Efficient Execution Plans for Distributed Skyline Query Processing

Paper by

Joao B. Rocha Junior, Akrivi Vlachou

Christos Dolkeridis, Kjetil Norvag

Presentation by

Ritika (15111036)

Swapnil Mhamane(15111044)

Published in EDBT/ICDT '11

System Overview

- Each server stores autonomously a fraction of the data.
  - Horizontal partition of data.
- All servers need to process the skyline query.
- Each server S_i can directly connect to any other server S_i.

Naïve Approach

- A skyline query can be initiated by any server (S_{org}).
- Skyline query is processed by sending the query to all servers S_i.
- Each server Si reports its local skyline set SKY_i to originator.
- S_{org} gathers local skyline set and computes global skyline set
- Use of transitivity property
- Single hop execution

Motivation

- Execution plan defines the order in which the individual skyline queries are processed on different servers
- Servers need not be contacted can be pruned
- Discarding some data points locally based on skyline points from preceding server
- Negative point : Local skyline queries on consecutive server is a blocking operation
- Observation :
  - There exist dependency between two servers
  - Optimizations by exploiting these dependencies.

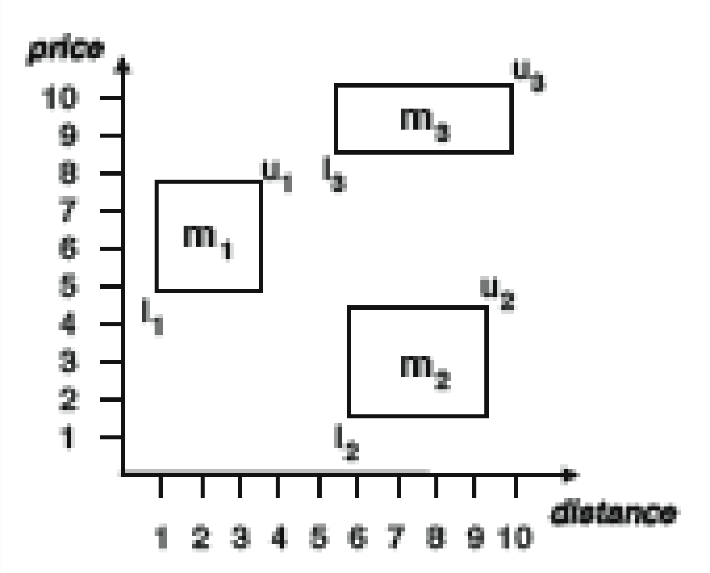
Relation among MBR

- A hyper-rectangle m_i (l_i, u_i).
- Given two hyper-rectangles m_i and m_j

  (1) m_i dominates m_j, if $u_i \prec l_j$;

  (2) m_i partially dominates m_j, if $l_i \prec u_j$, but $u_i \nprec l_j$;

  (3) m_i and m_j are incomparable, if $l_i \nprec u_j$ and $l_j \nprec u_i$.



- Enclosed dominance area V_{ij} of m_i on m_j : the volume of mj that is dominated by the lower left corner l_i of m_i
- Pruning power (PP_{ij}) of m_i on m_j :

  PP_{ij} = Vij/Vj

Main phases of SkyPlan

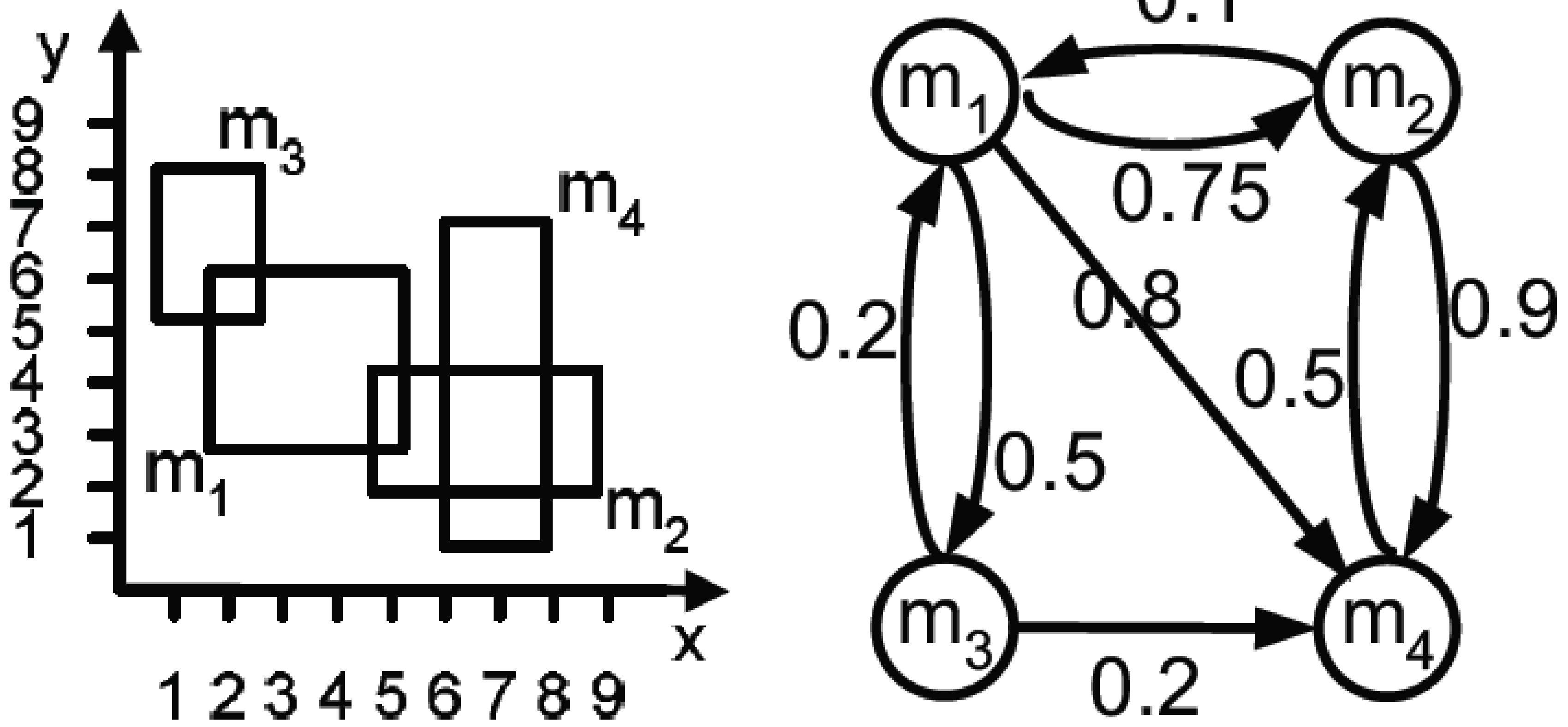
- Collect the MBRs of all servers.
- Builds a weighted directed graph
- The graph is transformed into an execution plan
- Execute plan recursively

Skyline dependency graph

- SD-graph : the weighted directed graph G(N, E,w)
- Each node n_i ∊ N corresponds to a non-dominated MBR m_i.
- E is a set of ordered pairs e_{ij} = (i, j), where m_i, m_j are MBRs, and m_i partially dominates m_j
- w is a weight function :
  - Normalized pruning power

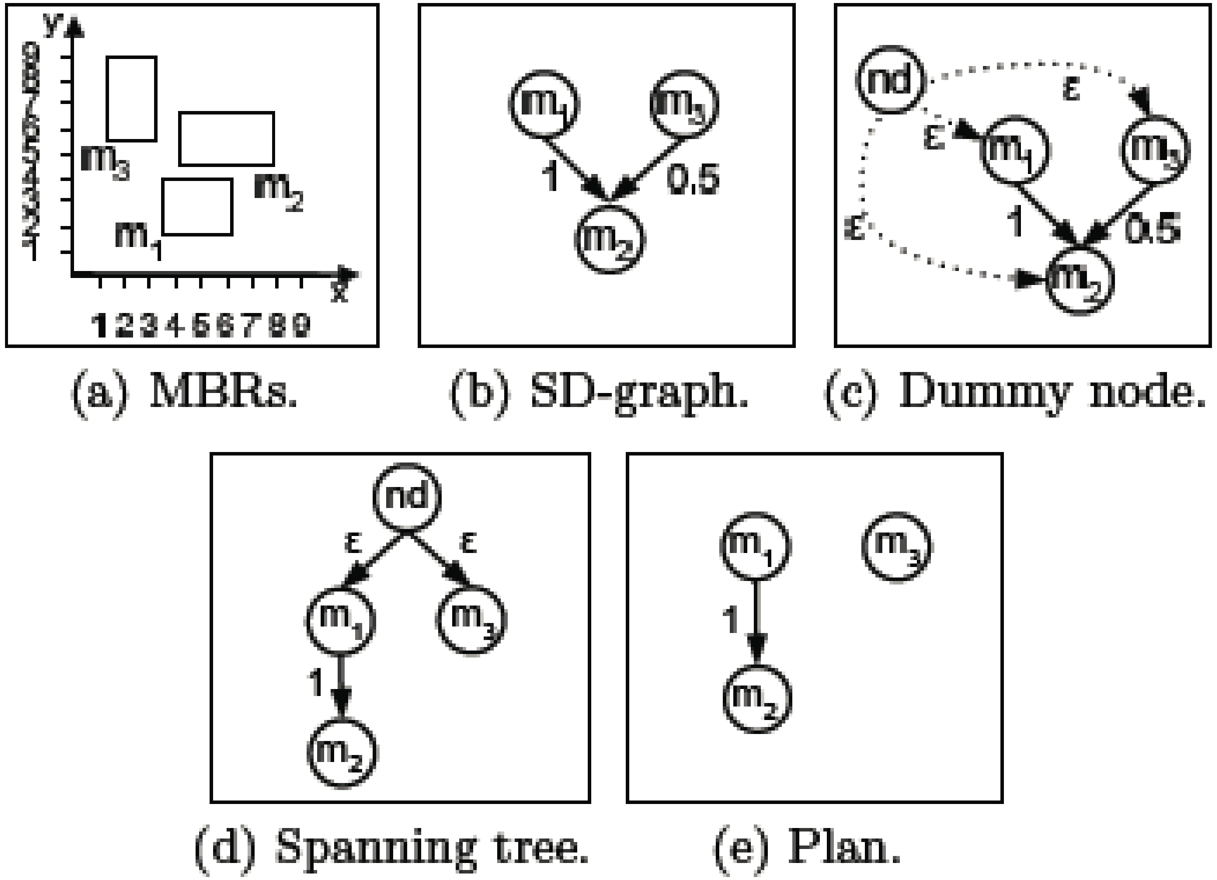
$$w_{ij} = |m_j|/|D| * PP_{ij}$$

SD-graph example



(a) Set of MBRs. (b) SD-graph.

Finding Maximum Spanning Tree



(a) MBRs. (b) SD-graph. (c) Dummy node.

(d) Spanning tree. (e) Plan.

Execution Plan

- The quality Q(P) of an execution plan P(N,E)

$$Q(P) \ = \sum e_{ij} \in E \ \ w_{ij}$$

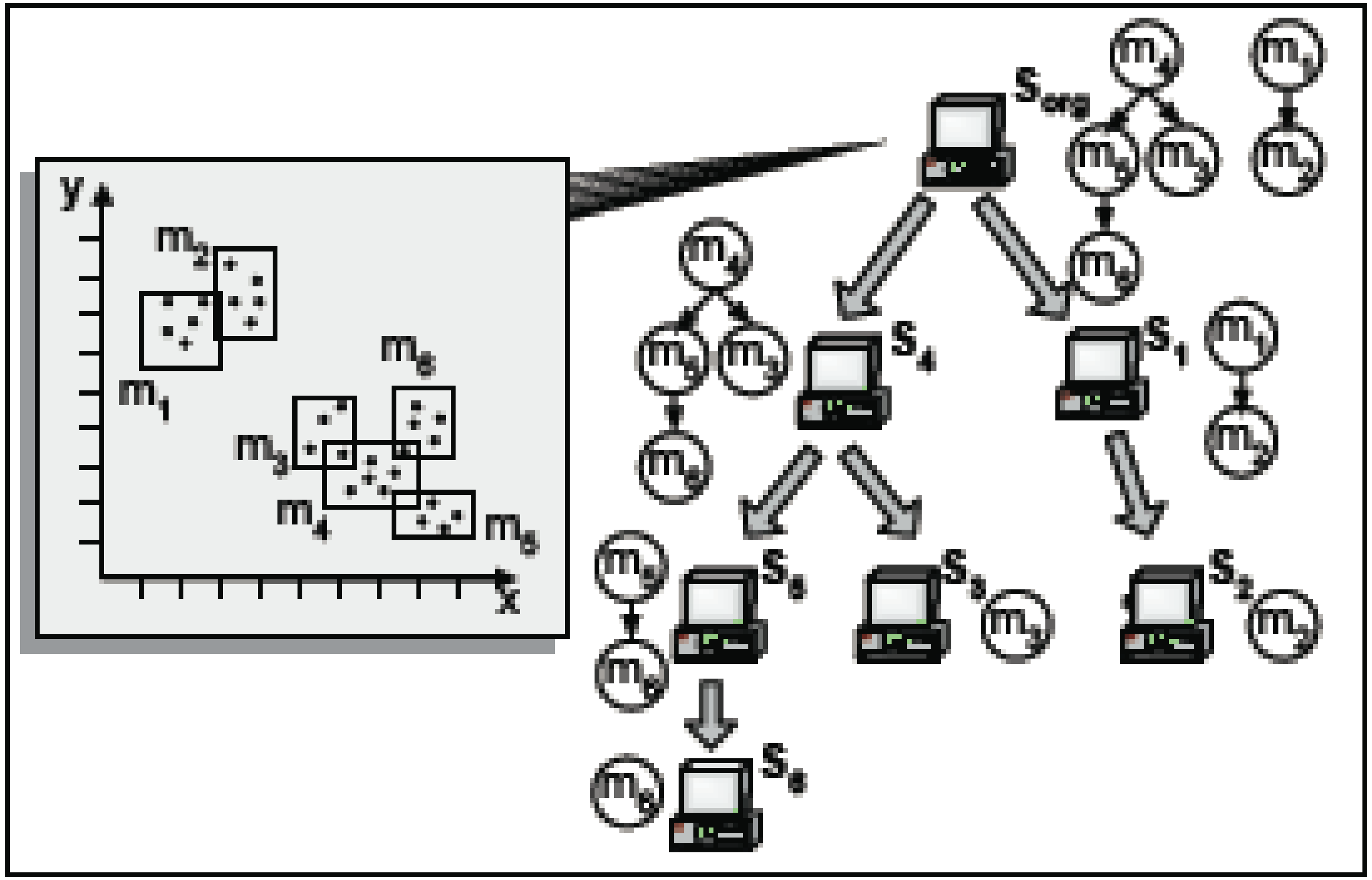
- Thus, an execution plan Pi is better than another plan Pj in terms of quality if: Q(Pi) > Q(Pj).

- Maximum pruning execution plan
- Execution plan : Set of directed weighted trees (Forest)
- Each tree is processed in parallel

Algorithm: QueryProcessing(Si,F,P)

```
INPUT: Filter points F = {f1, . . . , fk},Execution plan P.
OUTPUT: Local skyline
m ← P.getRootMBR()
sky ← computeSkyline(m)
P’ ← refinePlan(P, sky)
F’ ← refineFilters(F, sky)
S’ ← P’.getNextServers()
for (∀$S_j$ ∈ S’) do $sky_j$ ← QueryProcessing($S_j$ , F’,$P_j$‘)
sky ← mergeSkyline(sky, $sky_j$ )
return sky
```

Example



K-hop execution plan

- Another objective is to restrict the number of hops in query processing
    - Bounding latency in consecutive execution
- K-hop execution plan : Execution plan with height at most k
- Hop constrained maximum spanning tree problem
- **Algorithm** :
    - Find longest path in generated execution plan
    - Replace pruning power edge with other edge reducing length by at least 1
    - If there exist more than one such edges, choose edge with maximum pruning power
    - Repeat until length is reduced to k

Summary

- Paper targets the problem of deriving efficient execution plan for distributed skyline computation
- It proposes the novel framework called SkyPlan, that maps the dependencies between the queries into graph and generates cost aware execution plan.
- Aim was to maximize pruning power in consecutive queries while keep increment in parallelism
- It proposes distributed query execution mechanism that allows continuous refinement of plan during in-network query processing.

References

- João B. Rocha-Junior∗ , Akrivi Vlachou, Christos Doulkeridis, and Kjetil Nørvåg "Efficient execution plans for distributed skyline query processing"

Thank you ...!