Range Queries over a Compact Representation of Minimum Bounding Rectangles

N. R. Brisaboa[1] M. R. Luaces[1] G. Navarro[2] D. Seco[1]

[1]Database Laboratory, University of A Coruña, Spain
[2]Department of Computer Science, University of Chile

November 1, 2010

Outline

1. Motivation
2. SW-Tree
3. Experiments
4. Conclusions and Future Work

Motivation

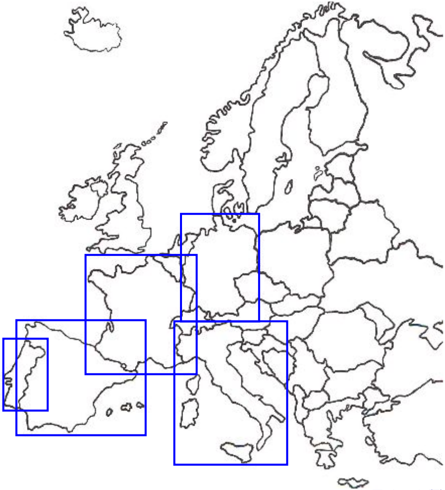
- Spatial indexes are a key component in GIS
  - Large collections of geographic data
  - Geographic operations are very complex
    - Sequential search is not feasible
- Filter/Refine Strategy
  - Minimum Bounding Rectangle (MBR)

Motivation

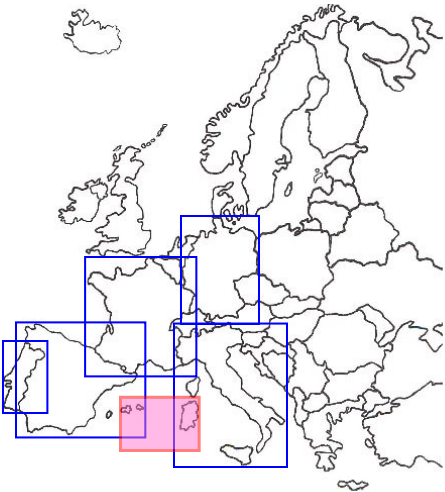
Motivation

Motivation

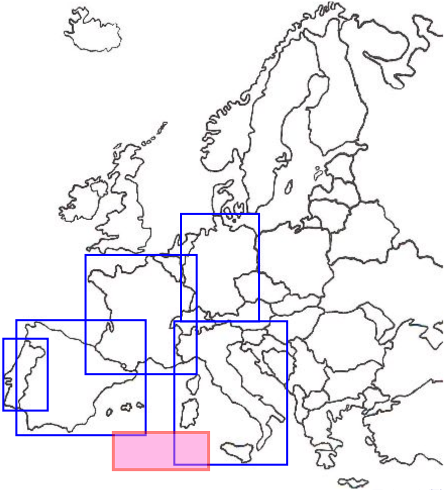
Motivation



Motivation



Motivation



Motivation

- Typical requirements of spatial indexes:
  - Dynamic operations: inserts, deletes, updates, ...
  - Secondary storage management
    - Space consumption is a less important issue
  - ...
- Nowadays, some of these requirements have changed
  - Static data collections are useful in many domains
  - Memory hierarchy evolution
    - Reduction of the main memory cost
    - New levels (flash memory)
- Our goal is a new spatial access method: SW-Tree
  - Static geographic data collections
  - Main memory: compact
  - Efficiency similar to classical indexes

Motivation

- Typical requirements of spatial indexes:
  - Dynamic operations: inserts, deletes, updates, ...
  - Secondary storage management
    - Space consumption is a less important issue
  - ...
- Nowadays, some of these requirements have changed
  - Static data collections are useful in many domains
  - Memory hierarchy evolution
    - Reduction of the main memory cost
    - New levels (flash memory)
- Our goal is a new spatial access method: SW-Tree
  - Static geographic data collections
  - Main memory: compact
  - Efficiency similar to classical indexes

Motivation

Quote

"The time difference between accessing a piece of information in RAM vs reading it from disk is similar to the time difference between picking up a pen from this desk and taking a plane to Spain and picking up a pen from my desk"

Motivation

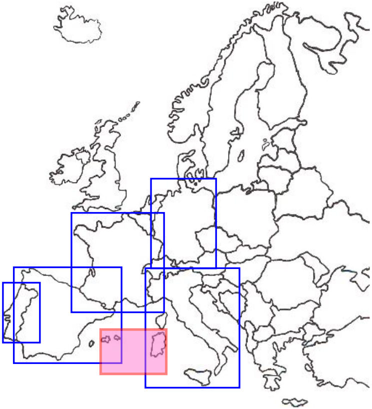
- Typical requirements of spatial indexes:
  - Dynamic operations: inserts, deletes, updates, ...
  - Secondary storage management
    - Space consumption is a less important issue
  - ...
- Nowadays, some of these requirements have changed
  - Static data collections are useful in many domains
  - Memory hierarchy evolution
    - Reduction of the main memory cost
    - New levels (flash memory)
- Our goal is a new spatial access method: SW-Tree
  - Static geographic data collections
  - Main memory: compact
  - Efficiency similar to classical indexes

Motivation

- Typical requirements of spatial indexes:
  - Dynamic operations: inserts, deletes, updates, ...
  - Secondary storage management
    - Space consumption is a less important issue
  - ...
- Nowadays, some of these requirements have changed
  - Static data collections are useful in many domains
  - Memory hierarchy evolution
    - Reduction of the main memory cost
    - New levels (flash memory)
- Our goal is a new spatial access method: SW-Tree
  - Static geographic data collections
  - Main memory: compact
  - Efficiency similar to classical indexes

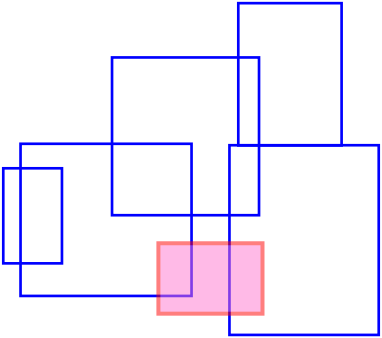
SW-Tree

- Remind the problem...



SW-Tree

- ... and forget the refinement step



Overview

- Orthogonal problem
  - Work with the rank of the coordinates
- Decomposition of a d-dimensional problem into its d dimensions ($d = 2$)
  - Solve d (one-dimensional) subproblems and intersect their results
- Transform the original space
  - A one-dimensional interval can be represented as a 2-dimensional point

Overview

- Orthogonal problem
  - Work with the rank of the coordinates
- Decomposition of a d-dimensional problem into its d dimensions ($d = 2$)
  - Solve d (one-dimensional) subproblems and intersect their results
- Transform the original space
  - A one-dimensional interval can be represented as a 2-dimensional point

Overview

- Orthogonal problem
  - Work with the rank of the coordinates
- Decomposition of a d-dimensional problem into its d dimensions ($d = 2$)
  - Solve d (one-dimensional) subproblems and intersect their results
- Transform the original space
  - A one-dimensional interval can be represented as a 2-dimensional point

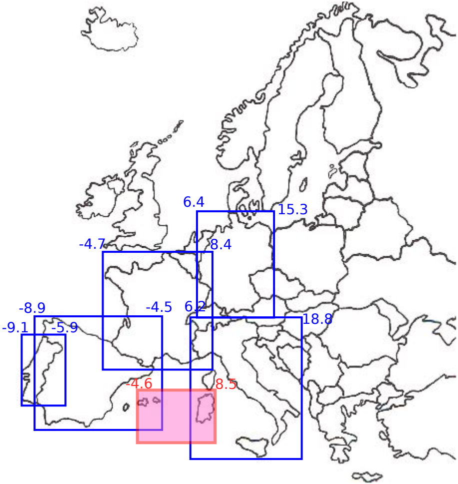
Overview

- Orthogonal problem
  - Work with the rank of the coordinates
- Decomposition of a d-dimensional problem into its d dimensions ($d = 2$)
  - Solve d (one-dimensional) subproblems and intersect their results
- Transform the original space
  - A one-dimensional interval can be represented as a 2-dimensional point

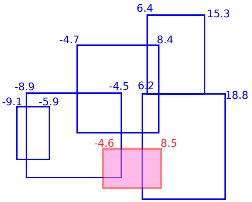
Orthogonal Problem

- Gabow et al. (1984)
- Work with the rank of the coordinates
- Practical solution:
  - Store the real coordinates into sorted arrays
  - Perform binary searches to translate real queries to the rank space

Orthogonal Problem

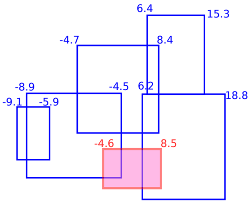


Orthogonal Problem



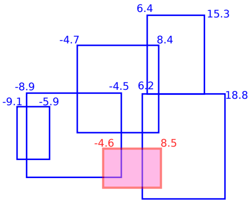
1	2	3	4	5	6	7	8	9	10
-9.1	-8.9	-5.9	-4.7	-4.5	6.2	6.4	8.4	15.3	18.8

Orthogonal Problem



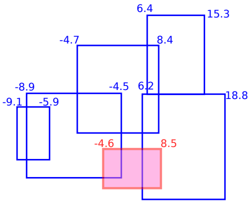
1	2	3	4	5	6	7	8	9	10
-9.1	-8.9	-5.9	-4.7	-4.5	6.2	6.4	8.4	15.3	18.8
				-4.6			8.5		

Orthogonal Problem



- Coordinates encoding:
  - Scaling
  - Differential compression

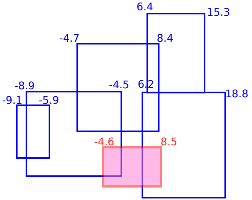
Orthogonal Problem



- Coordinates encoding:
  - Scaling
  - Differential compression

1	2	3	4	5	6	7	8	9	10
-9.1	-8.9	-5.9	-4.7	-4.5	6.2	6.4	8.4	15.3	18.8

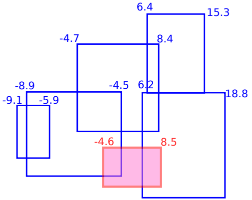
Orthogonal Problem



- Coordinates encoding:
  - Scaling
  - Differential compression

1	2	3	4	5	6	7	8	9	10
-9.1	-8.9	-5.9	-4.7	-4.5	6.2	6.4	8.4	15.3	18.8
-91	-89	-59	-47	-45	62	64	84	153	188

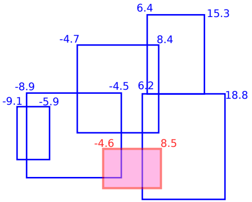
Orthogonal Problem



- Coordinates encoding:
  - Scaling
  - Differential compression

1	2	3	4	5	6	7	8	9	10
-9.1	-8.9	-5.9	-4.7	-4.5	6.2	6.4	8.4	15.3	18.8
-91	-89	-59	-47	-45	62	64	84	153	188
0	2	30	12	2	107	2	20	69	35

Orthogonal Problem

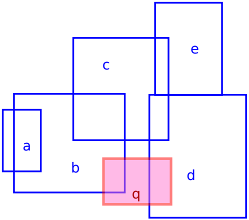


- Coordinates encoding:
  - Scaling
  - Differential compression

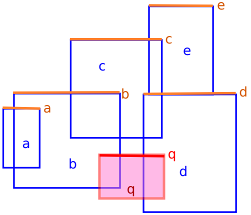
1	2	3	4	5	6	7	8	9	10
-9.1	-8.9	-5.9	-4.7	-4.5	6.2	6.4	8.4	15.3	18.8
-91	-89	-59	-47	-45	62	64	84	153	188
0	2	30	12	2	107	2	20	69	35
$\phi(0)$	$\phi(2)$	$\phi(30)$	$\phi(12)$	$\phi(2)$	$\phi(107)$	$\phi(2)$	$\phi(20)$	$\phi(69)$	$\phi(35)$

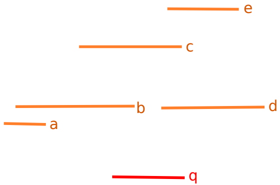
$\phi()$ coding integers function (e.g. γ-codes, δ-codes, Rice, Vbytes)

Decomposition

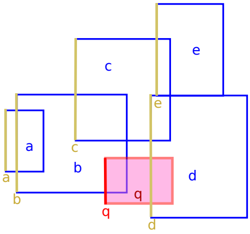


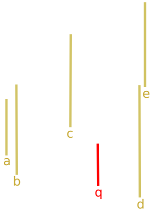
Decomposition



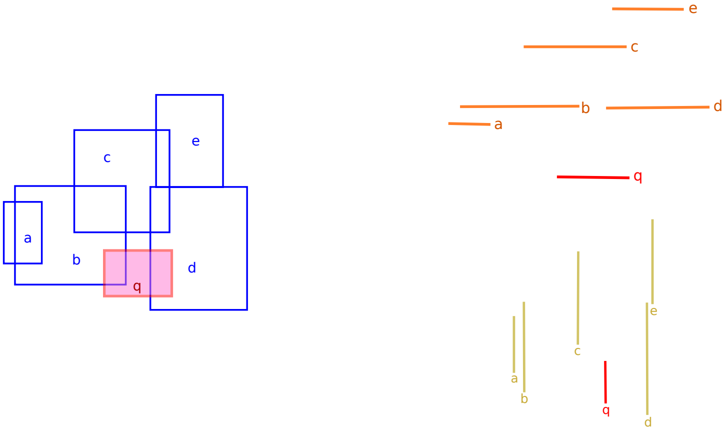


Decomposition

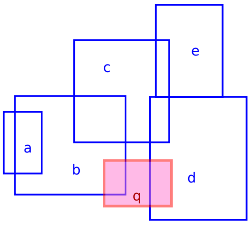


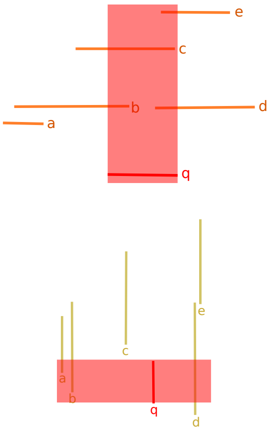


Decomposition



Decomposition





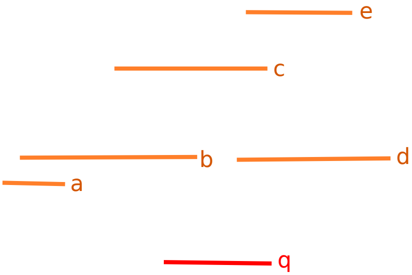
Decomposition

- Decomposition of a d-dimensional problem into its d dimensions ($d = 2$)
- d-dimensional range query decomposition:
  - d one-dimensional interval intersection problems
- Interval Intersection:
  - Interval trees, Segment trees, Priority trees ($\Omega(\log n + m)$)
  - Schmidt'09 ($O(1 + m)$)

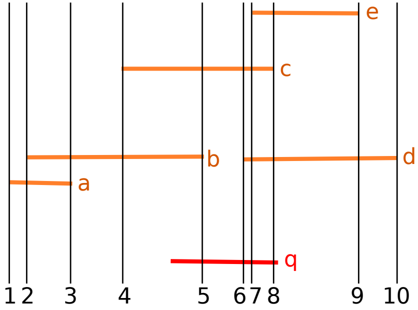
Decomposition

- Decomposition of a d-dimensional problem into its d dimensions ($d = 2$)
- d-dimensional range query decomposition:
  - d one-dimensional interval intersection problems
- Interval Intersection:
  - Interval trees, Segment trees, Priority trees ($\Omega(\log n + m)$)
  - Schmidt'09 ($O(1 + m)$)

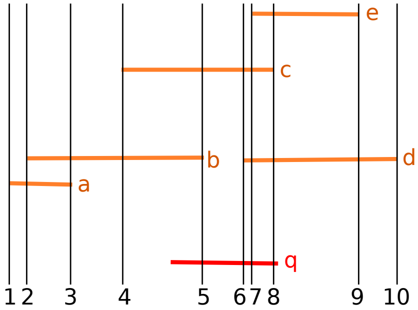
Transformation



Transformation

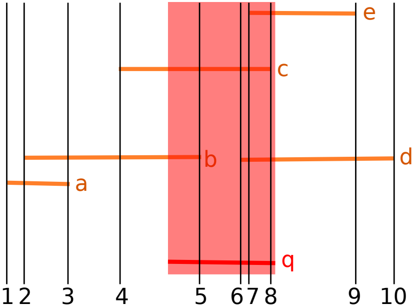


Transformation



l/u	1	2	3	4	5	6	7	8	9	10
1			a							
2					b					
3										
4								c		
5										
6										d
7									e	
8										
9										
10										

Transformation



l/u	1	2	3	4	5	6	7	8	9	10
1			a							
2					b					
3										
4								c		
5										
6										d
7									e	
8										
9										
10										

Transformation

- A one-dimensional interval can be represented as a 2-dimensional point
- Solve an interval intersection query in the original space is equivalent to solve a two-sided range query in the transformed space:
  - $q = [l^q, u^q]$
  - $(l_i, u_i) / l_i \leq u^q \ \wedge \ u_i \geq l^q$
- Two-dimensional range reporting:
  - Wavelet tree $(O(m \log(n/m) + m))$
  - K-d-tree $(O(\sqrt{n} + m))$
  - Alstrup et al. $(O(\log\log(n) + m))$
  - Bose et al. $(O(\frac{m \log(n)}{\log\log(n)}))$

Transformation

- A one-dimensional interval can be represented as a 2-dimensional point
- Solve an interval intersection query in the original space is equivalent to solve a two-sided range query in the transformed space:
  - $q = [l^q, u^q]$
  - $(l_i, u_i) / l_i \leq u^q \;\wedge\; u_i \geq l^q$
- Two-dimensional range reporting:
  - Wavelet tree $(O(m \log(n/m) + m))$
  - K-d-tree $(O(\sqrt{n} + m))$
  - Alstrup et al. $(O(\log \log(n) + m))$
  - Bose et al. $(O(\frac{m \log(n)}{\log \log(n)}))$

Wavelet Tree-based Solution

- Many alternatives:

Data Structure	Worst-case search time
Interval, Segment, and Priority trees	$\Omega(\log n + m)$
Schmidt'09	$O(1 + m)$
K-d-tree	$O(\sqrt{n} + m)$
Alstrup et al.	$O(\log\log(n) + m)$
Bose et al.	$O(\frac{m \log(n)}{\log\log(n)})$
Wavelet tree	$O(m \log(n/m) + m)$

- Which are the virtues of the wavelet tree?
  - Good space/time trade-off (space: $n \log n + o(n \log n)$ bits)
  - No significant implementation overhead
  - Most operations in the rank space (competitive against K-d-tree)

Wavelet Tree-based Solution

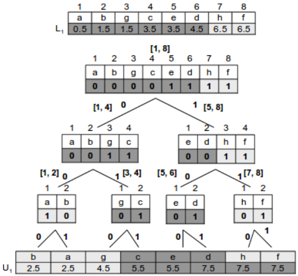
- Many alternatives:

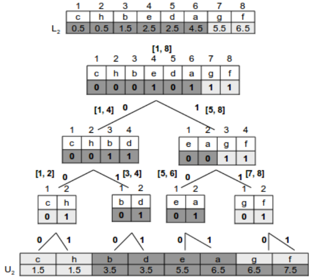
Data Structure	Worst-case search time
Interval, Segment, and Priority trees	$\Omega(\log n + m)$
Schmidt'09	$O(1 + m)$
K-d-tree	$O(\sqrt{n} + m)$
Alstrup et al.	$O(\log\log(n) + m)$
Bose et al.	$O(\frac{m\log(n)}{\log\log(n)})$
Wavelet tree	$O(m\log(n/m) + m)$

- Which are the virtues of the wavelet tree?
  - Good space/time trade-off (space: $n\log n + o(n\log n)$ bits)
  - No significant implementation overhead
  - Most operations in the rank space (competitive against K-d-tree)

Wavelet Tree-based Solution

- SeCoGIS'09: *A New Point Access Method based on Wavelet Trees*
- Permutation in the order of the points in each dimension
- Balanced binary tree
- Constant time operation: $rank_1(B, i)$

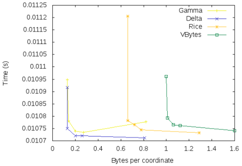




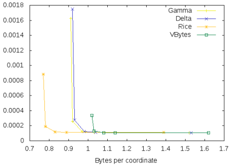
Experimental Environment

- Structures
  - R*-tree, STR R-tree (Spatial Index Library)
  - SW-tree
- Datasets
  - Synthetic (1,000,000 MBRs each)
    - Uniform
    - Gauss (world size $= 1,000 \times 1,000$, $\mu = 500$, $\sigma = 200$)
    - Zipf (world size $= 1,000 \times 1,000$, $\rho = 1$)
  - Real
    - EIEL (569,534 MBRs from buildings in A Coruña)
    - TIGER (2,249,727 MBRs from California roads)
- Experiments in:
  - Intel Pentium 4 3.00GHz with 4GB of RAM
  - GNU/Linux kernel 2.6.27
  - gcc 4.3.2 and -O9 optimizations
  - Time represents CPU user-time

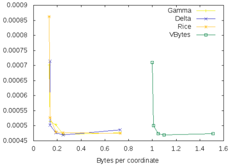
Coordinates Encoding



(a) Zipf

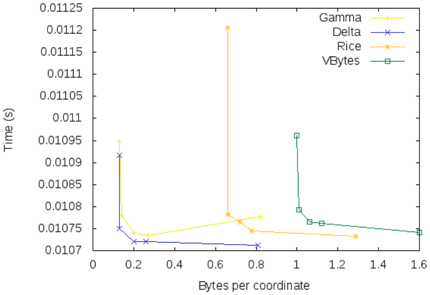


(b) EIEL

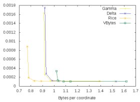


(c) Tiger

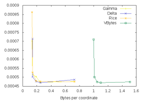
Coordinates Encoding



(a) Zipf

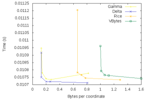


(b) EIEL

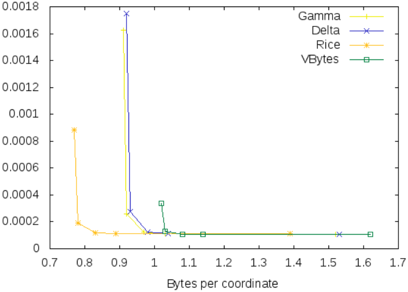


(c) Tiger

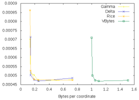
Coordinates Encoding



(a) Zipf

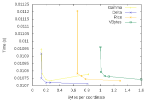


(b) EIEL

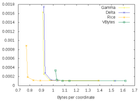


(c) Tiger

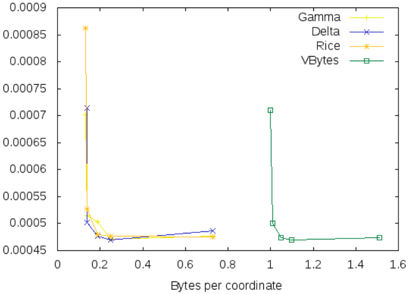
Coordinates Encoding



(a) Zipf

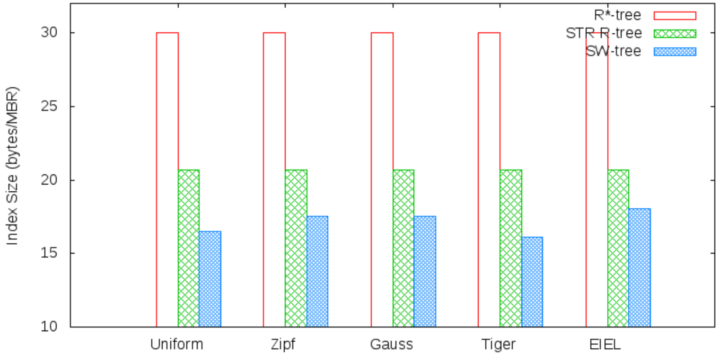


(b) EIEL

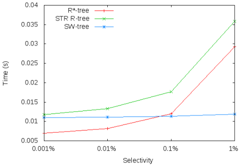


(c) Tiger

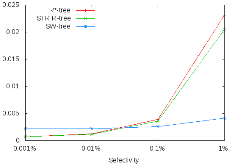
Space Comparison



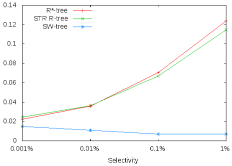
Time Comparison (Synthetic Datasets)



(a) Uniform

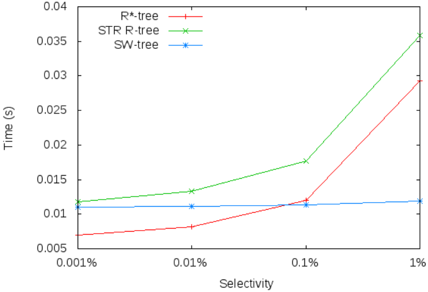


(b) Gauss

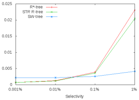


(c) Zipf

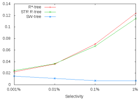
Time Comparison (Synthetic Datasets)



(a) Uniform

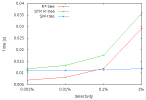


(b) Gauss

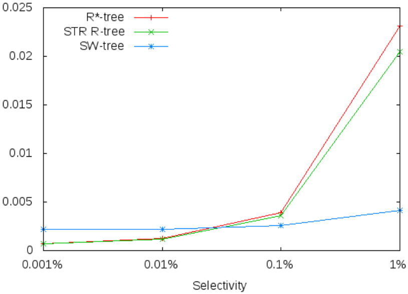


(c) Zipf

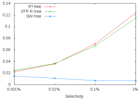
Time Comparison (Synthetic Datasets)



(a) Uniform

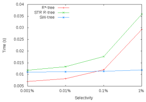


(b) Gauss

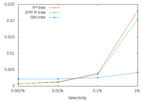


(c) Zipf

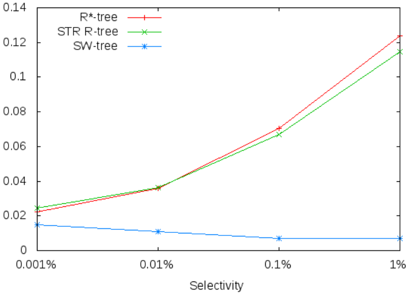
Time Comparison (Synthetic Datasets)



(a) Uniform

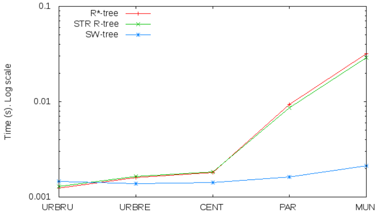


(b) Gauss

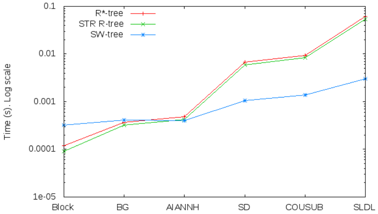


(c) Zipf

Time Comparison (Real Datasets)



(a) EIEL



(b) Tiger

Conclusions and Future Work

Conclusions

- Compact structure to index semi-static collections of MBRs
- Good space/time trade-off

Conclusions and Future Work

Conclusions

- ~~Compact structure to index semi-static collections of MBRs~~
- General technique to index semi-static collections of MBRs
- ~~Good space/time trade-off~~
- Different space/time trade-offs
- Closing the gap with other very active research fields (information retrieval and text compression)

Conclusions and Future Work

Future Work

- Lossy compressed spatial indexes (CR-tree)
- Dynamic bitmaps supporting *rank*
- Other operations: k-nearest neighbors, spatial join

The End

Questions?