# Efficient Skyline Maintenance
for Streaming Data
with Partially-Ordered Domains

Yuan Fang
Institute for Infocomm Research

Chee-Yong Chan
National University of Singapore

DASFAA10 @ Tsukuba, Japan
2 April 2010

Outline

- Introduction
- STARS
- STARS$^+$
- SkyGrid
- Experiments
- Conclusion

Outline

- Introduction
  - Concept
  - Problem settings
- STARS
- STARS$^+$
- SkyGrid
- Experiments
- Conclusion

Definitions

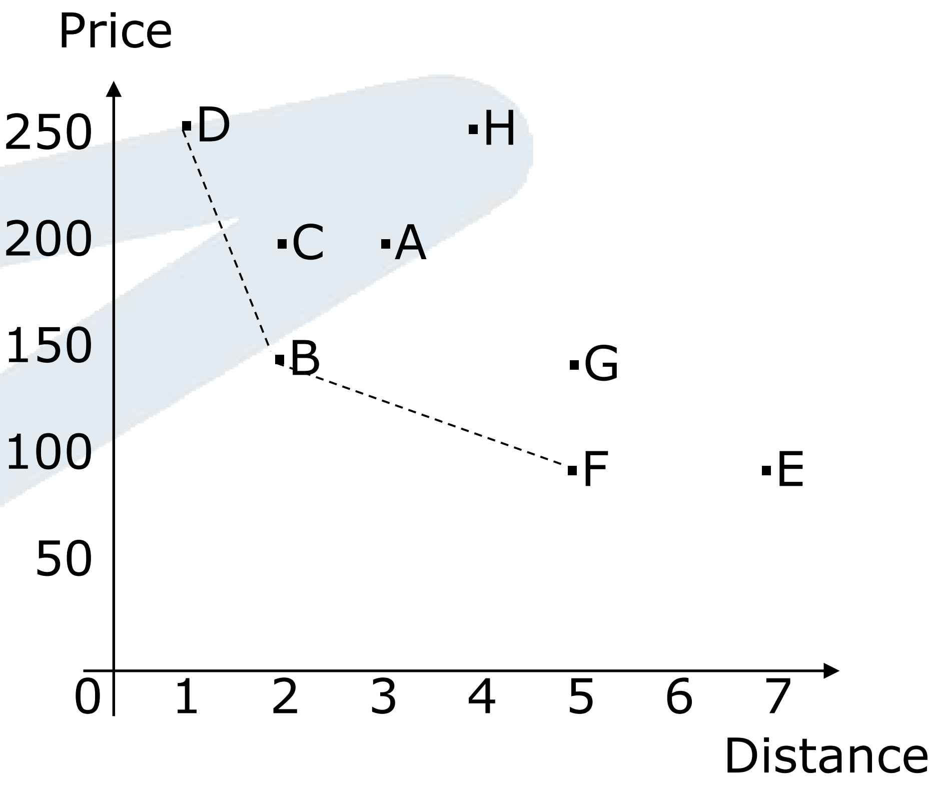
- A tuple X **dominates** Y iff for a set of relevant attributes A:
  - (1) X is better than or equal to Y in every attribute in A; and
  - (2) X is better than Y in at least one attribute in A.
- The **skyline** consists of:
  - all tuples not dominated by any other tuple.

Hotel example

- Tourist looking for a hotel
  - Cheap
  - Close to the city
- Relevant attributes
  - Price
  - Distance
- A hotel X dominates Y iff:
  - (1) X.price ≤ Y.price; and
  - (2) X.distance ≤ Y.distance; and
  - (3) at least one of (1) and (2) is strict.

Hotel example

Hotel	Price ($)	Distance to city (km)
A	200	3
B	150	2
C	200	2
D	250	1
E	100	7
F	100	5
G	150	5
H	250	4



Data domain

- Total-order
  - A linear ordering of every value
  - E.g., price, grade
- Partial-order
  - Lack a total linear ordering
  - Values can be comparable and incomparable
  - Good for hierarchies, preferences, intervals
  - E.g., user prefers yellow to red and blue to red, but there is no preference between yellow and blue.

Query context

- Offline
  - For disk-resident data (relatively static)
  - Answer query on demand
- Online
  - For streaming data (fast-changing)
  - Infeasible to answer the query from scratch
  - Maintain a skyline continuously

Problem settings

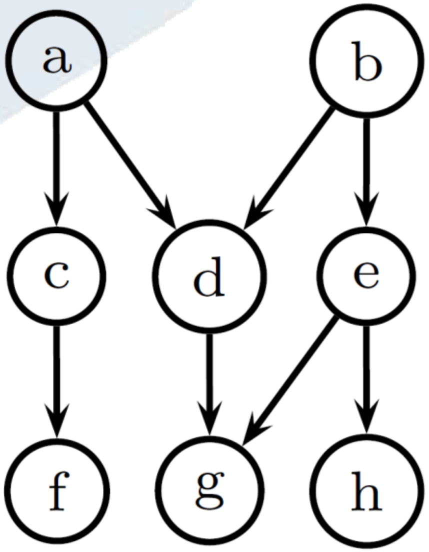
- Data domain: partially-ordered
- Query context: streaming
  - Count-based sliding window model
  - Maintain a buffer of size N
  - New tuple inserted into buffer
  - Oldest tuple deleted from buffer
- Baseline: Streaming Arrangment Skyline (STARS)
- Contributions
  - STARS$^+$
  - SkyGrid

Outline

- Introduction
- STARS
- STARS+
- SkyGrid
- Experiments
- Conclusion

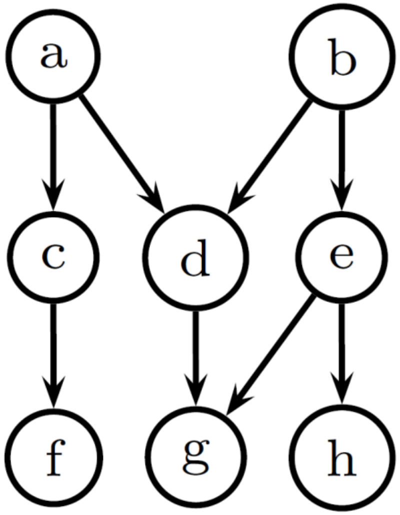
Domain representation

- Directed acyclic graph (DAG)
- Vertex – values
- Edge – relationship
  - a dominates d
  - d dominates g
  - d, e incomparable



Domain transformation

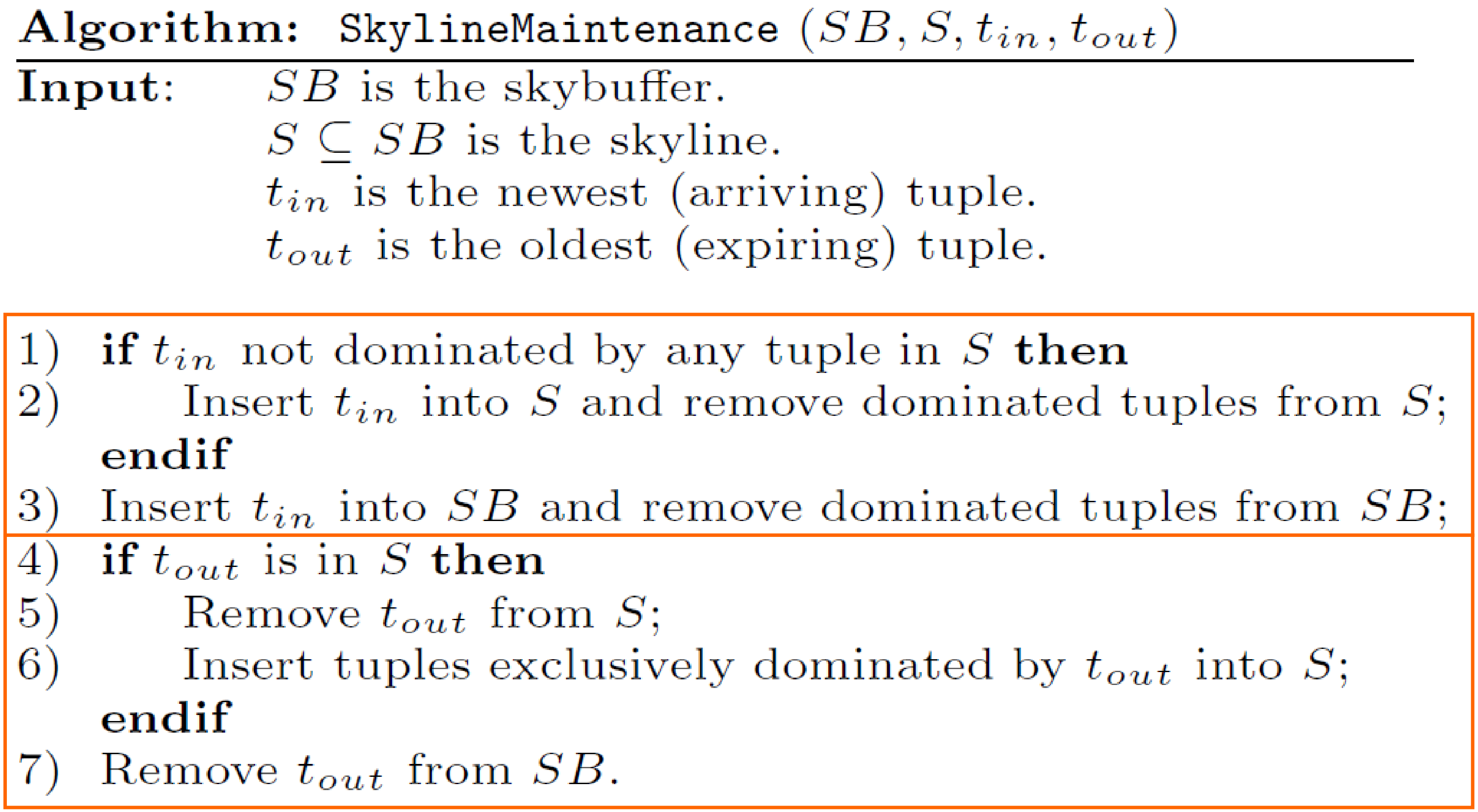
- Values → *topological sorting* orders
- A topological sort of a DAG is:
  - A linear ordering of all the vertices
  - For any directed edge, source vertex is always listed before the destination vertex
  - Denote vertex v's position by r(v)
- If r(v) ≥ r(v'), v cannot dominate v'
  - Inverse is not true
- Example: a, b, c, …, h
  - r(c) > r(a)
  - c does not dominate a



SkyBuffer

- Discard irrelevant tuples from buffer
- Given t’ and t, if:
  - (1) t’ is younger than t, and
  - (2) t’ dominates t.
- t can never get promoted to the skyline
- t is irrelevant
- SkyBuffer: the relevant part of the buffer

Skyline maintenance



Algorithm: `SkylineMaintenance` (SB, S, t_{in}, t_{out})

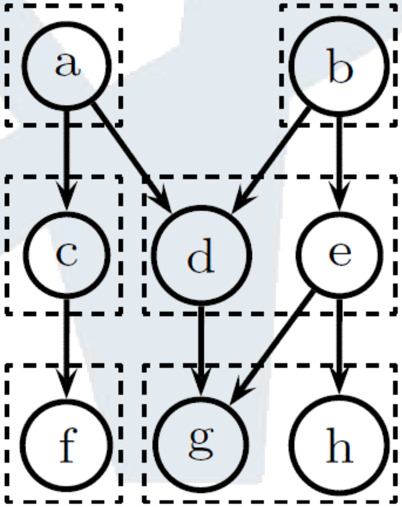
Input: SB is the skybuffer.
$S \subseteq SB$ is the skyline.
t_{in} is the newest (arriving) tuple.
t_{out} is the oldest (expiring) tuple.

```
if $t_{in}$ not dominated by any tuple in $S$ then
    Insert $t_{in}$ into $S$ and remove dominated tuples from $S$;
endif
Insert $t_{in}$ into $SB$ and remove dominated tuples from $SB$;
if $t_{out}$ is in $S$ then
    Remove $t_{out}$ from $S$;
    Insert tuples exclusively dominated by $t_{out}$ into $S$;
endif
Remove $t_{out}$ from $SB$.
```

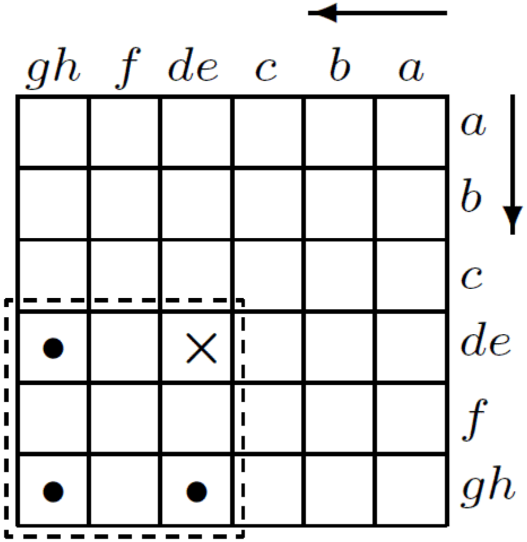
Buffer organization

- Main query: to find tuples dominated by a query tuple
- Multi-dimensional grid
- Value grouping for scalability
- Focused search

(a) DAG for $\mathcal{D}$



(b) Skybuffer grid



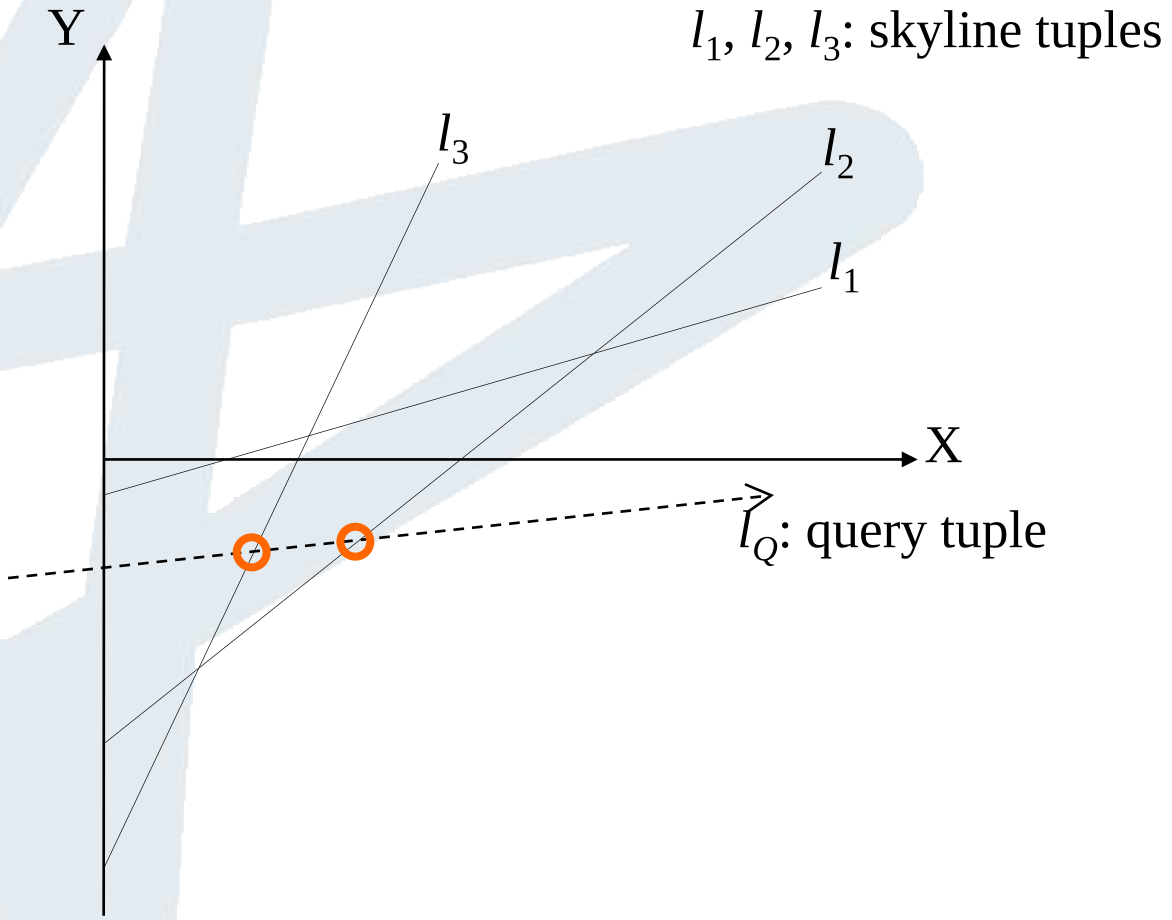
Skyline organization

- Main query: to check whether any tuple dominates a query tuple
- Each tuple mapped to a line

  $$y = r(a) \cdot x - r(b),$$

  where a, b are two arbitrarily attributes
- Skyline: a geometric arrangement
  - Only need to check lines intersecting with the query line on the positive half of the x-axis
  - Doubly-Connected-Edge-List (DCEL)

Progressive query of skyline tuples



Outline

- Introduction
- STARS
- STARS$^+$
  - Dominating Tuple
  - Empty Cell
  - Minmax
- SkyGrid
- Experiments
- Conclusion

STARS$^+$

- An improved STARS
- Drawbacks of STARS
  - Expensive exclusive dominance checking
  - Sparse buffer grid
  - Inefficient geometric arrangement
    - Arbitrary tuple-line mapping
    - Unknown pruning power
    - Quadratic space complexity

Exclusive dominance checking

- Required for every tuple in the buffer dominated by an expiring skyline tuple
- Each checking requires a query to skyline
- Many such queries in a single update when a skyline tuple expires

Dominating Tuple optimization

- Eager approach
  - When a tuple comes in, find all dominating skyline tuples
- Lazy approach
  - Defer computation until needed
- **Semi-eager approach**
  - Only remember one dominating tuple
  - Virtually no extra computation
  - "Dominating Tuple" optimization

Complexity analysis

- Time complexity for a given buffer tuple
- s: size of the skyline.
- Assume independent attribute values.
- Assume exclusive dominance checking is required for every expiring skyline.

	Lazy	Eager	Semi-eager
Exclusive dominance checking	O(s)	O(1)	O(ln(s))
Pre-computation overhead	O(1)	O(s)	O(1)

Sparse buffer grid

- Most of the tuples in the buffer are irrelevant
- Only SkyBuffer affects skyline
- The buffer grid is very sparse
  - Assume independent attribute values
  - SkyBuffer size: $O(\ln^d N)$
  - Density: $\rho = O(\ln^d N/g^d)$
  - E.g., $\rho = 0.022$ when $N = 10^5$, $d = 4$, $g = 30$
  - Most of the cells are empty

d: # of dimensions
g: # of buckets/dim
N: buffer size

A*STAR I²R

Empty Cell optimization

- Maintain d – 1 additional structures (index grids)
  - Keep track of # of tuples in the grid regions.
  - Each index grid C_i ($1 \le i \le d-1$) is i-dimensional
  - C_i maintains # of tuples in the regions identified by first i dimentions
- During Focused search
  - Candidate cells are examined by enumerating the cell coordinates
  - Early termination of the enumeration if an empty region is detected

Problems in geometric arrangement

- d > 2 is common in real life
- Arbitrary attribute selection in line mapping
  - Performance gap can exceed 20%
  - No heuristic to optimize this selection
- Only utilize two attributes for mapping
  - Intuitively, using more attributes is likely to provide better pruning power

Minmax optimization

- Consider a d-tuple $t = (a_1, \dots, a_d)$
- Minmax maps t to the line

  $y = C \cdot x - D$, where

  $C = \max(r(t.a_1), \dots, r(t.a_d))$,

  $D = \min(r(t.a_1), \dots, r(t.a_d))$
- Proven correctness for pruning lines
- An intuitively better heuristic
  - Utilize all attributes
  - Two extreme values may help prune more

Outline

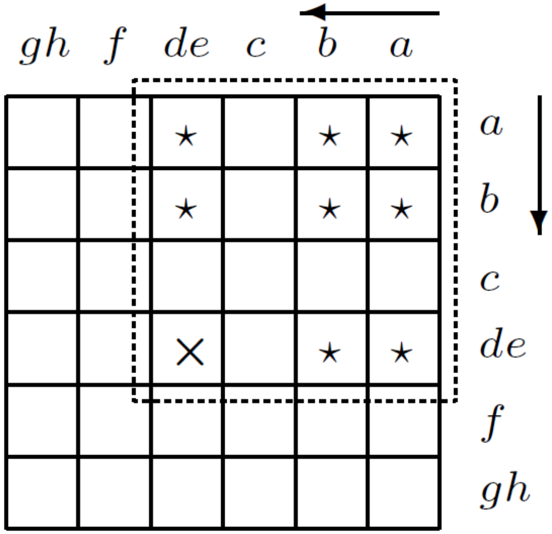
- Introduction
- STARS
- STARS$^+$
- SkyGrid
- Experiments
- Conclusion

Why geometric arrangement

- Pruning ratio
  - Assume independent attribute values
  - Suppose no “progressiveness”
  - Proven to be less than ½
- Space complexity
  - Quadratic space $O(s^2)$

SkyGrid

- Eliminate geometric arrangement
- SkyGrid: everything in one place
  - Both SkyBuffer and skyline in the same grid
  - Distinguish skyline with a status bit
  - Utilize two sets of index grids for Empty Cell optimization
- Allows focused search
  - On the other direction



Benefits of SkyGrid

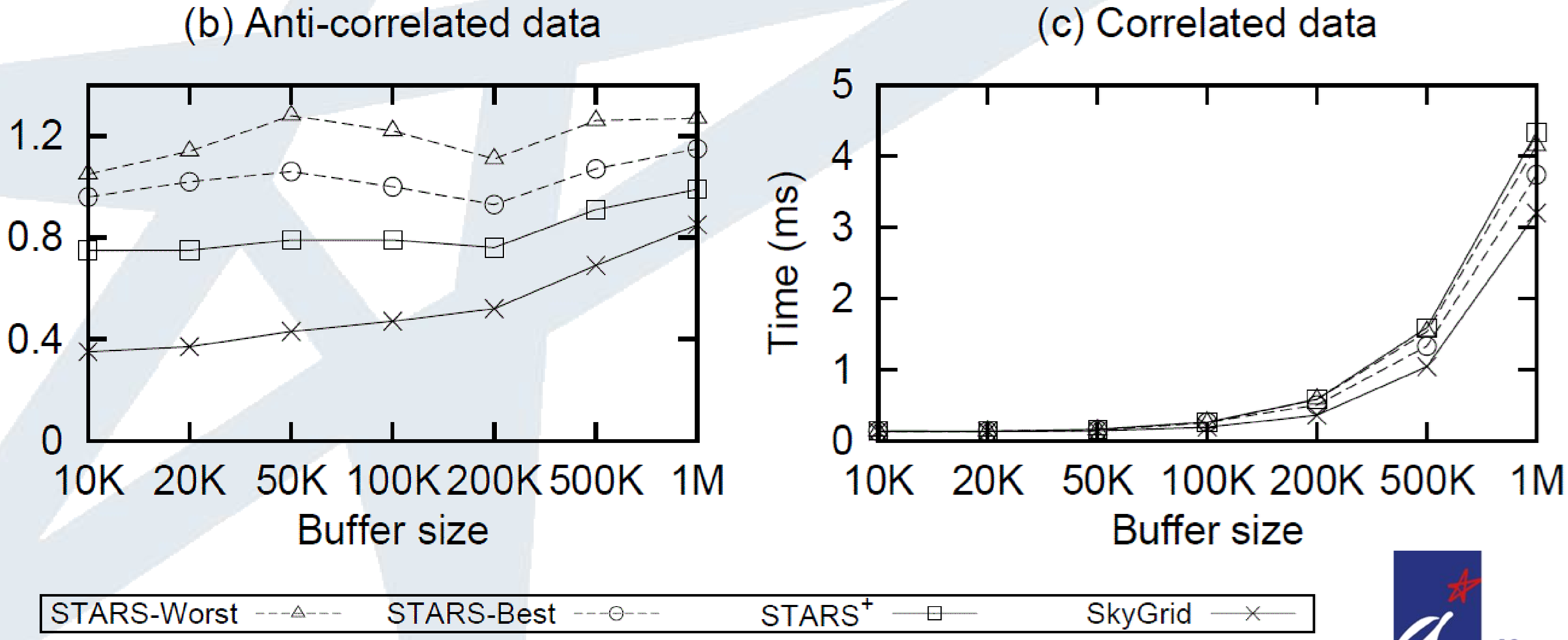
- Potentially higher pruning ratio
- No extra space required for skyline
  - Except for the linear requirement of status bit
- Simplified skyline maintenance
  - No manipulation of geometric arrangment (O(s))
  - Only update a status bit (O(1))

Outline

- Introduction
- STARS
- STARS+
- SkyGrid
- Experiments
- Conclusion

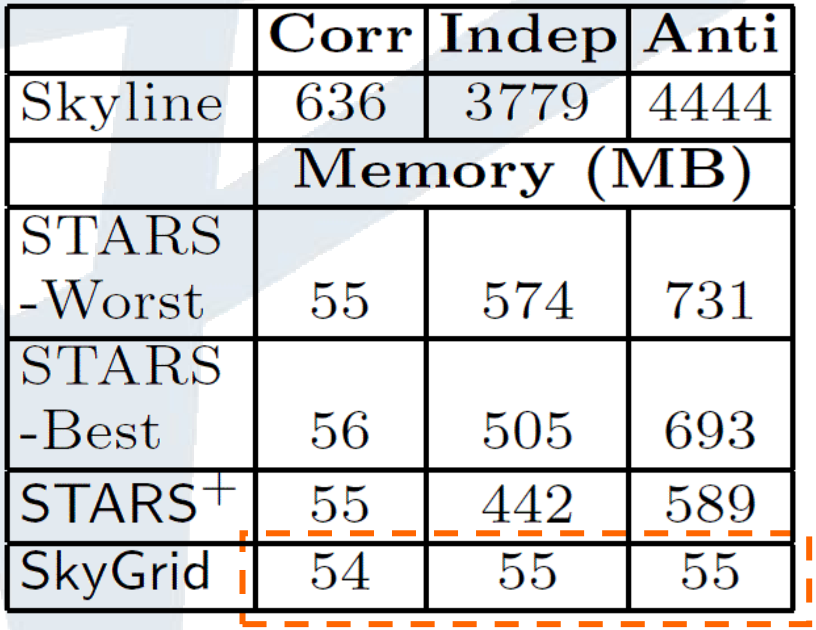
Amortized update time

- STARS-Best, STARS-worst, STARS$^+$, SkyGrid
- With two orthogonal optimizations in all (DT/EC)



Space requirement

- Buffer size = 100K
- SkyGrid uses the least memory



	Corr	Indep	Anti
Skyline	636	3779	4444
	Memory (MB)		
STARS -Worst	55	574	731
STARS -Best	56	505	693
STARS$^+$	55	442	589
SkyGrid	54	55	55

Outline

- Introduction
- STARS
- STARS$^+$
- SkyGrid
- Experiments
- Conclusion

Conclusion

- Two new approaches for streaming data on partially-ordered domains
  - STARS$^+$
  - SkyGrid
- Both outperforms STARS
- The surprising result: SkyGrid, being the simplest, is the best approach.

Questions?

References

1. S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In ICDE, pages 421–430, 2001.
2. C.-Y. Chan, P.-K. Eng, and K.-L. Tan. Stratified computation of skylines with partially-ordered domains. In SIGMOD, pages 203–214, 2005.
3. C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang. Finding k-dominant skylines in high dimensional space. In SIGMOD, pages 503–514, 2006.
4. Y. Fang, C.-Y. Chan. Efficient Skyline Maintenance for Streaming Data with Partially-Ordered Domains. In DASFAA, pages 322-336, 2010.
5. D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: an online algorithm for skyline queries. In VLDB, pages 275–286, 2002.
6. K. C. Lee, B. Zheng, H. Li, and W.-C. Lee. Approaching the skyline in Z order. In VLDB, pages 279–290, 2007.
7. X. Lin, Y. Yuan, W. Wang, and H. Lu. Stabbing the sky: Efficient skyline computation over sliding windows. In ICDE, pages 502–513, 2005.
8. D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In SIGMOD, pages 467–478, 2003.
9. D. Sacharidis, S. Papadopoulos, and D. Papadias. Topologically-sorted skyline for partially-ordered domains. In ICDE, 2009.
10. N. Sarkas, G. Das, N. Koudas, and A. K. Tung. Categorical skylines for streaming data. In SIGMOD, pages 239–250, 2008.
11. K.-L. Tan, P.-K. Eng, and B. C. Ooi. Efficient progressive skyline computation. In VLDB, pages 301–310, 2001.
12. Y. Tao and D. Papadias. Maintaining sliding window skylines on data streams. IEEE TKDE, 18(3):377–391, 2006.