Learn-to-Index on multidimensional data

Group 4

DENG Nan, MA Xiaolei, ZHANG Hexiao, ZHU Yuzhang, SHU Yi

Introduction

- **Index**
  - An essential tool for high performance database query
  - B-tree, Hash, Bitmaps…
  - Index can be seen as model
    - Maps a key to the position of a record within a sorted array
- **Learn-to-Index (LTI)**
  - Search for a key quickly
  - Less memory space
- **Application Scenario**
  - Ride-sharing platforms
  - Real estate platforms

Related Work Prior to Papers

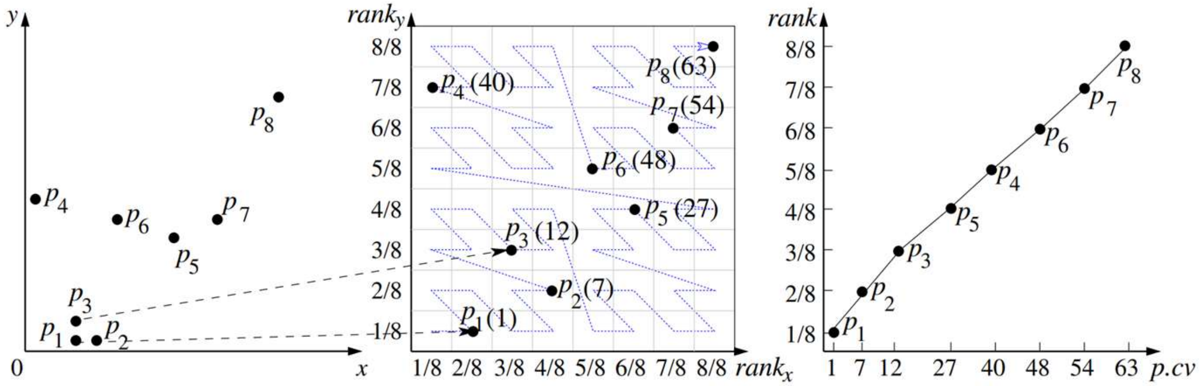
- For 1-dimensional data
  - Recursive Model Index (RMI) [1]
- For spatial data and queries
  - traditional indexing models: R-trees, kd-trees, and quadtrees
  - Z-order Model (ZM) [2]
    - Does not support KNN & data updates

[1] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis.2018. The Case for Learned Index Structures. In SIGMOD. 489–504.

[2] Haixin Wang, Xiaoyi Fu, Jianliang Xu, and Hua Lu. 2019. Learned Index for Spatial Queries. In MDM. 569–574.

Three methods for spatial indexes with machine learning

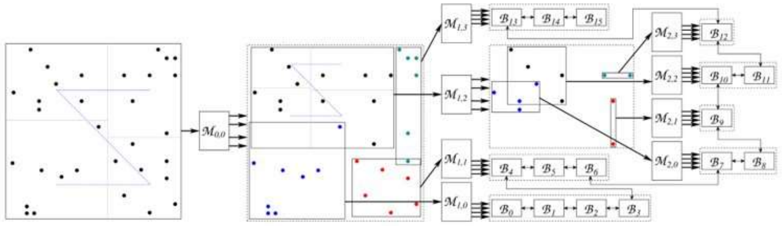
Effectively Learning Spatial Indices (RSMI)

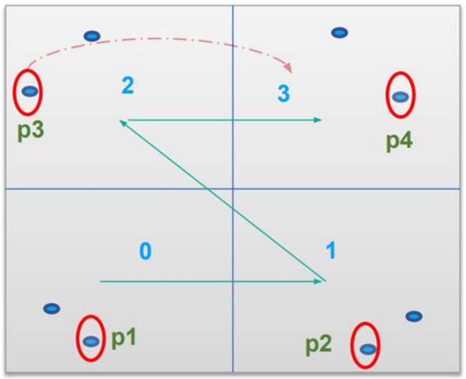


(a) Original space **(b)** Rank space **(c)** CDF

Spatial index based on ordering the data points by a rank space-based transformation

- Simplify the indexing functions to be learned
- M(search keys) = disk block Ids (location)





Point	p1	p2	p3	p4
Initial partition Id	0	1	2	3
Model predicted Id	0	1	3	3
Learned partition Id	0	1	3	3

For scaling to large datasets, proposes the Recursive Spatial Model Index (RSMI).

- Recursively partitions a dataset
- Partitioning is learned over the distribution of data

Steps:

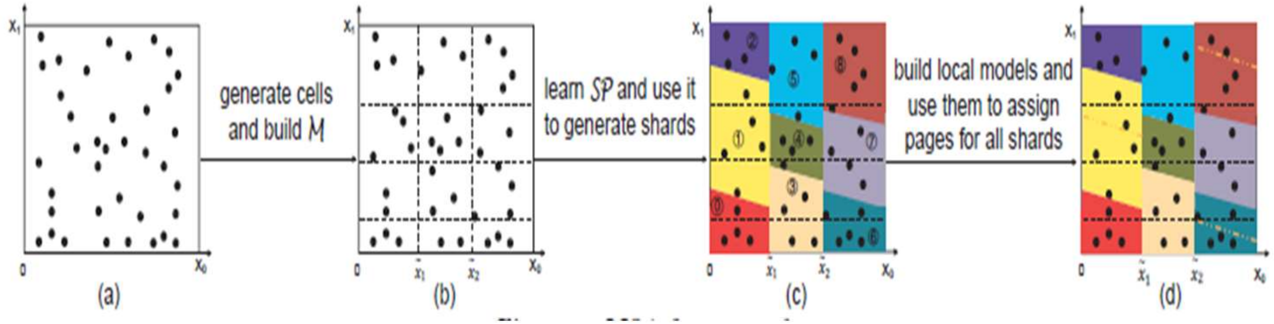
- Initially distribute the data into equal sized partitions
- Use a Space Filling Curve (SFC) to assign Ids
- Learn the partition Ids using a model M0,0
- Rearrange the data based on the prediction of M0,0
- Recursively repartition until each partition can be learned with a simple model

Effectively Learning Spatial Indices (RSMI)

Discussion:

- Focusing on point data.
- Depend on space filling curves.
- Support update, but periodical rebuilding is needed.
- Support point, window and KNN queries. But window query and KNN query are not exact. Experiments show that recalls are over 87% across various settings.

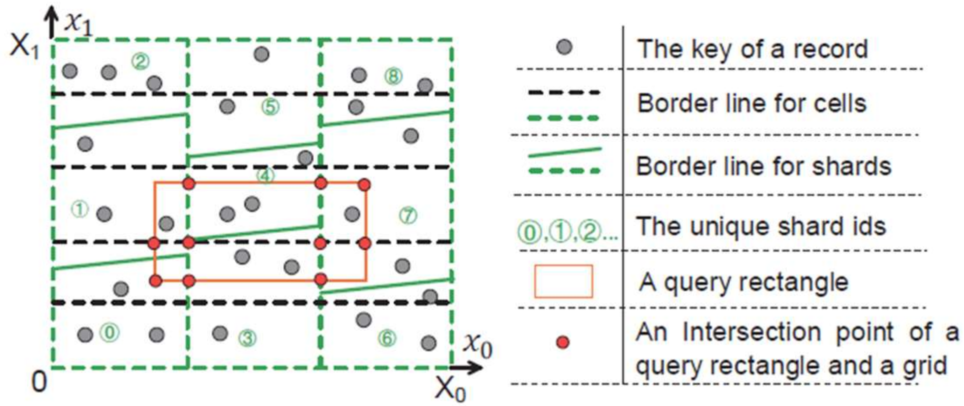
LISA: Learned Index Structure for Spatial Data



Core idea: Using machine learning models to generate searchable data layout in disk pages.

- Representation of grid cells
- A partially monotonic mapping function
- A monotonic shard prediction function
- Local models

LISA: Learned Index Structure for Spatial Data



Query process

- Given a query rectangle
- Get all cells that overlap with this rectangle
- Use monotonic mapping function to get 1-dimensional mapped value
- Obtain the corresponding shards that overlap with the mapped value by monotonic shard prediction function
- Access the data by local models

Discussion of LISA

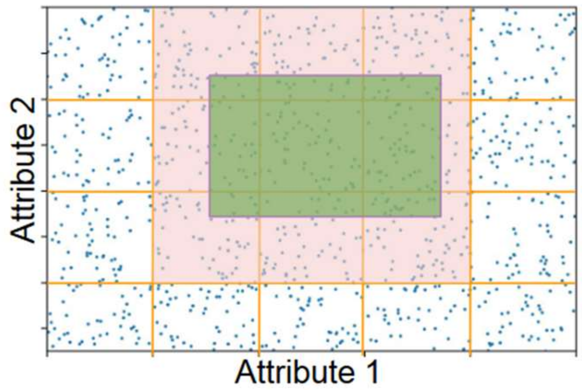
Ads:

- Assuring the correctness of the query result by monotonic functions
- Low additional overhead required to perform an index scan

Dis:

- In high-dimensional spaces, the number of cells required to partition the space becomes exponentially large, resulting in a large number of empty or sparsely populated cells.
- It is time consuming to construct index and execute the query.

Learning Multi-dimensional Indexes



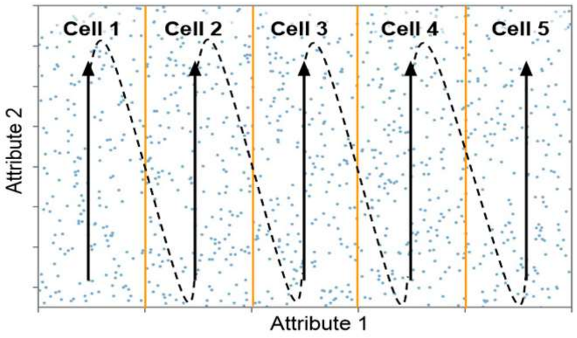
- Proposes Flood, a learned variant of basic grid index.
- Creates (d-1) dimensional grid and sort cells by the remaining dimension.
- Learned optimizations in several aspects:
  - Query time cost estimation
  - Dimension ordering selection
  - Cell flattening for average distribution of data in cells
  - Piecewise linear model for estimation of sort dimension values

Average time of projection

$$Time(D,q,L) = w_p N_c + w_r N_c + w_s N_s$$

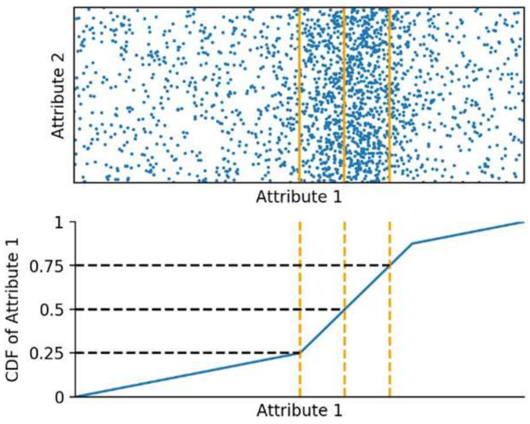
Average time of refinement

Average time of scan



Two steps to optimize dimension ordering:

- Predict the weights in the query time cost using random forest regression
- Enumerate all the dimensions as sort dimension, and order remaining dimensions by query selectivity. Use gradient descent to find best column number on each remaining dimension.



$$\frac{1}{|V|}\sum_{v\in V} D(v) - P(v) \leq \delta$$

Tradeoff between size and speed

Two methods to optimize cells and refinement:

- Flatten the cells to handle skewed data
  - RMIs to model CDF
  - Stabilizes query time by making number of points in cell average
- Piecewise Linear Model (PLM) built on sort dimension to replace binary search
  - Error bound is controlled by creating a new slice when average error exceeds value

Discussion of Flood

- Can speedup existing indexes though with limited improvement given limited search space.
- Limited to the grid index layout. Not applicable to tree-based or z-order indexes.
- Can only find nearby cells in order to perform kNN query. Excluded from evaluation.

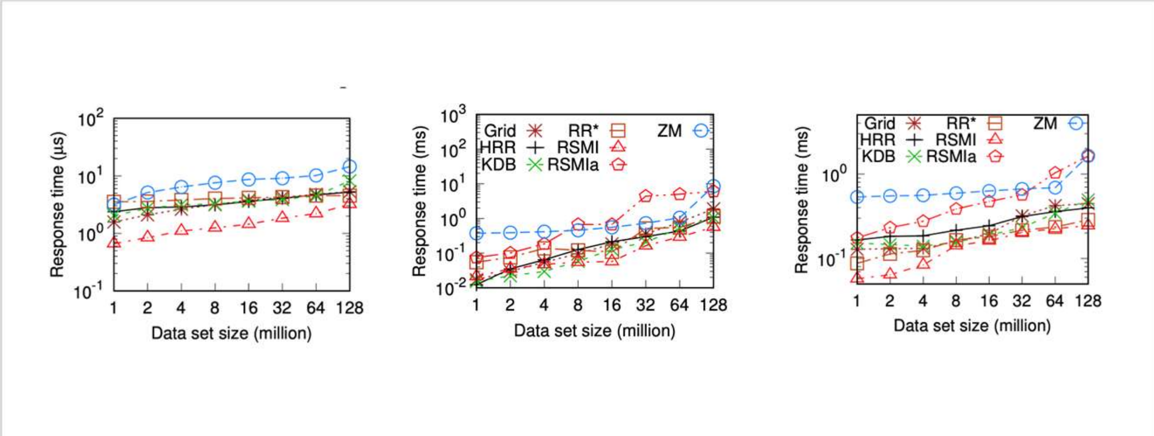
Method	Update Support	Query Support	ML Method	Learning Objective
RSMI	Yes (Periodic Rebuilding)	Point, Window and KNN (Not Accruent)	Deep Neural Networks	Fit the Cumulative Distribution Function (CDF) to map Space-Filling Curve (SFC) values to ranks
LISA	Yes (Flexible)	Point, Window, KNN	Piecewise Linear Models	Predict rank from output of mapping function M
Flood	No	Not KNN	Tree-based Models	Learn optimized data layout directly using query samples

Evaluation and Results

Index	Competitors	Datasets	Metrics
RSMI	ZM, Grid, KDB, HRR, RR*	2 real-world and 3 synthetic dataset	Avg. response time, number of block accesses, recall
LISA	Baseline, R-tree, R*-tree, KD-tree, ZM	2 real-world and 10 synthetic datasets	Size, IO, IO ratio, size ratio
Flood	Full Scan, Clustered Single-Dimensional Index, Grid Files, Z-Order Index, UB-tree, Hyperoctree, k-d tree, R*-Tree	3 real-world and 1 synthetic dataset	Total query time, index creation time, disk seeks, disk blocks

RSMI

- Experiments on real and synthetic data sets with more than 100 million points show that the proposed learned indices and query algorithms are highly effective and efficient. Query processing using RSMI is more than an order of magnitude faster than the use of R-trees or a recently proposed learned index, while the window and kNN query results are highly accurate, i.e., over 87% across a variety of settings.



LISA

- Extensive experiments demonstrate that LISA clearly outperforms R-tree and other traditional spatial indexes in terms of storage consumption and IO cost for range and kNN queries. Moreover, LISA supports data insertion and deletion operations efficiently.

Table 4: Performance of response time

Method	3D Uniform		ImageNet	
	CPU time (ms)	Response time (ms)	CPU time (ms)	Response time (ms)
R-tree	1.34	11.26	3.85	39.50
R*-tree	1.31	11.08	3.61	37.79
KD-tree	729	765.5	5,655	6,378
ZM	1.97	246.4	2.32	1,173
LISA	1.43	**10.07**	5.08	**27.22**

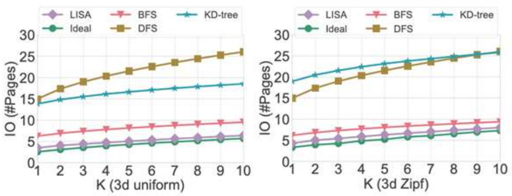


Figure 13: IO cost on ***K*****NN query**

Flood

- Flood achieves up to three orders of magnitude faster performance for range scans with predicates than state-of-the-art multi-dimensional indexes or sort orders on real-world datasets and workloads.

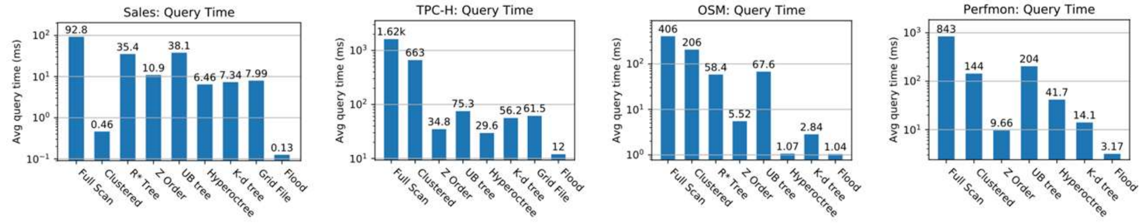


Figure 7: Query speed of Flood on all datasets. Flood's index is trained automatically, while every other index is manually optimized for each workload to achieve the best performance. We excluded the R-tree for cases for which it ran out of memory. Note the log scale.

	sales	tpc-h	osm	perfmon
Flood Learning	10.3	33.4	44.5	33.3
Flood Loading	4.12	29.6	8.03	22.0
Flood Total	14.4	63.0	52.5	55.3
Clustered	2.11	16.2	4.85	11.6
Z Order	7.82	86.7	24.9	72.6
UB tree	8.28	81.9	26.0	69.5
Hyperoctree	2.47	42.2	31.4	54.8
K-d tree	8.45	140	36.9	250
Grid File	10.6	121	N/A	N/A
R* tree	259	N/A	1340	N/A

Table 4: Index Creation Time in Seconds

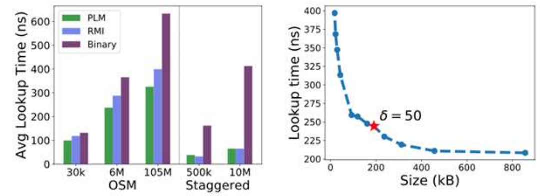


Figure 17: (a) A comparison of three per-cell CDF models on two 1-D datasets. (b) The size-speed tradeoff for the PLM, with our configuration marked.

Potential Issues

- **Efficiently Supporting Updates**
- **Support for Other Spatial Operations and Data Structures**
- **Choosing the Right ML Models**
- **Concurrency Support**
- **Benchmarking Learned Multidimensional Indexes**

Thank you / Q & A