Module 8: Miscellaneous Topics

Lecture 39: Skyline Queries

Prev topic Next topic Prev page Next page

The Lecture Contains:

Skyline queries

- Each object in database D has k' attributes
- Skyline query over $k \leq k'$ attributes
- Informally, return all objects in database that are not dominated by any other object

Skyline queries

- Each object in database D has k' attributes
- Skyline query over $k \leq k'$ attributes
- Informally, return all objects in database that are not dominated by any other object
- For each skyline attribute, a dominating function is defined
  - Can be any comparison operator, denoted by $\succ$
  - If it is at least equal, then it is denoted by $\succeq$
- An object O_p dominates another object O_q if

$$O_p \succ O_q \iff \forall i, O_{pi} \succeq O_{qi} \text{ and } \exists j, O_{pj} \succ O_{qj}$$

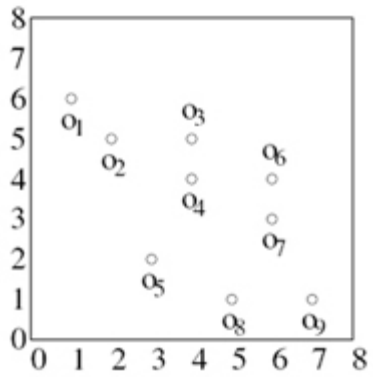
- The skyline query returns the set of objects S such that for all objects $O_p \in S$ and all objects $O_q \notin S$,

$$O_p \in S \iff \nexists O_q \in D, O_q \succ O_p$$

$$O_q \notin S \iff \exists O_p \in S, O_p \succ O_q$$

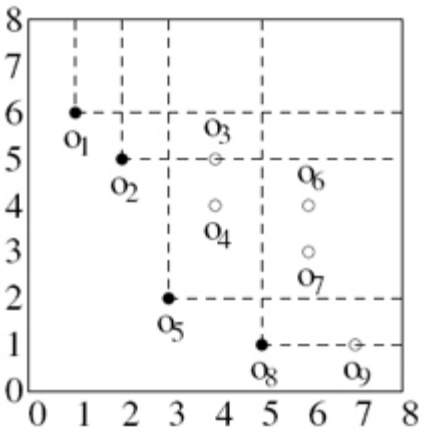
Explanation

- If object O_p dominates object O_q, then for *all* monotone preference functions, O_p is better than O_q
- Moreover, for every skyline object O_p, there exists a monotone preference f such that O_p optimizes f
- If preference function is *linear*, problem becomes *convex hull* computation



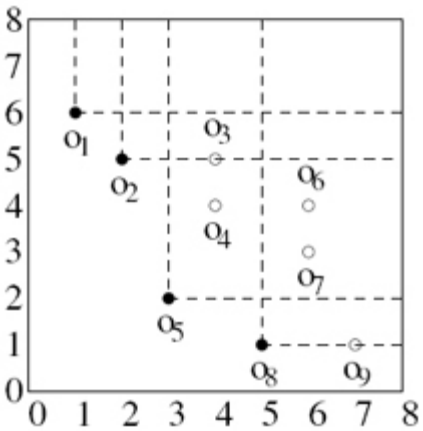
Explanation

- If object O_p dominates object O_q, then for *all* monotone preference functions, O_p is better than O_q
- Moreover, for every skyline object O_p, there exists a monotone preference f such that O_p optimizes f
- If preference function is *linear*, problem becomes *convex hull* computation



Explanation

- If object O_p dominates object O_q, then for *all* monotone preference functions, O_p is better than O_q
- Moreover, for every skyline object O_p, there exists a monotone preference f such that O_p optimizes f
- If preference function is *linear*, problem becomes *convex hull* computation
- Also known as Pareto curve or maximal vector problem



Block-nested-loop (BNL) algorithm

- Naïve algorithm
  - Compare each object with every other object
  - Only if no object dominates it, then it is a sklyline
  - $O(n^2)$

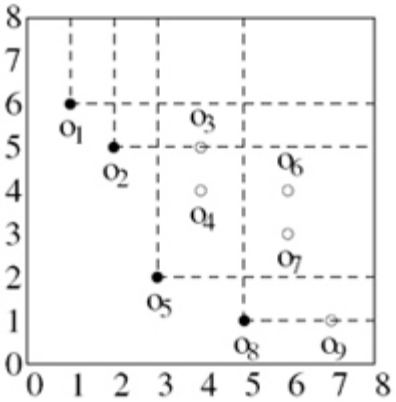
Block-nested-loop (BNL) algorithm

- Naïve algorithm
  - Compare each object with every other object
  - Only if no object dominates it, then it is a sklyline
  - $O(n^2)$
- Maintain a window of objects not *yet* dominated
- Check every new object against the list
  - If object is dominated, prune it
  - If object dominates some point(s) in window, prune those object(s)
  - If neither, add to window
- Window contains the skyline objects
- What if window does not t in memory?

Block-nested-loop (BNL) algorithm

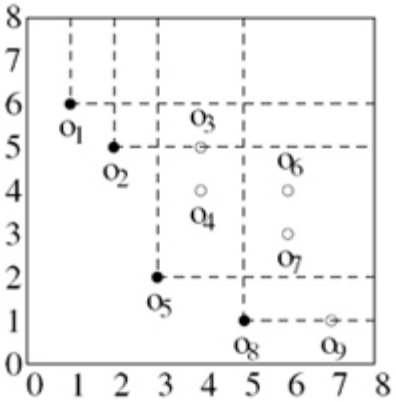
- Naïve algorithm
  - Compare each object with every other object
  - Only if no object dominates it, then it is a sklyline
  - $O(n^2)$
- Maintain a window of objects not *yet* dominated
- Check every new object against the list
  - If object is dominated, prune it
  - If object dominates some point(s) in window, prune those object(s)
  - If neither, add to window
- Window contains the skyline objects
- What if window does not t in memory?
  - Put into temporary le and note timestamp
  - Objects added afterwards are not guaranteed to be skylines
  - Make extra pass using the temporary le as input

Example



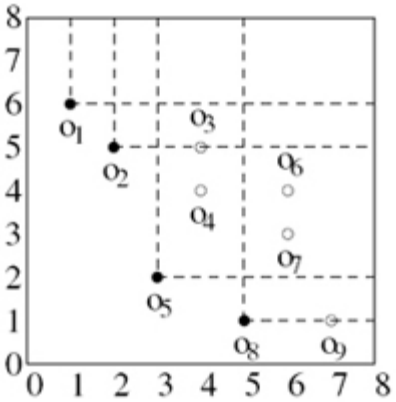
Window : Φ

Example



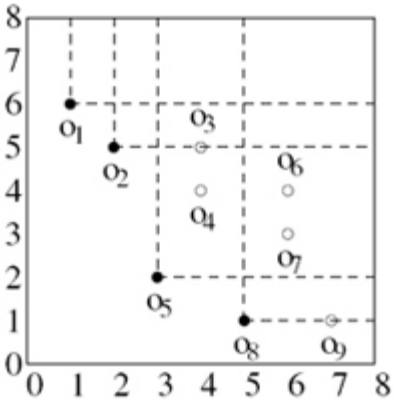
Window : O_1

Example



Window : O_1 O_2

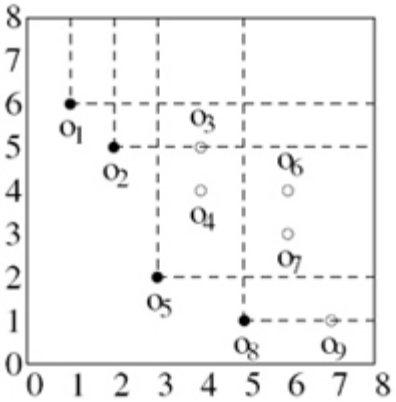
Example



Window : O_1 O_2

Pruned : O_3

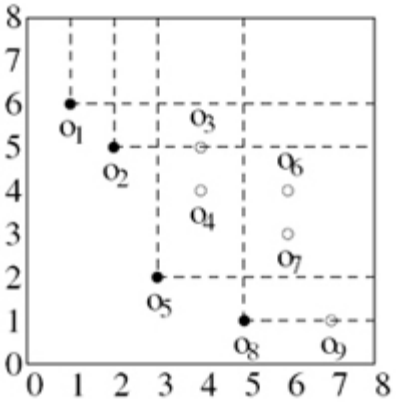
Example



Window : O_1 O_2 O_4

Pruned : O_3

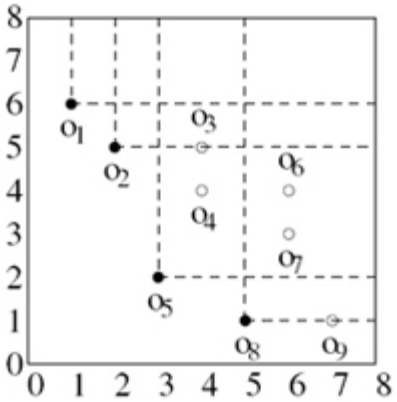
Example



Window : O_1 O_2 O_5

Pruned : O_3 O_4

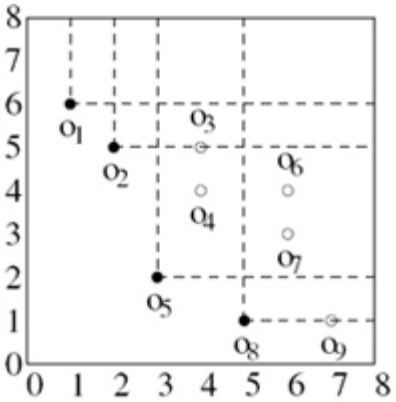
Example



Window : O_1 O_2 O_5

Pruned : O_3 O_4 O_6 O_7

Example

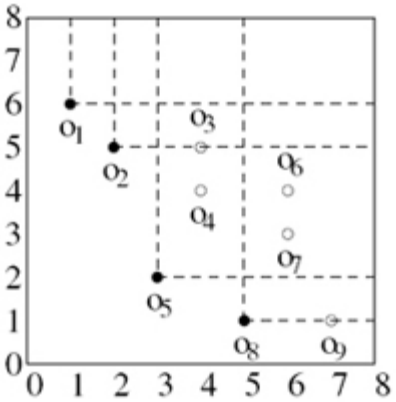


Window : O_1 O_2 O_5

Pruned : O_3 O_4 O_6 O_7

Temporary file : O_8

Example

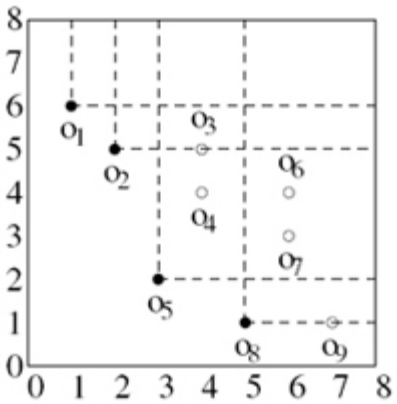


Window : O_1 O_2 O_5

Pruned : O_3 O_4 O_6 O_7

Temporary file : O_8 O_9

Example



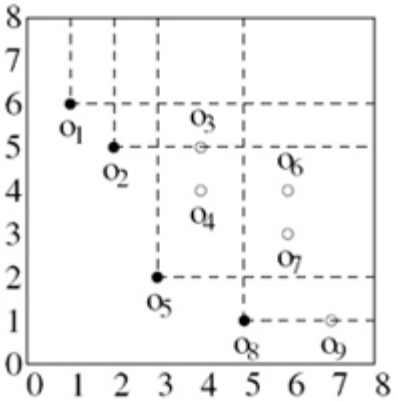
Window : O_1 O_2 O_5

Pruned : O_3 O_4 O_6 O_7

Temporary file : O_8 O_9

Next pass with O_8, O_9

Example



Window : O_1 O_2 O_5

Pruned : O_3 O_4 O_6 O_7

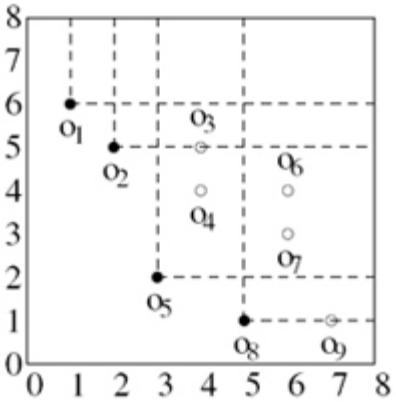
Temporary file : O_8 O_9

Next pass with O_8, O_9

Window : O_8

Pruned : O_9

Example



Window : $O_1\ O_2\ O_5$

Pruned : $O_3\ O_4\ O_6\ O_7$

Temporary file : $O_8\ O_9$

Next pass with O_8, O_9

Window : O_8

Pruned : O_9

Skyline set : $O_1\ O_2\ O_5\ O_8$

Sort-filter-skyline (SFS) algorithm

- Sort objects by some monotone scoring function
- Use the ordered list as input for BNL algorithm

Sort-filter-skyline (SFS) algorithm

- Sort objects by some monotone scoring function
- Use the ordered list as input for BNL algorithm
- Example : entropy function $E(O_i) = \sum_{j=1}^{d} \ln(O_{ij} + 1)$

Object	Value	Entropy
O_1	(1,6)	2.6390
O_2	(2,5)	2.8903
O_3	(4,5)	3.4011
O_4	(4,4)	3.2188
O_5	(3,2)	2.4848
O_6	(6,4)	3.5553
O_7	(6,3)	3.3321
O_8	(5,1)	2.4848
O_9	(7,1)	2.7725

Sorted order produces $O_5, O_8, O_1, O_9, O_2, O_4, O_7, O_3, O_6$

Produces much better scanning order