Complex Spatio-Temporal Pattern Queries

Marios Hadjieleftheriou, George Kollios
Boston University
{marioh, gkollios}@cs.bu.edu

Petko Bakalov, Vassilis Tsotras
University of California, Riverside
{pbakalov, tsotras}@cs.ucr.edu

Presented by Benjamin Krogh

Department of Computer Science, Aalborg

May 30, 2012

Originally presented at the VLDB conference 2005

Agenda

Motivation

Context

1. Large amount of trajectory data is produced daily
2. Analysis of such data requires development of sophisticated techniques

Motivation

Context

1. Large amount of trajectory data is produced daily
2. Analysis of such data requires development of sophisticated techniques

Example queries we want to answer:

1. Identify all vehicles in proximity of location p_0, at time t_0, and in proximity of location p_1 at time t_1 (STP With Time)
2. Identify all vehicles in proximity of location p_0, and later in proximity of location p_1 (STP With Order)

Motivation

Context

1. Large amount of trajectory data is produced daily
2. Analysis of such data requires development of sophisticated techniques

Example queries we want to answer:

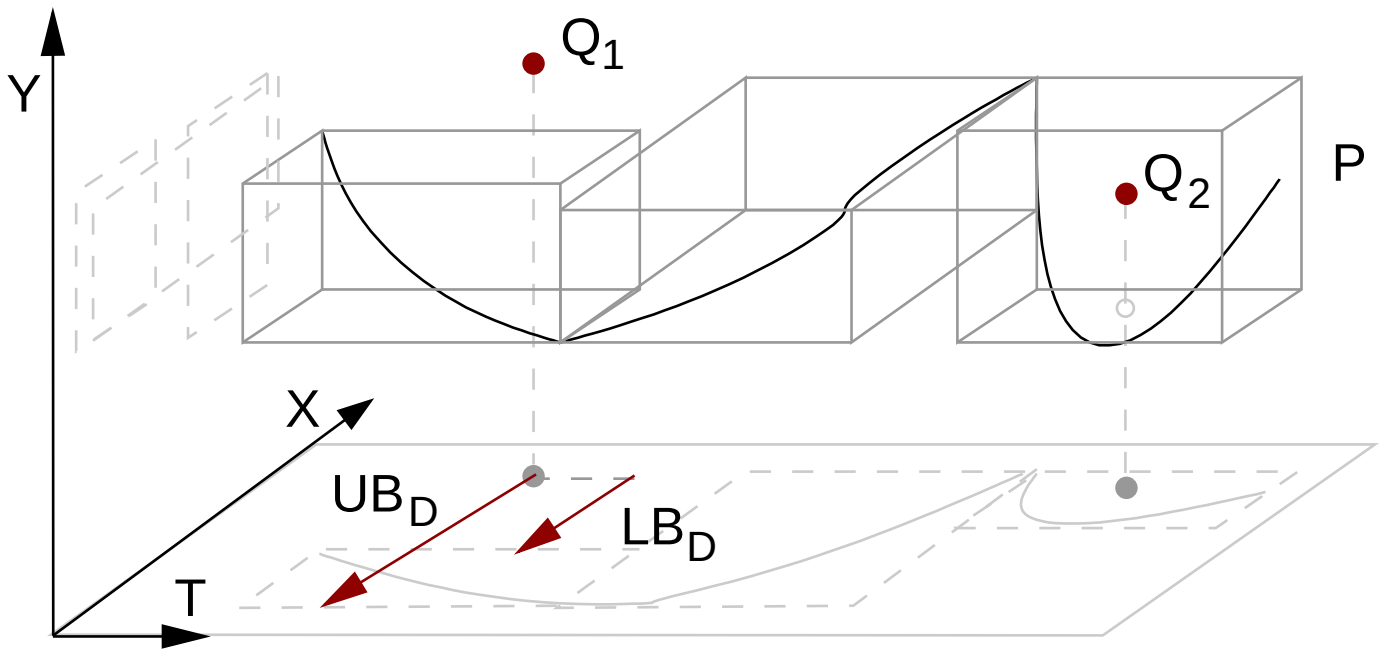
1. Identify all vehicles in proximity of location p_0, at time t_0, and in proximity of location p_1 at time t_1 (STP With Time)
2. Identify all vehicles in proximity of location p_0, and later in proximity of location p_1 (STP With Order)

Existing solutions (using spatial indexes) degenerate to linear scan!

Preliminaries

Trajectory Indexing

1. Split each trajectory into smaller chunks
2. Take the MBR of each chunk
3. Index MBRs using spatial index



Preliminaries, cont

Query

$Q = \{(Q_1, T_1), (Q_2, T_2) \ldots (Q_M, T_M)\}$

- Q_i is a $NN(q_i)$ or range query
- T_i is a time instant, or a time interval (or empty).

Preliminaries, cont

Query

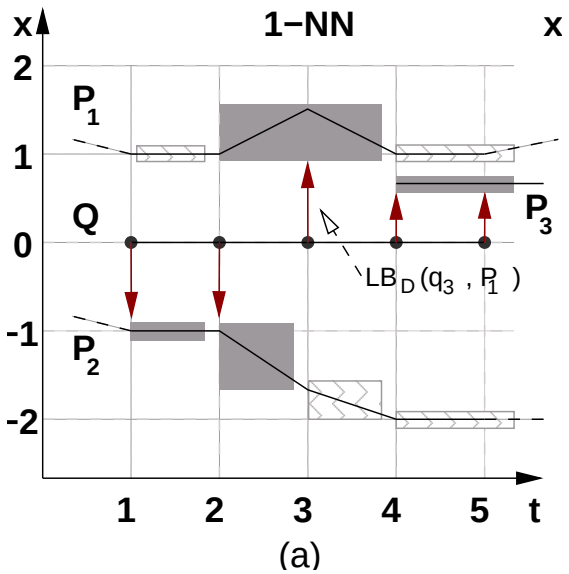
$\mathrm{Q} = \{(Q_1, T_1), (Q_2, T_2) \ldots (Q_M, T_M)\}$

- Q_i is a $NN(q_i)$ or range query
- T_i is a time instant, or a time interval (or empty).

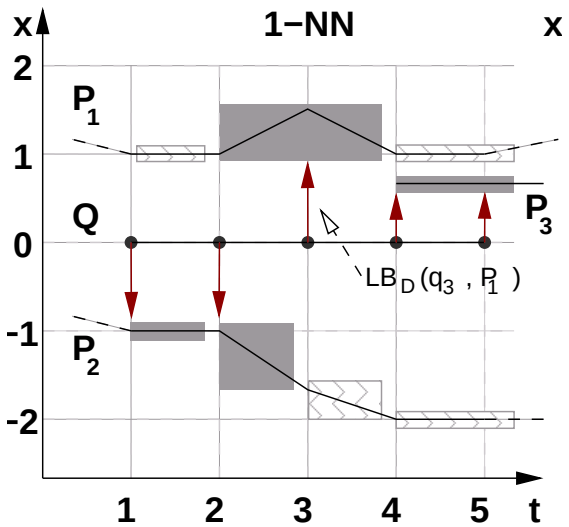
Notation

- $D(\mathrm{Q}, P)$ is distance from query Q to trajectory P
- $LB_D(\mathrm{Q}, P)$ is lower bound distance from Q to P

STP With Time (NN)



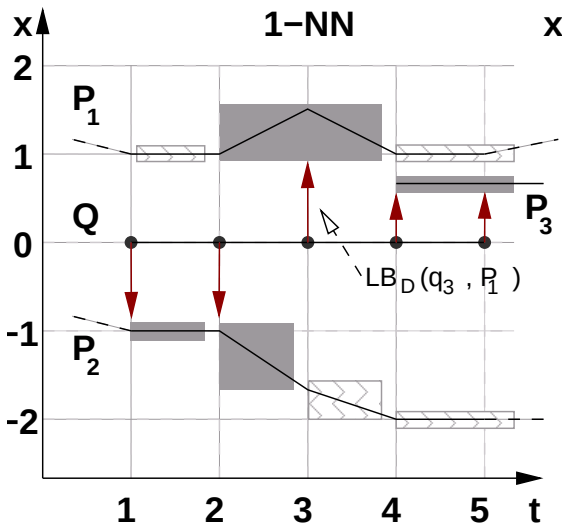
STP With Time (NN)



Sketch of Algorithm

1. Set k to 1

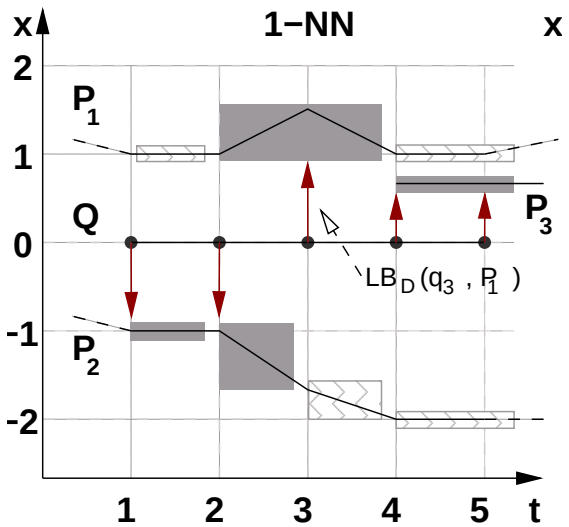
STP With Time (NN)



Sketch of Algorithm

1. Set k to 1
2. For each point q_i retrieve the k-NN MBRs.

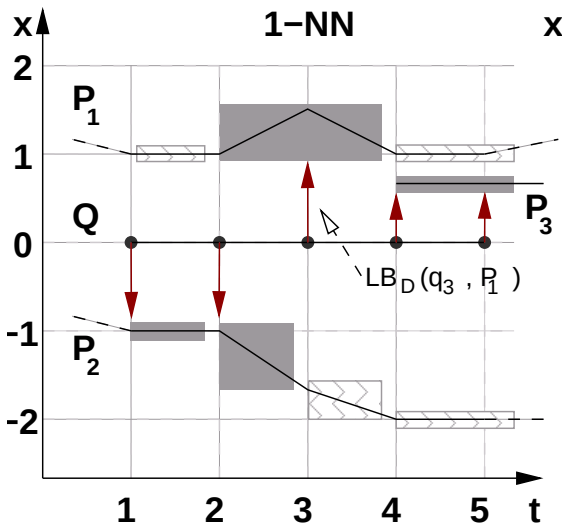
STP With Time (NN)



Sketch of Algorithm

1. Set k to 1
2. For each point q_i retrieve the k-NN MBRs.
3. If enough MBRs have been retrieved for a trajectory P to *cover* Q, and $D(\mathrm{Q}, P) < LB_D(P_i, \mathrm{Q})$; return P

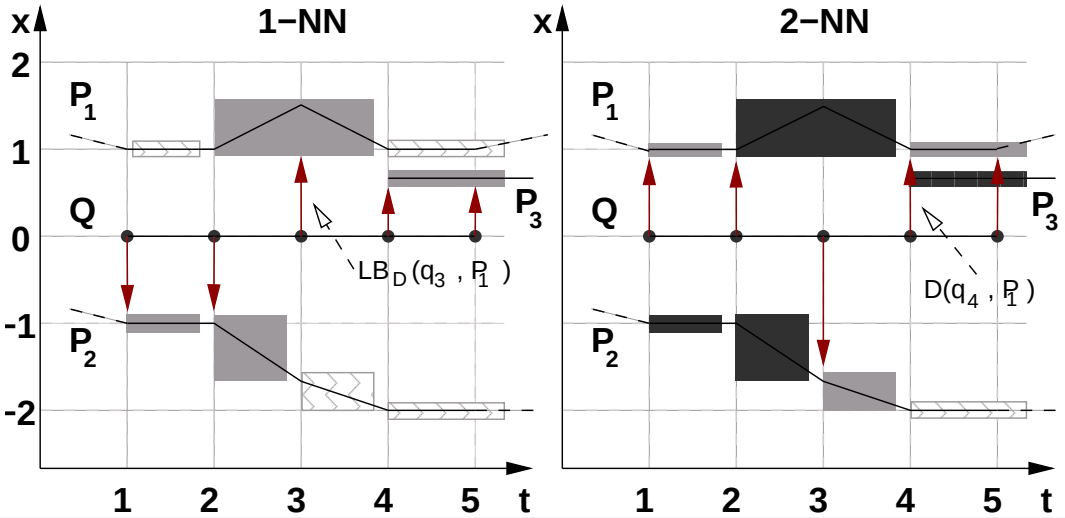
STP With Time (NN)



Sketch of Algorithm

1. Set k to 1
2. For each point q_i retrieve the k-NN MBRs.
3. If enough MBRs have been retrieved for a trajectory P to *cover* Q, and $D(\mathrm{Q}, P) < LB_D(P_i, \mathrm{Q})$; return P
4. Update pruning threshold; increment k; goto 2

STP With Time (NN)



Sketch of Algorithm

1. Set k to 1
2. For each point q_i retrieve the k-NN MBRs.
3. If enough MBRs have been retrieved for a trajectory P to *cover* Q, and $D(\mathrm{Q}, P) < LB_D(P_i, \mathrm{Q})$; return P
4. Update pruning threshold; increment k; goto 2

STP With Order (NN) - Preliminaries

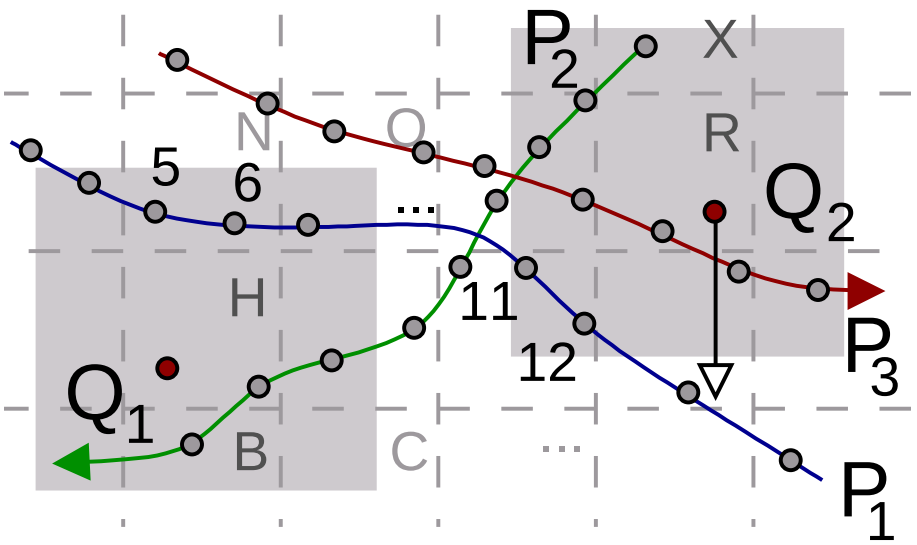
Problem

To answer an STP query With Order, basically requires one to project out the temporal dimension. Degenerates to a linear scan over data.

Solution

1. Create a grid over the 2D-space
2. With each cell, store a list of intersecting trajectories

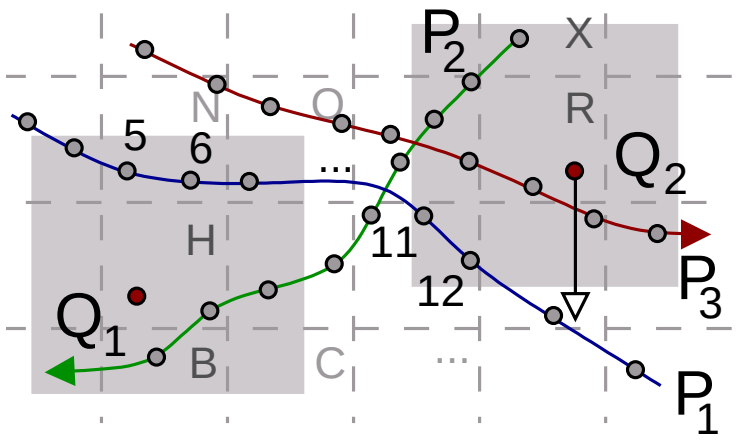
STP With Order (NN) - Preliminaries - cont



Trajectory Lists

CID	*List*
B	$\{(P_2, t_9)\}$
H	$\{(P_2, t_8)\}$
N	$\{(P_1, t_5), (P_1, t_6), (P_3, t_2)\}$
...	...

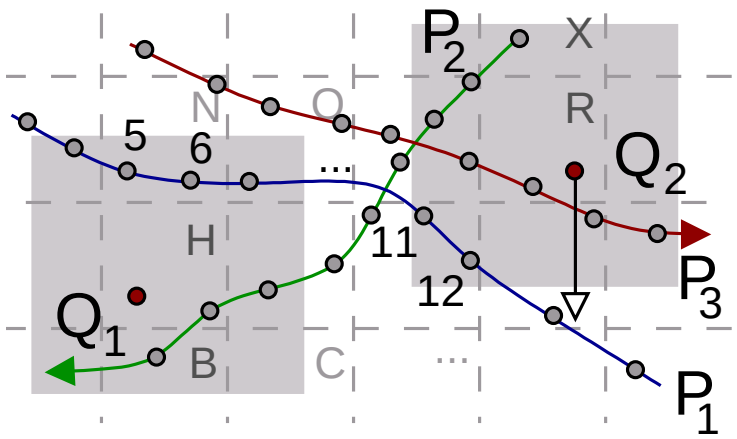
STP With Order (NN)



Sketch of Algorithm

1. For each query point q_i, initialize an empty list l_i

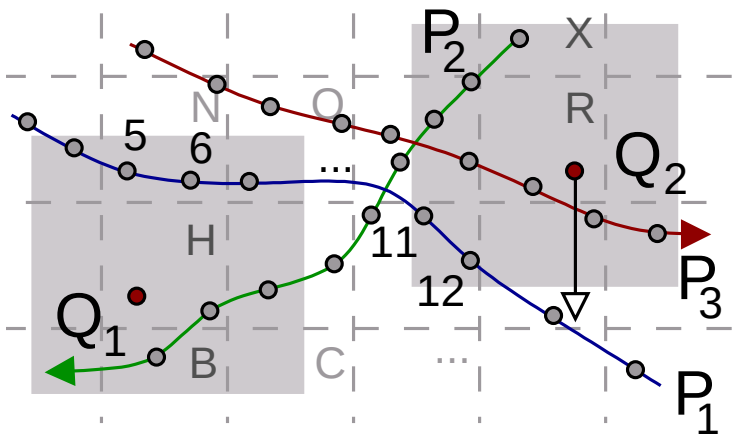
STP With Order (NN)



Sketch of Algorithm

1. For each query point q_i, initialize an empty list l_i
2. For each q_i, $Expand(q_i, l_i)$

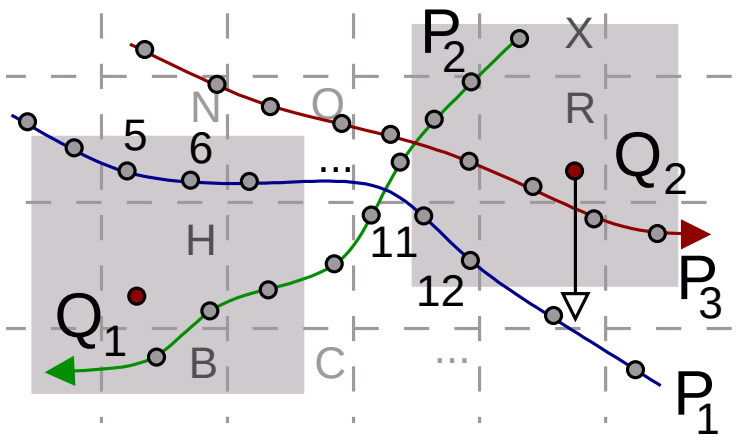
STP With Order (NN)



Sketch of Algorithm

1. For each query point q_i, initialize an empty list l_i
2. For each q_i, $Expand(q_i, l_i)$
3. Check if any trajectory P covers the query $\mathbb{Q}$, while satisfying order

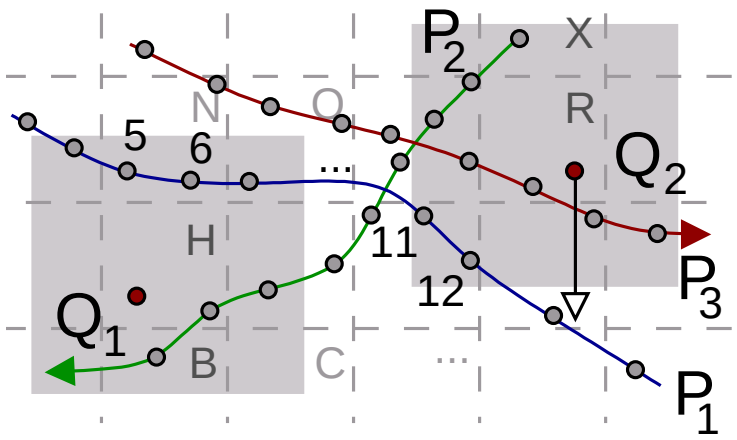
STP With Order (NN)



Sketch of Algorithm

1. For each query point q_i, initialize an empty list l_i
2. For each q_i, $Expand(q_i, l_i)$
3. Check if any trajectory P covers the query $\mathbb{Q}$, while satisfying order
4. If true, calculate new pruning threshold $D(\mathbb{Q}, P)$

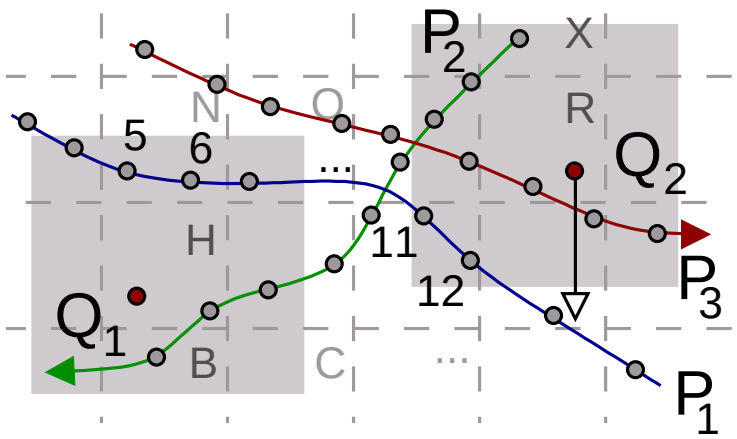
STP With Order (NN)



Sketch of Algorithm

1. For each query point q_i, initialize an empty list l_i
2. For each q_i, $Expand(q_i, l_i)$
3. Check if any trajectory P covers the query Q, while satisfying order
4. If true, calculate new pruning threshold $D(\mathrm{Q}, P)$
5. Prune result sets using new threshold

STP With Order (NN)



Sketch of Algorithm

1. For each query point q_i, initialize an empty list l_i
2. For each q_i, $Expand(q_i, l_i)$
3. Check if any trajectory P covers the query $\mathbb{Q}$, while satisfying order
4. If true, calculate new pruning threshold $D(\mathbb{Q}, P)$
5. Prune result sets using new threshold
6. If empty stop; goto 2 otherwise

Performance Evaluation

Test Setup

1. Spatial area is 1000×1000 miles.
2. Synthetic data-set with 5×10^5 moving objects, generated over a real road network
3. 3×10^5 trajectories, each approximated by 20 MBRs, stored in R-tree
4. R-tree uses 6.1×10^4 pages

Test Queries

Random Patterns based on consecutive nodes in the road network, with time constraints

Relevant Patterns based on parts of the trajectories in the data-set, but skewed in time and space

Performance Evaluation - STP With Time

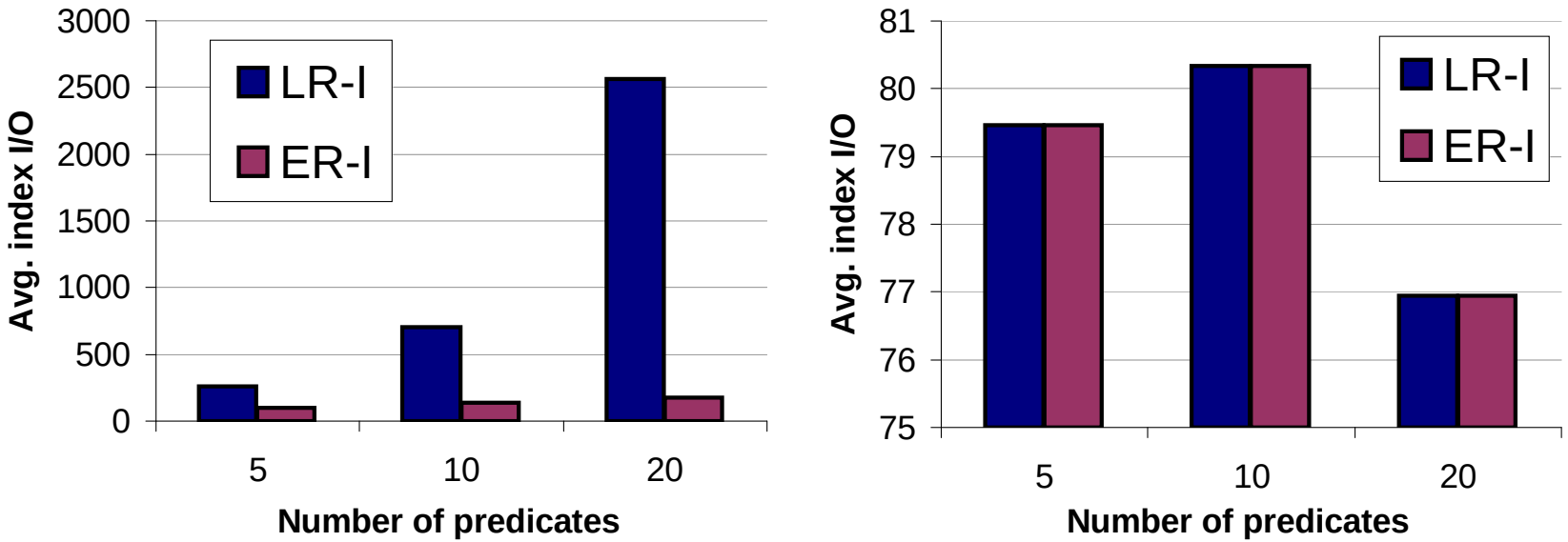
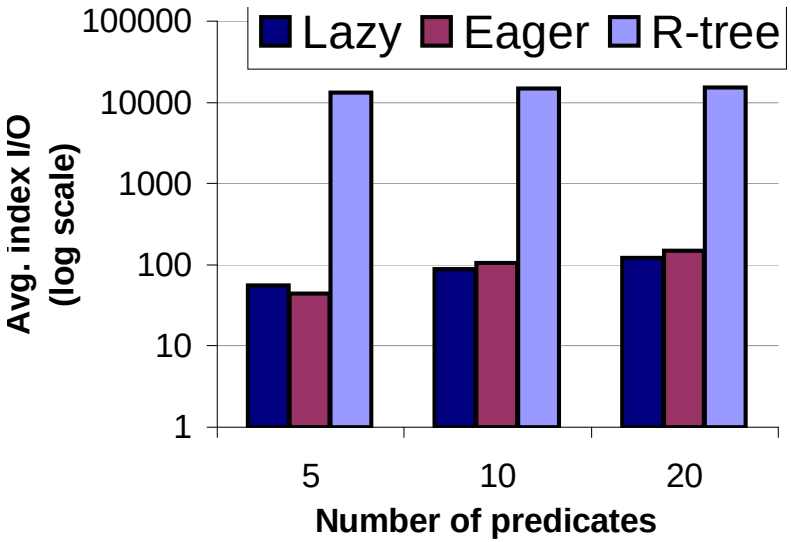


Figure: Left: Random. Right: Relevant.

Performance Evaluation - STP With Order



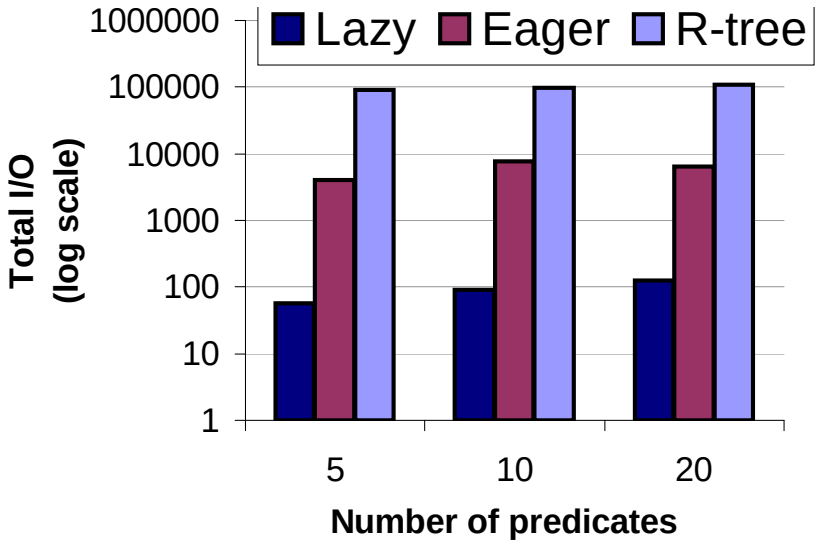


Figure: Both: Relevant.

Conclusion

1. Introduced and formalized a new type of spatio-temporal pattern query
2. Introduced spatial structures and algorithms reduces IO costs significantly vs state of the art
3. Experimentally validated

Critique

Strong Points

1. Relevant problem!
2. Likely much better than state of the art
3. Extensive testing, tweaking several parameters

Weak Points

1. Too complicated presentation of simple algorithms
2. Errors in several algorithms
3. Only tested with synthetic data set