USC Viterbi School of Engineering

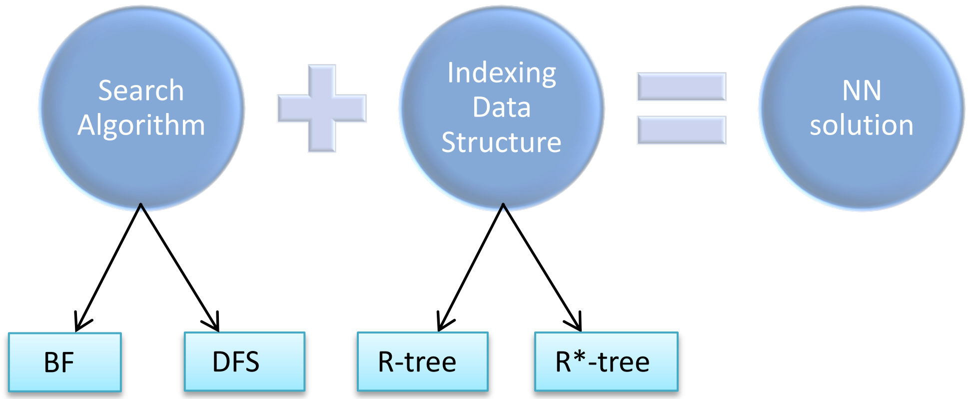
Reverse kNN search in Arbitrary Dimensionality

Instructor: Cyrus Shahabi

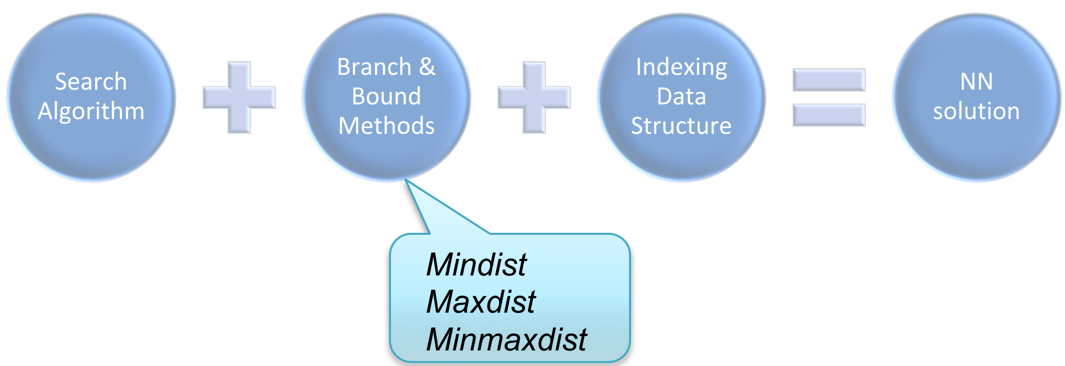
USC Viterbi School of Engineering

Algorithms for finding NN

- Elementary methods:

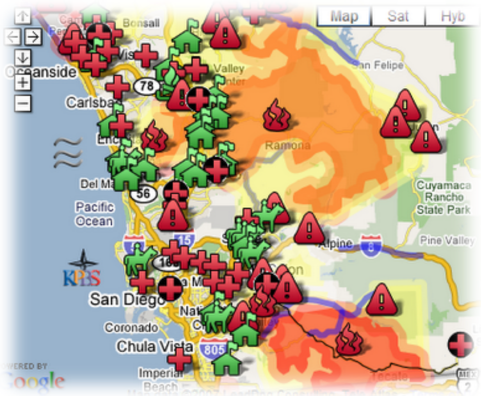


- More advanced methods:



Reverse Nearest Neighbors Queries

What are the fire locations I'm nearest to?



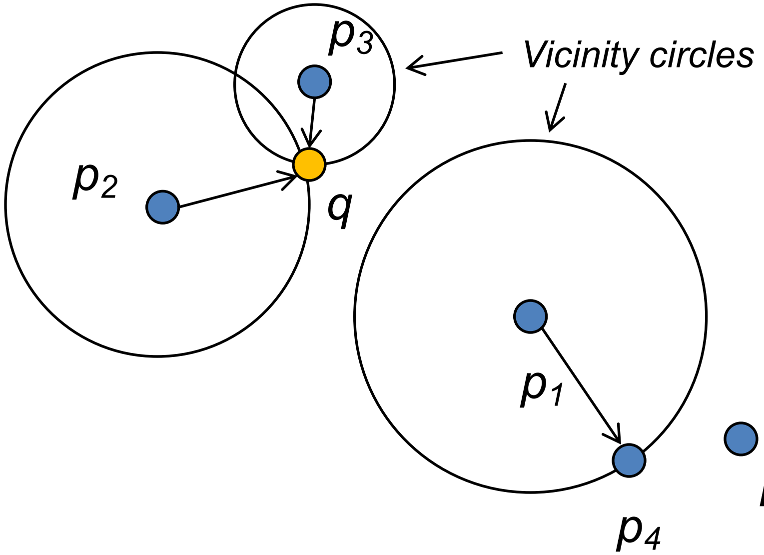
Which houses I'm the closest restaurant to?

RNN Definition

- A data point *p* is the reverse nearest neighbor of query point *q*, if there is no point *p'* such that *dist(p', p)< dist(q, p),* i.e. *q* is the NN of *p*.

$NN(p_2)=NN(p_3)=q$

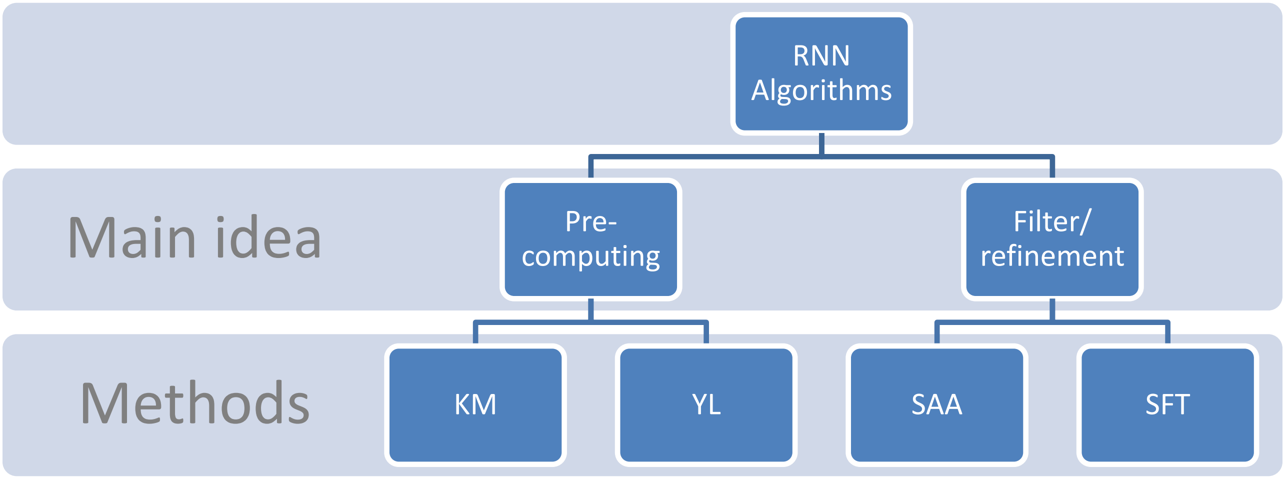
$RNN(q)= \{p_2, p_3\}$



- Is RNN a symmetric relation?

USC Viterbi School of Engineering

Related Work



USC Viterbi School of Engineering

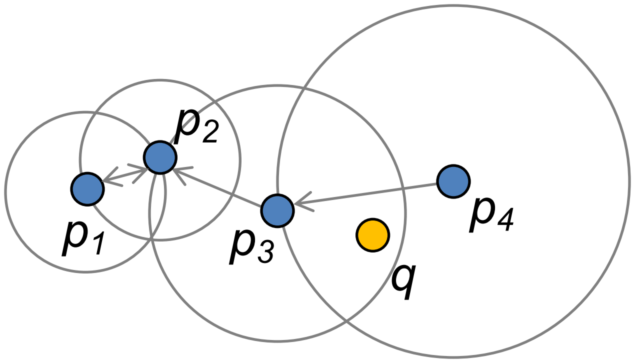
KM

- Original RNN method

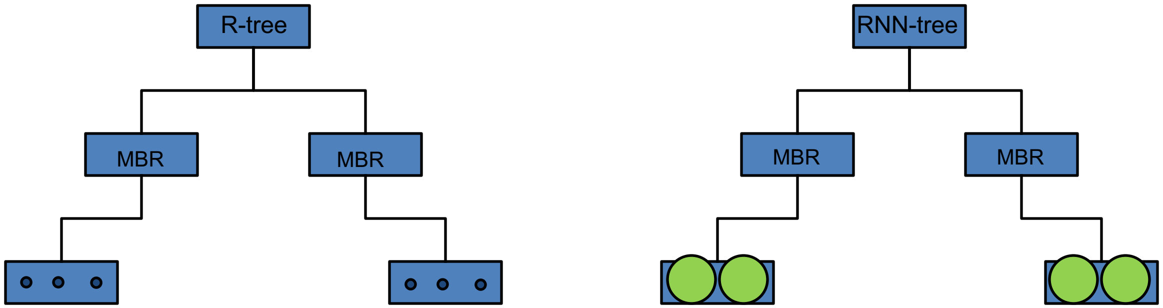
  For all *p:*

1. Pre-compute NN(*p*)
2. Represent *p* as a vicinity circle
3. Index the MBR of all circles by an R-tree

   (Named RNN-tree)
4. RNN(*q*)= all circles that contain *q*



- Needs two trees: RNN-tree & R-tree



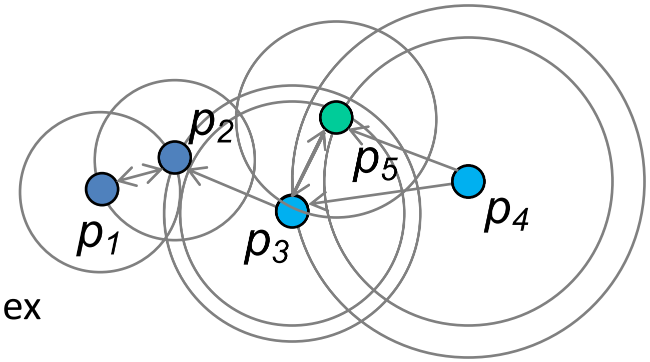
USC Viterbi School of Engineering

KM (Cont.)

- YL: Merges the trees
- What happens if we insert p_5?

  RNN(p_5)=?

  1. Find all points that have p_5 as their new NN
  2. Update the vicinity circles of those points in the index
  3. Compute NN(p_5) and insert the corresponding circle in the index
- Drawbacks?



Techniques that rely on pre-processing cannot deal efficiently with updates

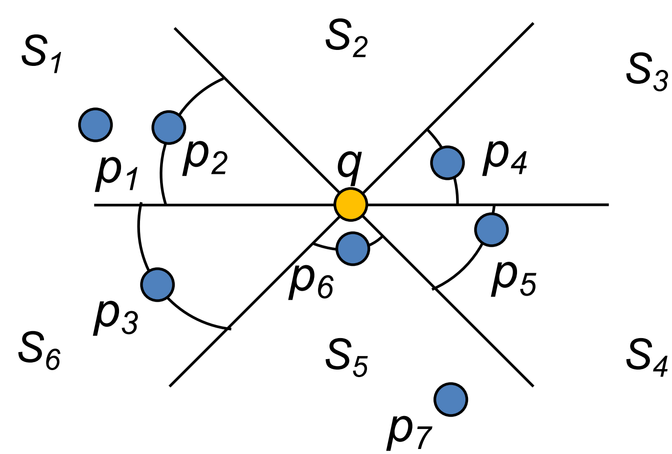
USC Viterbi School of Engineering

SAA

- Elimination of the need for pre-computing all NNs in filter/ refinement methods
- SAA:
  - Divide the space around query into six equal regions
  - Find NN(q) in all regions (candidate keys) (prove by contradiction: p1 rNN(q) but p2 not!)
  - Either (i) or (ii) holds for each candidate key p
    - (i) p is in RNN(q)
    - (ii) No RNN(q) in S_i
  - RNN(q)= {p_6}

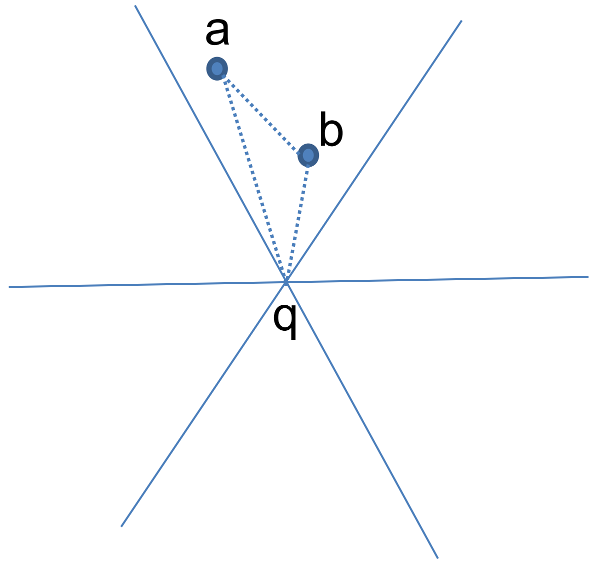
Filter

Refine



- Any Drawbacks?

The number of regions increases exponentially with the dimensionality



Since the angle between $edge_{qa}$ and $edge_{qb}$ is smaller than 60 degree, then the $edge_{ab}$ is NOT the largest edges in the triangle of abq.

Further, since NN(q) = b, i.e., $edge_{qb} < edge_{qa}$. Thus, $edge_{qb}$ is also NOT the largest edges in the triangle abq.

Therefore, $edge_{qa}$ should be the largest edge! And thus, NN(a) is definitely NOT q.

SFT

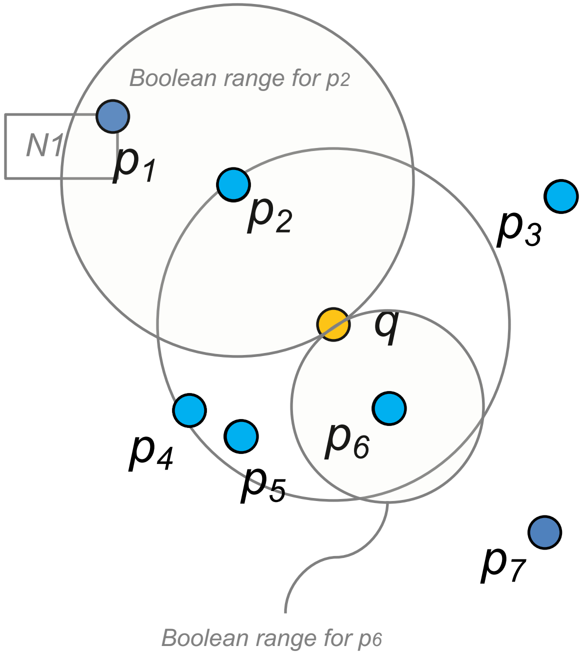
Filter

Refine

1. Find the *k*NNs of the query *q* (*k* candidates)
2. Eliminate the points that are closer to other candidates than *q*.
3. Apply *Boolean range queries* to determine the actual RNNs

- A Boolean range query terminates as the first data point is found

- Drawbacks?



False misses
Choosing a proper *k*

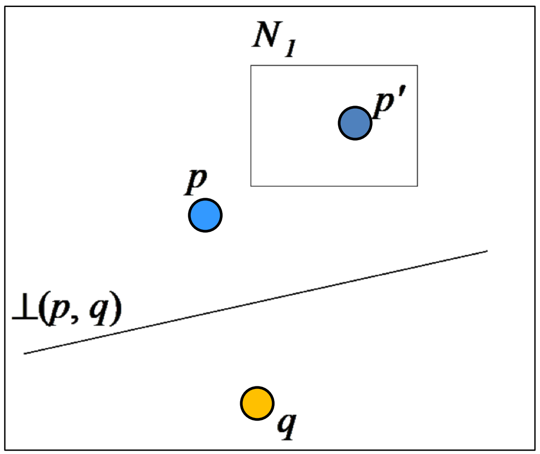
- Concluding former methods:

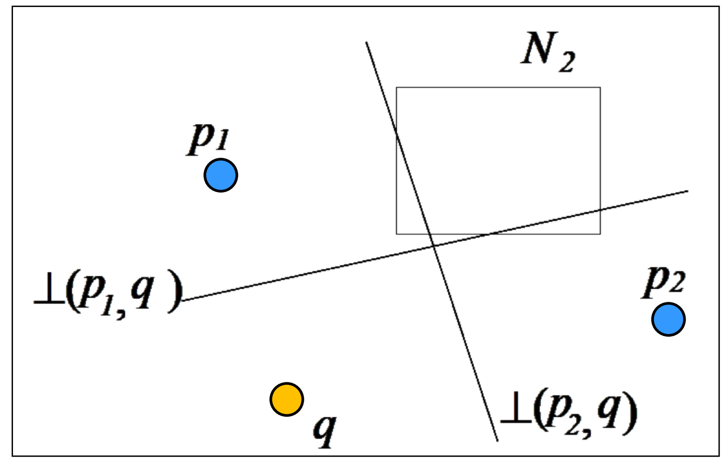
	Dynamic data	Arbitrary dimensionality	Exact result
KM, YL	No	Yes	Yes
SAA	Yes	No	Yes
SFT	Yes	Yes	No

- SAA is good for 2d geographical applications → we'll use it later
- This paper: how to come up with a "pruning" idea to avoid opening useless boxes in r-tree for RNN?

Half-plane pruning

- Draw perpendicular bisector of pq
- Can *p′* be an RNN of *q*?

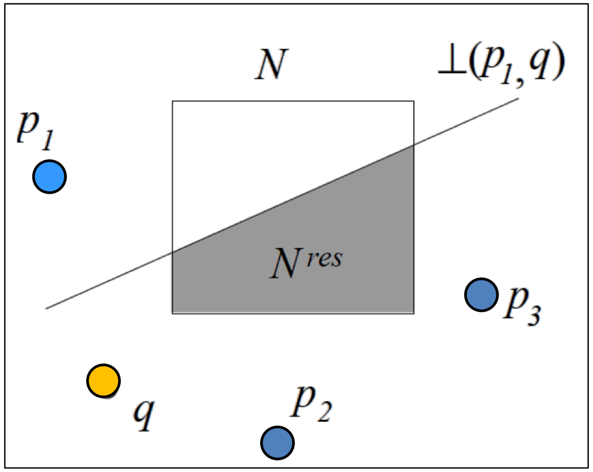


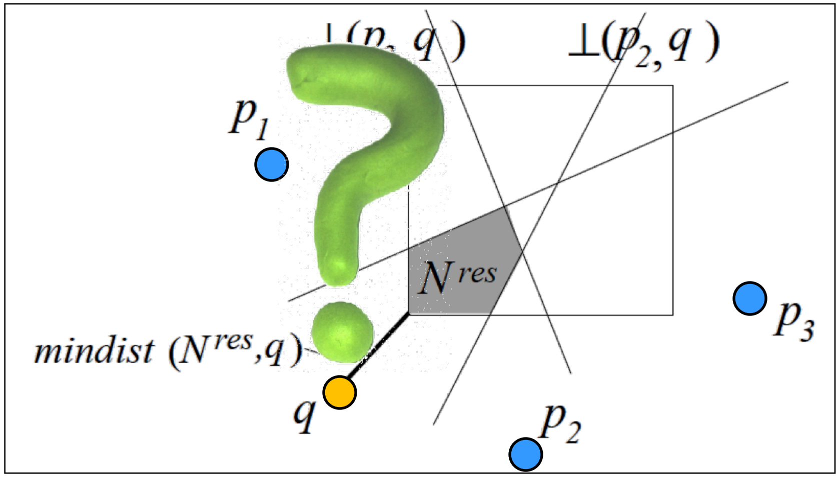


- If $p_1, p_2, \ldots, p_n$ are *n* data points, then any node whose MBR falls inside $\cup_{i=1..n} \perp(p_i, q)$ cannot contain any RNN result.

- E.g., points inside N2 would have either p1 or p2 as their NN, hence they are not RNN of q

- Pruning an R-tree MBR:



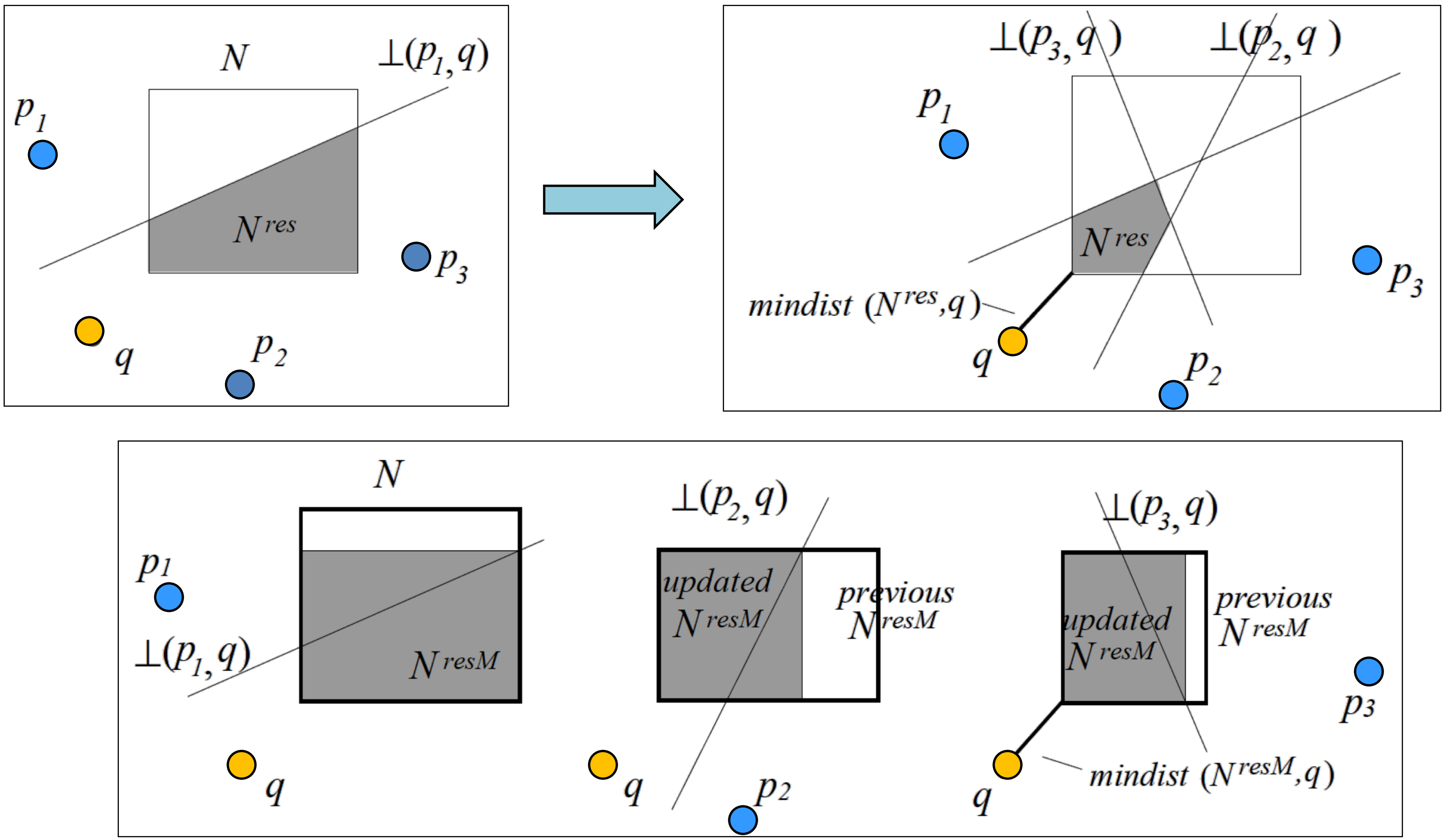


- Drawbacks?

$O(n^2)$ processing time in terms of bisector trimming for computing N^{res}

Computation of intersections does not scale with dimensionality

- Approximating the residual MBR



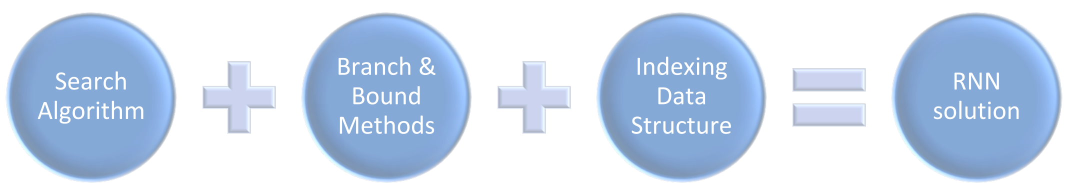
USC Viterbi School of Engineering

- An MBR can be pruned if its residual region is empty
- The approximation is a superset of the real residual region
- We can prune an MBR if its approximate residual is empty
- Good news:

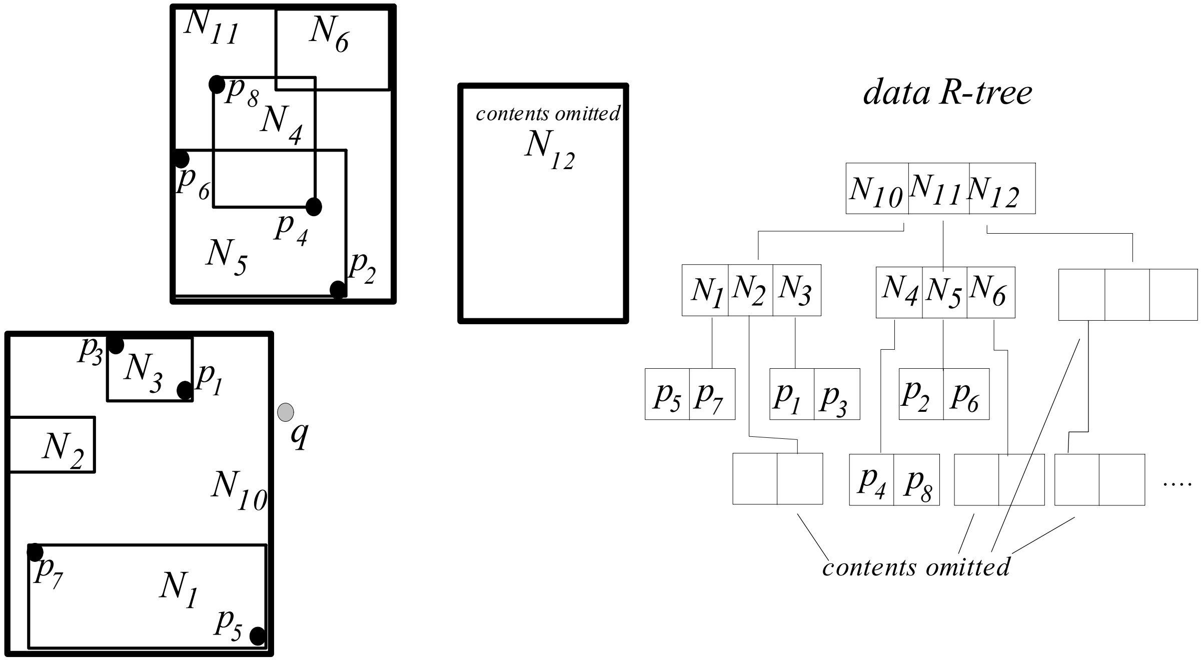
$O(n)$ processing time for computing N^{resM}
No more hyper-polyhedrons to make the intersection computation complex

TPL Algorithm

- The big picture
  - Uses best-first search
  - Utilizes one R-tree as the data structure
  - Includes filtering/ refinement phases
  - Uses candidate points to prune entries
  - Filters visited entries to obtain the set S_{cnd} of candidates
  - Adds pruned entries to set S_{rfn}
  - S_{rfn} is used in the refinement step to eliminate false hits

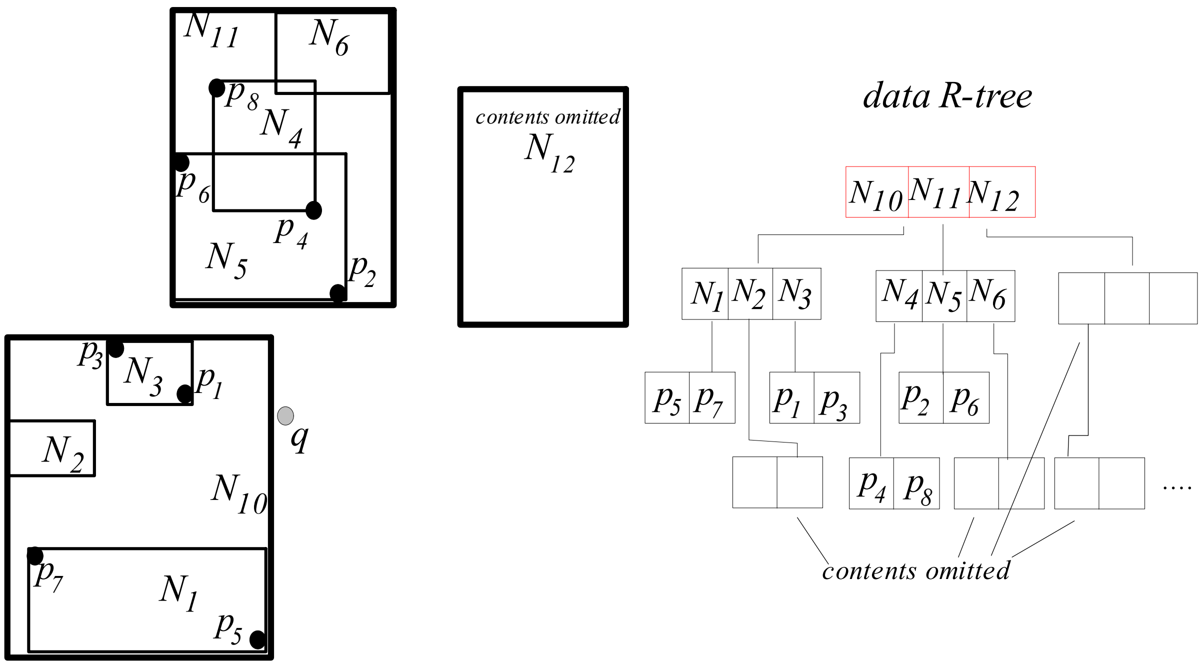


TPL Example



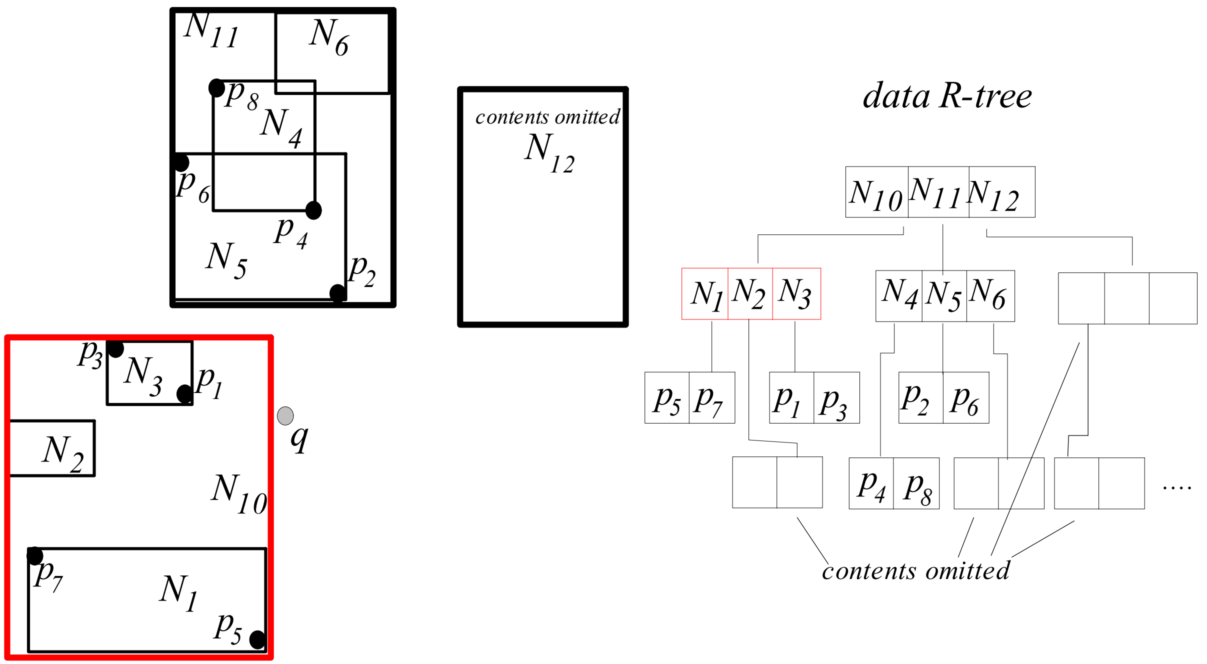
- * Figures of this example are obtained from [2]

Filtering step



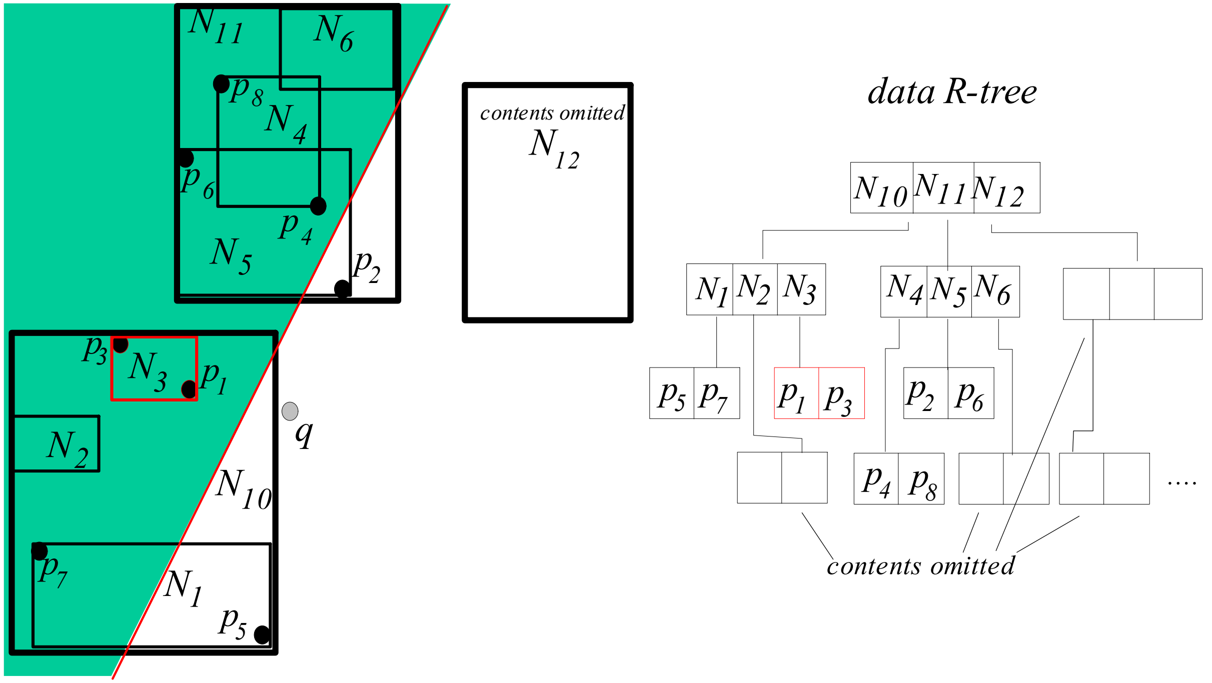
Action	Heap	S_{cnd}	S_{rfn}
Visit root	$\{N_{10}, N_{11}, N_{12}\}$	{}	{}

USC Viterbi School of Engineering



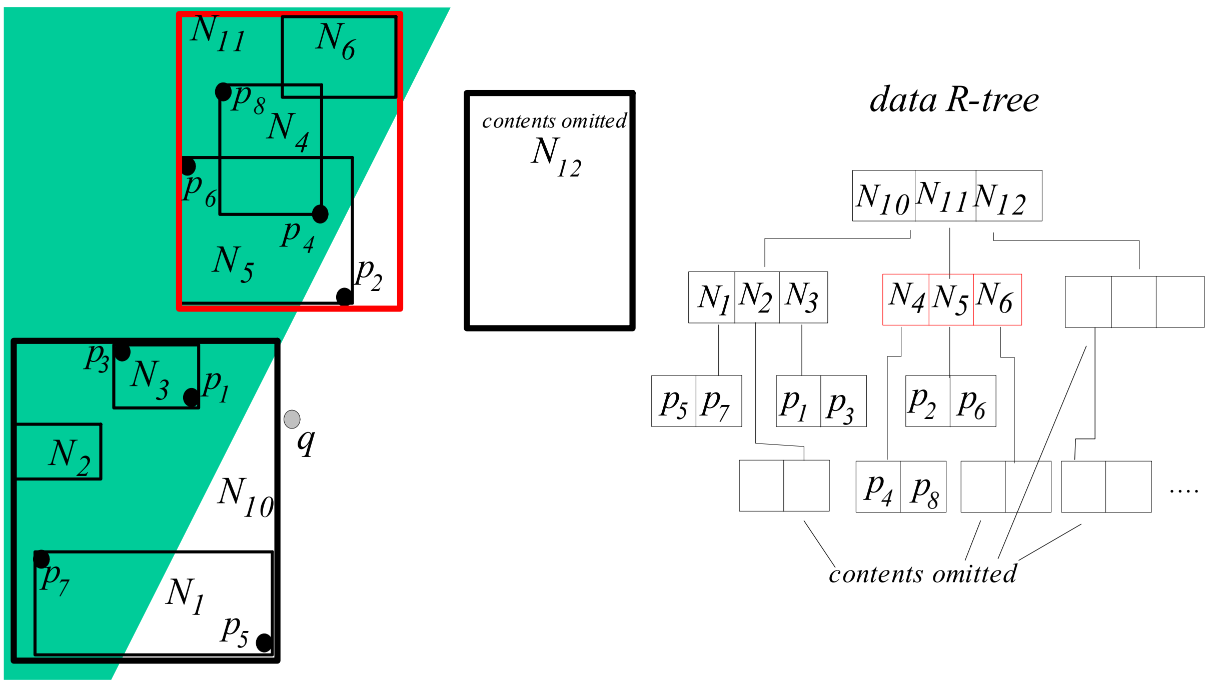
Action	Heap	S_{cnd}	S_{rfn}
Visit N_{10}	$\{N_3, N_{11}, N_2, N_1, N_{12}\}$	{}	{}

USC Viterbi School of Engineering



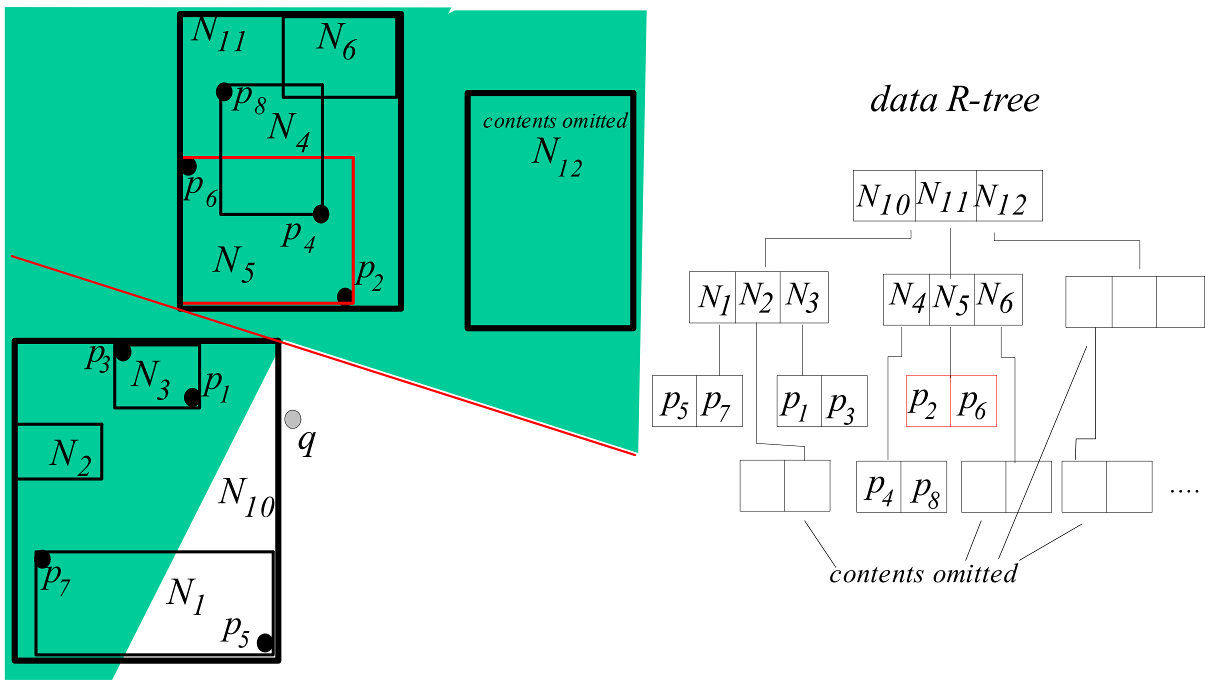
Action	Heap	S_{cnd}	S_{rfn}
Visit N_3	$\{N_{11}, N_2, N_1, N_{12}\}$	$\{p_1\}$	$\{p_3\}$

USC Viterbi School of Engineering

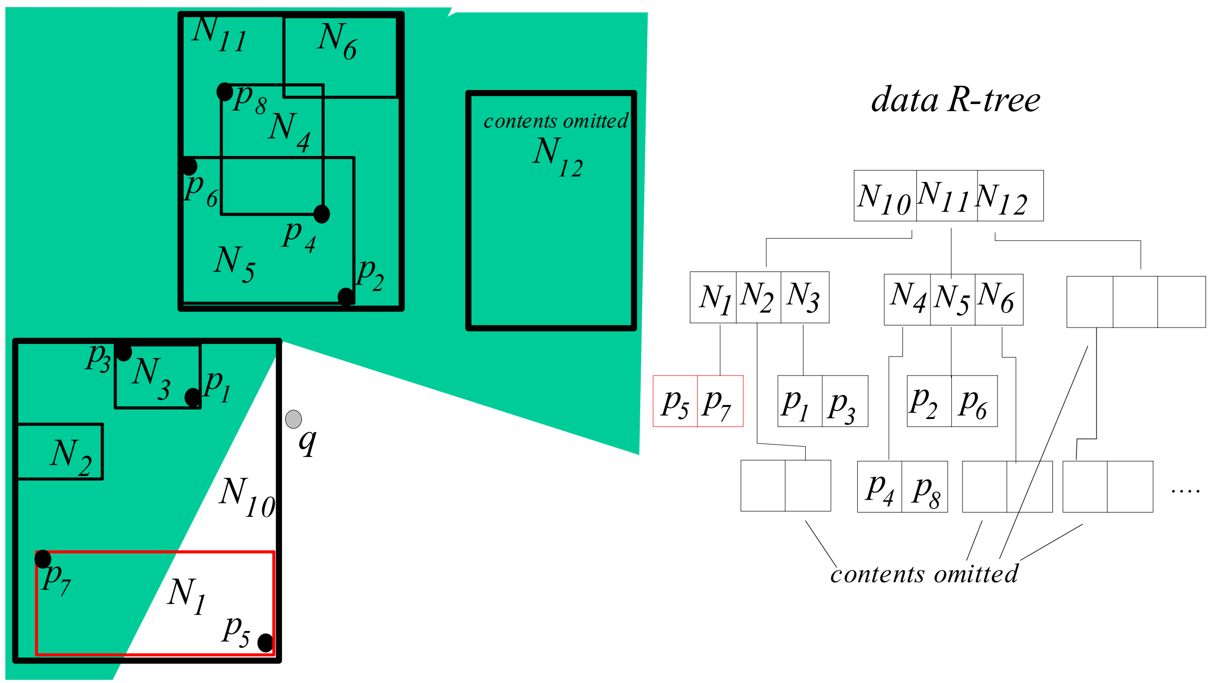


Action	Heap	S_{cnd}	S_{rfn}
Visit N_{11}	$\{N_5, N_2, N_1, N_{12}\}$	$\{p_1\}$	$\{p_3, N_4, N_6\}$

USC Viterbi School of Engineering

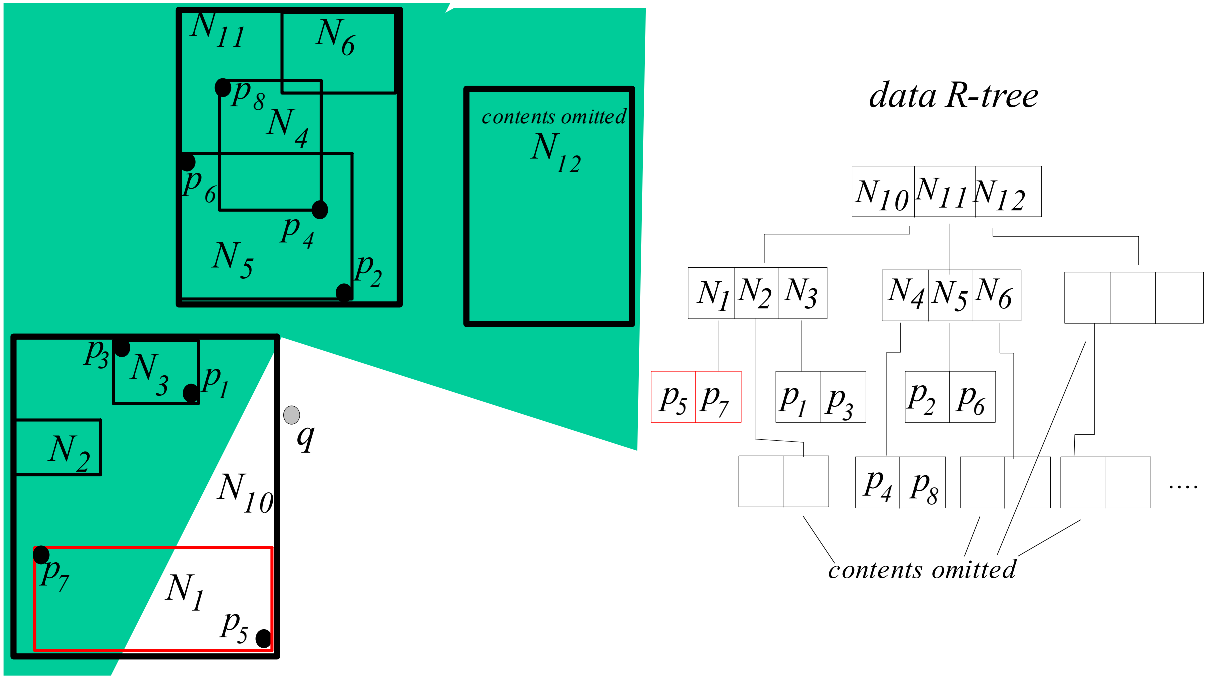


Action	Heap	S_{cnd}	S_{rfn}
Visit N_5	$\{N_2, N_1, N_{12}\}$	$\{p_1, p_2\}$	$\{p_3, N_4, N_6, p_6\}$



Action	Heap	S_{cnd}	S_{rfn}
Visit N_1	$\{N_{12}\}$	$\{p_1, p_2, p_5\}$	$\{p_3, N_4, N_6, p_6, N_2, p_7\}$

USC Viterbi School of Engineering



Action	Heap	S_{cnd}	S_{rfn}
	{}	$\{p_1, p_2, p_5\}$	$\{p_3, N_4, N_6, p_6, N_2, p_7, N_{12}\}$

USC Viterbi School of Engineering

Refinement Heuristics

- Let P_{rfn} be the set of points and N_{rfn} be the set of nodes in S_{rfn}
- A candidate point can be eliminated if it is closer to another candidate point than to the query
- A point p from S_{cnd} can be discarded as a false hit if either of the following hold:

  (i) there is a point $p' \in P_{rfn}$ such that $dist(p, p') < dist(p, q)$

  (ii) There is a node MBR $N \in N_{rfn}$ such that $minmaxdist(p, N) < dist(p, q)$

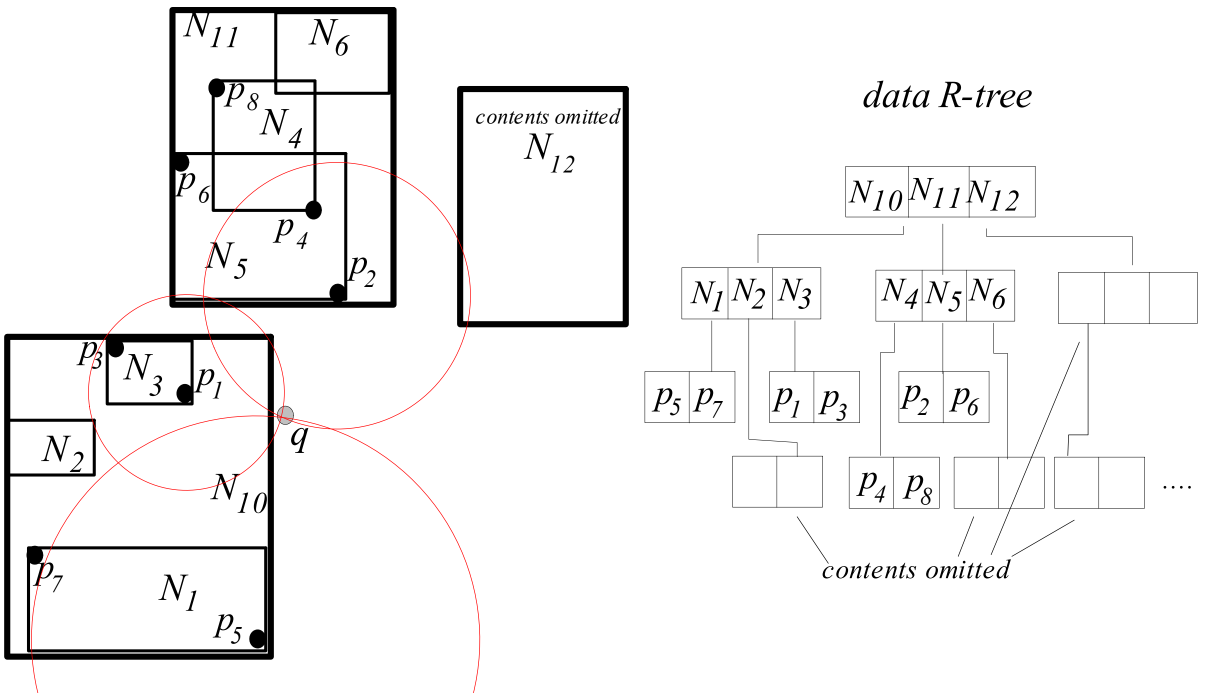
- A point p from S_{cnd} can be reported as an actual result if the following conditions hold:

  (i) There is no point $p' \in P_{rfn}$ such that $dist(p, p') < dist(p, q)$

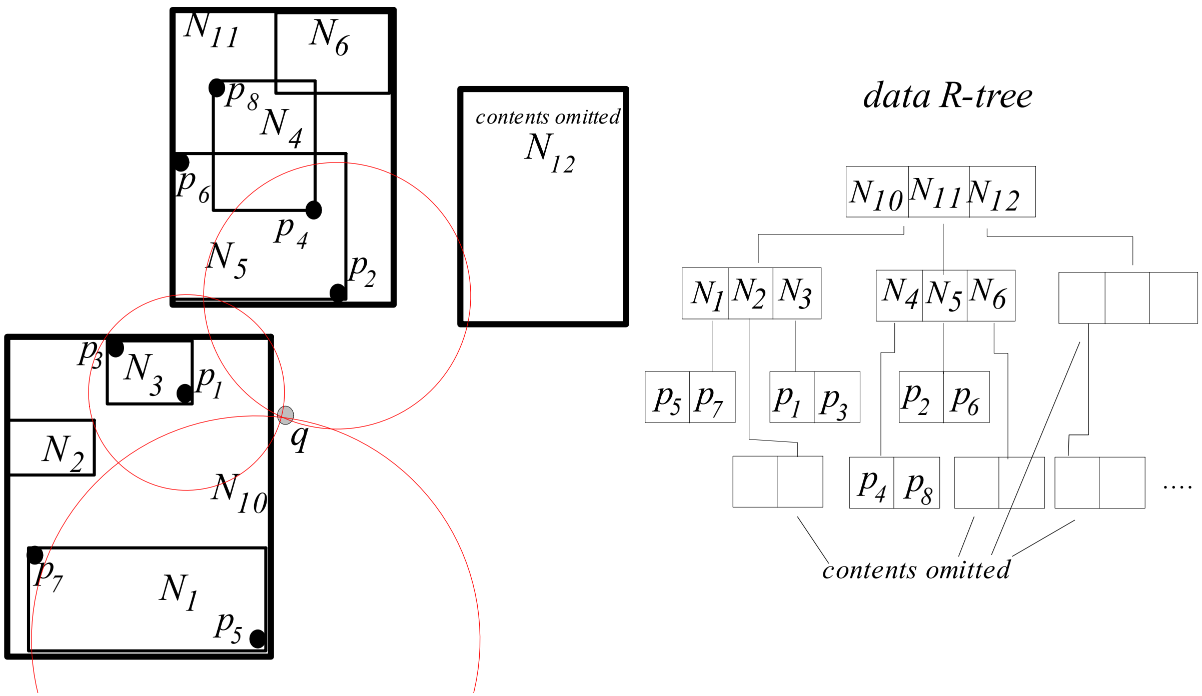
  (ii) For every node $N \in N_{rfn} : mindist(p, N) \geq dist(p, q)$

- If none of the above works, visit all node MBRs $N \in N_{rfn}$ where $mindist(p, N) < dist(p, q)$ and use the mentioned heuristics considering the newly visited entries

USC Viterbi School of Engineering



Action	Scnd	Srfn	Actual results
	$\{p_1, p_2, p_5\}$	$\{p_3, N_4, N_6, p_6, N_2, p_7, N_{12}\}$	{}
Invalidate p_1	$\{p_2, p_5\}$	$\{N_4, N_6, N_2, N_{12}\}$	{}
Validate p_5	$\{p_2\}$	$\{N_4, N_6, N_2, N_{12}\}$	$\{p_5\}$
Remove N_6, N_2	$\{p_2\}$	$\{N_4, N_{12}\}$	$\{p_5\}$

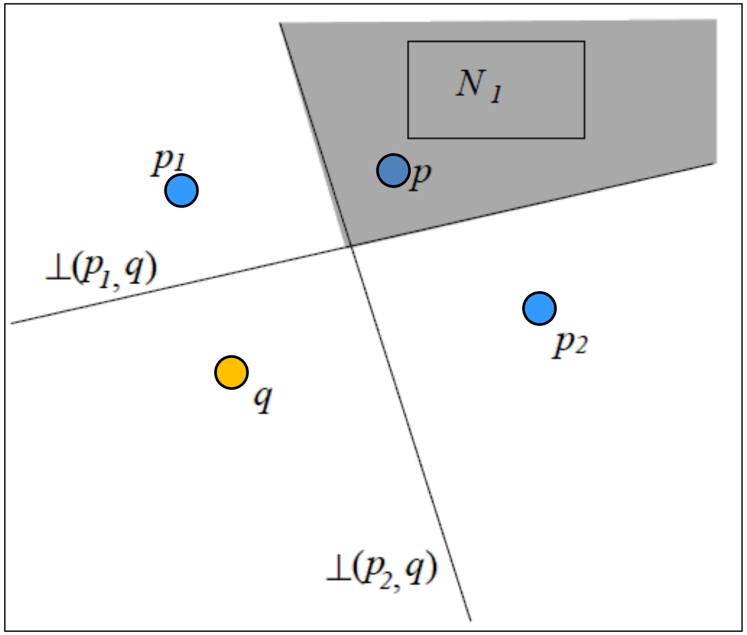


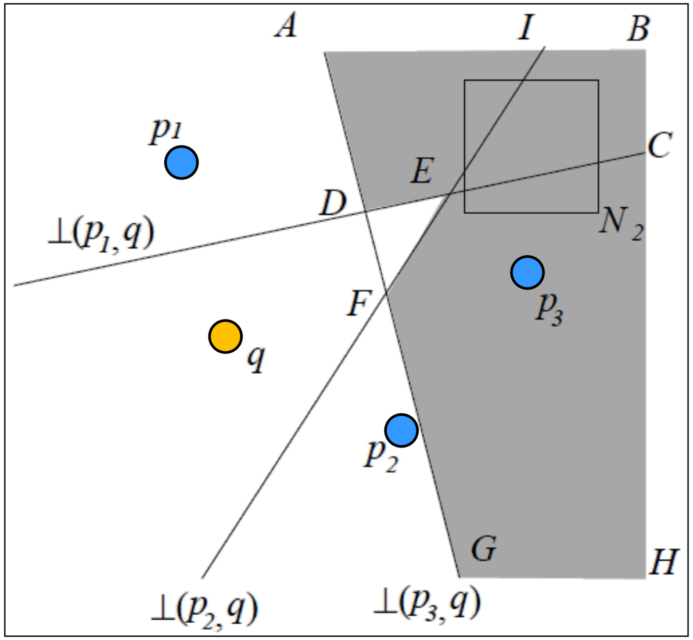
Action	*Scnd*	*Srfn*	Actual results
	$\{p_2\}$	$\{N_4, N_{12}\}$	$\{p_5\}$
Access N_4	$\{p_2\}$	$\{p_4, p_8, N_{12}\}$	$\{p_5\}$
Invalidate p_2	$\{\}$	$\{N_{12}\}$	$\{p_5\}$

USC Viterbi School of Engineering

RkNN pruning

- Return all points that have *q* as one of their *k* nearest neighbors





- Let $\{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ be a subset of $\{p_1, p_2, \ldots, p_n\}$. Each of the $\binom{n}{k}$ subsets, prunes the area $\cap_{i=1\sim k} \mathrm{PL}\sigma_i(\sigma_i, q)$

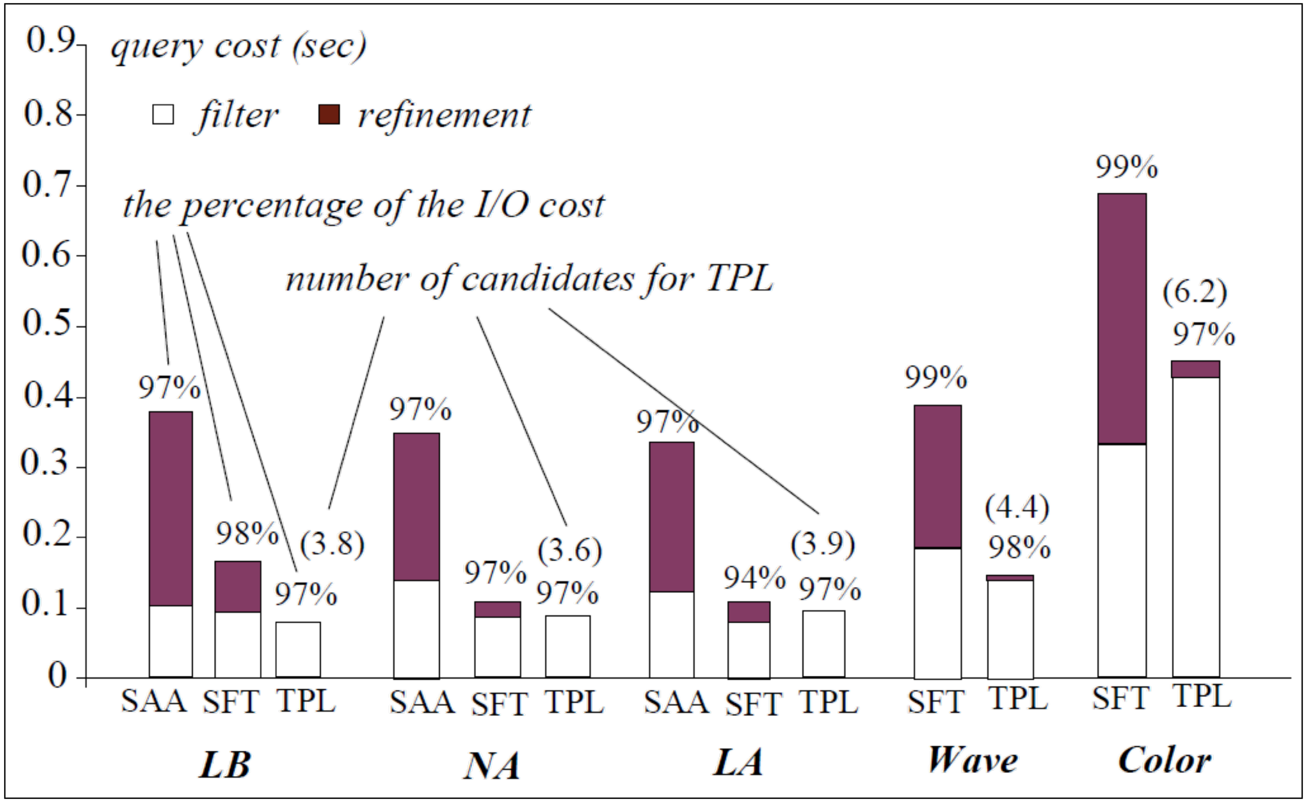
kTPL Algorithm

- Same filtering as TPL
- Same refining with the following exceptions:
  - A point can be pruned if *k* points are found within distance *dist(p,q)* from *p*
  - A counter is associated with each point (initialized to k) and decreases when such a point is found
  - A candidate is eliminated if counter= 0
  - No prior knowledge of number of points in a node, so no application of $minmaxdist(p,N) < dist(p,q)$ in pruning
  - A point *p* can be pruned if a node *N* is found such that

$$maxdist(p,N) < dist(p,q) \text{ and } min\_card(N) \geq \text{counter}(p)$$

Experiments

- RNN queries on real data



Conclusion

- TPL is good in that it
  - Supports arbitrary values of k
    - KM
  - Can deal efficiently with database updates
    - KM
  - Is applicable to data of dimensionality more than two
    - SAA
  - Retrieves exact results
    - SFT
  - Results in fast results!

References

1. **"Reverse *k*NN Search in Arbitrary Dimensionality". Y. Tao, D. Papadias, X. Lian.**
2. A presentation by Jalal Kazemitabar in csci587 Fall'2010