Approaching the Skyline in Z Order

Ken C. K. Lee[1] Baihua Zheng[2]

Huajing Li[1] Wang-Chien Lee[1]

[1] **Pennsylvania State University, USA**
[2] **Singapore Management University, Singapore**

Presented in VLDB 2007, University of Vienna, Austria

What is skyline query?

- **Definition**: Given a set of multi-dimensional data points, skyline query finds a set of data points not dominated by others.

- A data point p dominates another data point q if and only if p is **better than or as good as** q on all dimensions and p is **strictly better than** q on at least one dimension.

Skyline applications …

- Find **cheap** and **conference-site close** hotels

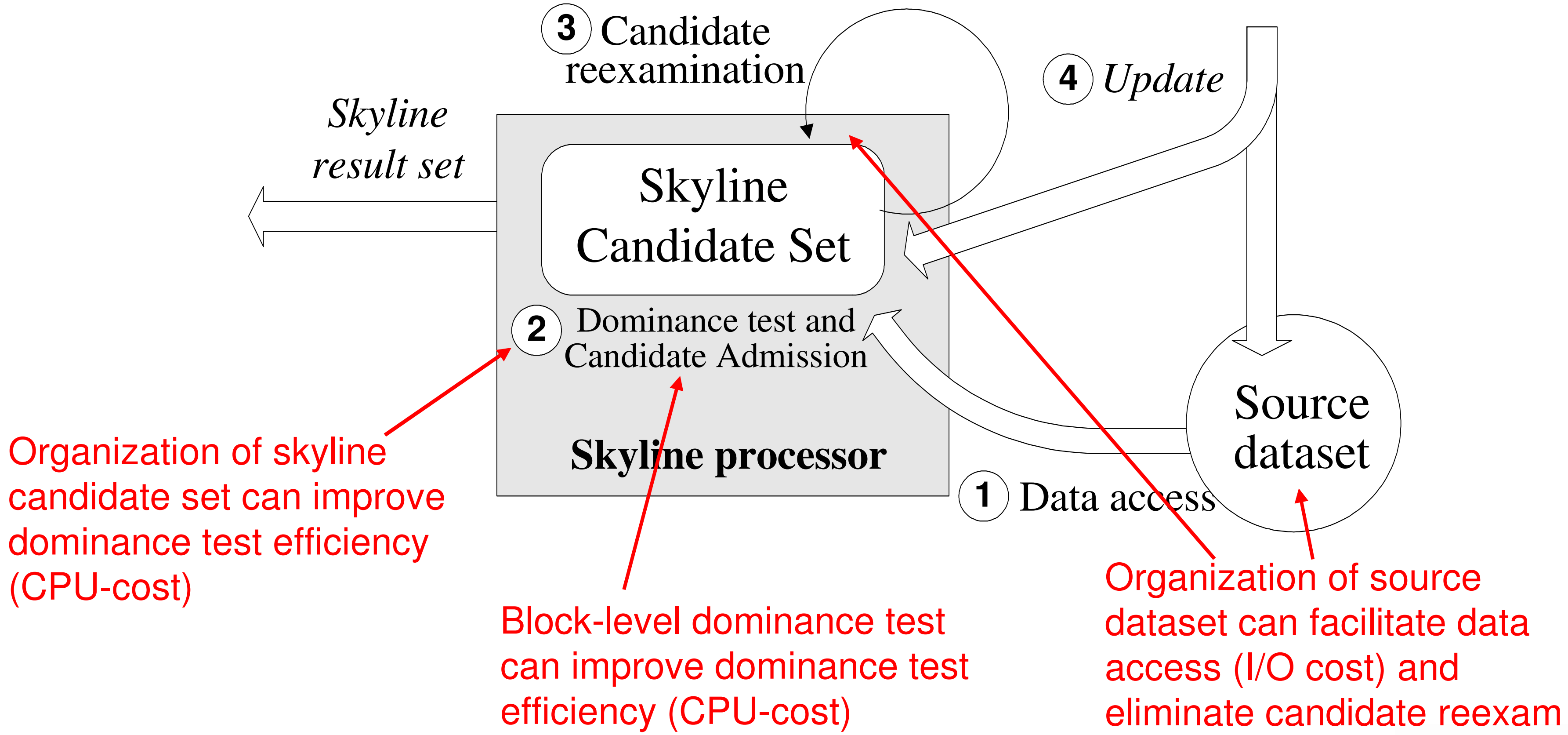
- Find **cheap** and **low mileage** secondhand cars

Challenges of skyline query processing

- Search efficiency
- Update efficiency
- Support of skyline query variants
  - k-dominant skyline

Our research objectives

- Develop a generic, unified and efficient processing framework to process skyline query.

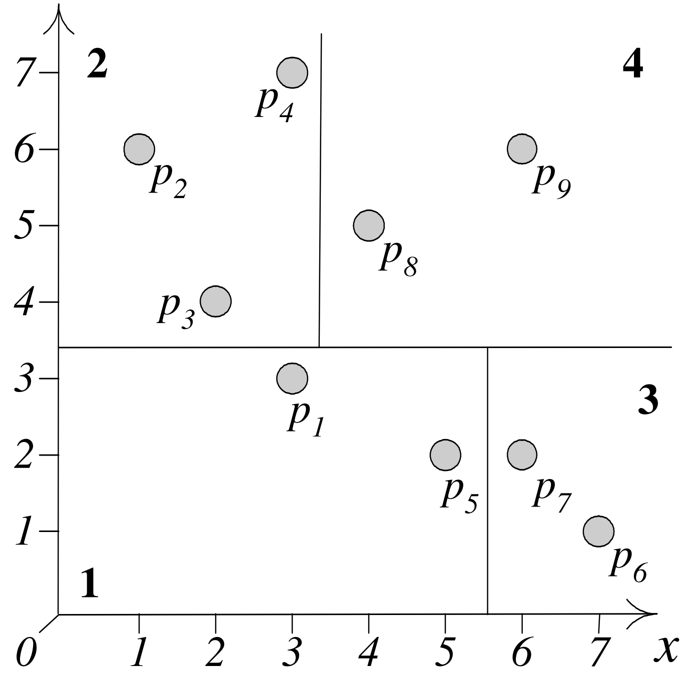


Related works

- Sorting-based approaches
  - Observation: accessing data points in any monotone function (entropy and sum of attributes) guarantees that dominating data points come before their dominated data points.
  - Approaches: Sort-Filter-Skyline [ICDE03], LESS [VLDB05]
  - Strength: no reexamination needed
  - Weakness: no indices on skyline candidates and data points, exhaustive dominance tests resulted.

Related works

- Divide-and-conquer (D&C) approach [ICDE01]
  - Partition data points along one dimension each time until the partition is small enough to be stored in main memory.
  - Determine skyline for each partition
  - Merge skyline from adjacent partition.



Related works

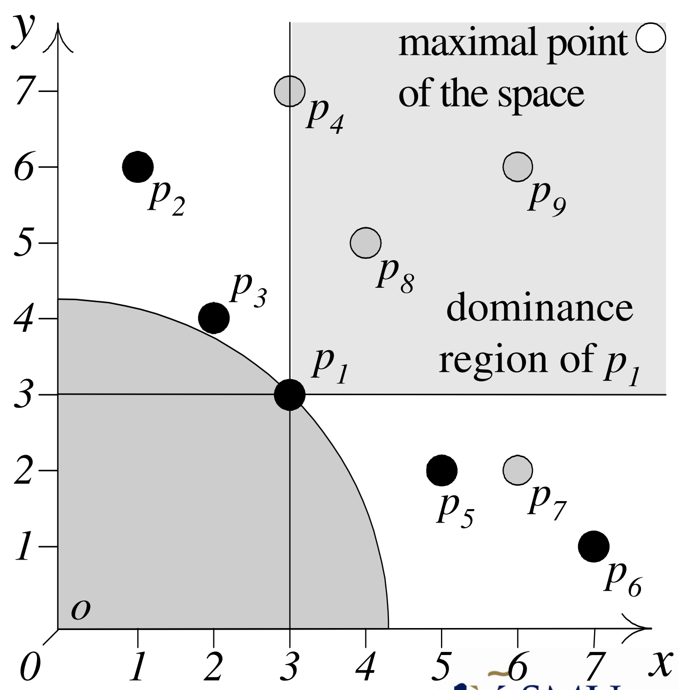
- Hybrid approaches
  – Combining D&C and sorting-based approaches
  – Representative approaches: NN [VLDB02] and BBS [SIGMOD03]

Observation:

1) The nearest neighboring point (e.g. p_1) should be a skyline
2) Other points behind it should be dominated.
3) The remaining points are incomparable and possibly other skyline points.

R-tree is used to index data points as it is good to support NN search.

BBS: use iterative NN search to reduce the repeated access of R-tree.



Related works

- Hybrid approaches

  R-tree: indexes data points to support NN search.

  BBS: iterative NN search to reduce the repeated access of R-tree.

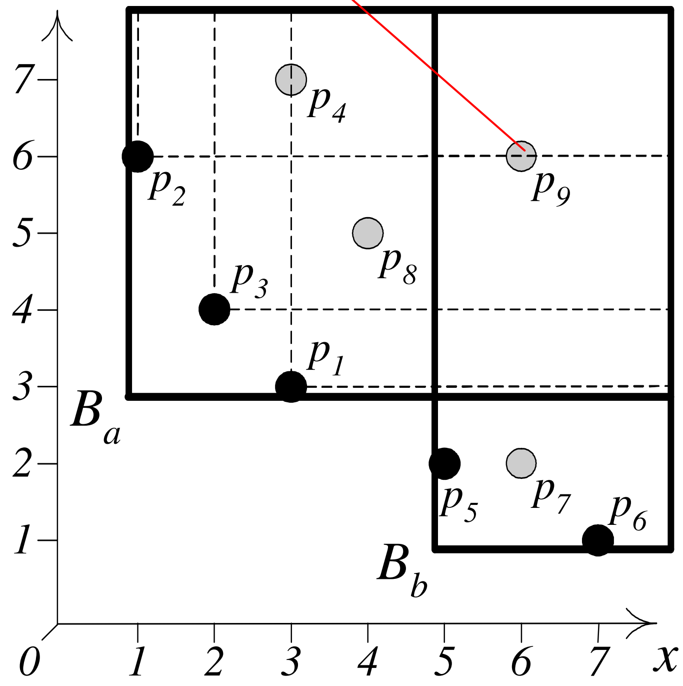
  a heap orders accessed data points

  High main memory contention to maintain a heap

  a main memory R-tree (mmR-tree) stores candidate skylines' dominance regions for dominance tests.

  Inefficient to support dominance tests

P_9 has to against B_a and B_b as it is enclosed by their MBBs.

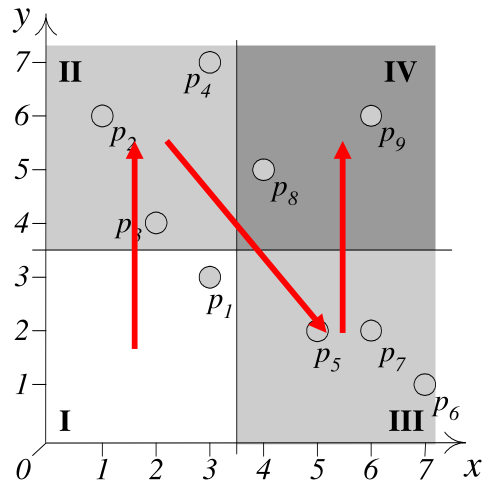


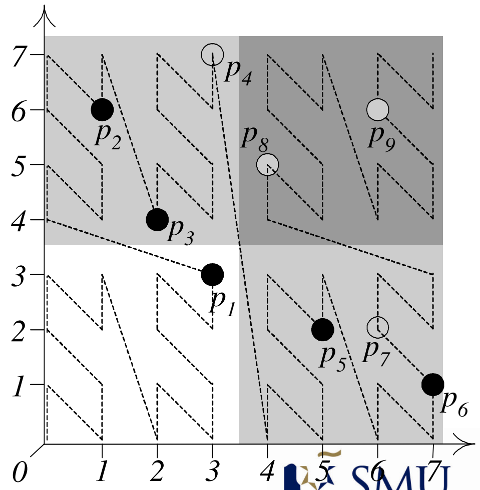
Skyline processing and Z Order

- Observations:
  - Partitioning a 2D space into 4 equi-sized subspaces
  - Data points in Region IV
    - should be dominated by any point in Region I and possibly dominated by those in Region II and Region III
  - Data points in Region II and Region III
    - may be dominated by those in Region I
    - are incomparable
- Possible access sequence for skyline points:
  - Region I ➔ Region II ➔ Region III ➔ Region IV, or
  - Region I ➔ Region III ➔ Region II ➔ Region IV

  ** These two sequence produce the same result.

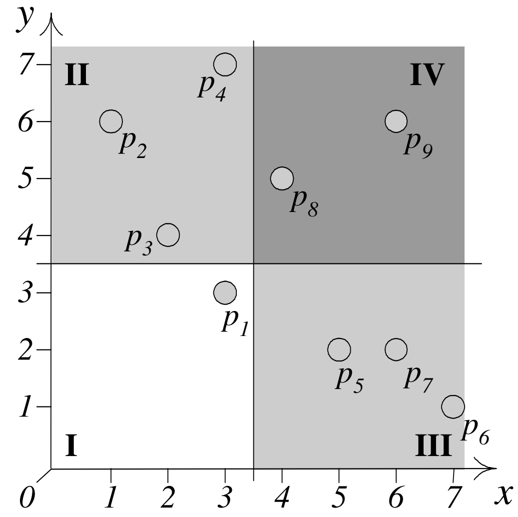
- Finally, it is Z Order space filling curve





Z-address

- Suppose attribute value domain range is $[0, 2^v - 1]$ each attribute is represented by a *v*-bit binary
- A point with *d* attributes is represented by *d* *v*-bit string
  - P8: (4, 5) = (100, 101)
  - P9: (6, 6) = (110, 110)
- Z-address is represented by *v* *d*-bit groups, with the *i*th d-bit group contributed by *i*th bit of each attribute value
  - P8: (4, 5) = (1 0 0, 1 0 1) -> 11 00 01
  - P9: (6, 6) = (1 1 0, 1 1 0) -> 11 11 00

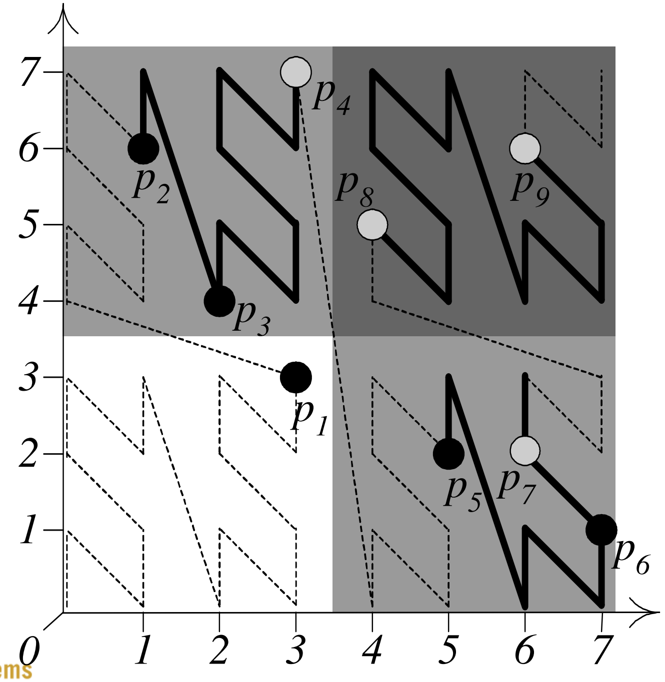


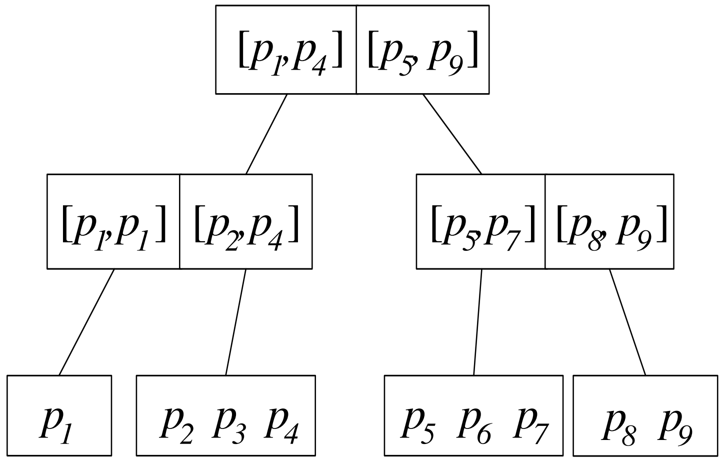
Why Z Order is better?

- In Z Order curve, data points are assigned Z-addresses
  - Monotone order (dominating data points always accessed before their dominated data points) ← transitivity property of skyline
  - Cluster in regions (incomparable data points are separate) ← incompatibility property of skyline

ZB-tree

- An B+-tree variant
- Z-addresses of data points are search keys
- Leaf level: individual data points
- Non-leaf level: ranges of Z-addresses
- Depth-first traversal == access data points in ascending Z-address order



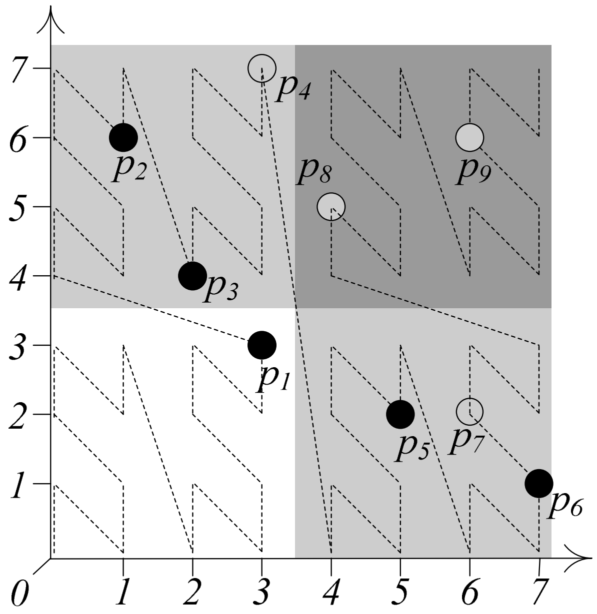


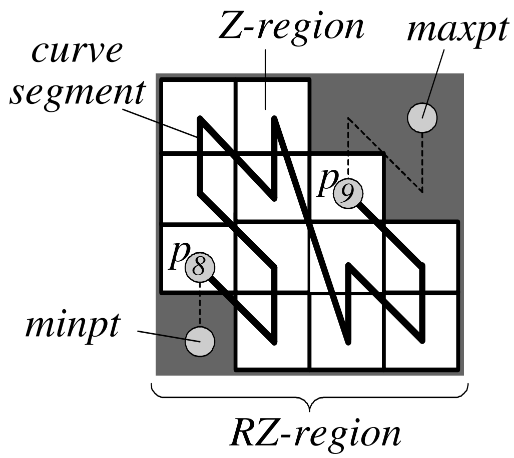
RZ-Region

- Node allocation criteria:
  - Small RZ-Region
- What is RZ-Region?
  - The smallest square area covering a segment along Z-order
- Example RZ-Region of [p_8, p_9]
  - P_8: 11 00 01
  - P_9: 11 11 00

  } **11** (common prefix)

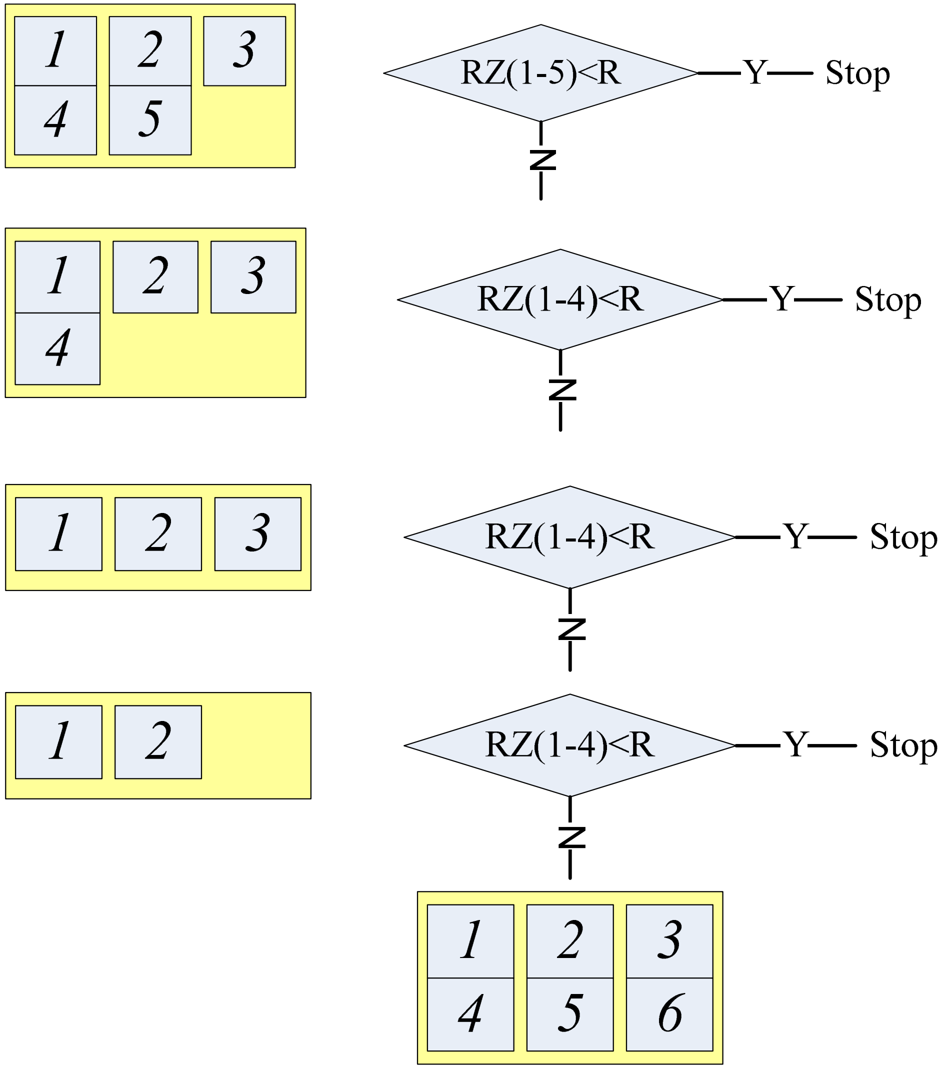
  - minpt: 11 0000 = (4, 4)
  - maxpt: 11 1111 = (7, 7)
- Properties of RZ-Region
  - $\forall z (\neq minpt) \in RZ,\ minpt \vdash z$
  - $\forall z (\neq maxpt) \in RZ,\ z \vdash maxpt$

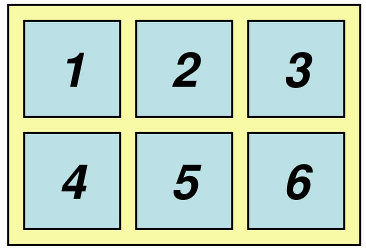




Node Allocation

Fanout [2,6]
R: RZ-region (1-6)

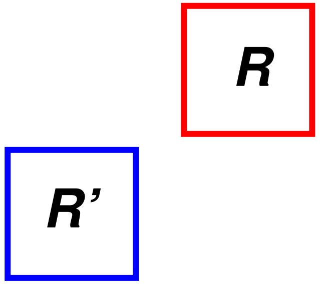




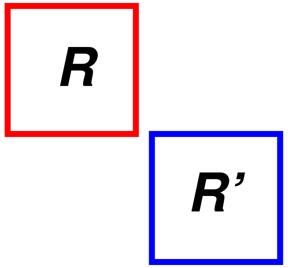
Z-Search

- Two ZB-tree: source, and skyline points
- Depth-first search
- Block based dominance tests

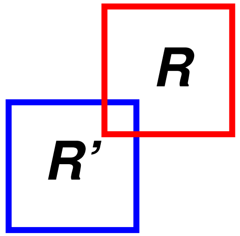
case 1: $R'.maxpt \vdash R.minpt \Rightarrow R' \vdash R$



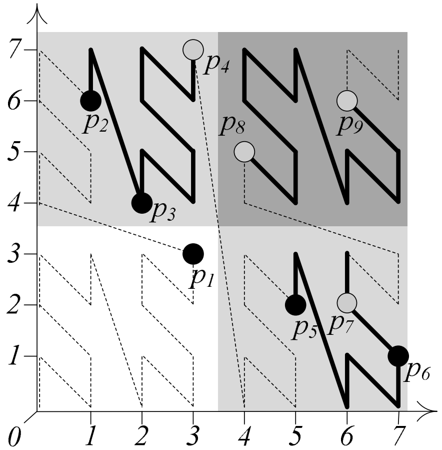
case 2: $R'.minpt \nvdash R.maxpt \Rightarrow R' \nvdash R$



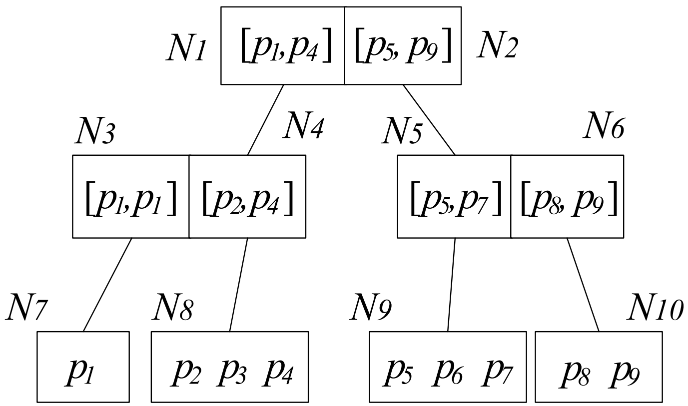
case 3: $R'.maxpt \nvdash R.minpt \wedge R'.minpt \vdash R.maxpt$



ZSearch (example)

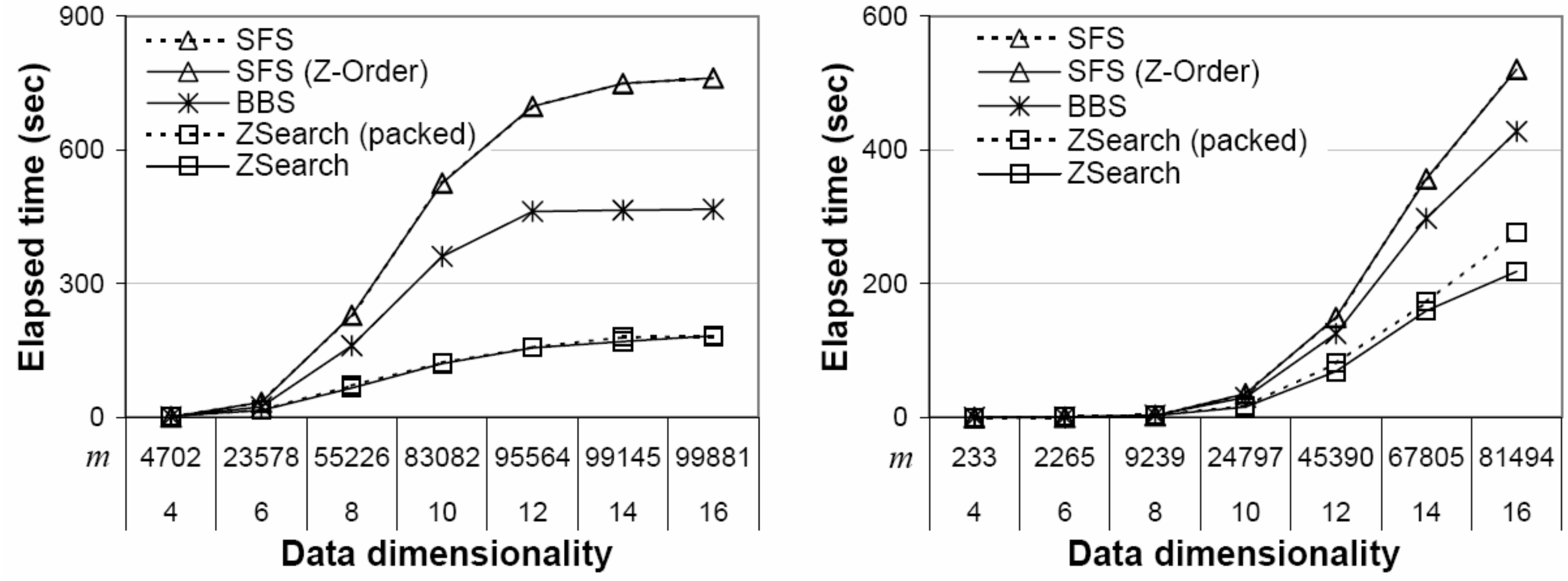


Skyline point ZBtree	ZBtree nodes
{}	*N1, N2*
{}	*N3, N4, N2*
{}	*N7, N4, N2*
{p1}	*N8, N2*
{p1},{p2,p3}	*N2*
{p1},{p2,p3}	*N5, N6*
{p1},{p2,p3},{p5,p6}	*N6*



Experiments

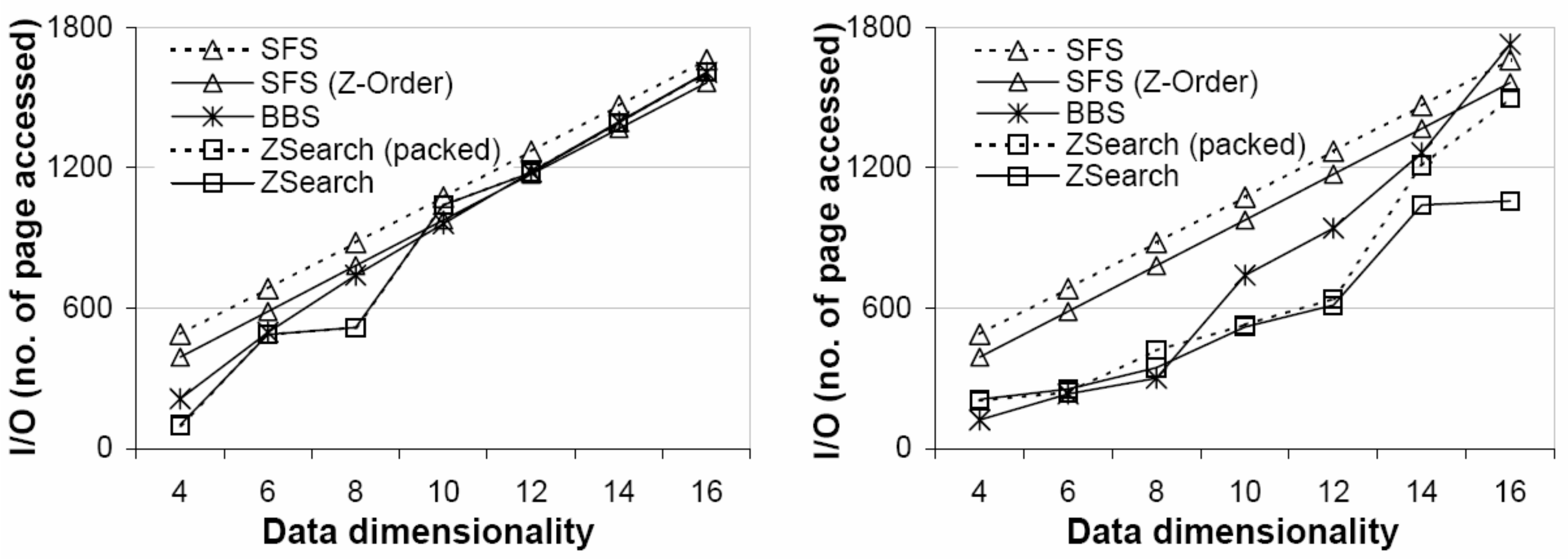
- Synthetic dataset
  - Distribution: anti-correlated, independent
  - Dimensionality: 4-16,
  - Cardinality: 100k



Elapsed time

Experiments

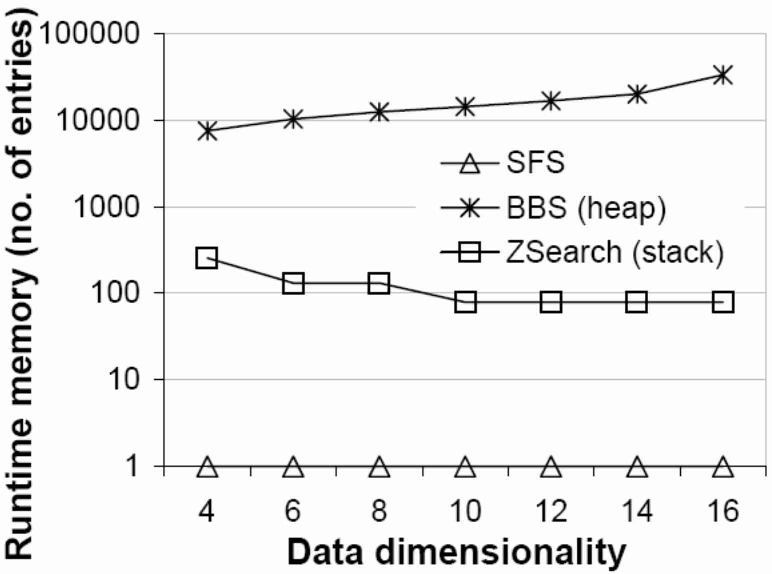
- Synthetic dataset
  - Distribution: anti-correlated, independent
  - Dimensionality: 4-16,
  - Cardinality: 100k

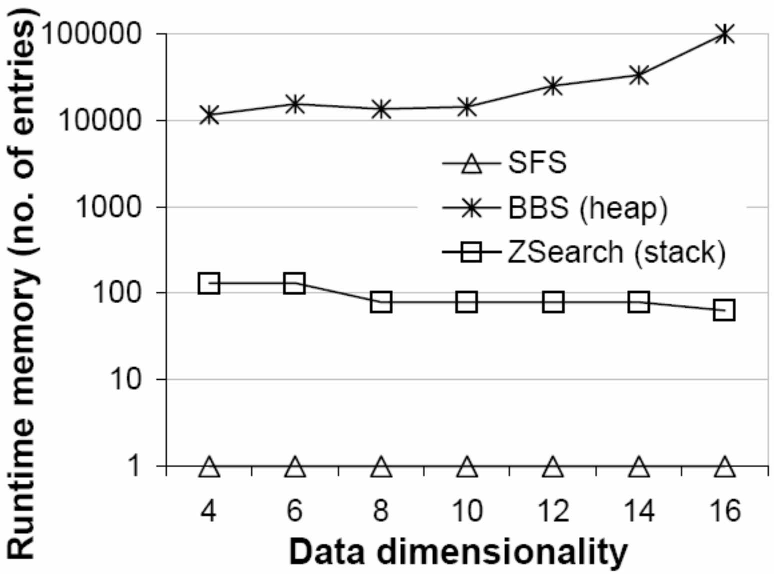


I/O Cost

Experiments

- Synthetic dataset
  - Distribution: anti-correlated, independent
  - Dimensionality: 4-16,
  - Cardinality: 100k





Runtime memory consumption

Experiments

- Real datasets
  - NBA - NBA player performance (dimensionality: 13, cardinality: 17k)
  - HOU - American family expenses on 6 categories (dimensionality: 6, cardinality: 127k)
  - FUEL - Performance of vehicles (e.g. mileage per gallon of gasoline) (dimensionality: 6, cardinality: 24k)

		Elapse Time			I/O cost		
Dataset	m	SFS	BBS	ZSearch	SFS	BBS	ZSearch
NBA	10816	2.933	3.364	1.723	228	230	131
HOU	5774	1.334	2.169	0.944	874	896	346
FUEL	1	0.031	0.001	0.001	164	3	3

ZUpdate

- Update:
  - insertion of new data points, and
  - deletion of data points that could be skyline points
- Challenges:
  - Insertion is straightforward; check if new data points are dominated by existing skyline. If no, put them as skyline
  - Deletion is complicated. Deletion of existing skyline may result in promotion of data points that are previously dominated
- Our solution
  - Based Z-order curve transitivity property, those potential skyline for promotion should be behind the deleted skyline point
  - Then by comparing candidate with skyline (RZ-regions), we identify new promoted skyline points

Experiments

- Real datasets, NBA, HOU and FUEL

Dataset	m	BBS-Update		DeltaSky		ZUpdate	
		del	ins	del	ins	del	ins
NBA	10816	78.37	4.18	42.25	5.09	14.21	1.27
HOU	5774	492.11	5.22	482.31	5.98	339.96	2.44
FUEL	1	0.10	0.001	0.15	0.008	0.10	0.001

Elapsed time

BBS-Update: [TODS05]
DeltaSky: [ICDE07]

k-ZSearch

- k-dominant skyline
  - Due to huge volume of result skyline points for high dimensionality, k-dominant skyline relax dominance conditions so some data points has a few good attributes can be dominated by others.
  - Notation: $a \vdash_k b$: a k-dominates b that for any *k* out of all dimensions, a has at least one attributes strictly better than b and a is better than or as good as b for the rest of attributes.
  - Challenges:
    - Data points can simultaneously dominate each others. (Transitivity property is no longer valid)
      - P2 (1, 6), and P8 (4,5)
  - Our solution:
    - Based on Z-Order curve clustering property, those cluster k-dominated are removed.
    - We adopt filter and reexamination framework to determine k-dominant skyline.

Experiments

- Real datasets: NBA, HOU, FUEL

Dataset	k	m	TSA	k-ZSearch
NBA	12	3794	7.931	2.696
	11	682	1.980	0.731
	10	79	0.322	0.171
HOU	5	22	0.815	0.226
	4	0	0.487	0.220
FUEL	5	1	0.063	0.001
	4	1	0.062	0.001

Elapsed time

TSA [SIGMOD06]

Our contribution

- Exploit a close relationship between skyline processing and Z-order
- ZB-tree, data index based on Z-order
- Develop a suite of algorithms based on ZB-tree
  - ZSearch – skyline search algorithm
    - more efficient than state-of-art search algorithms, such as BBS and SFS
  - ZUpdate – skyline result update algorithm
    - more efficient than existing available algorithms, such as BBS-Update and DeltaSky
  - K-ZSearch – k-dominant skyline search algorithm
    - more efficient than existing available algorithm such as TSA.

Q & A