University of São Paulo
São Carlos - S.P.

The MM-tree

A Memory-Based Metric Tree Without Overlap Between Nodes

Ives R. V. Pola
Caetano Traina Jr.
Agma J. M. Traina

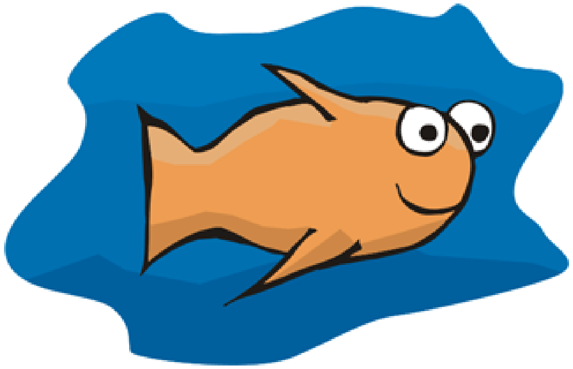
Agma J. M. Traina

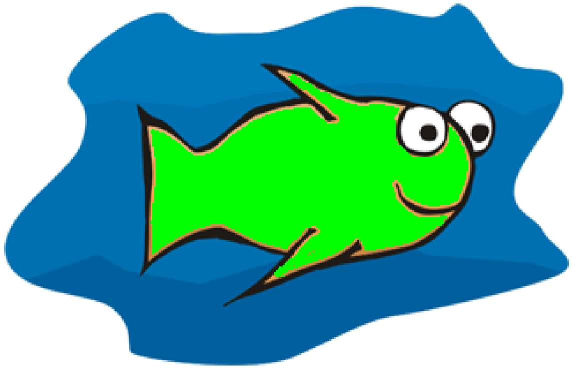
Outline

- Introduction
- Background
- Motivation
- The MM-tree
- Experiments and Results
- Conclusions

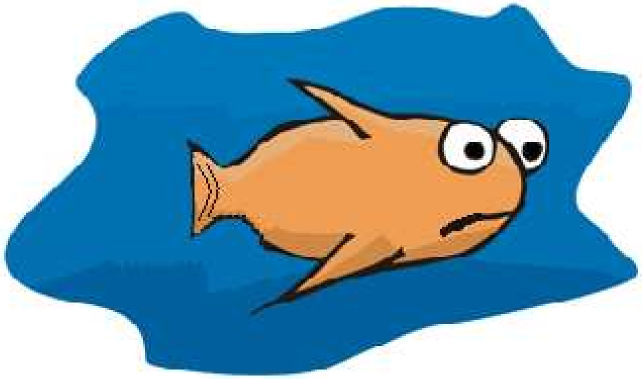
Introduction

Similarity





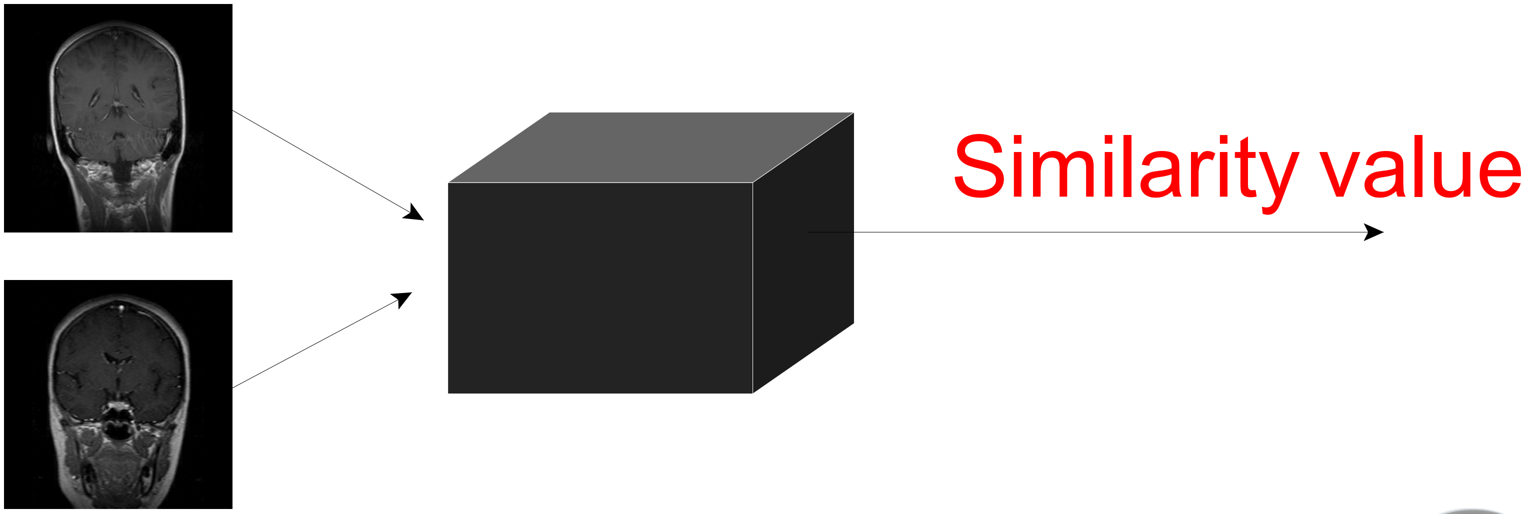
- Are they similar ?
  - Based on what?



Introduction

Similarity measure

- We can define a similarity function
  - Compare pairs of elements
  - Based on the elements attributes.



Similarity value

Background

Metric Space

- A metric space is defined by ***M*** < ***S***, d >

Let s_1, s_2, $s_3 \in$ ***S***, then **d: *S* x *S*** $\rightarrow \mathbb{R}^+$ must hold:

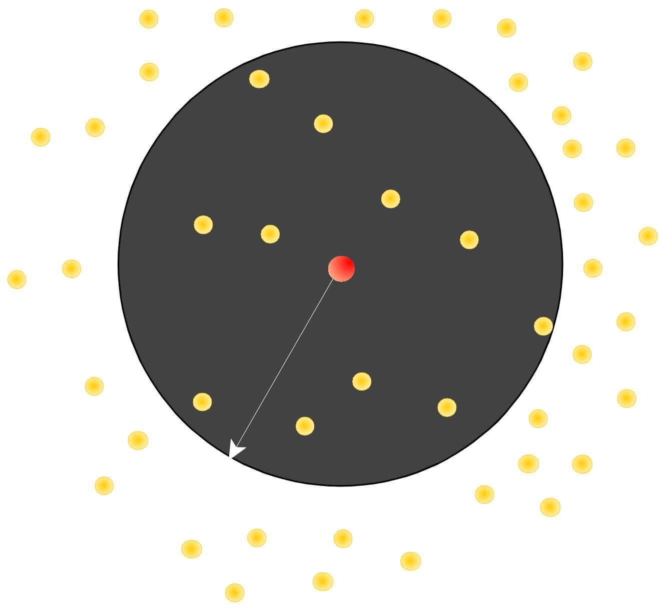
- Identity
  - $d(s_1,s_1) = 0$
- Simetry
  - $d(s_1, s_2) = d(s_2, s_1)$
- Non negativity
  - $0 < d(s_1, s_2) < \infty$, to s1 $\neq$ s2
- Triangular inequality
  - $d(s_1,s_2) \leq d(s_1,s_3) + d(s_3,s_2)$

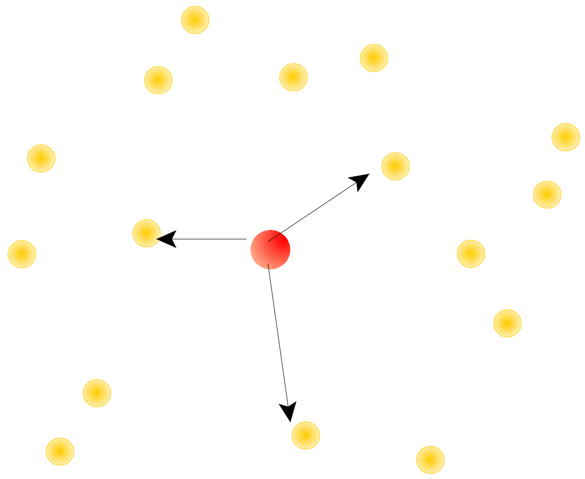
Background

Similarity queries

- Most commom:
  - Range query

K-nearest neighbor query





Background

Metric Access Methods

- Index data in a Metric Space
  - Distance-based trees
- Classification
  - Disk-based trees
    - Slim-tree, M-tree, MVP-tree, OMNI-family, DF-tree, DBM-tree
  - Main memory-based trees
    - GH-tree, VP-tree, GNAT, **MM-tree**
- Pruning of subtrees
  - Exploring the triangular inequality property.

Background

Memory-based vs Disk-based Metric Trees

- Advantages of Memory-based trees
  - The partition of space is flexible
    - Not fixed number of elements per node
  - Do not perform disk I/O
    - Fast to build
    - Fast to answer queries
- Disavantages
  - They are not persistent
  - There must be enough memory for data

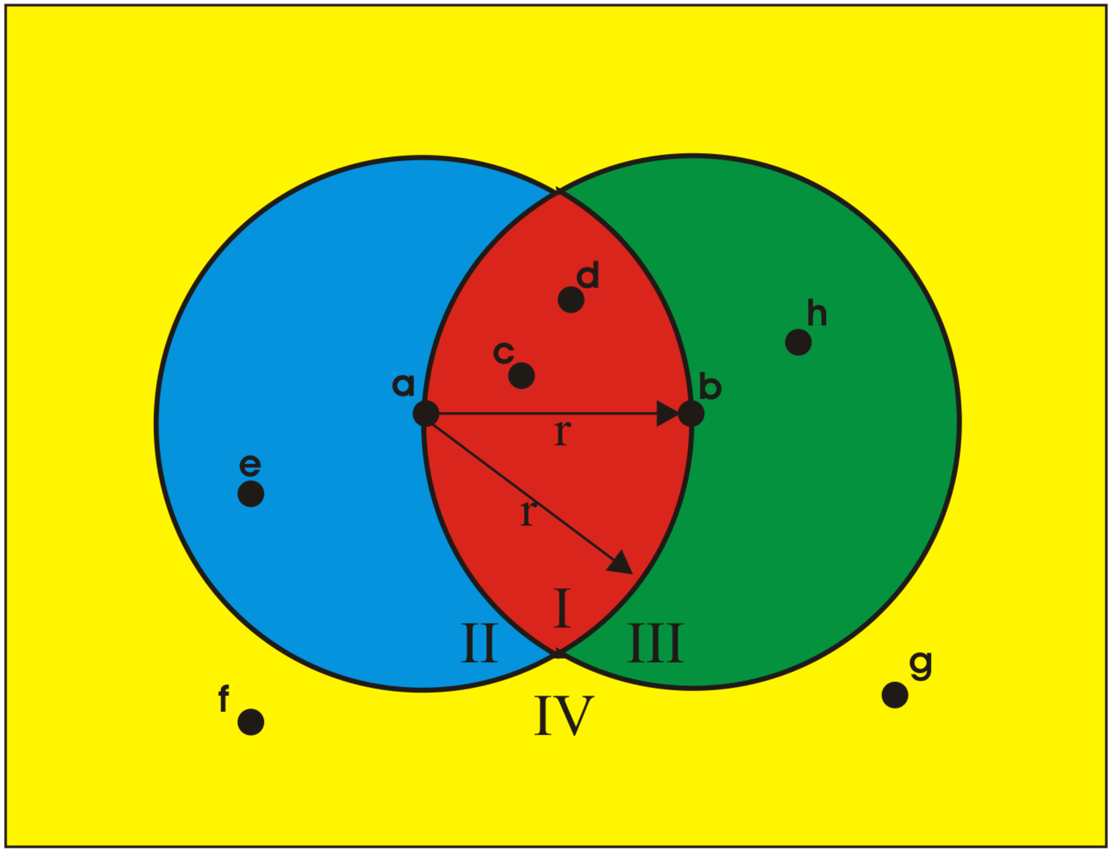
Motivation

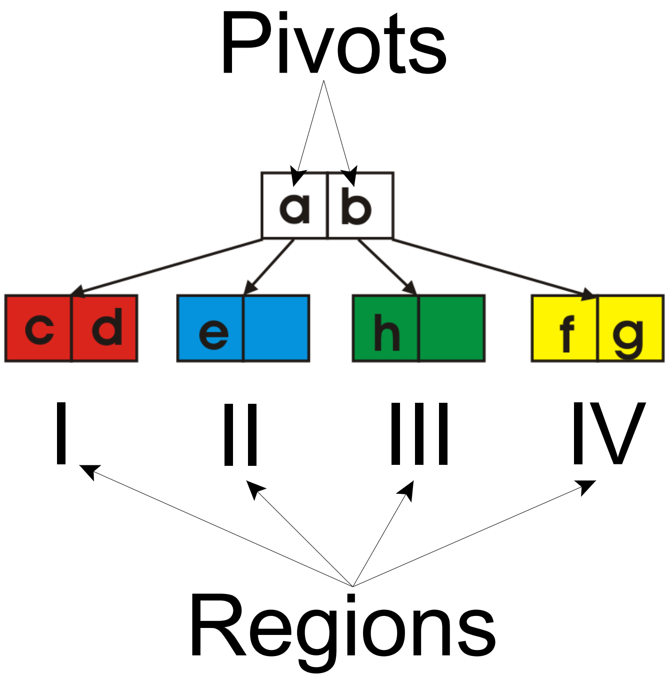
MM-tree

- A height-balanced tree
  - Reduces nodes retrieval on disk-based trees
    - Less disk accesses (they are computational expensive)
- But a main-memory tree
  - Do not perform disk access
  - We can choose to build a tree not fully balanced
    - In order to form disjoint regions

The MM-tree

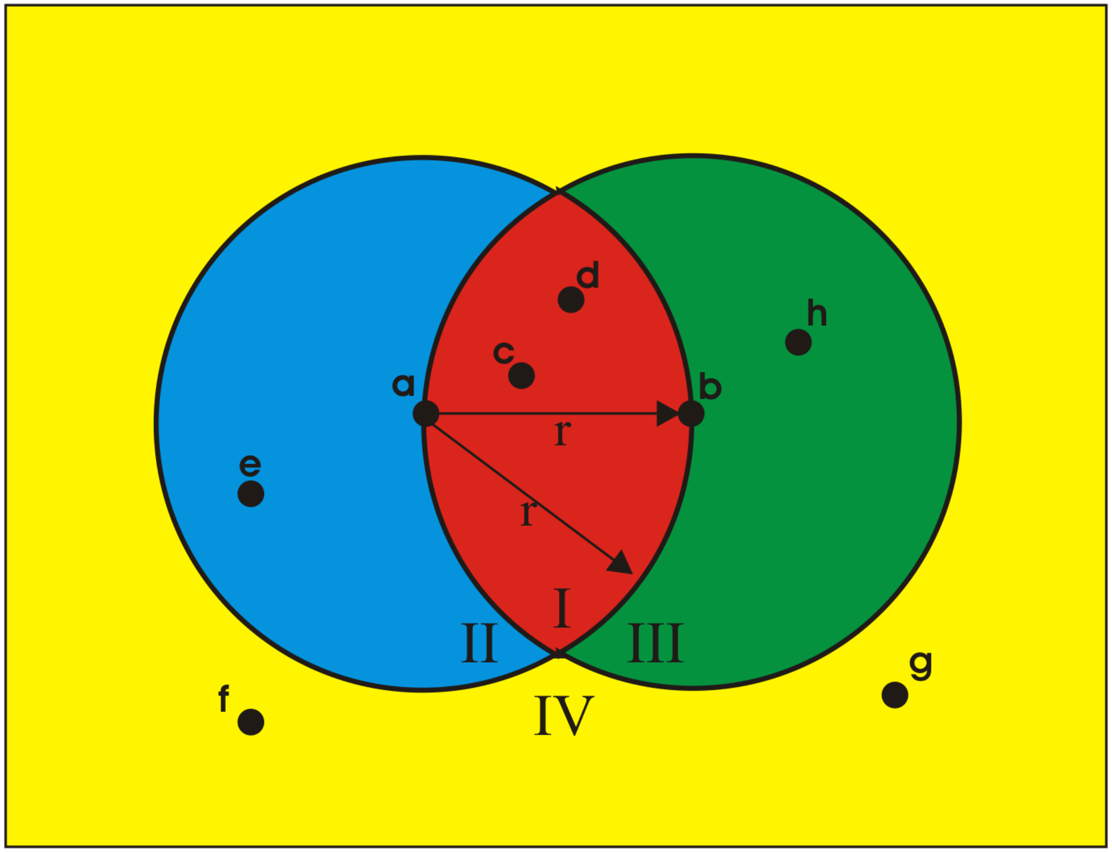
Two levels example

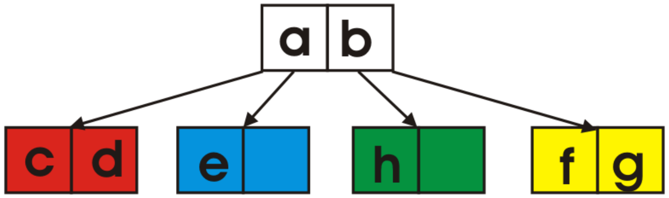




The MM-tree

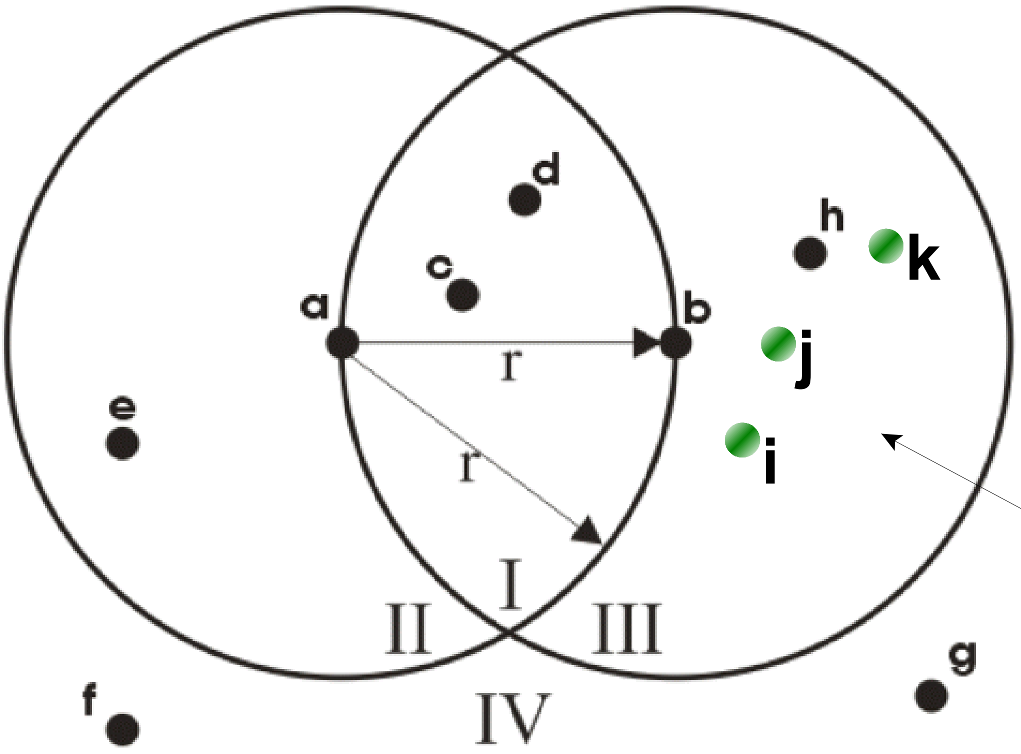
Building the tree

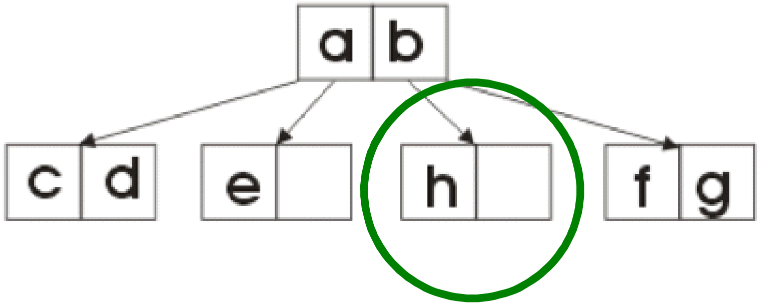




The MM-tree

Building the tree

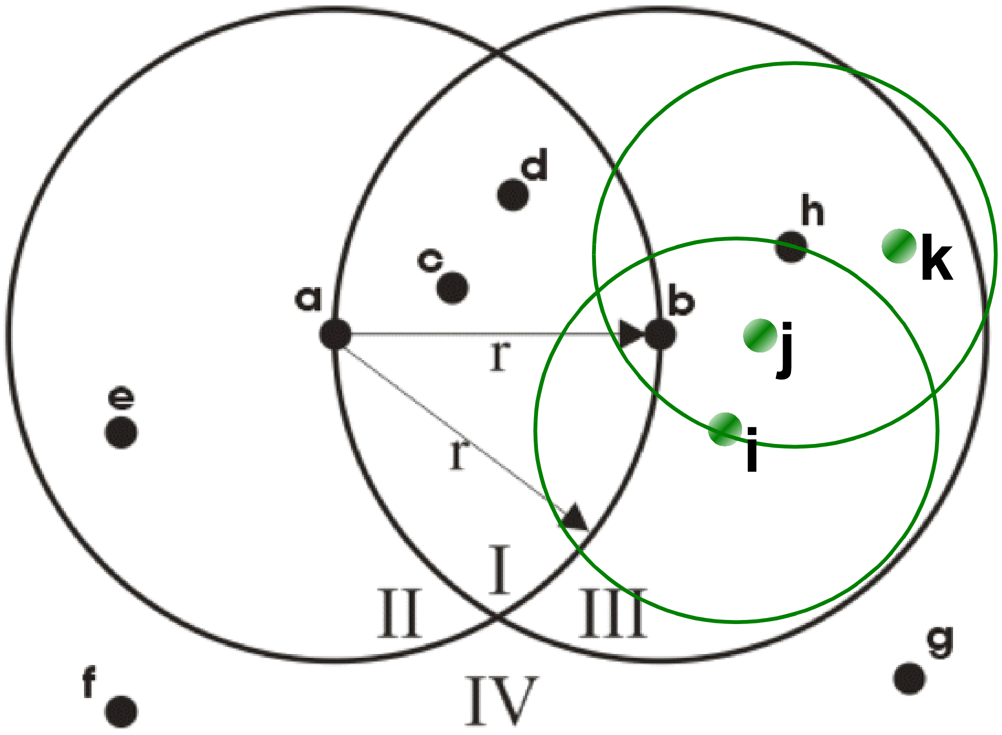


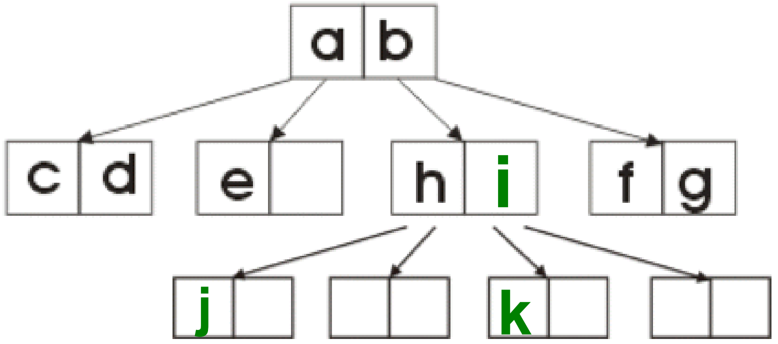


Insert: i, j, k

The MM-tree

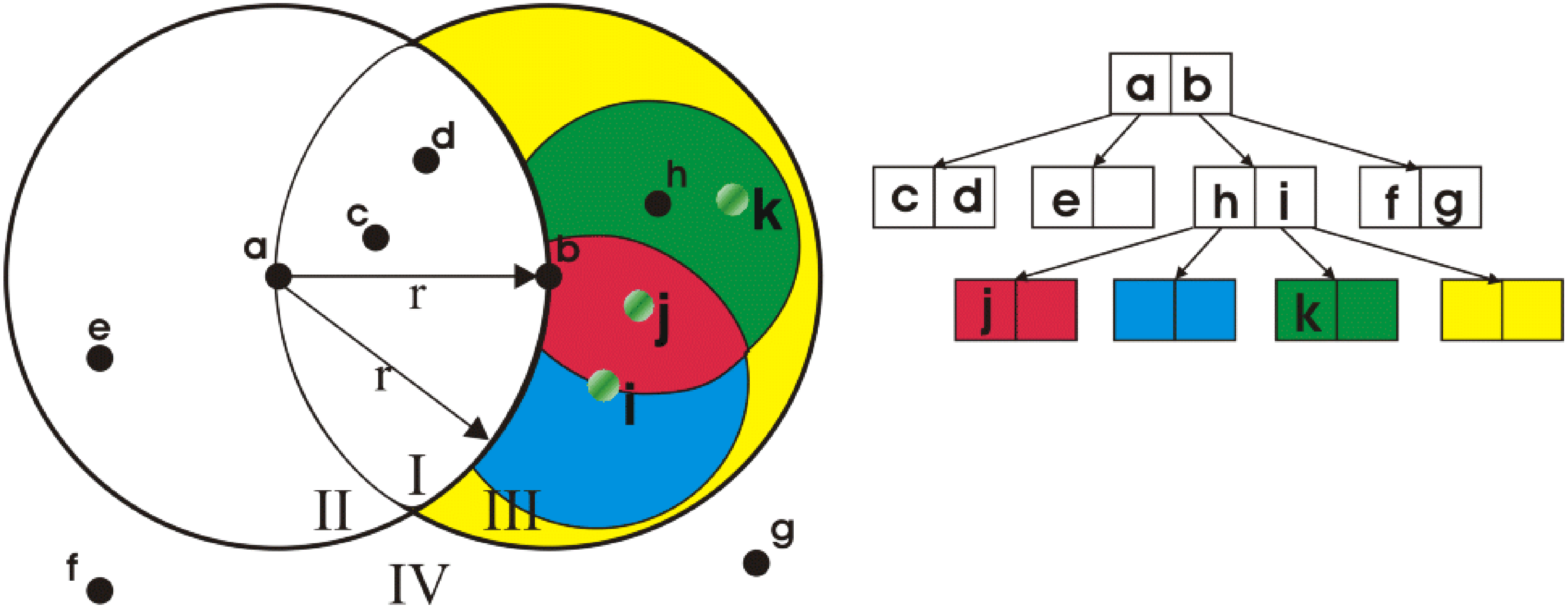
Building the tree





The MM-tree

Building the tree



The MM-tree

Memory metric tree

- It holds 2 elements per node
  - Divides the space into 4 disjoint regions
  - Only 2 distances per node are calculated

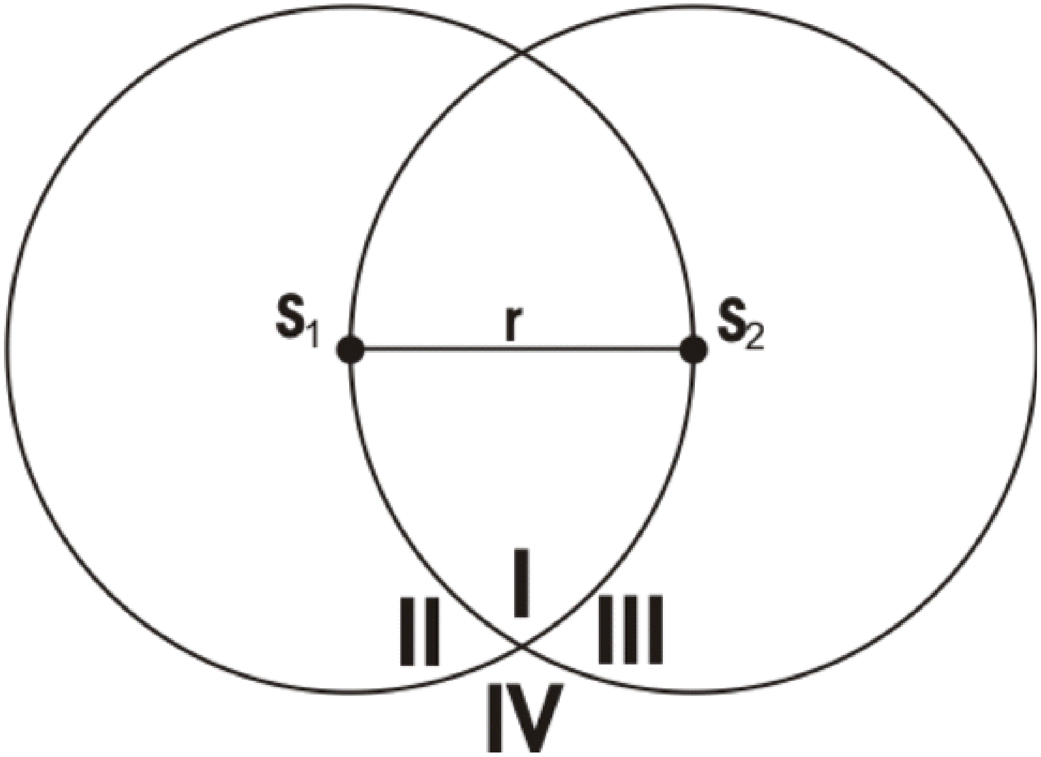
$$node[s_1, s_2, d(s_1, s_2), Ptr_1, Ptr_2, Ptr_3, Ptr_4]$$

The MM-tree

Inserting elements - choosing subtrees

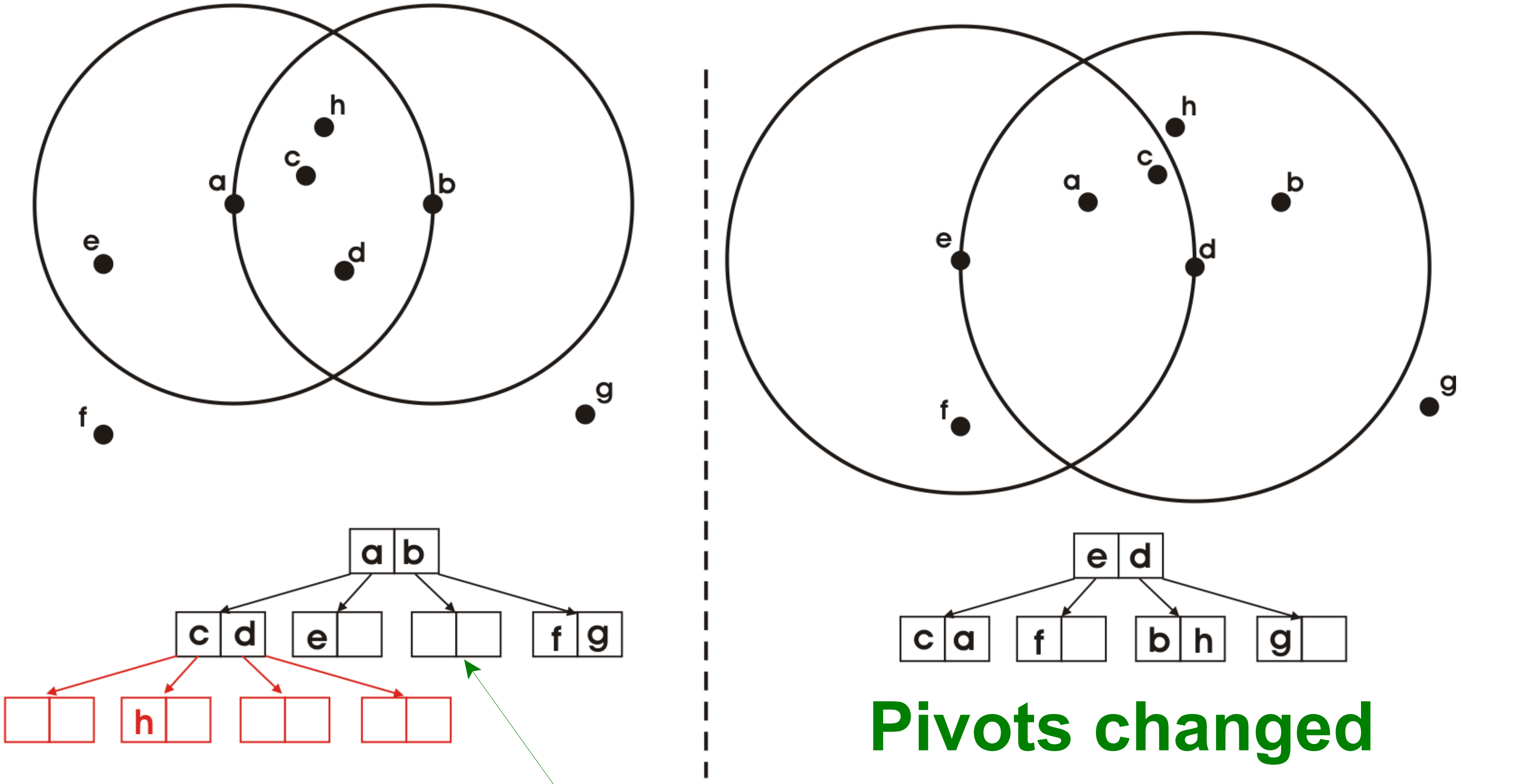
- **MM-tree is Dynamic**
  - **On Insertion**
    - **Which region the new element will belong to?**

$d(s_i, s_1)\ \theta\ r$	$d(s_i, s_2)\ \theta\ r$	Region
$<$	$<$	I
$<$	$\geq$	II
$\geq$	$<$	III
$\geq$	$\geq$	IV



The MM-tree

Balancing control on leaf nodes



Pivots changed

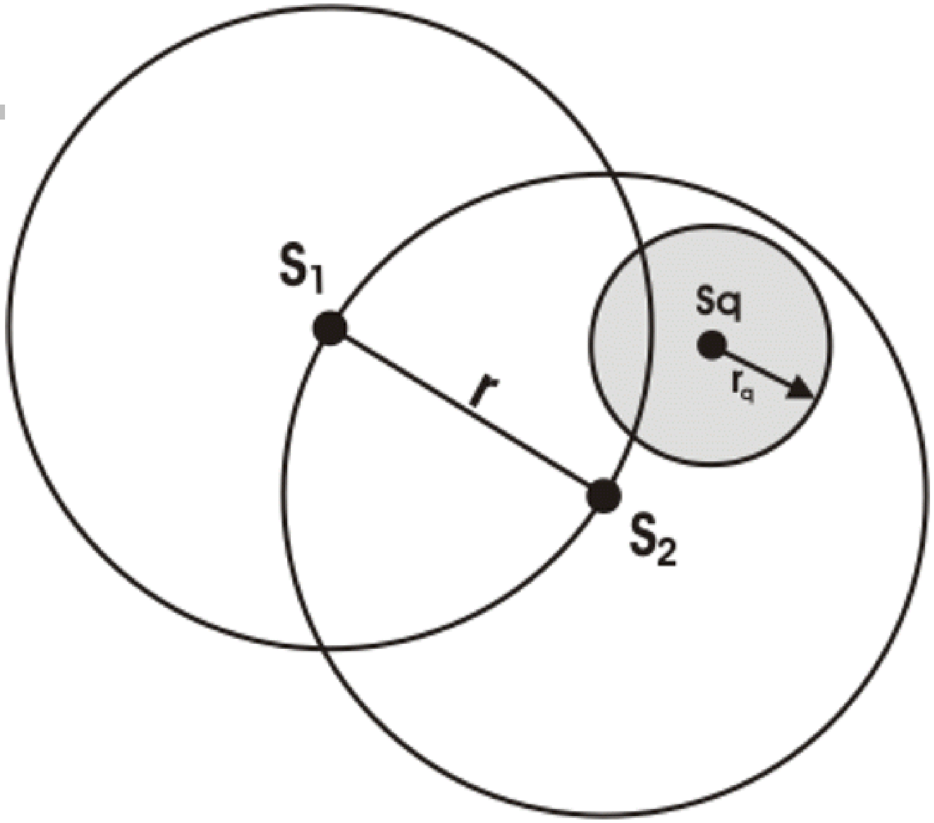
Extra level not needed

Empty node

The MM-tree

Range Queries

- Range Query (Sq, rq)
  - At each node detect which subtree to visit



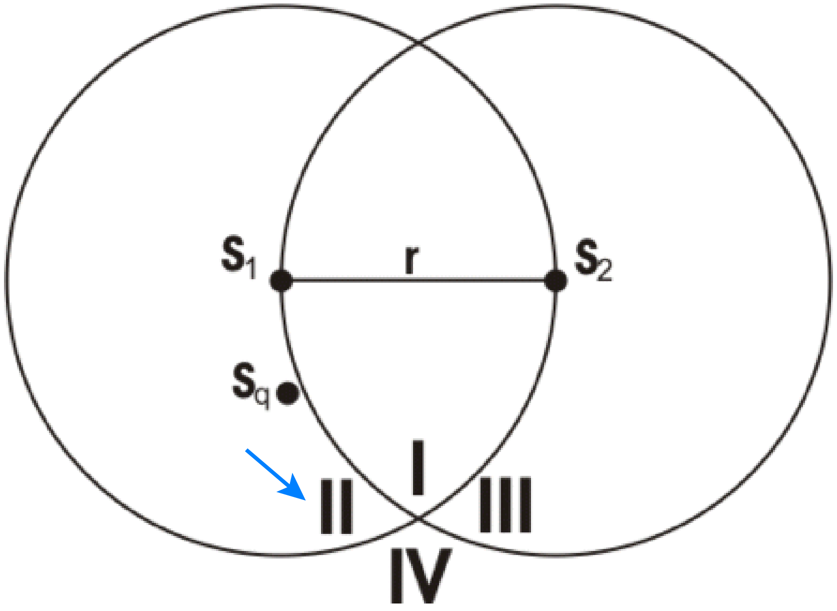
Visiting conditions:

Region I	$(d(s_q, s_2) < r_q + r) \;\wedge\; (d(s_q, s_1) < r_q + r)$
Region II	$(d(s_q, s_2) + r_q \geq r) \;\wedge\; (d(s_q, s_1) < r_q + r)$
Region III	$(d(s_q, s_2) < r_q + r) \;\wedge\; (d(s_q, s_1) + r_q \geq r)$
Region IV	$(d(s_q, s_2) + r_q \geq r) \;\wedge\; (d(s_q, s_1) + r_q \geq r)$

The MM-tree

Guided k-NN Query

- The sequence of subtrees visited depends on where the query center is.



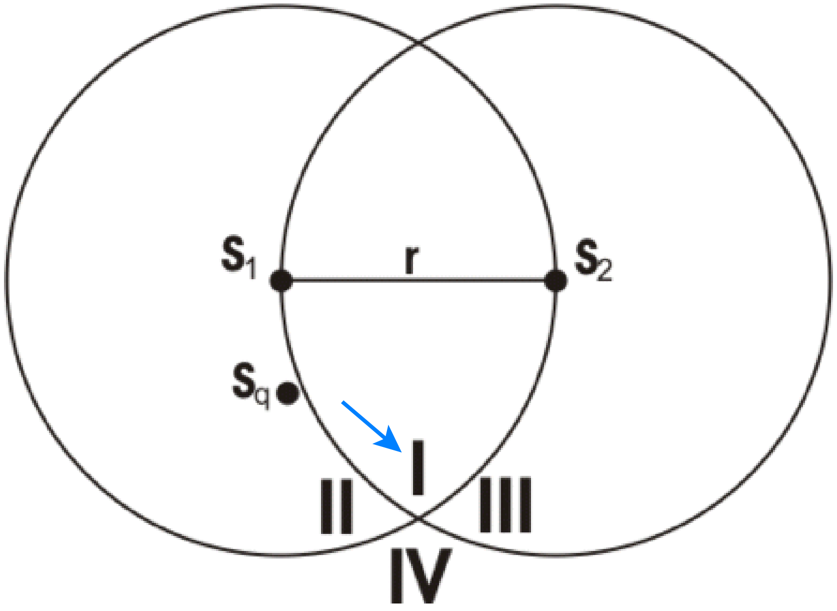
- Visit order:

II → I → (III,IV)

The MM-tree

Guided k-NN Query

- The sequence of subtrees visited depends on where the query center is.



- Visit order:

II → I → (III,IV)

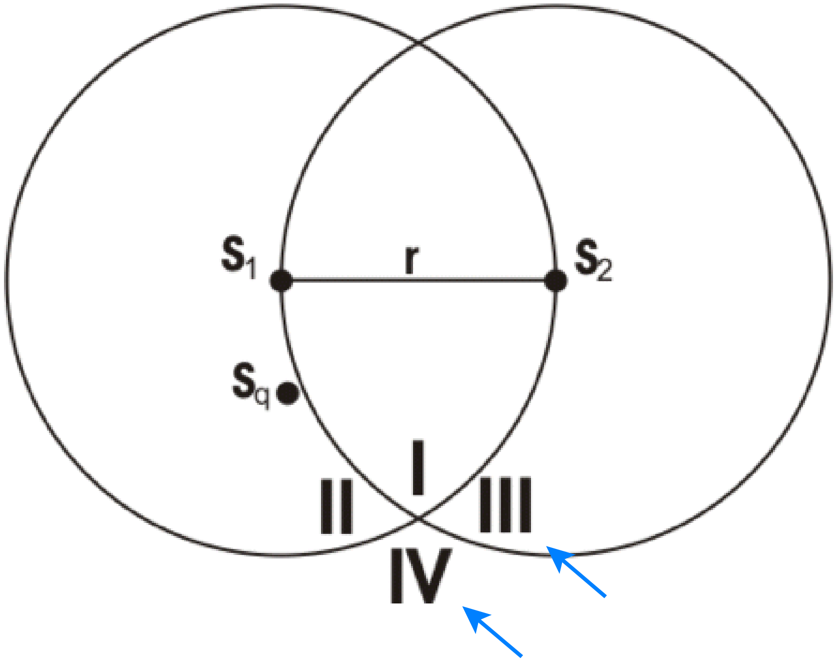
The MM-tree

Guided k-NN Query

- The sequence of subtrees visited depends on where the query center is.

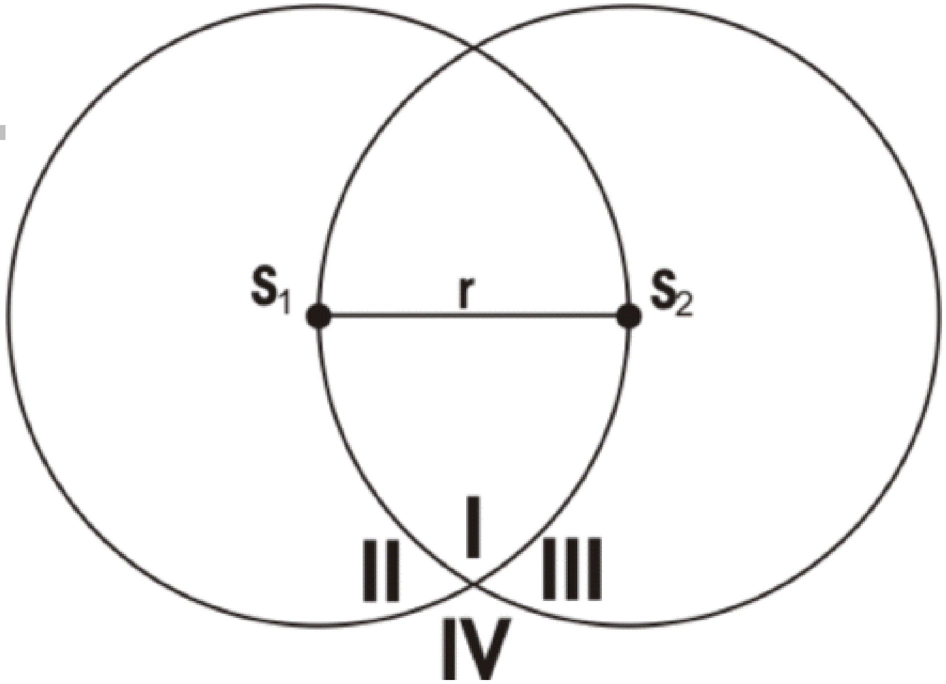
- Visit order:

II → I → (III,IV)

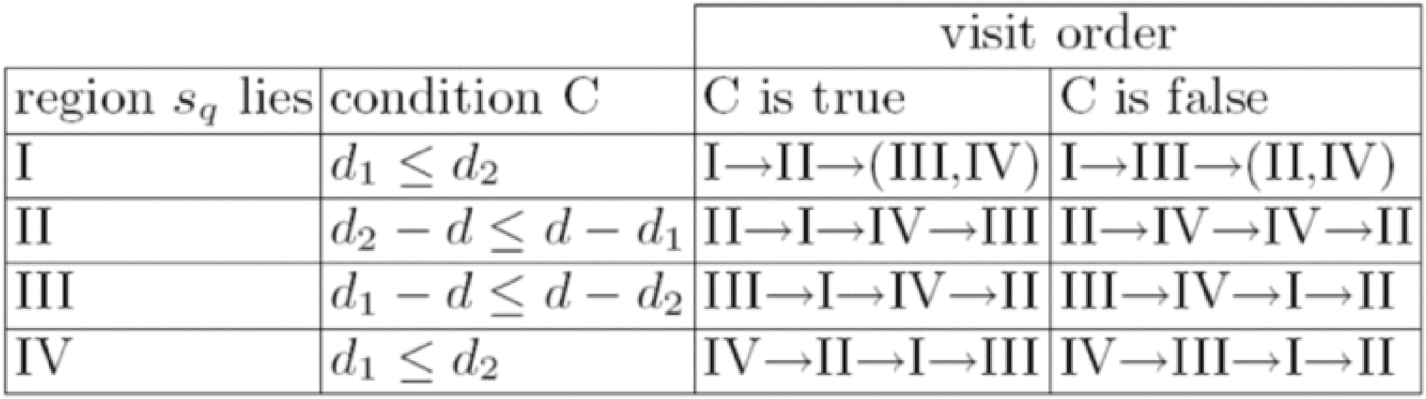


The MM-tree

Guided k-NN Query



- Generalizing, there are different sequences for each region

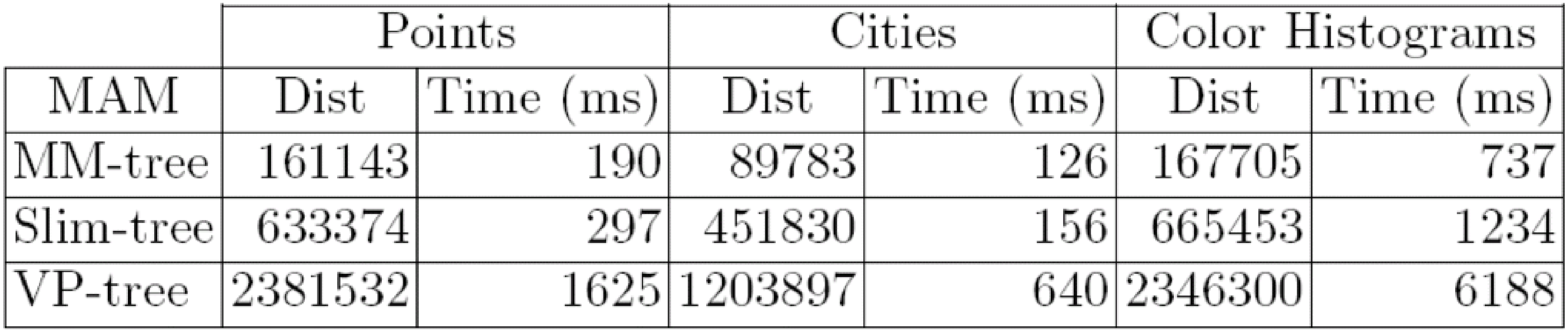


		visit order	
region s_q lies	condition C	C is true	C is false
I	$d_1 \leq d_2$	I→II→(III,IV)	I→III→(II,IV)
II	$d_2 - d \leq d - d_1$	II→I→IV→III	II→IV→IV→II
III	$d_1 - d \leq d - d_2$	III→I→IV→II	III→IV→I→II
IV	$d_1 \leq d_2$	IV→II→I→III	IV→III→I→II

Experiments

Construction Statistics

- The MM-tree was compared with
  - Slim-tree
  - VP-tree

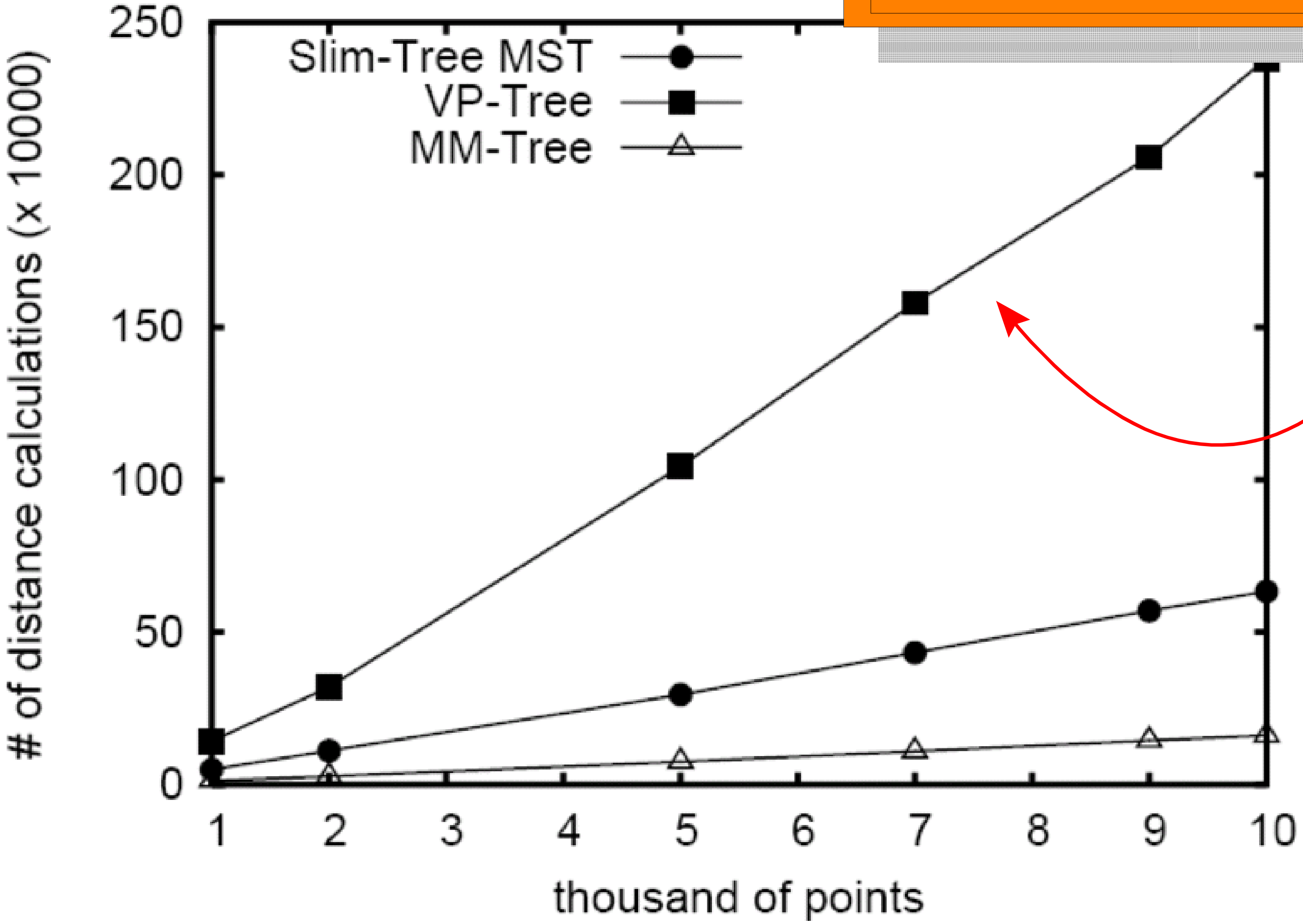


	Points		Cities		Color Histograms	
MAM	Dist	Time (ms)	Dist	Time (ms)	Dist	Time (ms)
MM-tree	161143	190	89783	126	167705	737
Slim-tree	633374	297	451830	156	665453	1234
VP-tree	2381532	1625	1203897	640	2346300	6188

Experiments

Construction Statistics

Number of Distances

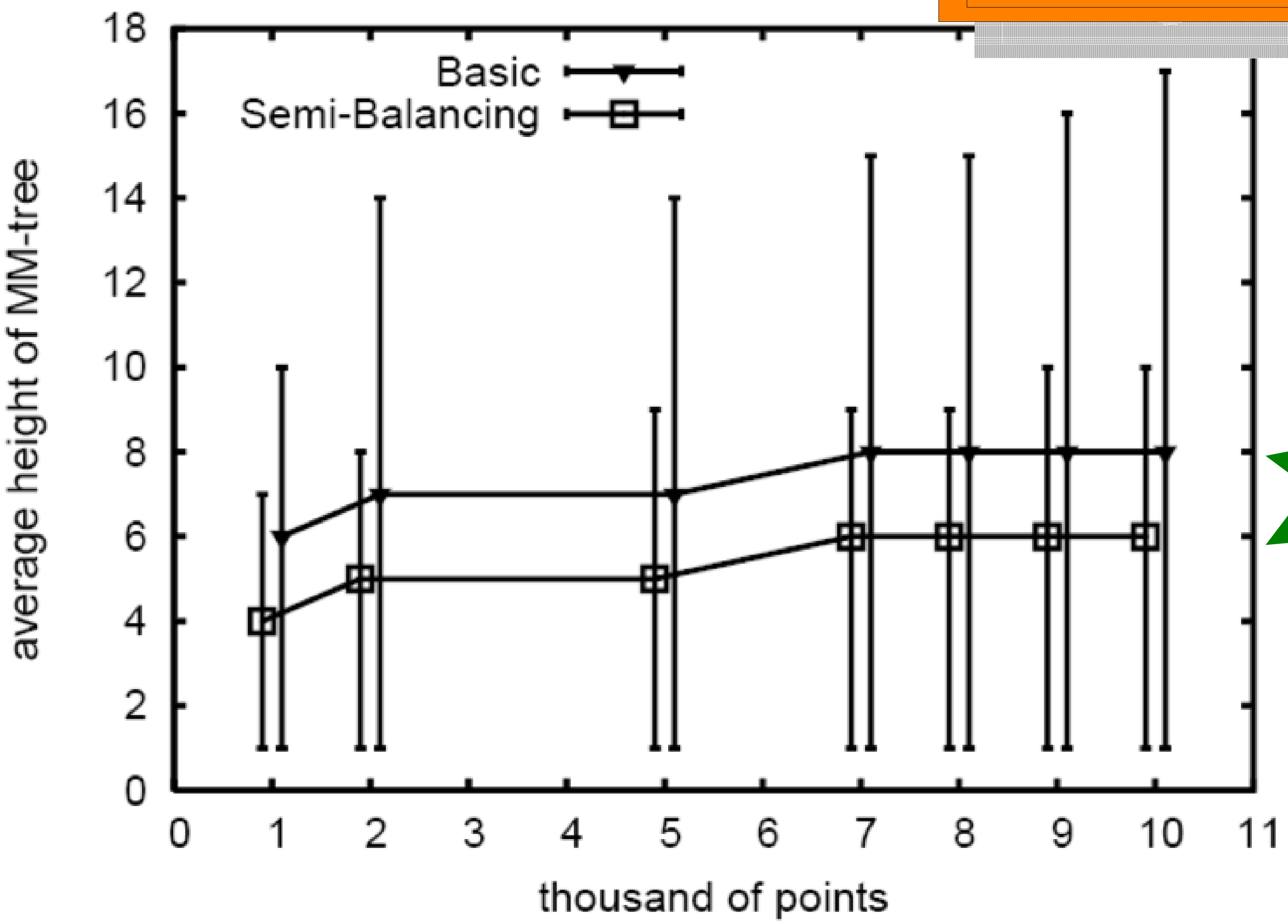


Not acceptable!

Experiments

MM-tree structure statistics

Height of MM-tree

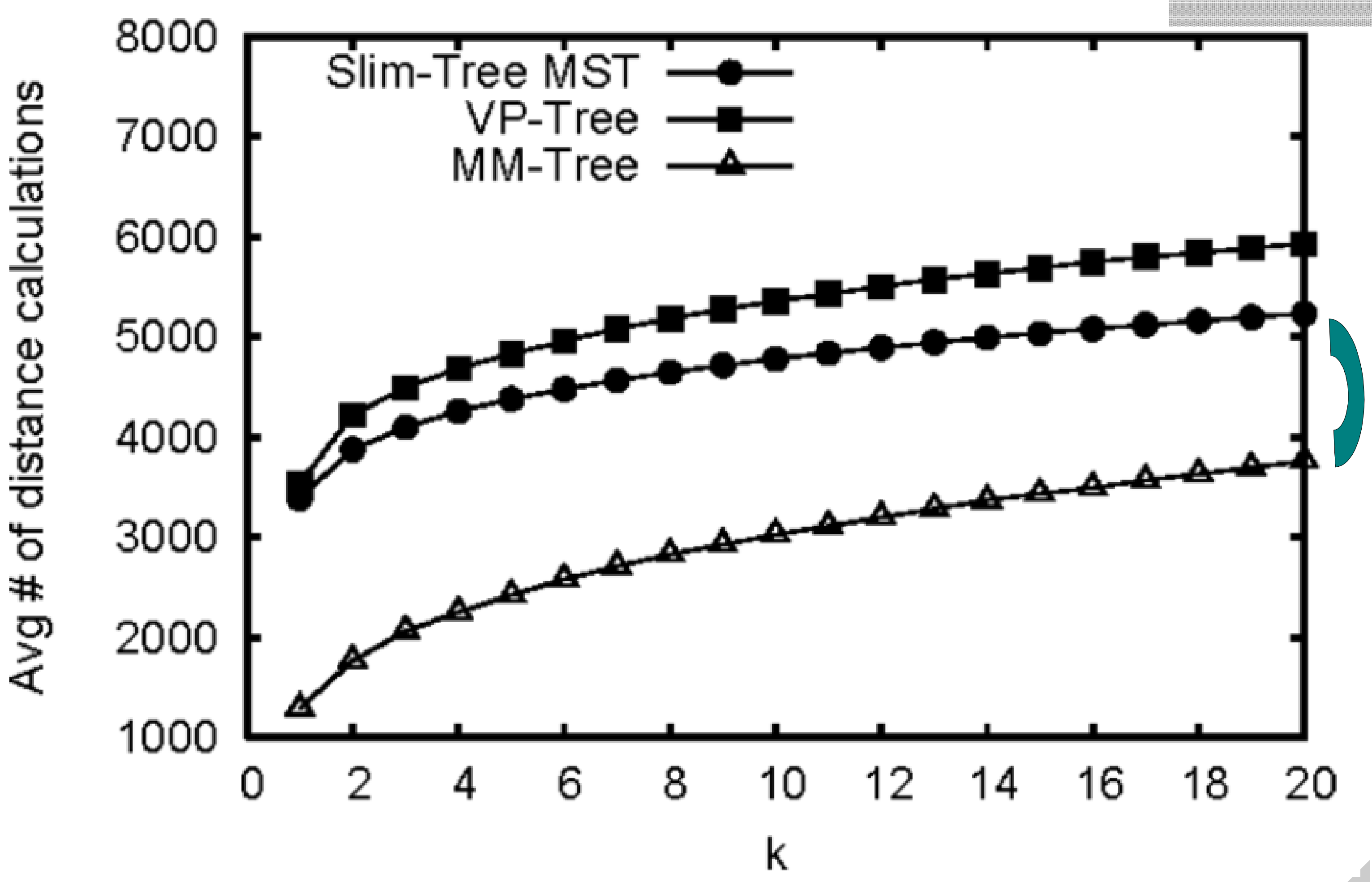


Two levels in average

Experiments

Color Histograms dataset

k-NN query

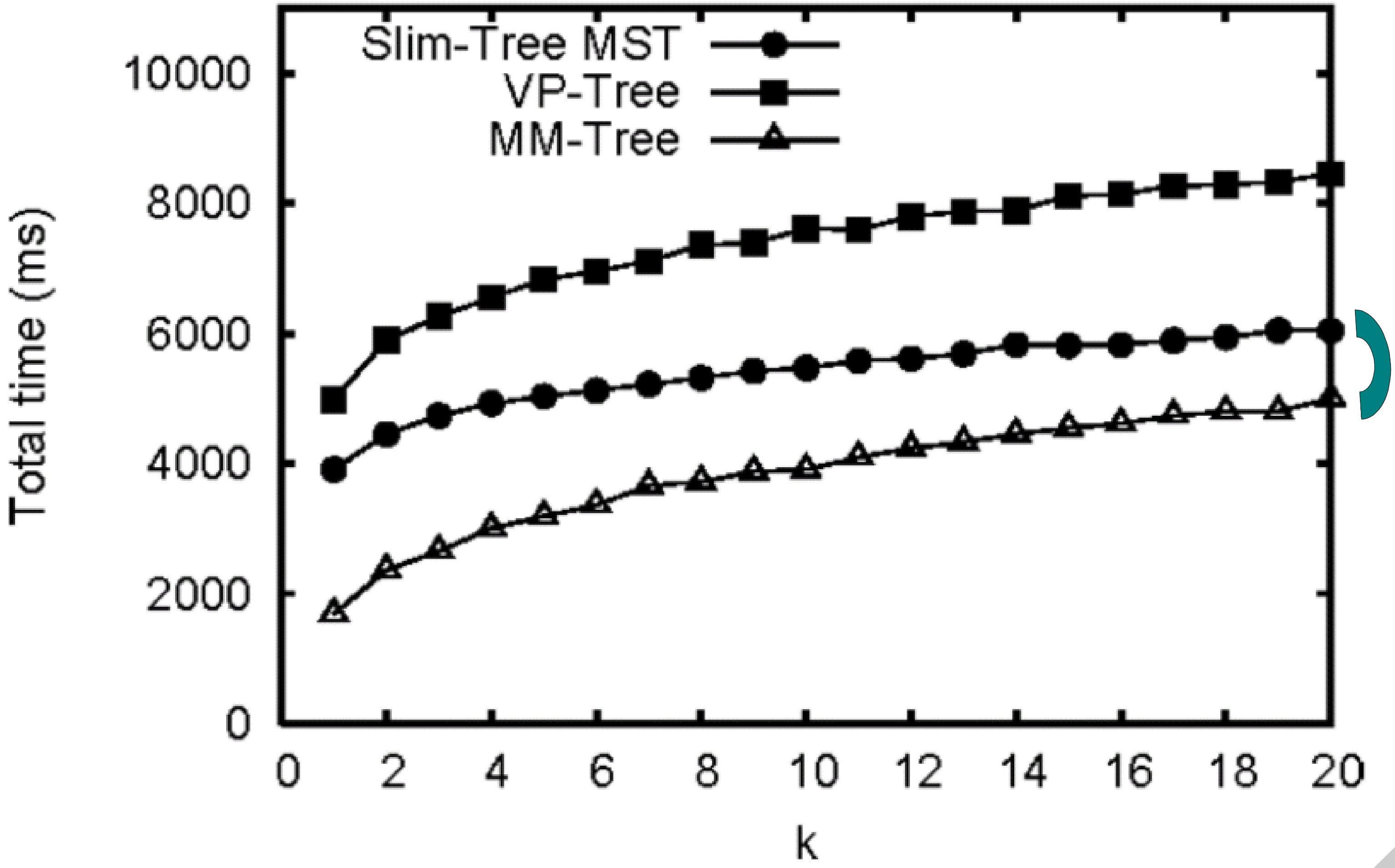


26%

Experiments

Color Histograms dataset

k-NN query

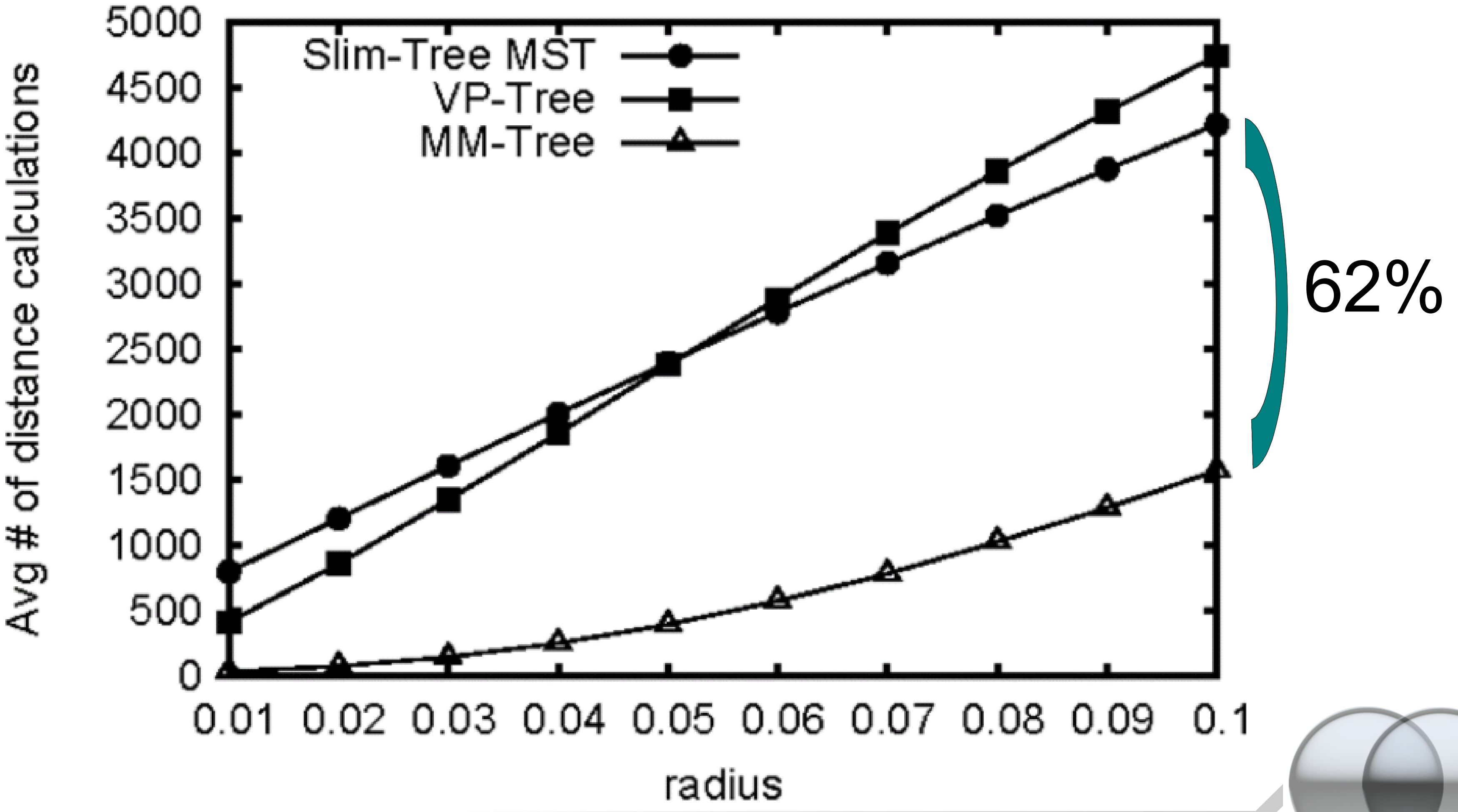


24%

Experiments

Color Histograms dataset

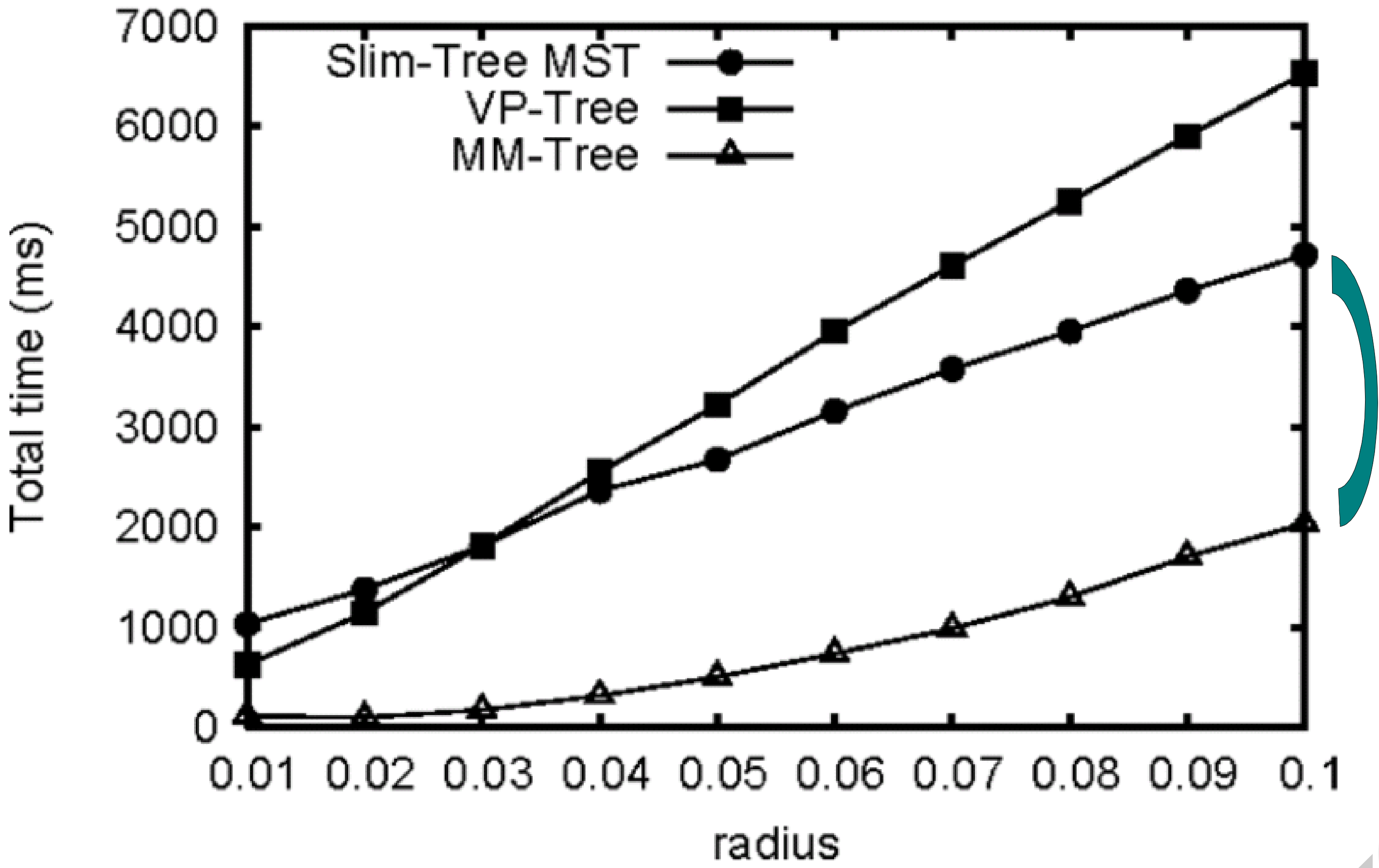
Range query



Experiments

Color Histograms dataset

Range query



60%

Conclusions

The MM-tree

- Useful for emerging applications that require the DBMS to provide fast ways to build indexes on data that fit in main memory
- The MM-tree is fast to build and provide fast similarity queries, partitioning the metric space into disjoint regions.
- Compared to the Slim-tree
  - KNN = 26% less distance calculations, 24% faster
  - RQ = 62% less distance calculations, 60% faster

The MM-tree

A Memory-Based Metric Tree Without Overlap Between Nodes

Thank You.

Open to questions.

Ives R. V. Pola
Caetano Traina Jr.
Agma J. M. Traina