Spatial Keyword Range Search on Trajectories

Yuxing Han[1], Liping Wang[1], Ying Zhang[2], Wenjie Zhang[3], Xuemin Lin[1,3]

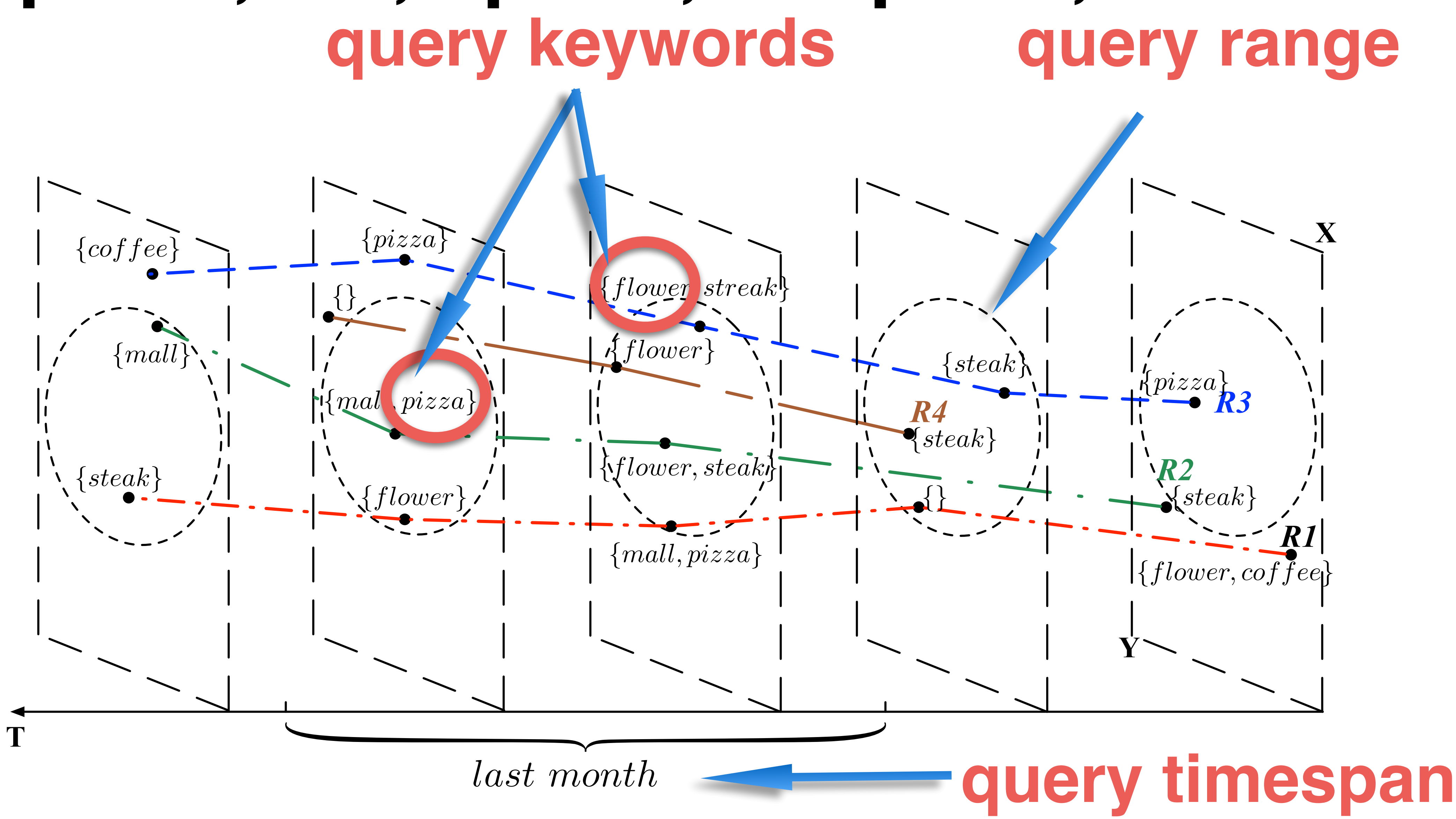
[1]East China Normal University, China
[2]University of Technology, Sydney
[3]The University of New South Wales, Sydney

Big Trajectory Data

- ✦Wireless Sensors with Global Position System
- ✦Driving Route Track Record
- ✦Social Network with Location-based Service

Motivation

It is meaningful to search trajectories based on three aspects, *i.e.*, spatio, temporal, textual.



Related Work

- Activity Trajectory Similarity Query (ATSQ)

[23] Zheng, K., Shang, S., Yuan, N. J., and Yang, Y. Towards efficient search for activity trajectories. In Data Engineering (ICDE), 2013 IEEE 29th International Conference on (2013), IEEE, pp. 230–241.

- Top-*k* Spatial Keyword Query (T*k*SK) on trajectory

[6] Cong, G., Lu, H., Ooi, B. C., Zhang, D., and Zhang, M. Efficient spatial keyword search in trajectory databases. arXiv preprint arXiv:1205.2880 (2012).

Problem Statement

Spatial Keyword Range search on Trajectories (SKRT):

Given a query region, a timespan and a set of keywords, we aim to retrieve trajectories that go through this region during query timespan, and contain *all* the query keywords.

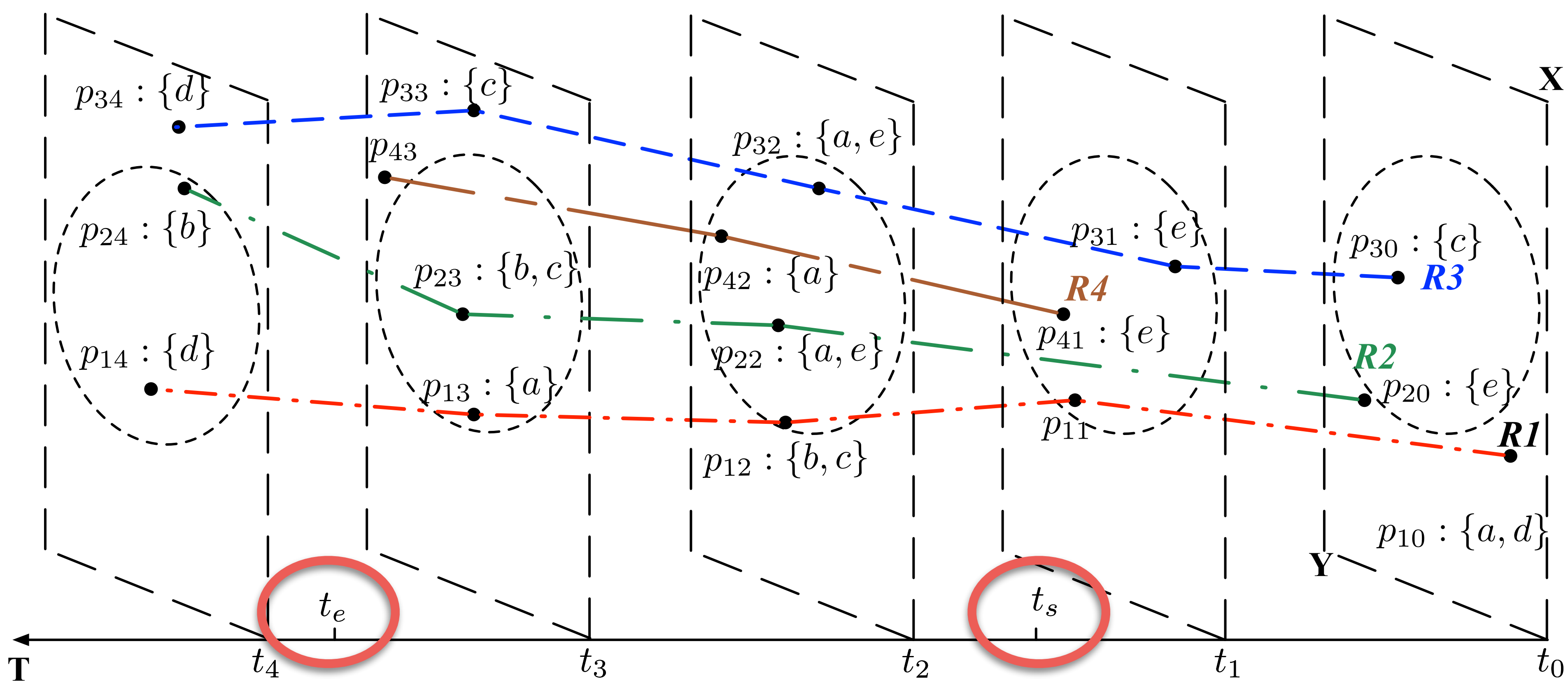
Example

$Q.R$: the space within dotted circle

$Q.T$ = [t_s,t_e]

$Q.\Phi$ = {a,c}

Trajectories Retrieved: R1, R2



Inverted Octree (IOC-Tree)

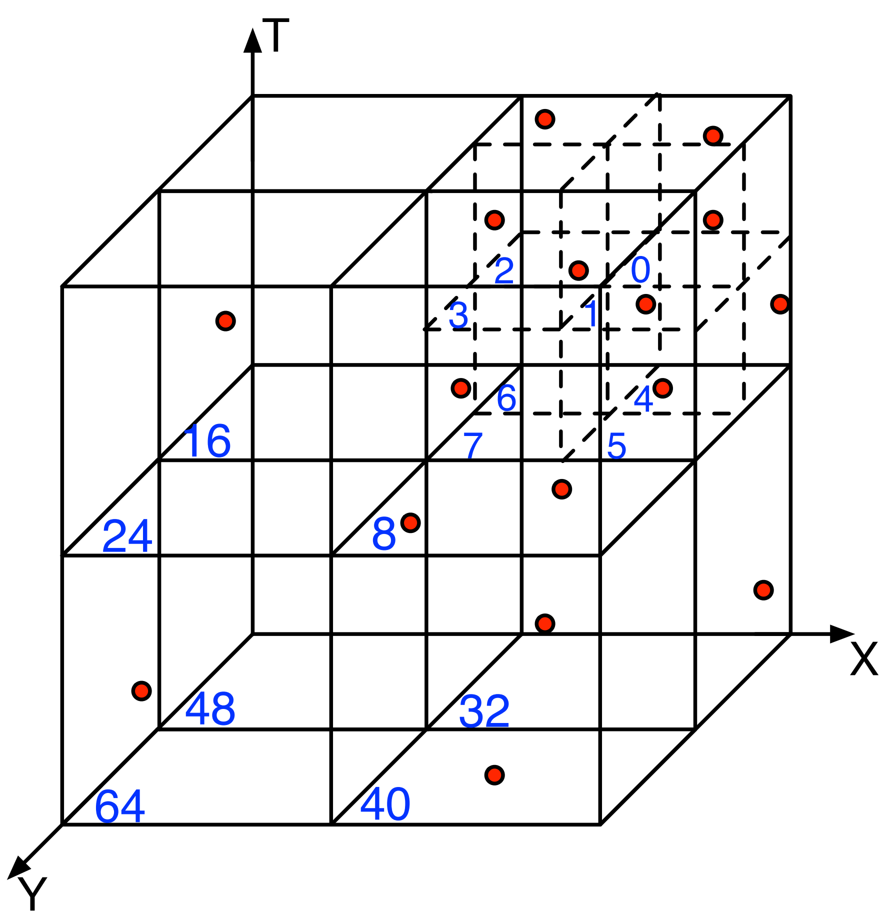
- inverted index
- octree
- morton code
- *signature* technique

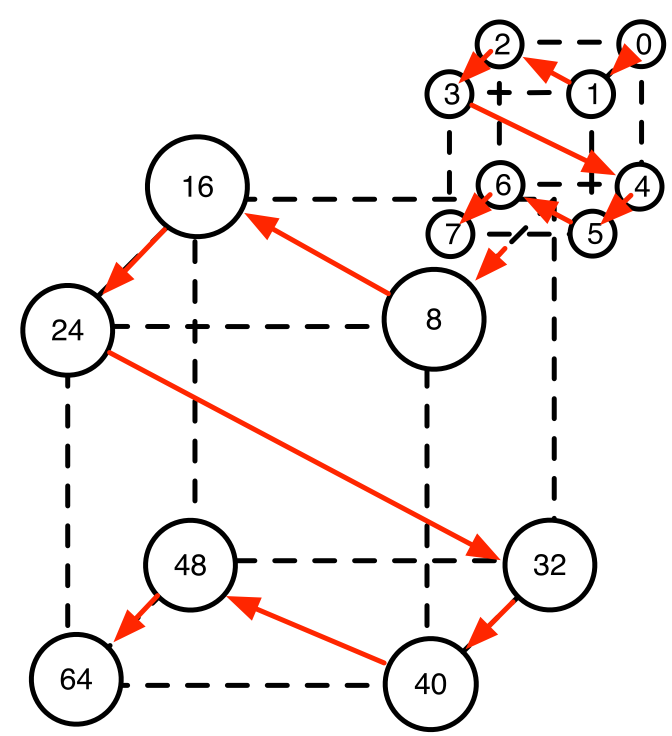
Inverted Index

WHY follow keyword-first-pruning ?

Octree & Morton Code

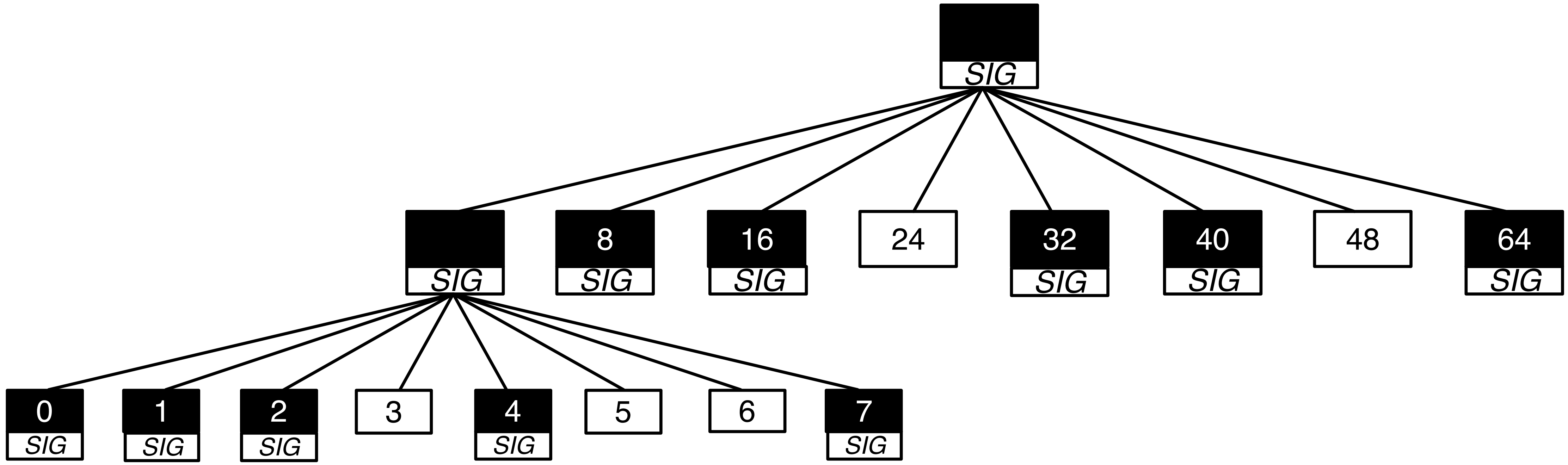
- octree: 3D analog of quadtree
- morton code: two nodes with high spatio-temporal proximity are assigned to the same page in the secondary storage





Signature

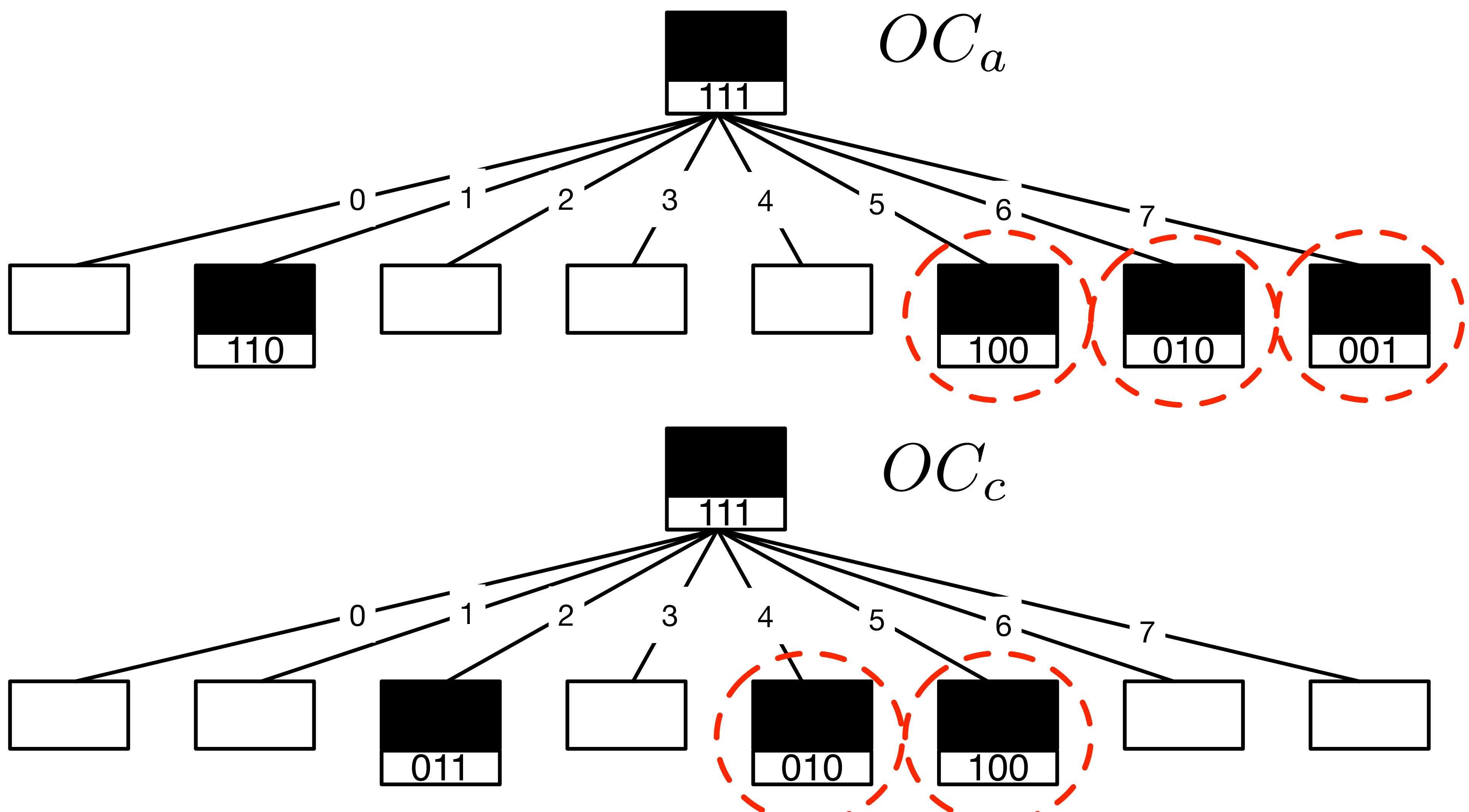
- for each node in octree, a *signature* is maintained to **summarize** the identifications of the trajectories that go through the corresponding spatio-temporal region.



IOC-Tree of Example

Table 2. Distribution of Trajectory Points

Node#	0	1	2	3	4	5	6	7
Points	p_{11}	p_{10} p_{31}	p_{30} p_{41}	p_{20}	p_{23} p_{24} p_{33}	p_{12} p_{13} p_{14}	p_{22} p_{32} p_{34}	p_{42} p_{43}



Algorithm

Main Idea: Prune as many trajectories as possible based on spatio, temporal, textual info by using IOC-Tree

- **Prune & Verification Strategy**

- **Three types of octree nodes:**
  a) ones locate outside the range,
  b) fully-covered nodes,
  c) partially-covered nodes.

Pruning

- Procedure *Prune* prunes non-promising nodes based on IOC-Tree
- *STRangeFilter*: only explore *black* nodes(i.e., non-empty nodes)
- deals with nodes level by level among related octrees

Procedure Prune(Q, L)

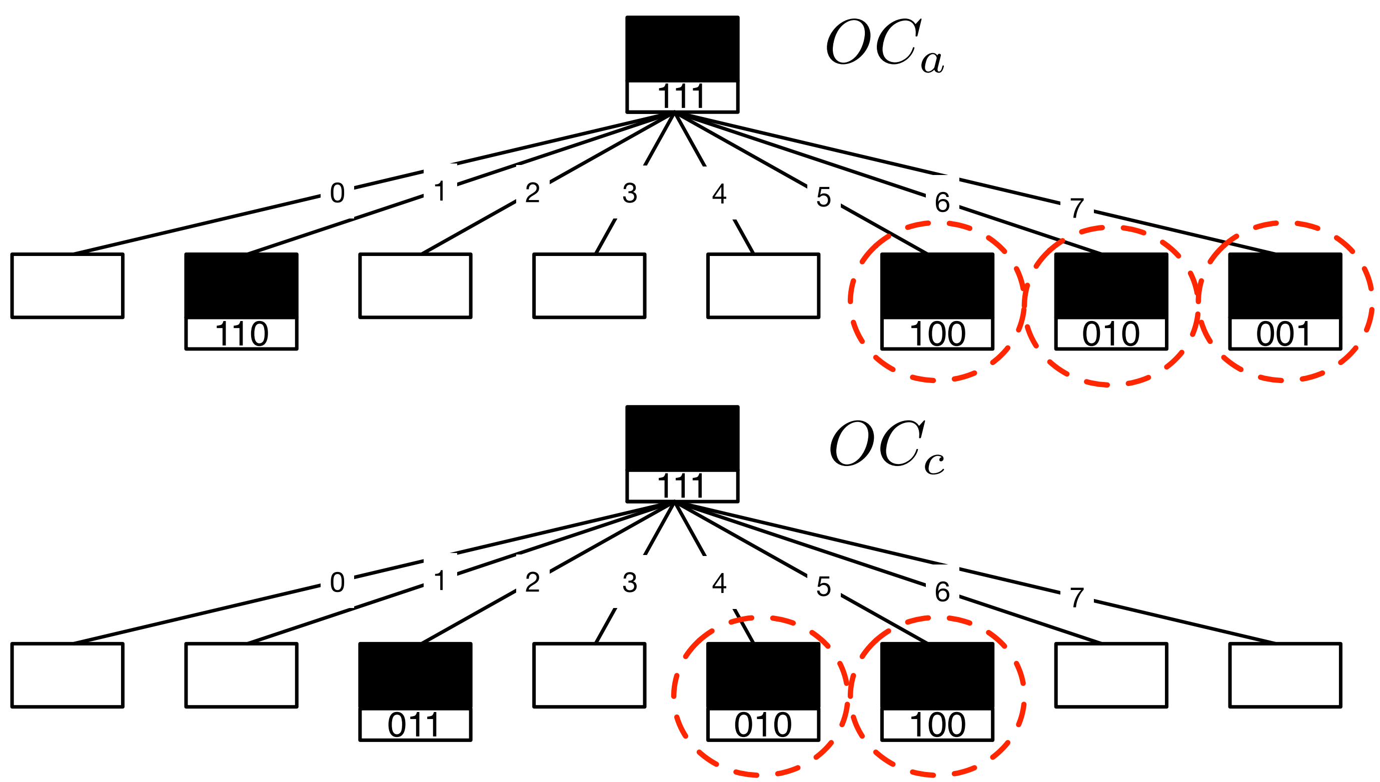
```
while L ≠ ∅ do
    STRangeFilter(Q.R, Q.T, L);
    foreach k_i ∈ Q.δ do
        SIG_i = bitwise-OR of signatures of nodes v ∈ L from OC_i;
    foreach node v ∈ L from OC_i do
        foreach SIG_j where j ≠ i do
            SignatureCheck(v, SIG_j);
    foreach node v ∈ L that survive from the signature test do
        Suppose v comes from OC_j;
        if v is a fully covered leaf node then
            add v into CN_j^f;
        else if v is a partially covered leaf node then
            add v into CN_j^p;
        else if v is non-leaf node then
            foreach child node v' of v do
                if v' is not a white node then
                    put v' into L;
        delete v from L;
foreach k_i ∈ Q.δ do
    determine TR(CN_i^f) and TR(CN_i^p);
```

Verification

- Procedure *Verification* aims at validating candidate trajectories
- Further validation should load corresponding cell on the disk

Procedure Verification(Q, $\mathcal{A}$)

$\mathcal{A} \leftarrow CT^f$;
foreach *trajectory* $Tr \in CT^p$ **do**
  find out a keyword set $\Psi = \{k_i | Tr \in \mathcal{TR}(CN_i^f)\}$;
  $\Omega \leftarrow Q.\Phi - \Psi$;
  foreach $\mathcal{TR}(CN_j^p)$ *where* $k_j \in \Omega$ **do**
    if $Tr \in \mathcal{TR}(CN)_j^p$ *and* ***LoadAndJudge***$(Tr, \mathcal{TR}(CN_j^p))$ **then**
      $\mathcal{A} \leftarrow \mathcal{A} \cup Tr$;



Extension

- Spatial Keyword Range search on Trajectories with Order-sensitive keywords (SKRTO)

Experiment

- ✦ Implemented in C++
- ✦ Windows 7
    - · Intel i5 CPU(3.10GHz)
    - · 8 GB main memory
- ✦ Three Real Trajectory Datasets:
    - a) two Foursquare check-in datasets from LA and NY
    - b) one geo-tagged tweets dataset
- ✦ Two Baselines:
    - a) GAT (ICDE'13) b) B^{ck}-tree (arxiv'12)

Dataset Statistics

Table 3. Dataset Statistics

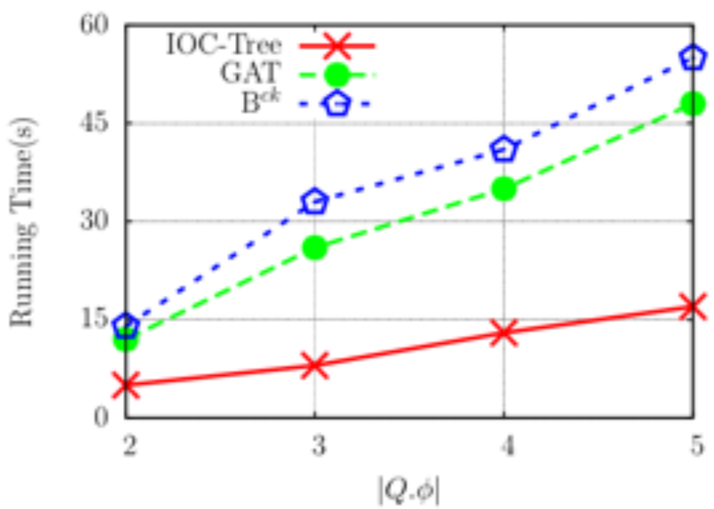
	LA	**NY**	**TW**
#trajectory	31,553	49,022	214,834
#location	215,614	206,416	1,287,315
#tag	3,175,597	3,068,401	28,645,905
#distinct-tag	100,843	89,665	1,852,141

Experimental Settings

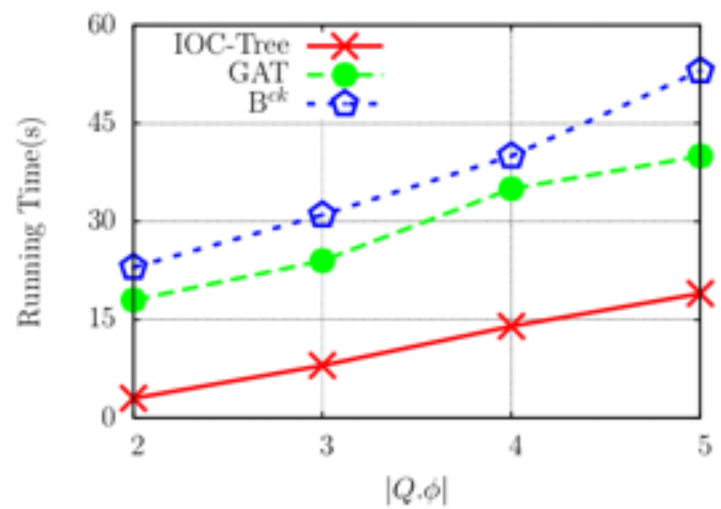
Table 4. Experimental Settings

	$\|\boldsymbol{Q.\phi}\|$	$\|\boldsymbol{Q.T}\|$ (month)	$\boldsymbol{\delta(Q.R)}$(km)
LA,NYC	2, $\underline{3}$, 4, 5	1, 2, $\underline{3}$, 4, 5	10, 20, $\underline{30}$, 40, 50
TW	2, $\underline{3}$, 4 ,5	0.5, 1, $\underline{1.5}$, 2, 2.5	5, 10, $\underline{15}$, 20, 25

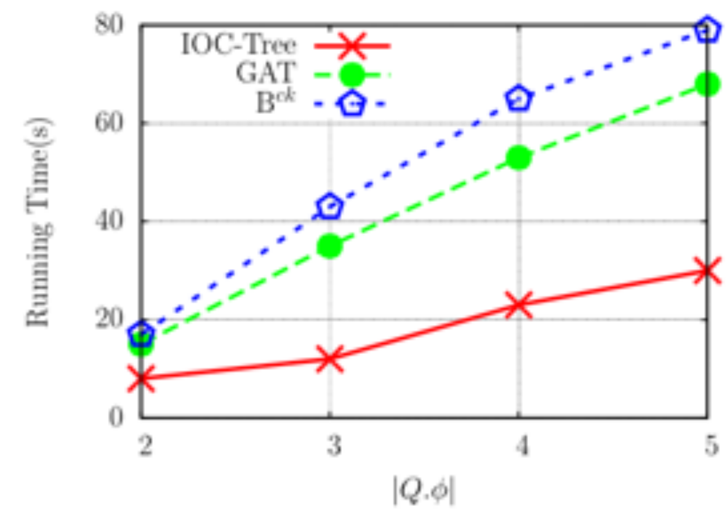
Varying number of query keywords



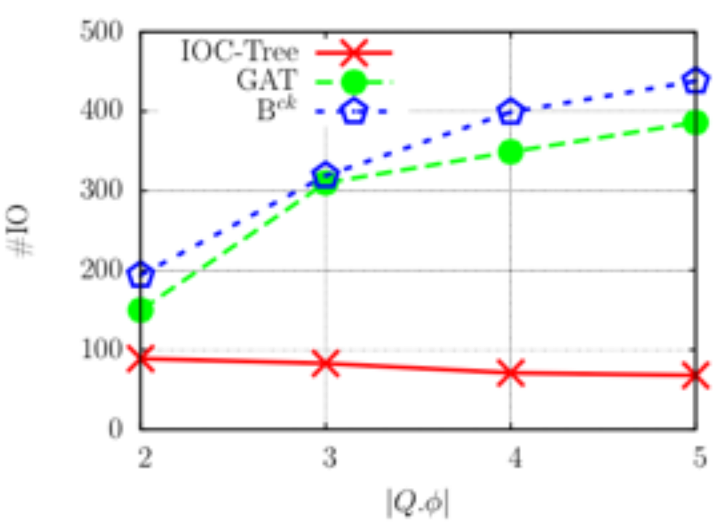
(a) $SKRT$ on LA



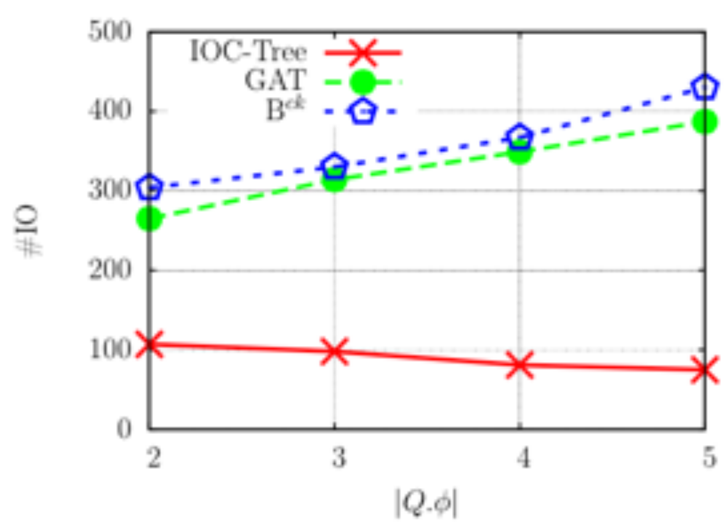
(b) $SKRT$ on NYC



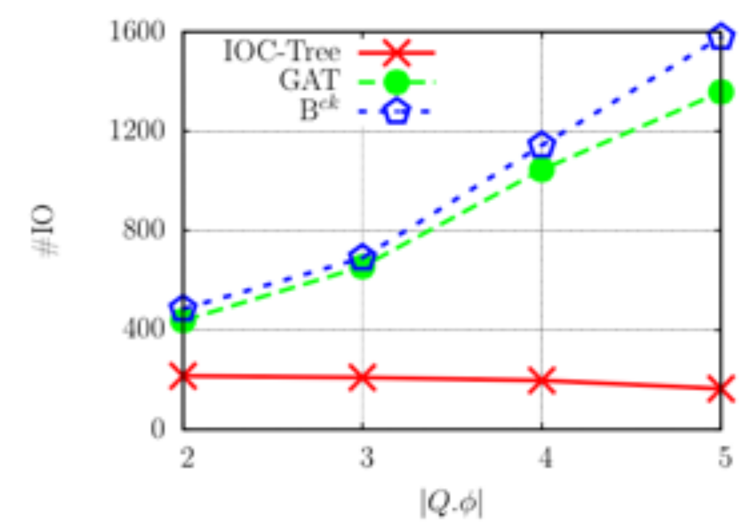
(c) $SKRT$ on TW



(a) $SKRT$ on LA

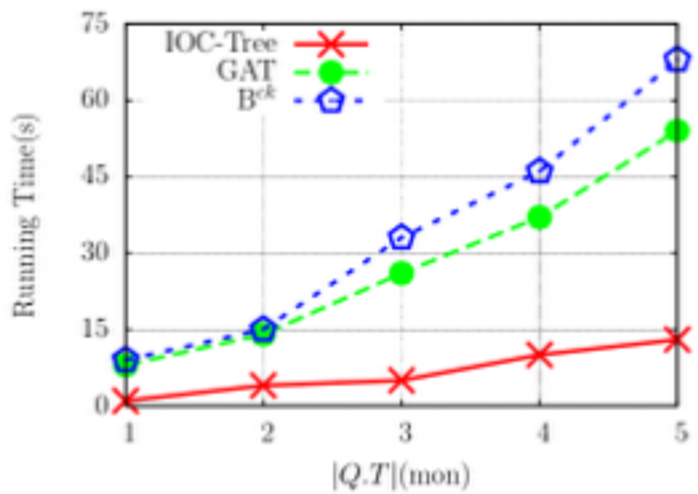


(b) $SKRT$ on NYC

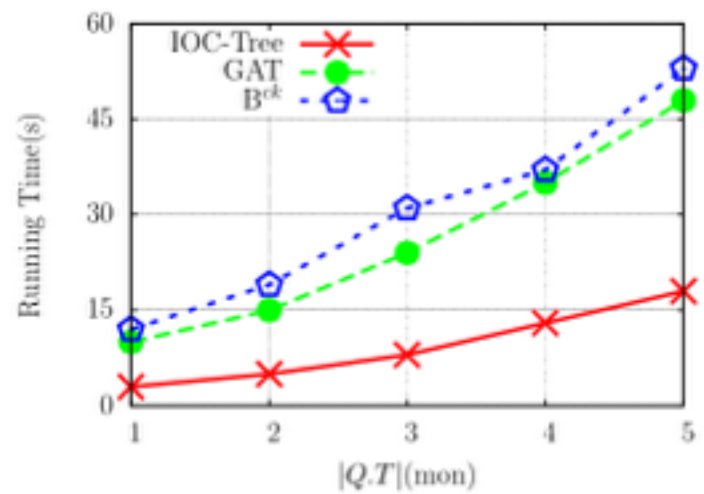


(c) $SKRT$ on TW

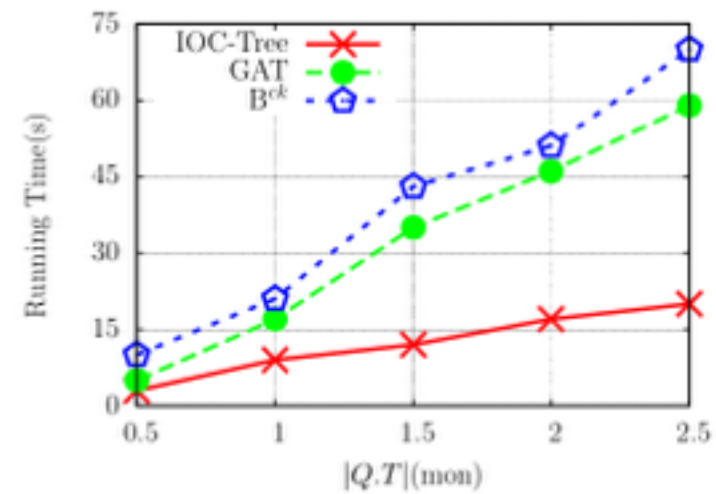
Varying query timespan



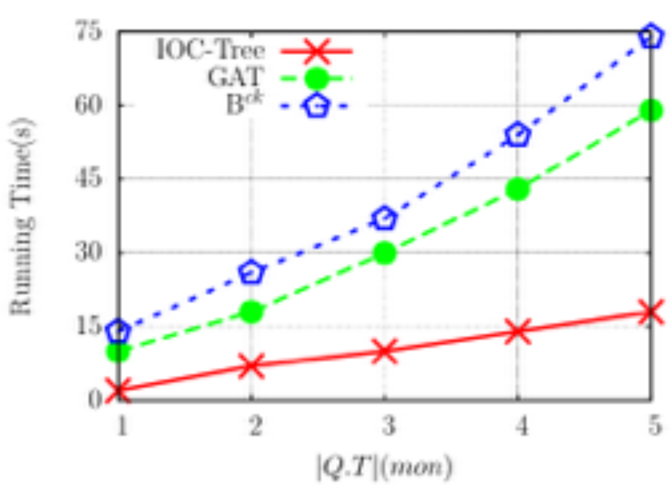
(a) $SKRT$ on LA



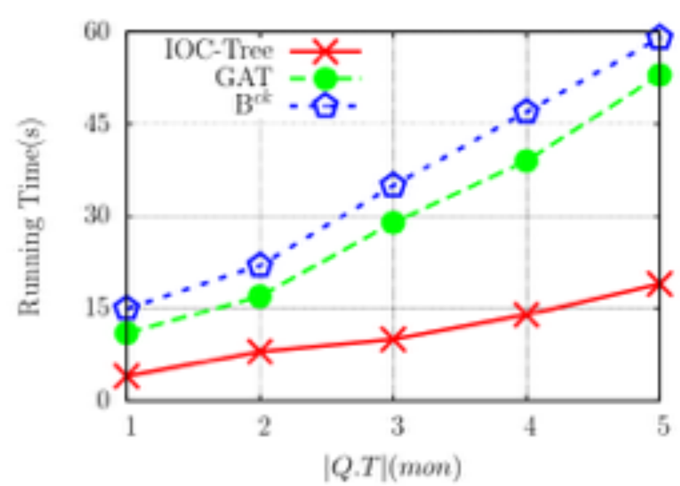
(b) $SKRT$ on NYC



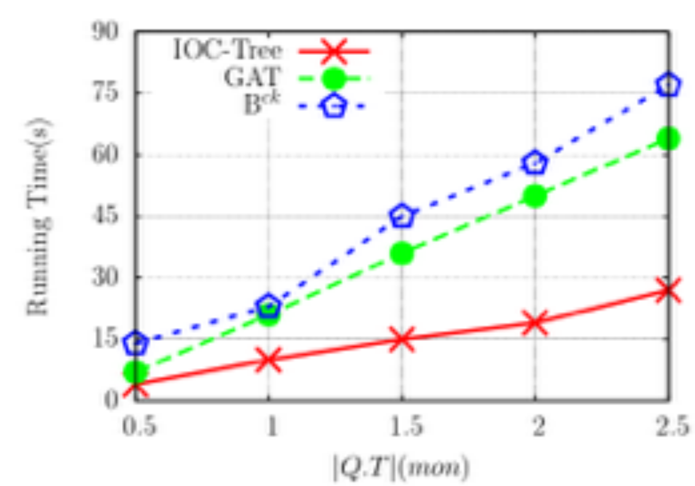
(c) $SKRT$ on TW



(a) $SKRTO$ on LA

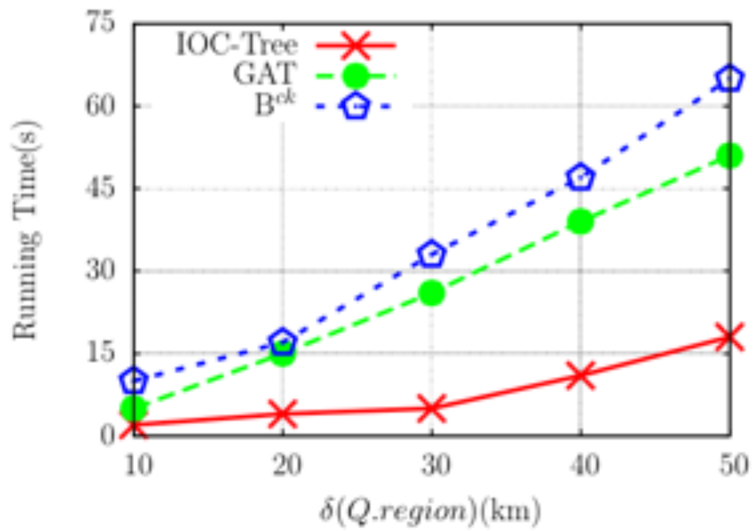


(b) $SKRTO$ on NYC

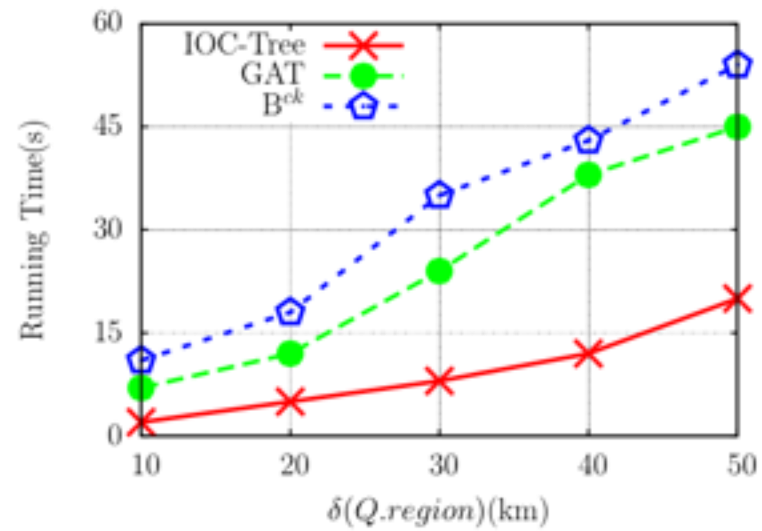


(c) $SKRTO$ on TW

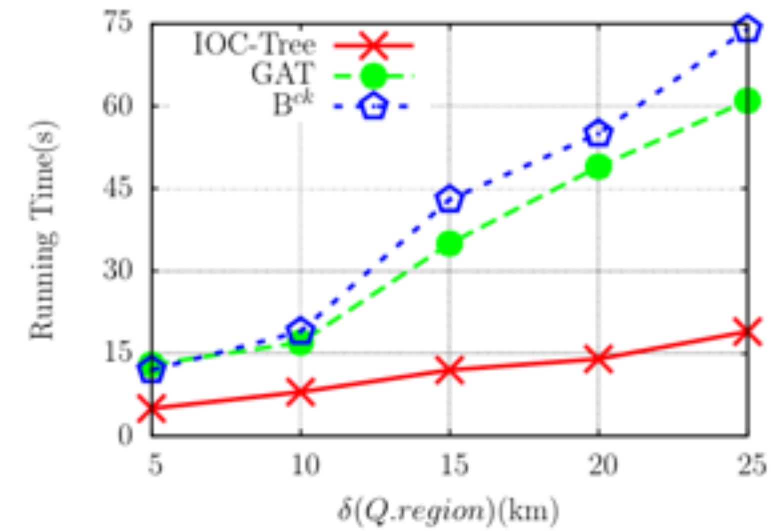
Varying query range



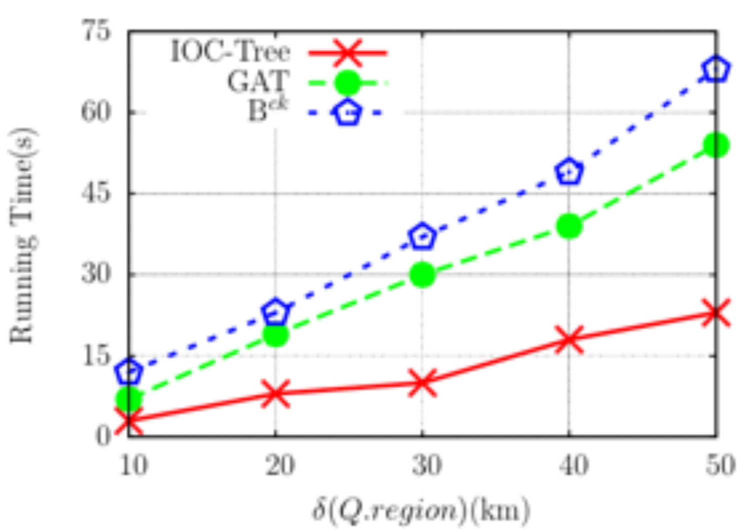
(a) *SKRT* on LA



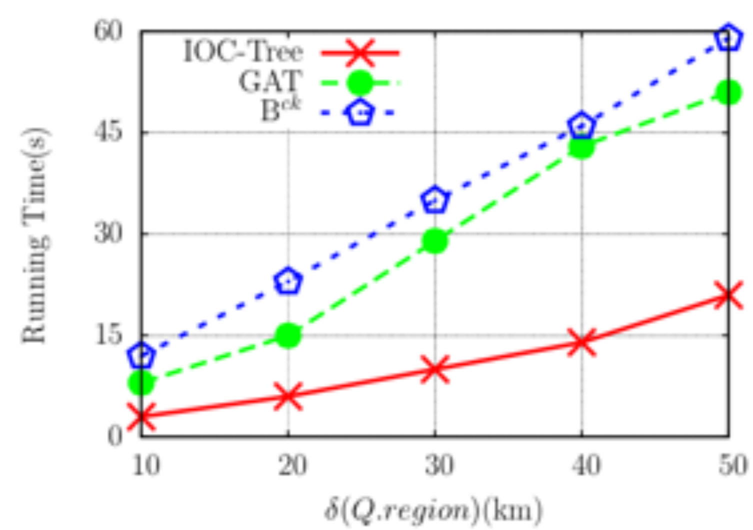
(b) *SKRT* on NYC



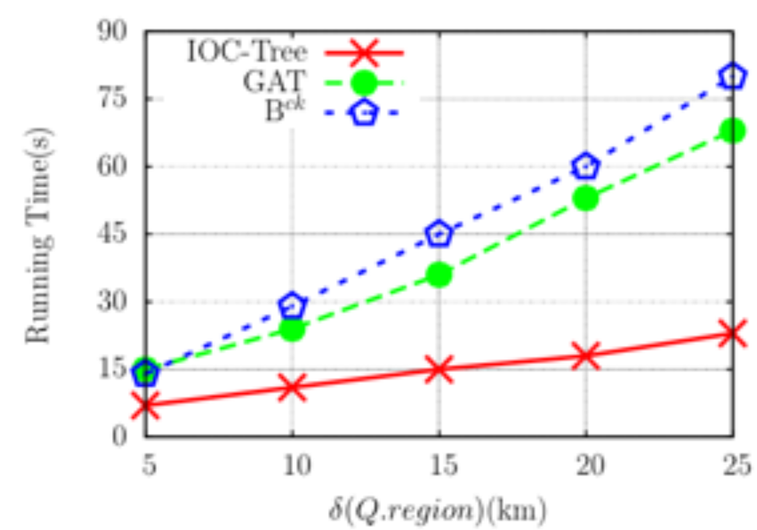
(c) *SKRT* on TW



(a) *SKRTO* on LA



(b) *SKRTO* on NYC



(c) *SKRTO* on TW

Conclusion

- ✦ Propose spatial keyword range search on trajectories
- ✦ Design IOC-Tree structure and query processing algorithm
- ✦ Extensive experiments on real datasets confirm the efficiency of our techniques

Thank you!