Optimizing Query Performance in Metric Spaces

Matej Antol

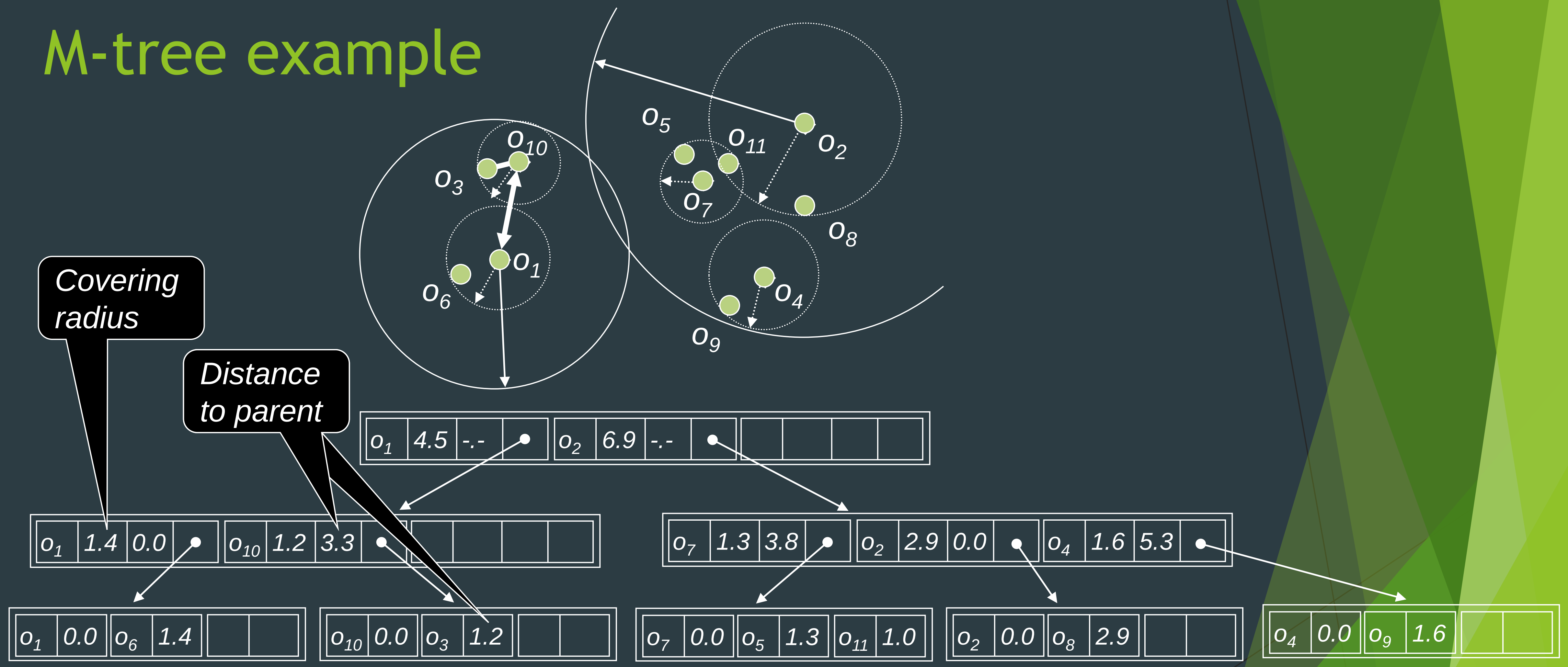
Content

- Background and motivation - indexing metric spaces and query evaluation
- Motivation for query evaluation optimization
- Approach #1 – Inverted Cache Index (ICI)
- Approach #2 – Hybrid strategies for priority queue creation
- Conclusions and future work

Indexing in metric spaces

- Indexes based on objects’ mutual distances
  - No coordinate system can be used to split data space
- Typically data-driven partitioning/clustering
  - M-tree
    - Clusters objects bottom up (like B-tree / R-tree)
  - M-index
    - Partitions space top down (recursive Voronoi partitioning)

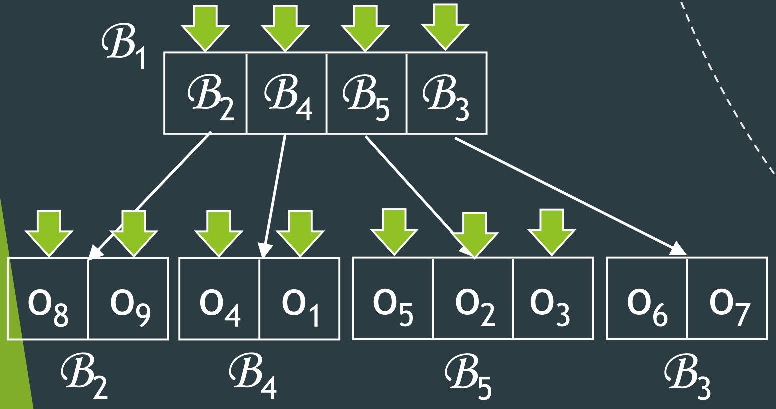
M-tree example



NN Search Algorithm

3-NN(*q*):

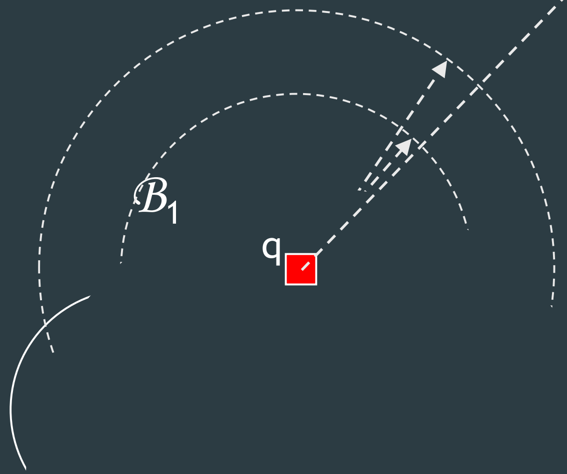
- Process $\mathcal{B}_1$
- Process $\mathcal{B}_2$
- Process $\mathcal{B}_4$
- Process $\mathcal{B}_5$
- Skip $\mathcal{B}_3$
- *PQ* is empty, quit.



Processing: done

Response: o_8, o_9, o_1

Final result

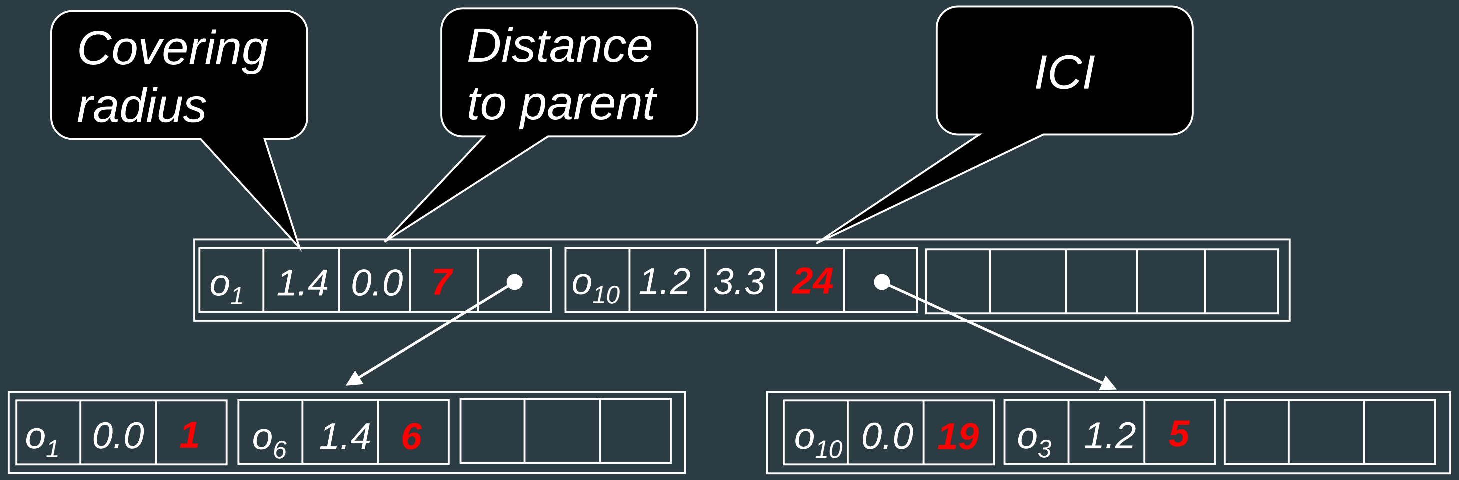


Motivation: Querying performance

- CoPhIR dataset
  - 1 million images, 5 MPEG7 features per image, one weighted distance function
- Querying using 1000 selected queries – 30NN query
  - 1 query enters around 1000 leaf nodes (in case when total number of l.n. is 1124)
  - Avg. number of leaf nodes containing answer objects is ~17
  - Avg. position of last positive leaf node is ~230
  - First positive leaf node is typically within first 5 visited leaf nodes

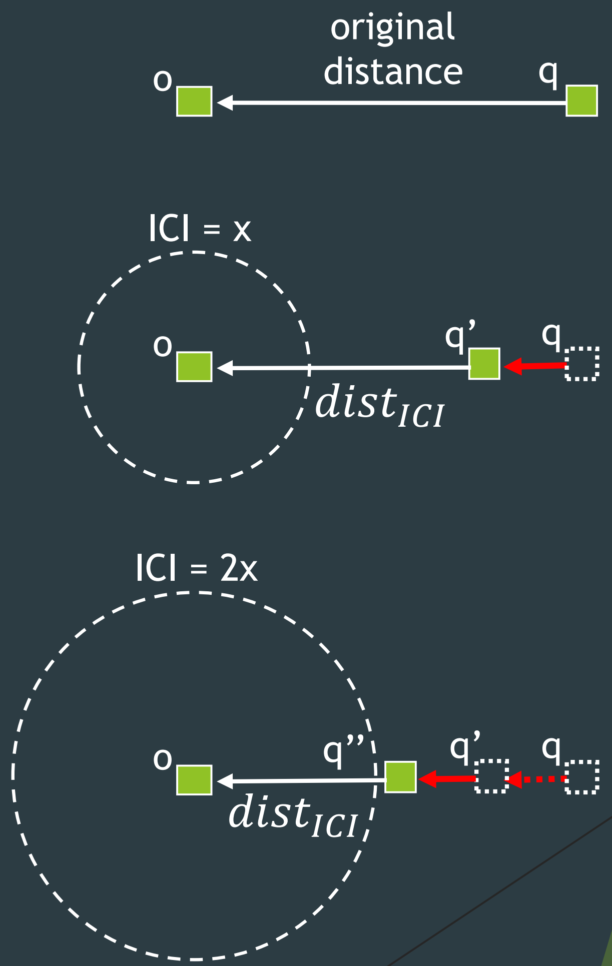
App. #1 - Inverted Cache Index (ICI)

- Remembering number of times an object/node was part of an answer to prior queries.
- ICI value of any node equals sum of ICI values of all its children



Inverted Cache Index (ICI)

- Use ICI with the distance between
  - query and node
  - query and object
- ICI can be depicted as a "mass" of the object/node
  - So creating "attraction force" that pulls the query closer
- Priority queue in kNN algorithm is ordered by this modified "distance"



Naïve ICI: Mass Distance (qd)

Formula of Naïve ICI:

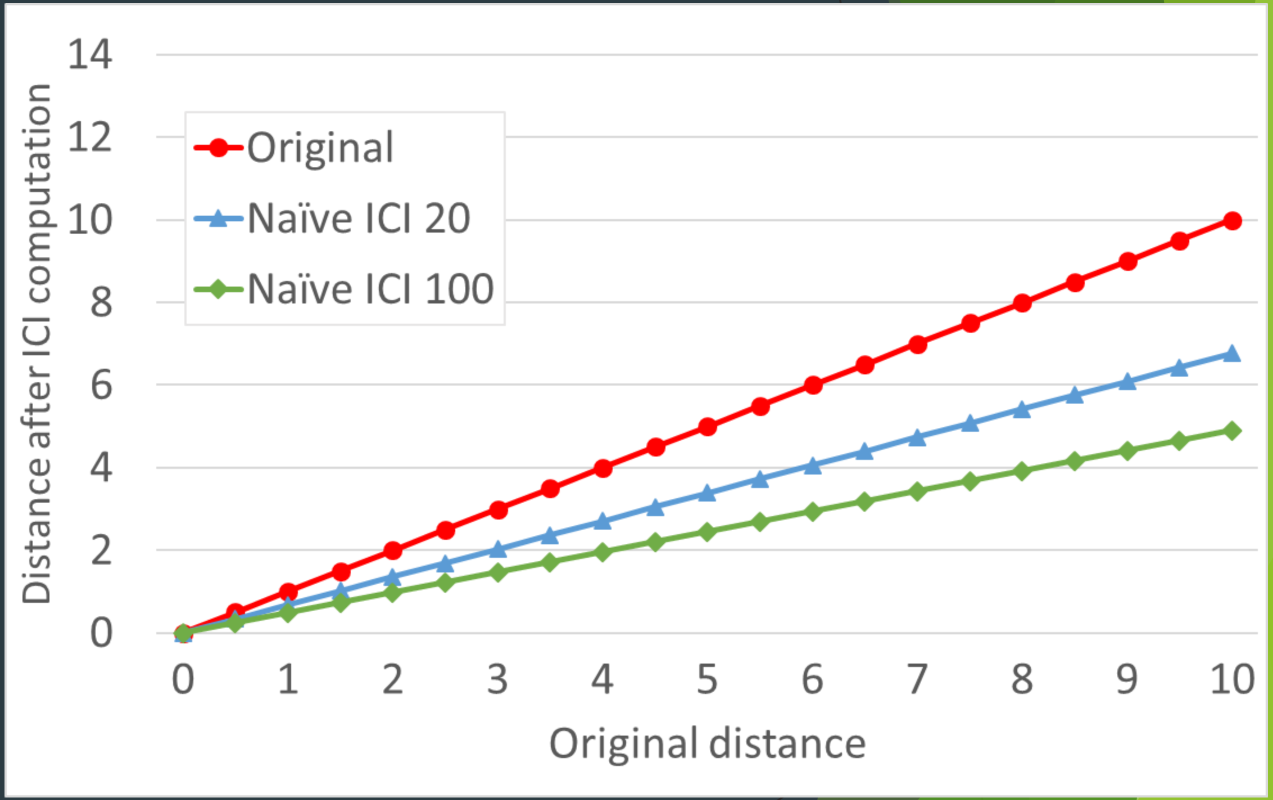
$$mass = \log_b(ICI + b)$$

$$dist_{ICI} = d/mass$$

Where:

b = selected log base

d = distance in original metric space



ICI = 20, b = 10

Extended ICI: Gravity Distance (qgd)

Formula of Extended ICI:

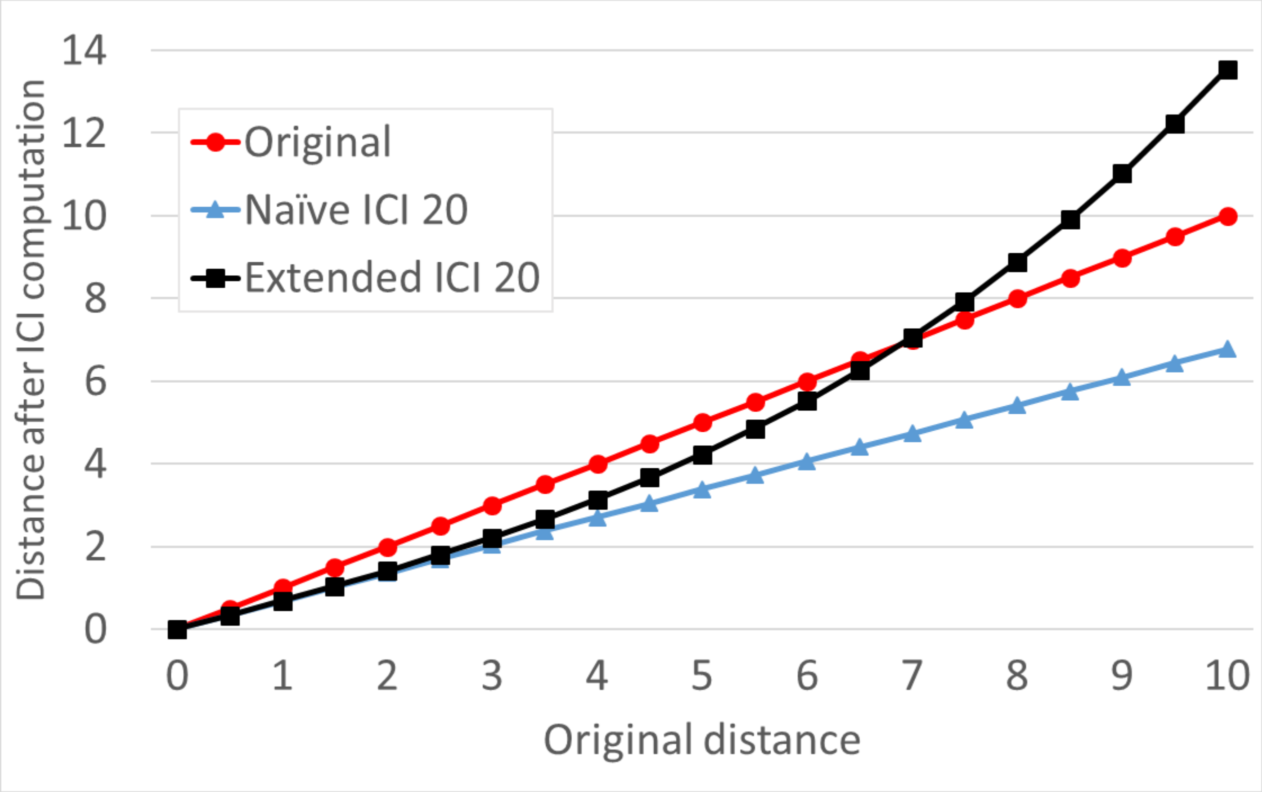
$$mass = \log_b(ICI + b)$$

$$mass_{gravity} = mass/\left((d/maxdist)^p + 1\right)$$

$$dist_{ICI} = d/mass_{gravity}$$

Where:

- d = distance in original metric space
- maxdist = maximum distance
- b = base of logarithm, mass growth
- p = power of normalized distance
  - (how strong gravitation force is)



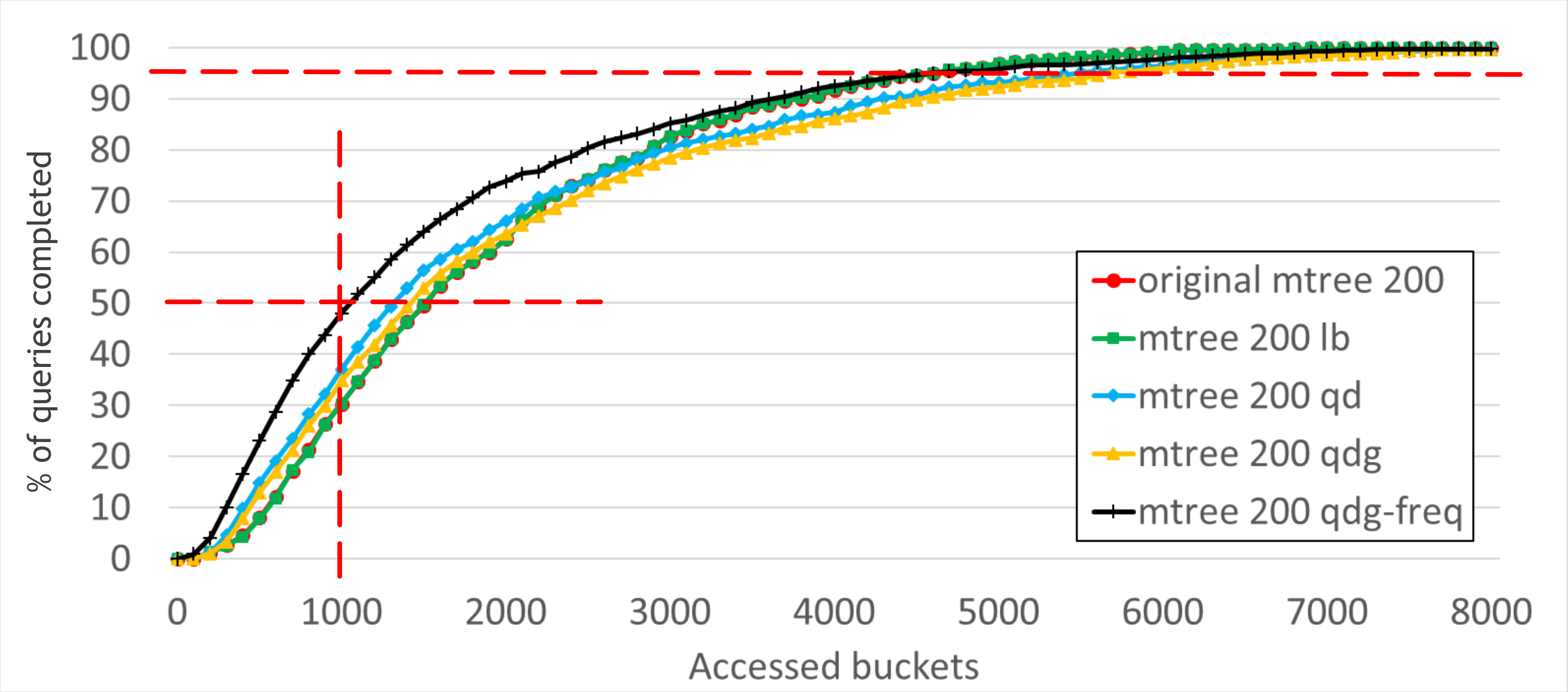
ICI = 20, b = 10,
p = 2, maxdist = 10

Experiment Protocol

- CoPhIR dataset
  - 1 million images, 5 MPEG7 features per image, one weighted distance function
- M-tree and M-index structures
  - varying leaf node capacity
- Queries
  - 1000 most repeated queries w.r.t. Google Analytics on Mufin Demo App
- Experiments
  1. Proof of concept – M-tree, l.n. cap = 200
     comparison of original results with naïve and various forms of extended ICI
  2. Results on M-tree, l.n. cap. = 2000, different learning and testing datasets
  3. Results on M-index, l.n. cap. = 2000, different learning and testing datasets
- Comparison measure
  - % of queries completed for 30-NN

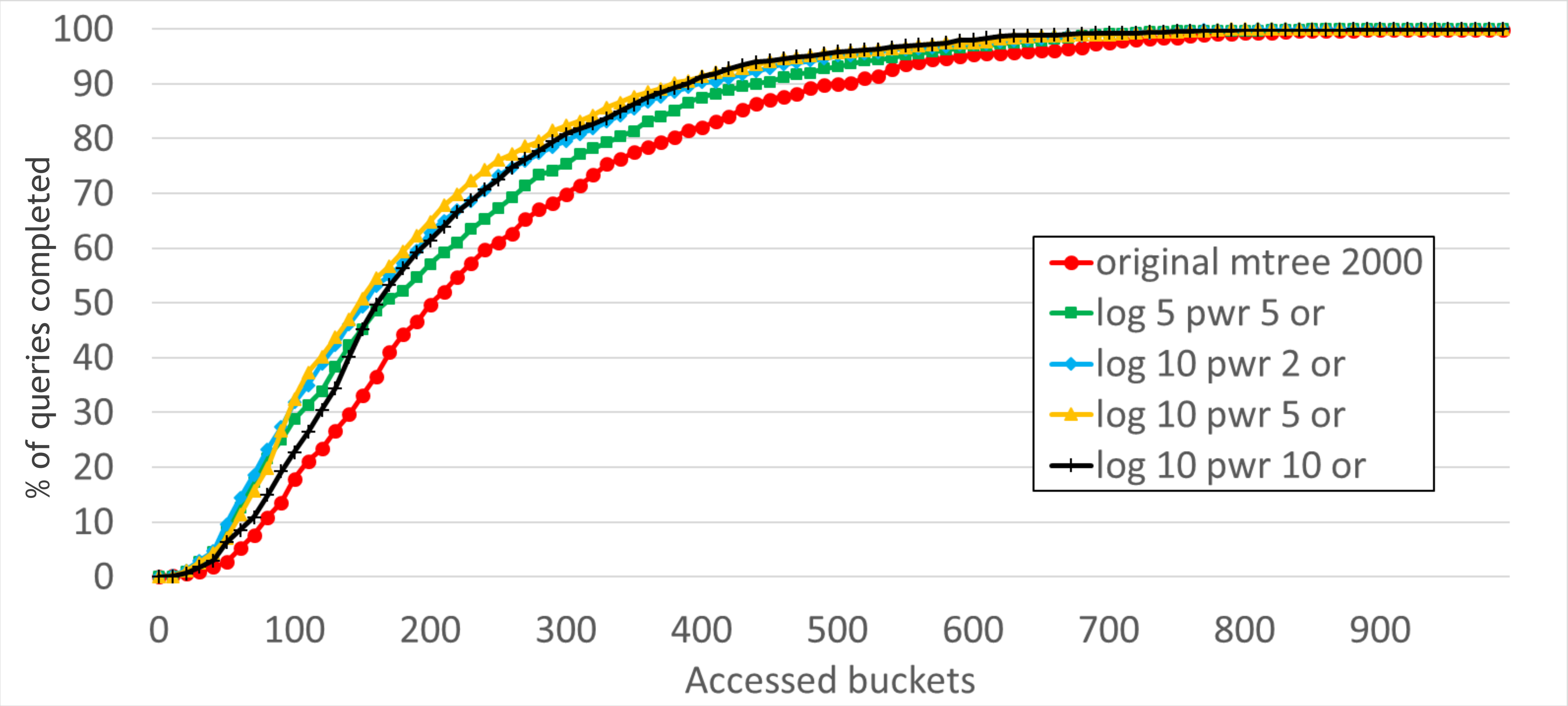
Proof of concept – M-tree

Leaf node capacity 200
Total leaf nodes 11 571
Different distance alteration approaches



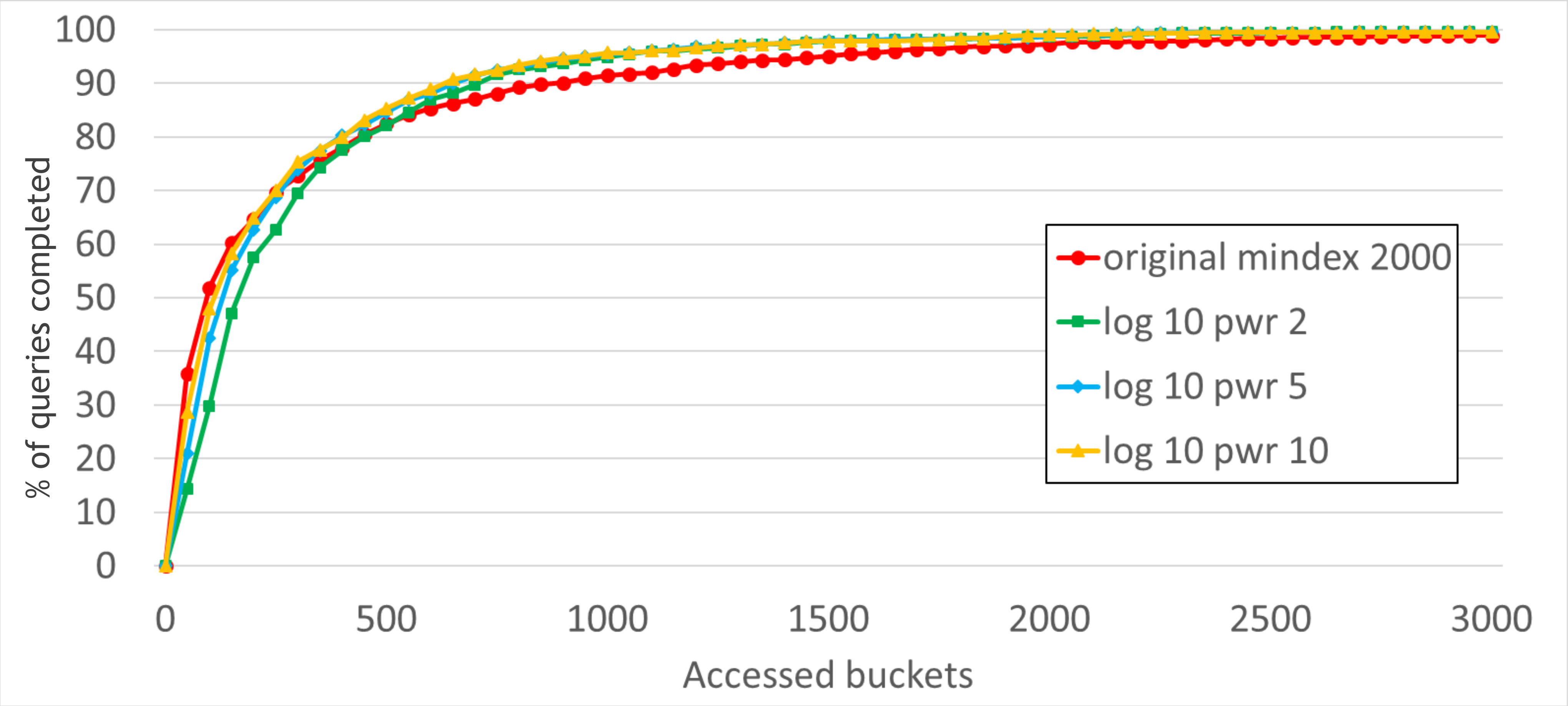
Results on M-tree

Leaf node capacity 2000
Learning on 1 year traffic (2009)
Testing on consequent 1 month traffic (1/2010)



Results on M-index

Leaf node capacity 2000
Learning on 1 year traffic (2009)
Testing on consequent 1 month traffic (1/2010)



Overall improvement

Indexing structure	log-pwr	Orig. visited l.n. 95% q. completed	ICI visited l.n. 95% q. completed	Improvement
M-tree 200	5-5	4600	4200	**8,7 %**
M-tree 2000	10-2	590	470	**20,5 %**
M-index 200	10-5	8000	6000	**25 %**
M-index 2000	10-5	1500	950	**37 %**

App. #2 - Hybrid strategies for priority queues

- Concluded from deeper analysis of queries in metric spaces

- Simple tool for processing and visualization of the data
  - Distances
  - Nodes radii
  - Number of objects within leaf nodes
  - ICI values
  - Distances according to different queries
    - lower bound
    - Precise
    - upper bound

Priority queues – current state

Three basic strategies are being used to create priority queues:

lower bound, upper bound and precise

LOWER BOUND

p1, p3, p2; prunes p4

visits small and further buckets later (p2)

UPPER BOUND

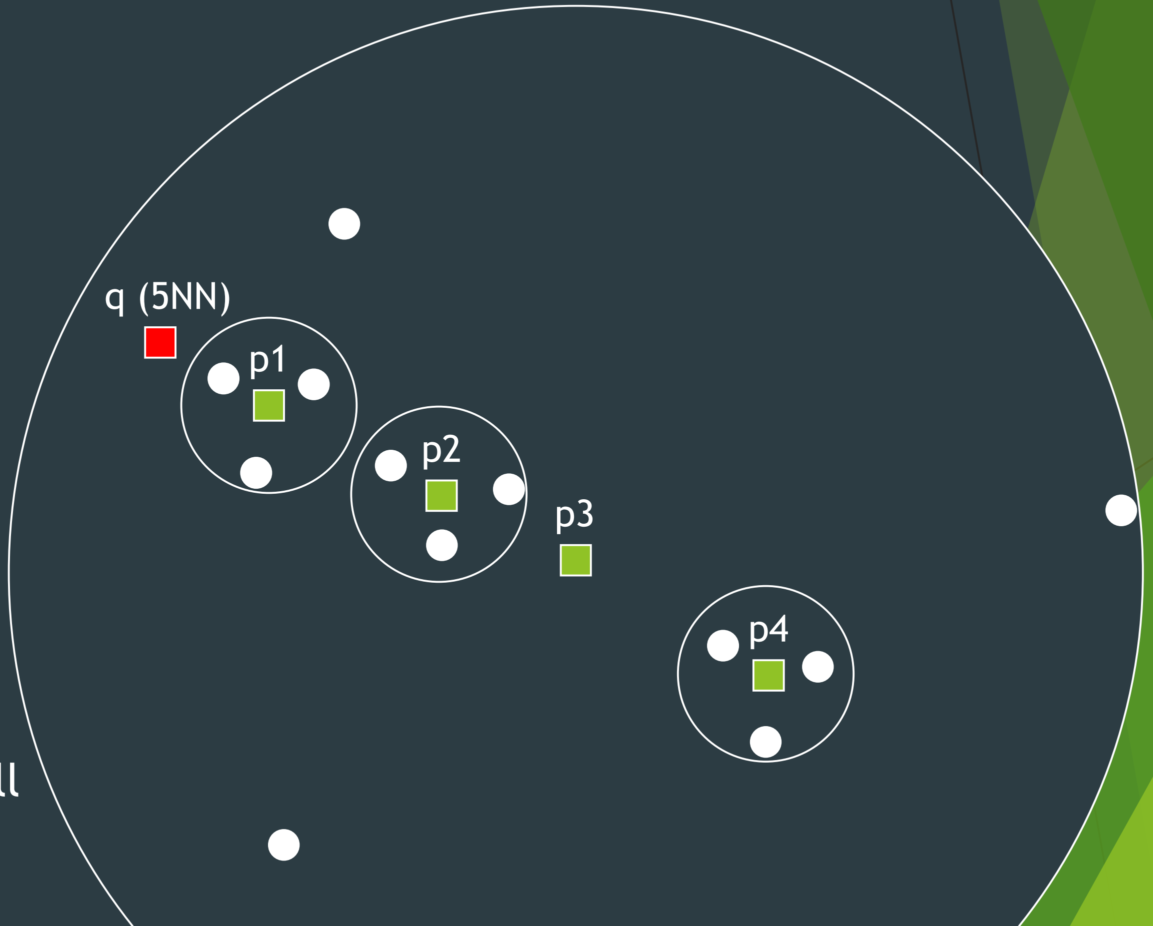
p1, p2, p4, p3

visits larger buckets later (p3)

PRECISE

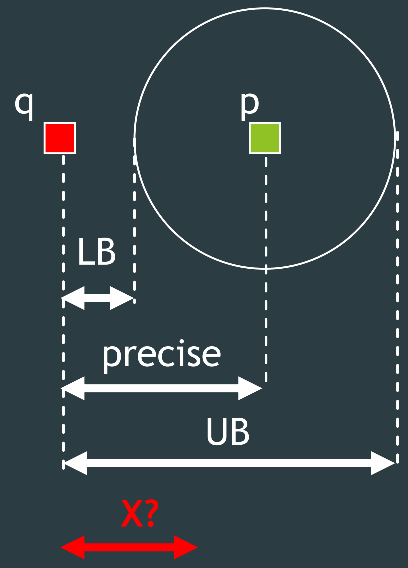
p1, p2, p3; prunes p4

does not take “density” into account at all



Priority queues – current state

- Current strategies are discrete – probably because of their intuitive representation
- Better priority queues can be constructed depending on density, size, no. of objects, etc…



LB = dist – rad

precise = dist

UB = dist + rad

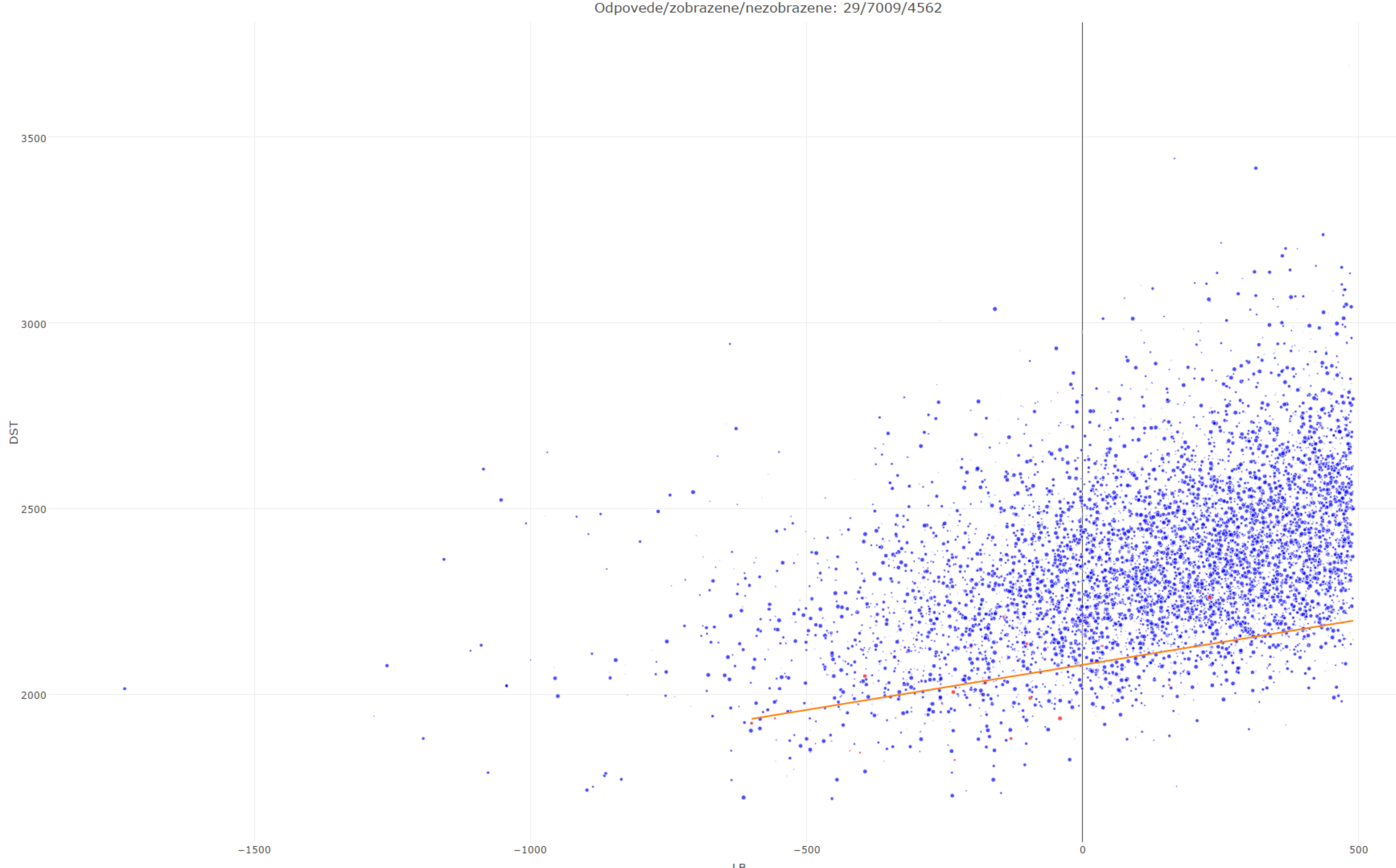
X = dist - 1/3 rad ?

…

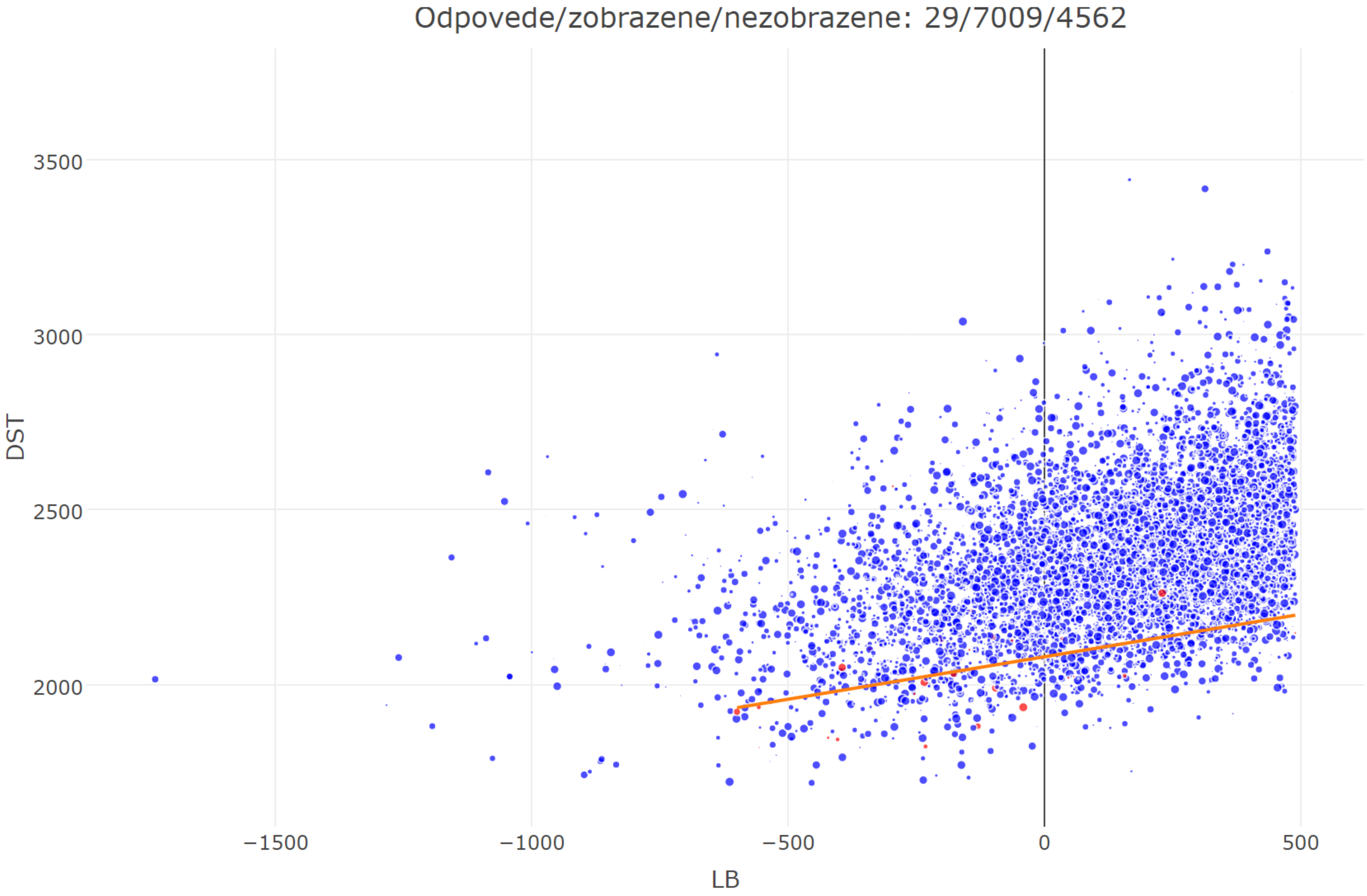
Priority queues performance analysis

- Best performing method (on our dataset) is lower bound
- We compared different parameters of the structure and queries

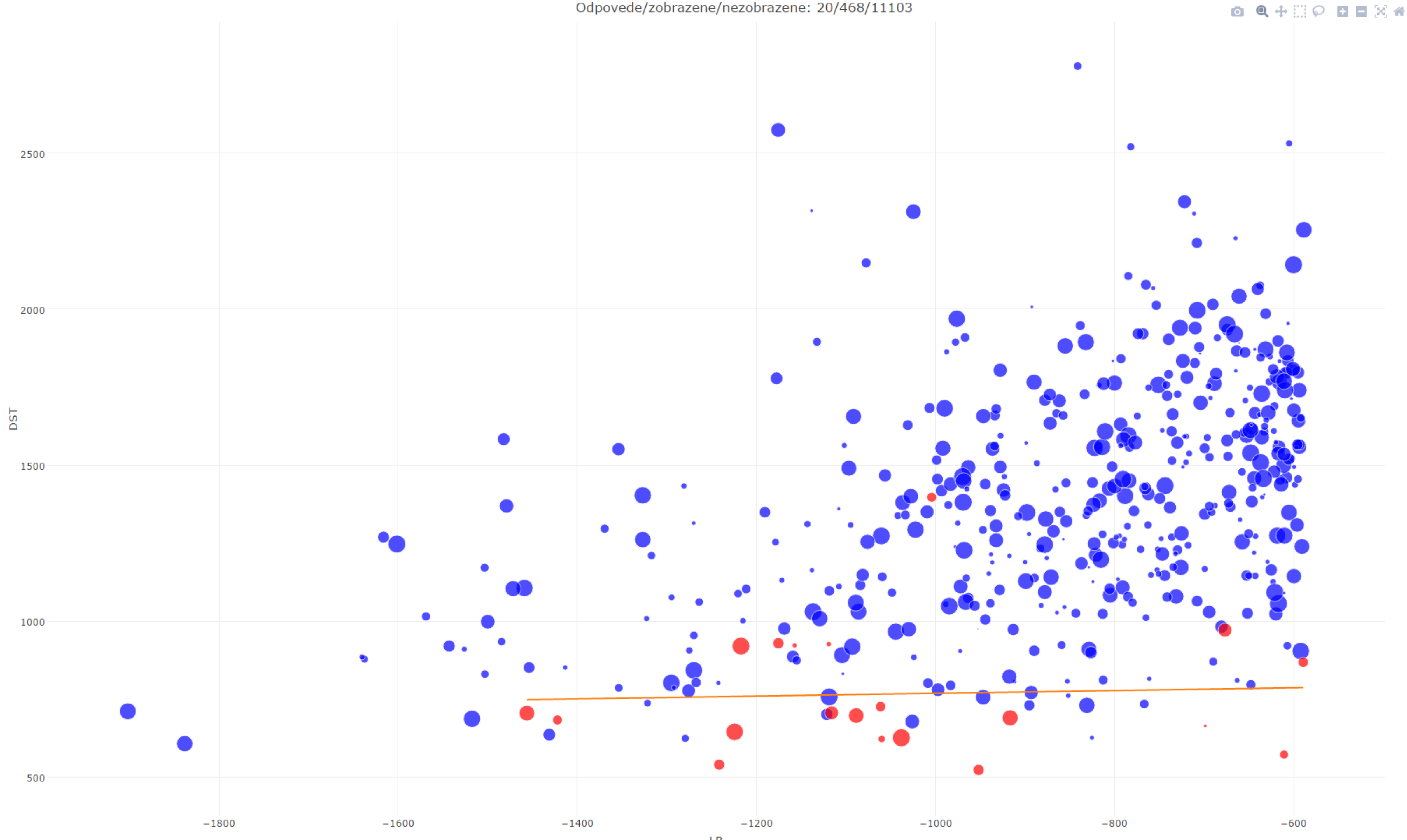
  (distances, nodes radii, number of objects, ICI values, distances to queries)
- Preliminary results show that suitable compound of 2 (or more) strategies can lead to better results



Odpovede/zobrazene/nezobrazene: 29/7009/4562
DST
3500
3000
2500
2000
−1500
−1000
−500
0
500



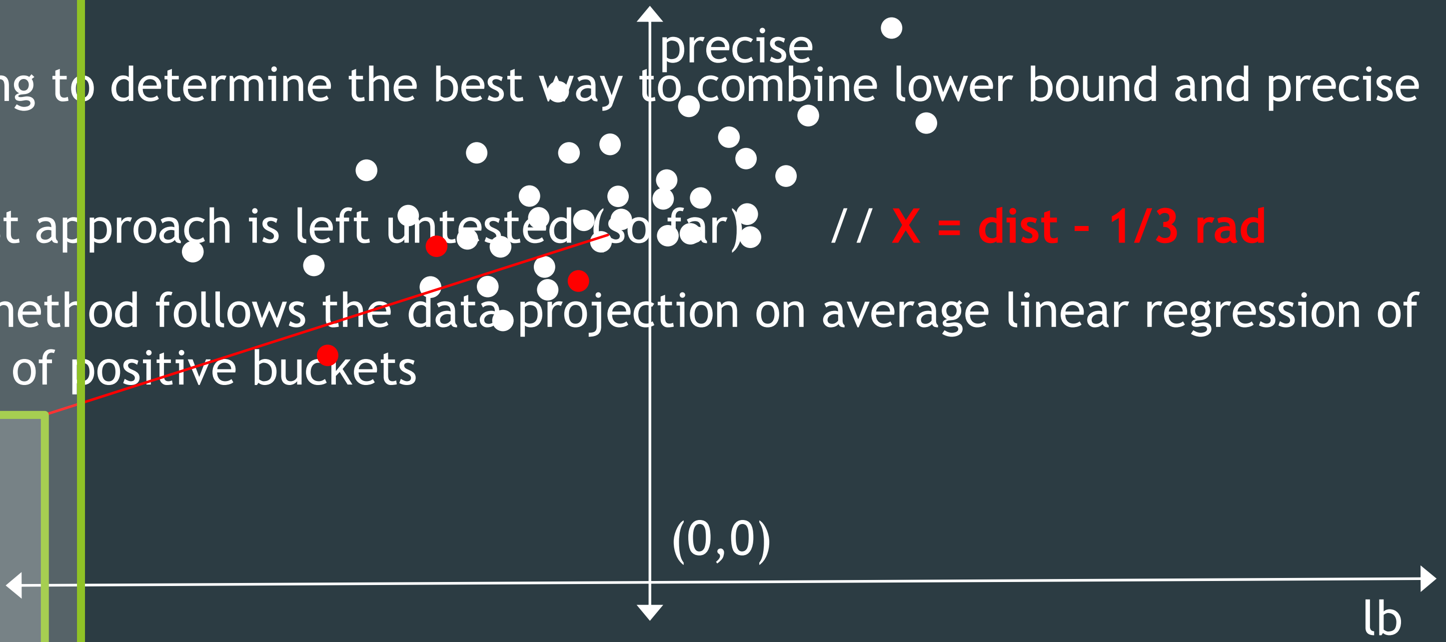
Odpovede/zobrazene/nezobrazene: 29/7009/4562
3500
3000
2500
2000
DST
−1500
−1000
−500
0
500
LB



Odpovede/zobrazene/nezobrazene: 20/468/11103
DST
2500
2000
1500
1000
500
−1800
−1600
−1400
−1200
−1000
−800
−600

Suggested method

- We are trying to determine the best way to combine lower bound and precise strategy
- The simplest approach is left untested (so far) // X = dist - 1/3 rad
- Suggested method follows the data projection on average linear regression of coordinates of positive buckets



order = max (k*(dst-rad) + q, dst)

dst = |p,q|

rad = size(p)

empirical (average) values: k=0,45 q=1900

// k*(dst-rad) + q denotes parametric form of linear regression (y = k*x + q)

Preliminary performance

- Total number of buckets in our structure is 11 571
- Last positive bucket has average position in priority queue 1 775
- Proof of concept setup in simulation has average position of last bucket around 1 343 (24% improvement)
- Proposed solution could outperform currently used strategies by tens of per cents
- Method is simple, clean and does not require any adjustments in indexing structures

Conclusions

- Inverted Cache Index
  - Shows improvement greater than 25% for a state-of-the-art indexing structures
  - Successful paper on ADBIS conference; paper was selected to be sent to impact journal
- Hybrid strategies for priority queues
  - New, yet untested method with promising future ☺

Future Work

- Inverted Cache Index
  - Journal paper
  - Application to approximate KNN query evaluation
- Hybrid strategies for priority queues
  - First tests on real data
  - Testing variety of strategies (alternatives to linear regression [max coordinates, …])
  - Determining the relation between structure parameters and strategy setup (dependence on density, avg dist, avg rad, number of objects, structure depth, …)

Thank you for your attention