15-826: Multimedia Databases and Data Mining

Lecture #6: Spatial Access Methods
Part III: R-trees

C. Faloutsos

Must-read material

- MM-Textbook, Chapter 5.2
- Ramakrishnan+Gehrke, Chapter 28.6

R-trees – impact:

- Popular method; like multi-d B-trees
- Guaranteed utilization; fast search (low dim’s)
- Used in practice:
  - Oracle spatial (R-tree default; z-order, too)
    docs.oracle.com/html/A88805_01/ado_intr.htm
  - IBM-DB2 spatial extender
  - Postgres: `create index ... using { rtree | gist }`
  - Sqlite3: [www.sqlite.org/rtree.html](http://www.sqlite.org/rtree.html)
Outline

Goal: ‘Find similar / interesting things’
• Intro to DB
• Indexing - similarity search
• Data Mining

Indexing - Detailed outline
• primary key indexing
• secondary key / multi-key indexing
• spatial access methods
  – problem dfn
  – z-ordering
  – R-trees
  – ...
• text
• ...

Indexing - more detailed outline
• R-trees
  – main idea; file structure
  – algorithms: insertion/split
  – deletion
  – search: range, nn, spatial joins
  – performance analysis
  – variations (packed; hilbert;...)


Reminder: problem

- Given a collection of geometric objects (points, lines, polygons, ...)
- organize them on disk, to answer spatial queries (range, nn, etc)

R-trees

- z-ordering: cuts regions to pieces -> dup. elim.
- how could we avoid that?
- Idea: try to extend/merge B-trees and k-d trees

(first attempt: k-d-B-trees)

- [Robinson, 81]: if $f$ is the fanout, split point-set in $f$ parts; and so on, recursively
(first attempt: k-d-B-trees)

- But: insertions/deletions are tricky (splits may propagate downwards and upwards)
- no guarantee on space utilization

R-trees

- [Guttman 84] Main idea: allow parents to overlap!

Antonin Guttman
[http://www.baymoon.com/~tg2/]

- \[\Rightarrow\] guaranteed 50% utilization
- \[\Rightarrow\] easier insertion/split algorithms.
- (only deal with Minimum Bounding Rectangles - MBRs)
R-trees

• eg., w/ fanout 4: group nearby rectangles to parent MBRs; each group -> disk page

A  C  D
B  E  F
G  H  J

P1  P2  P3  P4

A  C
B
D

F

G

H

I

J

P1  P2  P3  P4

A  B  C
D  E
F  G

H

I

J

A  B  C
D  E
F  G

H

I

J
R-trees - format of nodes

- {\(\text{MBR}; \text{obj-ptr}\)} for leaf nodes

```
x-low; x-high
y-low; y-high ...
 \text{obj ptr} ...
```

R-trees - format of nodes

- {\(\text{MBR}; \text{node-ptr}\)} for non-leaf nodes

```
x-low; x-high
y-low; y-high ...
 \text{node ptr} ...
```

R-trees - range search?
R-trees - range search

Observations:
- every parent node completely covers its 'children'
- a child MBR may be covered by more than one parent - it is stored under ONLY ONE of them. (ie., no need for dup. elim.)

R-trees - range search

Observations - cont'd
- a point query may follow multiple branches.
- everything works for any dimensionality
Indexing - more detailed outline

- R-trees
  - main idea; file structure
  - algorithms: insertion/split
  - deletion
  - search: range, nn, spatial joins
  - performance analysis
  - variations (packed; hilbert;...)

R-trees - insertion

- eg., rectangle 'X'

R-trees - insertion

- eg., rectangle 'X'
R-trees - insertion

• eg., rectangle ‘Y’

R-trees - insertion

• eg., rectangle ‘Y’: extend suitable parent.

R-trees - insertion

• eg., rectangle ‘Y’: extend suitable parent.
• Q: how to measure ‘suitability’?
R-trees - insertion

• eg., rectangle ‘Y’: extend suitable parent.
• Q: how to measure ‘suitability’?
• A: by increase in area (volume) (more details: later, under ‘performance analysis’)
• Q: what if there is no room? how to split?

R-trees - insertion

• eg., rectangle ‘W’ - focus on ‘P1’ - how to split?
R-trees - insertion

- eg., rectangle ‘W’ - focus on ‘P1’ - how to split?
- (A1: plane sweep, until 50% of rectangles)
- A2: ‘linear’ split
- A3: quadratic split
- A4: exponential split

R-trees - insertion & split

- pick two rectangles as ‘seeds’;
- assign each rectangle ‘R’ to the ‘closest’ ‘seed’
- Q: how to measure ‘closeness’?
R-trees - insertion & split

- pick two rectangles as 'seeds';
- assign each rectangle 'R' to the 'closest' 'seed'
- Q: how to measure 'closeness'?
- A: by increase of area (volume)
R-trees - insertion & split

- pick two rectangles as ‘seeds’;
- assign each rectangle ‘R’ to the ‘closest’ ‘seed’
- smart idea: pre-sort rectangles according to delta of closeness (ie., schedule easiest choices first!)

R-trees - insertion - pseudocode

- decide which parent to put new rectangle into (‘closest’ parent)
- if overflow, split to two, using (say,) the quadratic split algorithm
  – propagate the split upwards, if necessary
- update the MBRs of the affected parents.

R-trees - insertion - observations

- many more split algorithms exist (next!)
Indexing - more detailed outline

- R-trees
  - main idea; file structure
  - algorithms: insertion/split
  - deletion
  - search: range, nn, spatial joins
  - performance analysis
  - variations (packed; hilbert;...)

R-trees - deletion

- delete rectangle
- if underflow
  - ??

R-trees - deletion

- delete rectangle
- if underflow
  - temporarily delete all siblings (!);
  - delete the parent node and
  - re-insert them
R-trees - deletion

- variations: later (e.g., Hilbert R-trees w/ 2-to-1 merge)

Indexing - more detailed outline

- R-trees
  - main idea; file structure
  - algorithms: insertion/split
  - deletion
  - search: range, nn, spatial joins
  - performance analysis
  - variations (packed; hilbert;...)

R-trees - range search

pseudocode:
check the root
for each branch,
if its MBR intersects the query rectangle
  apply range-search (or print out, if this
  is a leaf)
R-trees - nn search

• Q: How? (find near neighbor; refine...)

R-trees - nn search

• A1: depth-first search; then, range query
R-trees - nn search

- A1: depth-first search; then, range query

R-trees - nn search

- A1: depth-first search; then, range query

R-trees - nn search

- A2: [Roussopoulos+, sigmod95]:
  - priority queue, with promising MBRs, and their best and worst-case distance
- main idea:
consider only P2 and P4, for illustration

worst of P2

best of P4

worst of P2

best of P4

=> P4 is useless for 1-nn
R-trees - nn search

• what is really the worst of, say, P2?

• A: the smallest of the two red segments!

R-trees - nn search

• variations: [Hjaltason & Samet] incremental nn:
  – build a priority queue
  – scan enough of the tree, to make sure you have the k nn
  – to find the (k+1)-th, check the queue, and scan some more of the tree

• ‘optimal’ (but, may need too much memory)
Indexing - more detailed outline

- R-trees
  - main idea; file structure
  - algorithms: insertion/split
  - deletion
  - search: range, nn, spatial joins
  - performance analysis
  - variations (packed; hilbert;...)

R-trees - spatial joins

Spatial joins: find (quickly) all counties intersecting lakes
R-trees - spatial joins

Spatial joins: find (quickly) all counties intersecting lakes

Assume that they are both organized in R-trees:
R-trees - spatial joins

for each parent P1 of tree T1
for each parent P2 of tree T2
    if their MBRs intersect,
        process them recursively (i.e., check their children)

R-trees - spatial joins

Improvements - variations:
- [Seeger+, sigmod 92]: do some pre-filtering; do plane-sweeping to avoid \( N1 \times N2 \) tests for intersection
- [Lo & Ravishankar, sigmod 94]: ‘seeded’ R-trees (FYI, many more papers on spatial joins, without R-trees: [Koudas+ Sevcik], e.t.c.)

Indexing - more detailed outline

• R-trees
  – main idea; file structure
  – algorithms: insertion/split
  – deletion
  – search: range, nn, spatial joins
  – performance analysis
  – variations (packed; hilbert;...)

CMU SCS

Copyright: C. Faloutsos (2014)
R-trees - performance analysis

• How many disk (=node) accesses we’ll need for
  – range
  – nn
  – spatial joins
• why does it matter?

• A: because we can design split etc algorithms accordingly; also, do query-optimization
R-trees - performance analysis

• motivating question: on, e.g., split, should we try to minimize the area (volume)? the perimeter? the overlap? or a weighted combination? why?

R-trees - performance analysis

• How many disk accesses for range queries?
  – query distribution wrt location?
    – “ “ wrt size?

R-trees - performance analysis

• How many disk accesses for range queries?
  – query distribution wrt location? uniform; (biased)
  – “ “ wrt size? uniform
R-trees - performance analysis

• easier case: we know the positions of parent MBRs, eg:

- Diagram showing an example of R-trees with parent MBRs.

R-trees - performance analysis

• How many times will P1 be retrieved (unif. queries)?

- Diagram showing the retrieval frequency of P1.

R-trees - performance analysis

• How many times will P1 be retrieved (unif. POINT queries)?

- Diagram illustrating the number of retrievals for P1 in POINT queries.
R-trees - performance analysis

• How many times will P1 be retrieved (unif. POINT queries)? A: x1*x2

R-trees - performance analysis

• How many times will P1 be retrieved (unif. queries of size q1xq2)?

R-trees - performance analysis

• How many times will P1 be retrieved (unif. queries of size q1xq2)?
R-trees - performance analysis

- How many times will P1 be retrieved (unif. queries of size $q_1 \times q_2$)?

$A: (x_1 + q_1) \times (x_2 + q_2)$
R-trees - performance analysis

- Thus, given a tree with \( N \) nodes (\( i=1, \ldots, N \)) we expect
  
  \[
  \#\text{DiskAccesses}(q_1,q_2) = \sum (x_{i,1} + q_1) \times (x_{i,2} + q_2) \\
  = \sum (x_{i,1} \times x_{i,2}) + q_2 \times \sum (x_{i,1}) + q_1 \times \sum (x_{i,2}) + q_1 \times q_2 \times N
  \]

R-trees - performance analysis

- Thus, given a tree with \( N \) nodes (\( i=1, \ldots, N \)) we expect
  
  \[
  \#\text{DiskAccesses}(q_1,q_2) = \sum (x_{i,1} + q_1) \times (x_{i,2} + q_2) \\
  = \sum (x_{i,1} \times x_{i,2}) + q_2 \times \sum (x_{i,1}) + q_1 \times \sum (x_{i,2}) + q_1 \times q_2 \times N
  \]

Observations:

- for point queries: only volume matters
- for horizontal-line queries: \((q_2=0)\): vertical length matters
- for large queries \((q_1, q_2 >> 0)\): the count \( N \) matters
R-trees - performance analysis

Observations (cont’ed)
• overlap: does not seem to matter
• formula: easily extendible to \( n \) dimensions
• (for even more details: [Pagel +, PODS93], [Kamel+, CIKM93])

Berndt-Uwe Pagel

R-trees - performance analysis

Conclusions:
• splits should try to minimize area and perimeter
• i.e., we want few, small, square-like parent MBRs
• rule of thumb: shoot for queries with \( q_1 = q_2 = 0.1 \) (or \( =0.5 \) or so).

R-trees - performance analysis

• How many disk (=node) accesses we’ll need for
  – range
  – nn
  – spatial joins
R-trees - performance analysis

Range queries - how many disk accesses, if we just now that we have
- \( N \) points in \( n \)-d space?
A: ?

R-trees - performance analysis

Range queries - how many disk accesses, if we just now that we have
- \( N \) points in \( n \)-d space?
A: can not tell! need to know distribution

R-trees - performance analysis

What are obvious and/or realistic distributions?
R-trees - performance analysis
What are obvious and/or realistic distributions?
A: uniform
A: Gaussian / mixture of Gaussians
A: self-similar / fractal. Fractal dimension ~ intrinsic dimension

R-trees - performance analysis
Formulas for range queries and k-nn queries: use fractal dimension [Kamel+, PODS94], [Korn+ ICDE2000] [Kriegel+, PODS97]

Indexing - more detailed outline
• R-trees
  – main idea; file structure
  – algorithms: insertion/split
  – deletion
  – search: range, nn, spatial joins
  – performance analysis
  – variations (packed; hilbert;...
R-trees - variations

Guttman’s R-trees sparked much follow-up work

- can we do better splits?
  - i.e., defer splits?

R-trees - variations

Guttman’s R-trees sparked much follow-up work

• can we do better splits?
  - i.e., defer splits?

Popular

A: R*-trees [Beckmann+, SIGMOD90]

Norbert Beckmann
Hans Peter Kriegel
Ralf Schneider
Bernhard Seeger
R-trees - variations

A: R*-trees [Beckmann+, SIGMOD90]
• defer splits, by forced-reinsert, i.e.: instead of splitting, temporarily delete some entries, shrink overflowing MBR, and re-insert those entries
• Which ones to re-insert?
• How many?

Q: Other ways to defer splits?
**R-trees - variations**

Q: Other ways to defer splits?
A: Push a few keys to the closest sibling node (closest = ?)

**R-trees - variations**

R*-trees: Also try to minimize area AND perimeter, in their split.
Performance: higher space utilization; faster than plain R-trees. One of the most successful R-tree variants.

**R-trees - variations**

Guttman’s R-trees sparked much follow-up work
- can we do better splits?
- what about static datasets (no ins/del/upd)?
  - Hilbert R-trees
- what about other bounding shapes?
R-trees - variations

• what about static datasets (no ins/del/upd)?
• Q: Best way to pack points?

A1: plane-sweep
- great for queries on ‘x’;
- terrible for ‘y’
R-trees - variations

• what about static datasets (no ins/del/upd)?
• Q: Best way to pack points?
• A1: plane-sweep
great for queries on ‘x’;
terrible for ‘y’
• Q: how to improve?

R-trees - variations

• A: plane-sweep on HILBERT curve!

R-trees - variations

• A: plane-sweep on HILBERT curve!
• (see [Kamel+, VLDB’94]
R-trees - variations

Guttman’s R-trees sparked much follow-up work
- can we do better splits?
- what about static datasets (no ins/del/upd)?
  - Hilbert R-trees - main idea
  - handling regions
  - performance/discussion
- what about other bounding shapes?

R-trees - variations

- What if we have regions, instead of points?
- I.e., how to impose a linear ordering (‘h-value’) on rectangles?

R-trees - variations

- What if we have regions, instead of points?
- I.e., how to impose a linear ordering (‘h-value’) on rectangles?
  - A1: h-value of center
  - A2: h-value of 4-d point (center, x-radius, y-radius)
  - A3: ...
R-trees - variations

• What if we have regions, instead of points?
• I.e., how to impose a linear ordering (‘h-value’) on rectangles?
• **A1: h-value of center**
• **A2: h-value of 4-d point**
  (center, x-radius, y-radius)
• A3: ...

R-trees - variations

• with h-values, we can have deferred splits, 2-to-3 splits (3-to-4, etc)
• experimentally: faster than R*-trees
  (reference: [Kamel Faloutsos vldb 94])

R-trees - variations

Guttman’s R-trees sparked **much** follow-up work
• can we do better splits?
• what about static datasets (no ins/del/upd)?
• what about other bounding shapes?
R-trees - variations

• what about other bounding shapes? (and why?)
• A1: arbitrary-orientation lines (cell-tree, Guenther)
• A2: P-trees (polygon trees) (MB polygon: 0, 90, 45, 135 degree lines)

R-trees - variations

• A3: L-shapes; holes (hB-tree)
• A4: TV-trees [Lin+, VLDB-Journal 1994]
• A5: SR-trees [Katayama+, SIGMOD97] (used in Informedia)

R-trees - conclusions

• Popular method; like multi-d B-trees
• guaranteed utilization; fast search (low dim’s)
• Used in practice:
  – Oracle spatial (R-tree default; z-order, too)
    docs.oracle.com/html/ADF8805_01/adn_intr.htm
  – IBM-DB2 spatial extender
  – Postgres: create index ... using ( rtree | gist )
  – Sqlite3: www.sqlite.org/rtree.html
• R* variation is popular
References

- Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, Bernhard Seeger: The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. ACM SIGMOD 1990: 322-331

References, cont’d

Other resources

• Code, papers, datasets etc:
  www.rtreeportal.org/
• Java applets and more info:
  donar.umiacs.umd.edu/quadtree/points/rtrees.html