The BUB-Tree (bounding UB-Tree) dealing with dead space

Dipl. Inform. Robert Fenk

Knowledge Bases Research Group (Prof. Bayer), Bavarian Research Center for Knowledge-Based Systems (FORWISS) Munich, Germany
Motivation

Objective: Multidimensional indexing

- real data is always skewed
  - data warehousing
  - spatial data

⇒ there is a lot of dead space

- UB-Tree partitions the whole universe

⇒ good performance in general, but queries on dead space suffer

- has the R-Tree a “better” partitioning?
● use Z-intervals instead of Z-separators
● store \((z_{\text{start}},z_{\text{end}})\) in a B-Tree
\[\Rightarrow\] higher index levels reflect Z-intervals of lower ones
UB-Tree

Random Insert

dead space covered by index

dead space no covered by index

Bulk Load

dead space covered by index

dead space no covered by index

BUB-Tree

R*-Tree

dead space covered by index

dead space no covered by index
**Insertion and Split**

- **find nearest Z-Region w.r.to Z-order and insert there**
  - cost: \textit{height} page reads, 1 page write, binary search on pages

- **fix region boundaries if inserting at start/end**
  - worst case cost: \textit{height} page reads & writes

- **split between tuples with biggest Z-distance**
  - the goal: minimize the space covered by the index!!!
    - honor minimum fill rate
    - split only when Z-gap volume exceeds a given % of Z-region volume
    - split only when Z-gap volume exceeds a given volume
Range Query

- we need two algorithms developed for UB-Trees
  - NextJumpIn(z): calculates next intersection point greater than z of the Z-curve with the query box
  - NextJumpOut(z): calculates the point greater than z (where Z is within the query box) on the Z-curve where it leaves the query box again
Summary

- “twice” the space requirements for index part
  - reduced index fanout,
    + but high potential for compression due to prefixes
  + „only“ populated parts of the universe are indexed
  + basic UB-Tree algorithms can be reused
  + still a disjoint space partitioning
  + logarithmic cost for basic operations, i.e. it is a dynamic index structure which is not true for the R*-Tree!
  
  + query performance equal or better to R*-Tree
  + also with dual space approach for GIS data