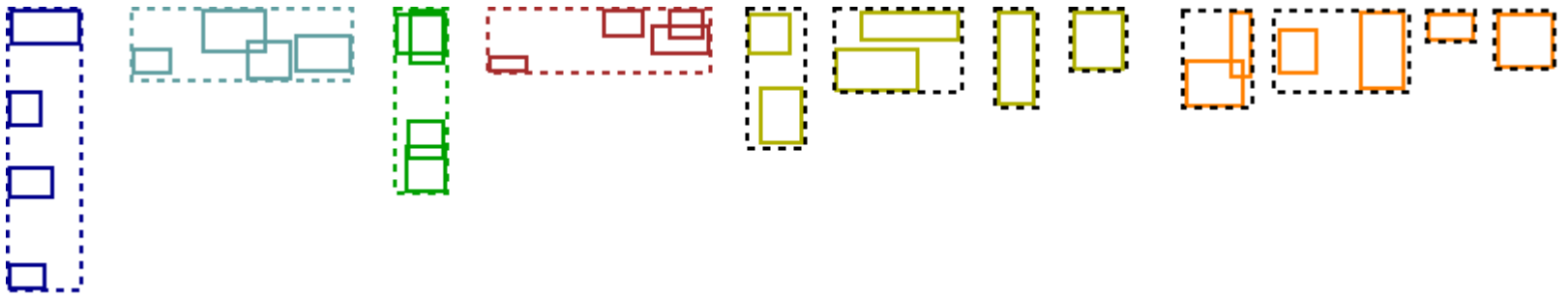
PR-tree

Constructed **bottom-up** using pseudo PR-trees.

1. Build pseudo PR-tree and get only the leaves.
2. Find the minimum bounding box for each leaf.
3. Build pseudo PR-tree of bounding box and get only the leaves.
4. Repeat from step 2 until the pseudo PR-tree has all leafs at level 1 (adjacent to the root).

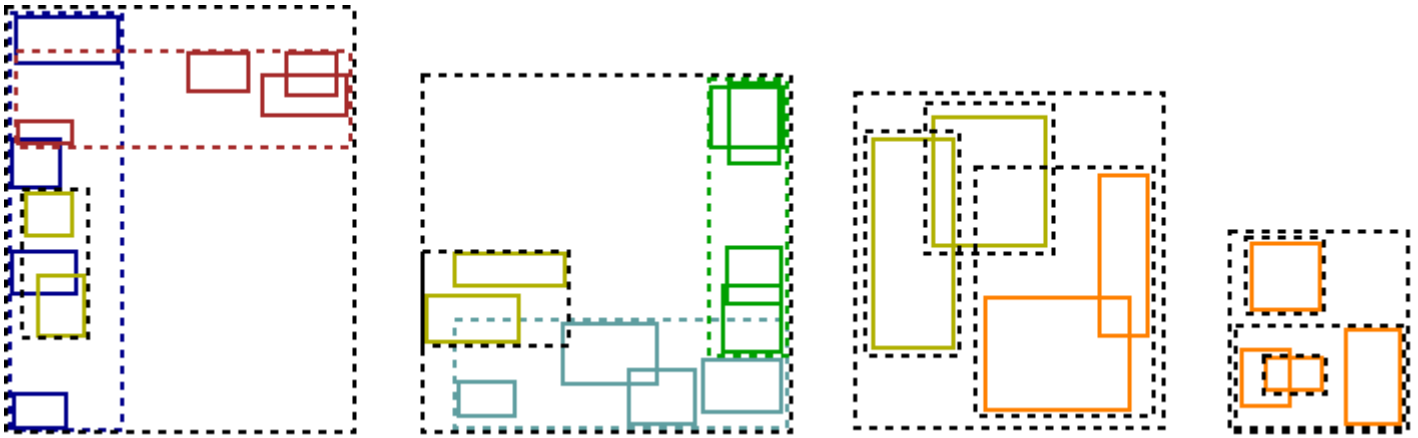
PR-tree

- Build pseudo PR-tree and get only the leaves.
- Find the minimum bounding box for each leaf.
- These will be the leaves of the PR-tree.



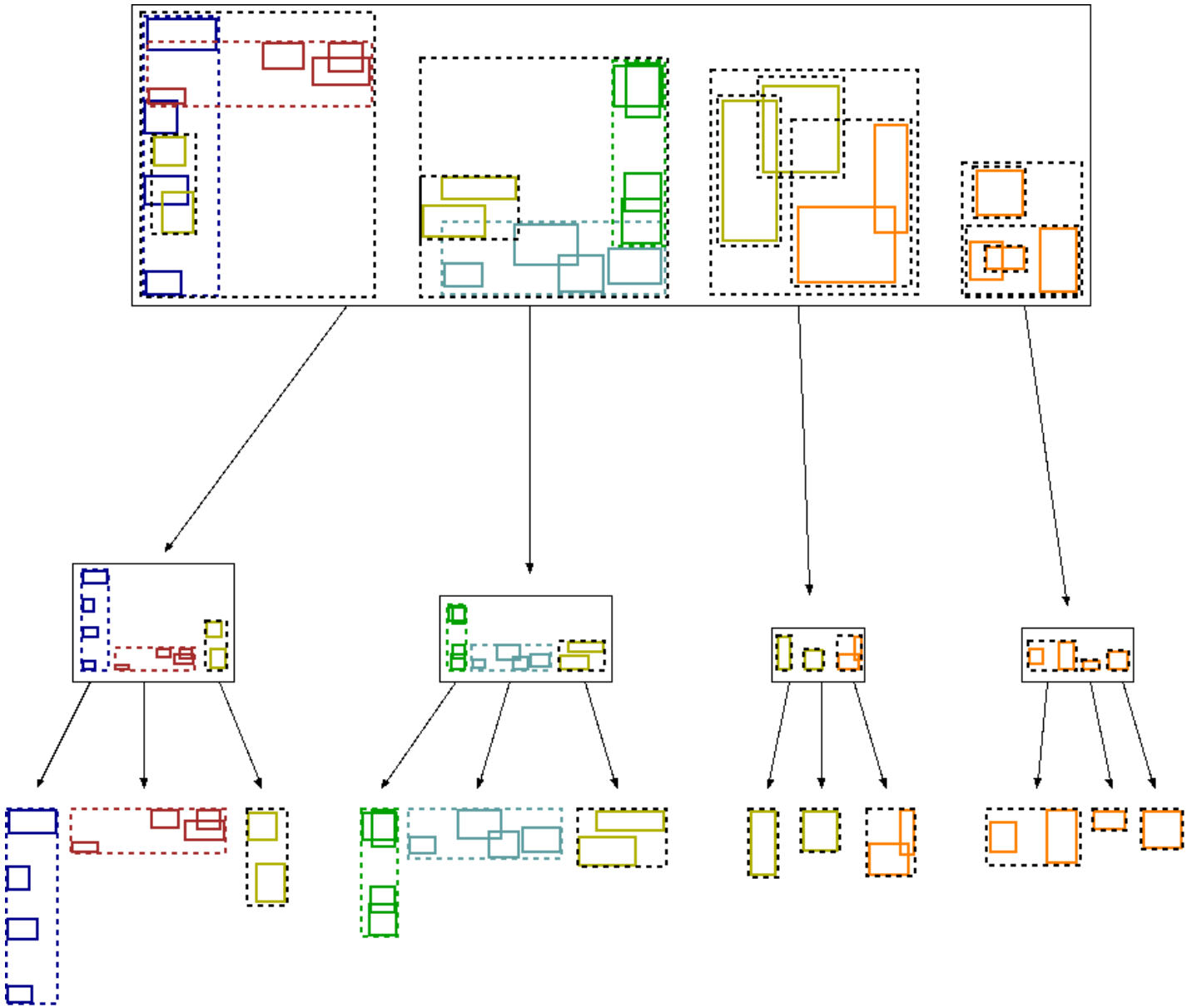
PR-tree

- Compute a pseudo PR-tree from the bounding boxes (on the previous slide) and get the leaves again: (internal data boxes included in the picture for improved readability)

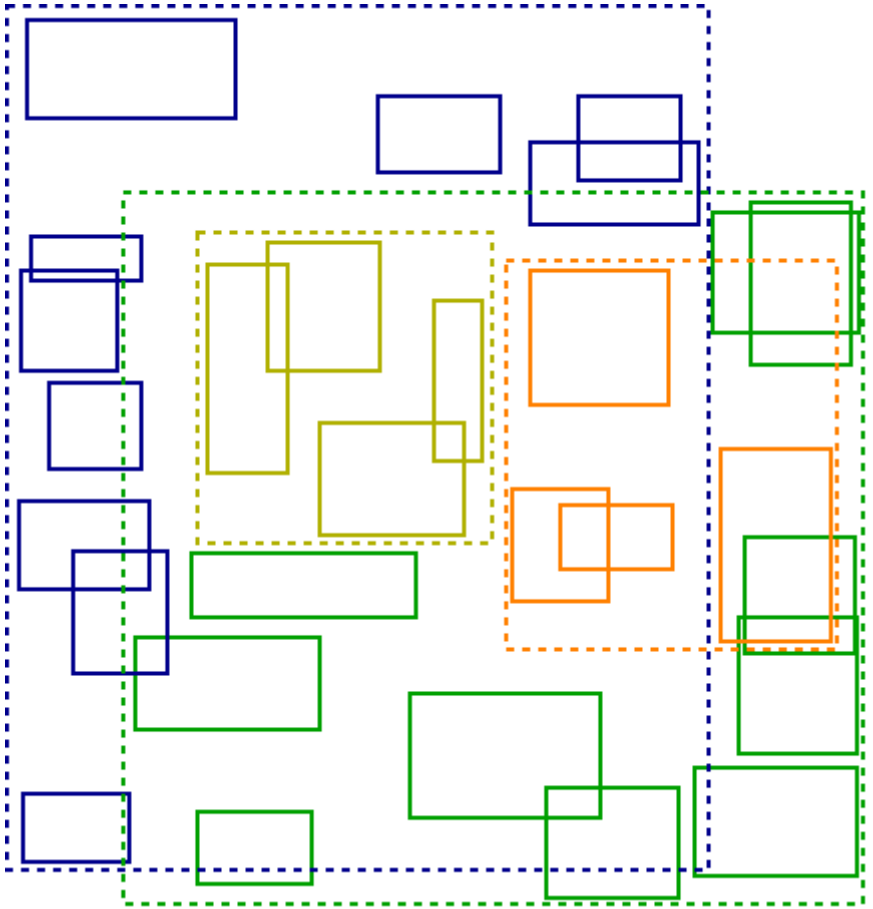


- As there are only four leaves, we can stop by simply putting them on level 1.

Finally the PR-tree:



The First Level in Detail:



- Huge overlap!
- (Notice color assignments in this picture are different from the previous pictures.)

Query Performance

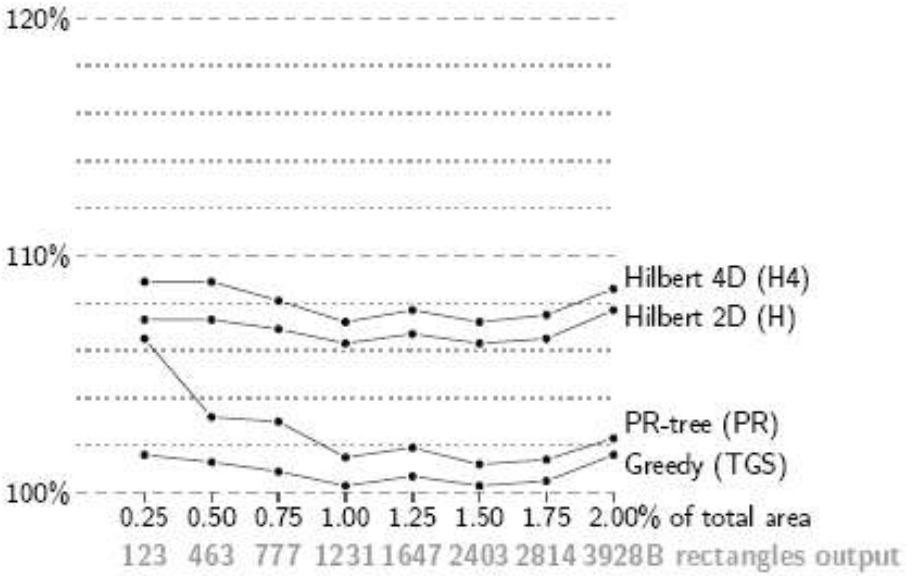


Figure 12: Query performance for queries with squares of varying size on the Western TIGER data. The performance is given as the number of blocks read divided by the output size T/B.

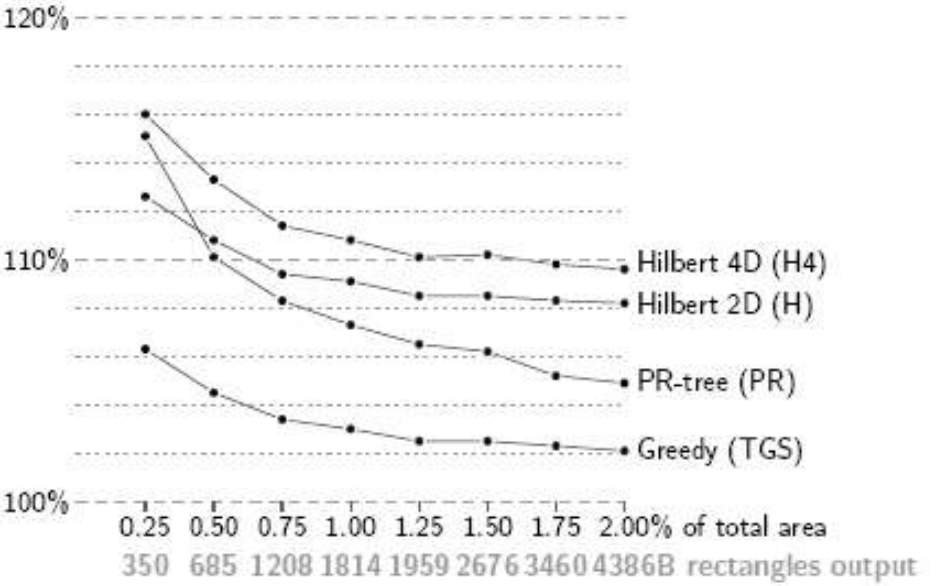


Figure 13: Query performance for queries with squares of varying size on the Eastern TIGER data. The performance is given as the number of blocks read divided by the output size T/B.

Query Performance

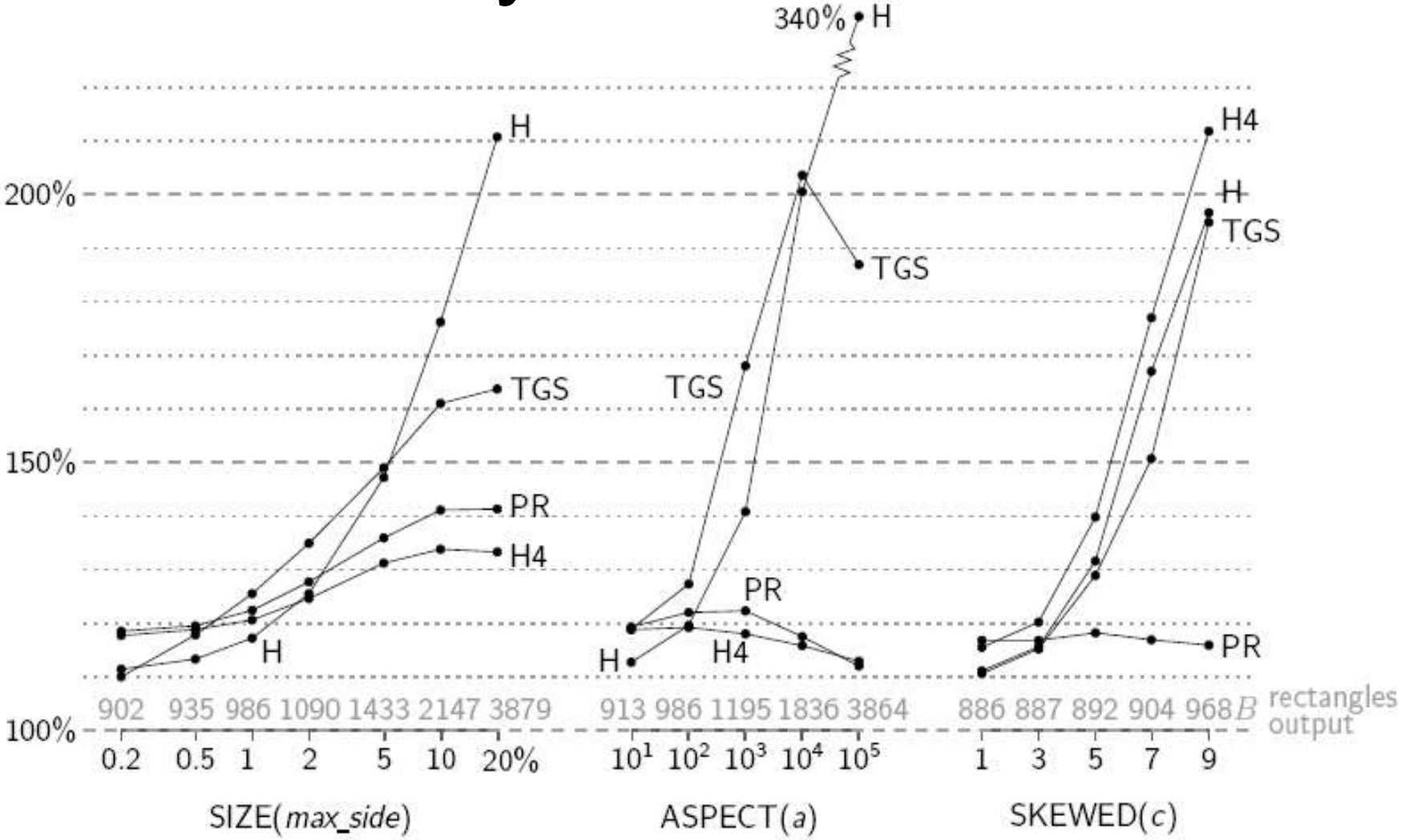


Figure 15: Query performance for queries with squares of area 0.01 on synthetic data sets. The performance is given as the number of blocks read divided by the output size T/B.

Build Performance

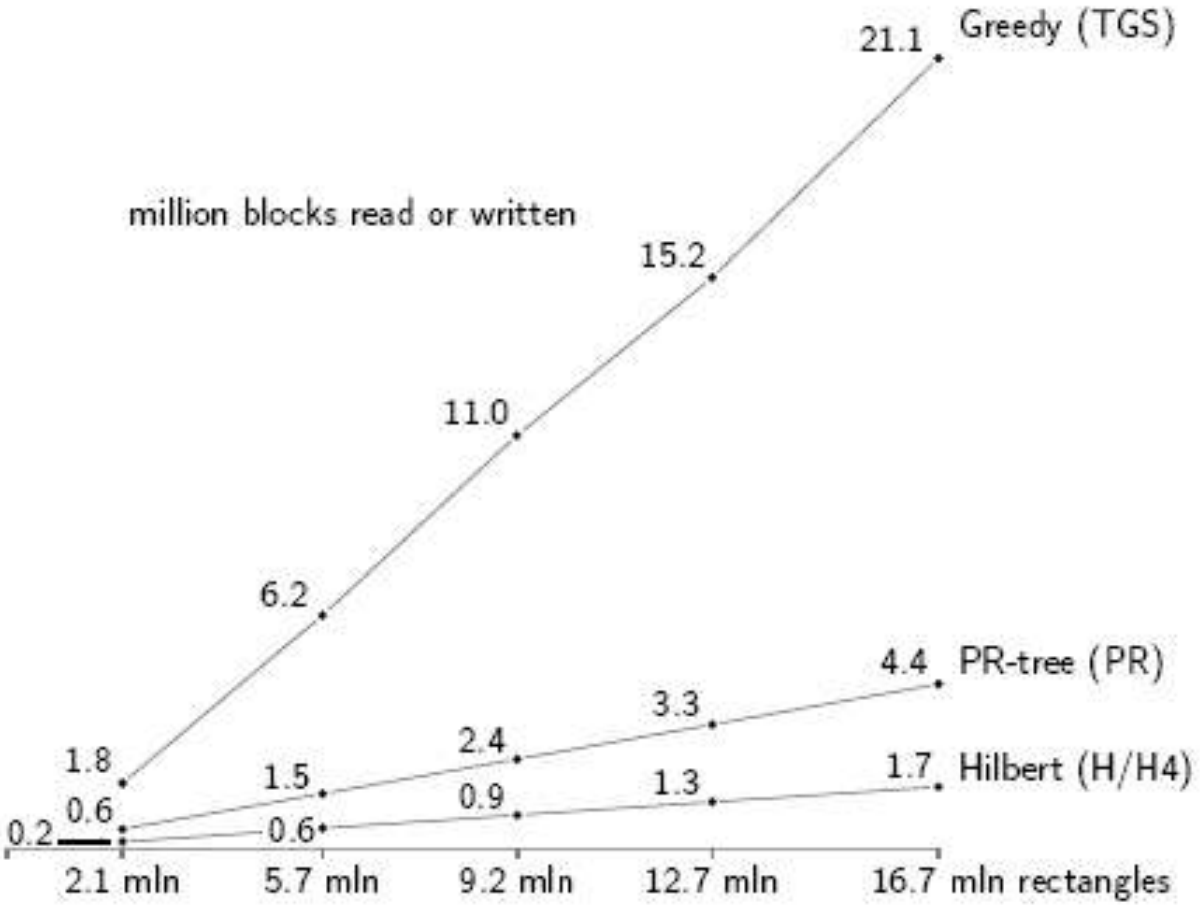


Figure 10: Bulk-loading performances on Eastern datasets (I/Os)