Efficient Processing of Preference Queries in Distributed and Spatial Databases

João B. Rocha-Junior

Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, January 2012
Outline

• Introduction and motivation
• Research questions
• Contributions
  – skyline queries in distributed systems
  – top-k spatial preference queries
  – top-k spatial keyword queries
• Conclusion
Traditional vs. preference queries

- Traditional queries
  - hard constraints, returning the exact result set or nothing
    - e.g., “select hotels price < 50” [empty]
    - e.g., “select hotels price < 100” [too many]

- Preference queries
  - take into account the wishes of the users to produce only the most important results
    - e.g., “select the cheapest hotels, stop after 3”

<table>
<thead>
<tr>
<th>Hotel</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>120</td>
</tr>
<tr>
<td>B</td>
<td>60</td>
</tr>
<tr>
<td>C</td>
<td>90</td>
</tr>
<tr>
<td>D</td>
<td>70</td>
</tr>
<tr>
<td>E</td>
<td>90</td>
</tr>
<tr>
<td>F</td>
<td>75</td>
</tr>
</tbody>
</table>
Preference queries

- **Qualitative**
  - the preferences are specified *directly* on the attributes of the tuples (objects)
  - a is better than b if the preference holds
  - query studied: skyline queries

- **Quantitative**
  - the preferences are specified *indirectly* through a score function on the attributes of the objects
  - a is better than b if $score(a) > score(b)$
  - queries studied: top-k spatial preference queries and top-k spatial keyword queries
Focus of the thesis

• Efficient processing of preference queries in distributed and spatial databases
  – novel techniques that avoid evaluating all objects at query time

• Contributions
  – skyline queries in distributed systems
  – top-k spatial preference queries
  – top-k spatial keyword queries, including the road networks version
Skyline

• Query
  – consist of dimensions to be minimized or maximized

• Returns
  – the set of tuples not dominated by any other

• Example
  – database of hotels
  – each object is represented as a point in the space
  – hotel: \((price, distance)\)
  – **skyline of price [min], distance [min]**
Skyline queries in distributed systems

- Each server stores autonomously its data
  - requires aggregating the skyline of each server

- Applications
  - global hotel reservations system
  - teaching hospital networks

each server maintains its own data locally
Top-k spatial preference

• **Query**
  – spatial neighborhood (area)
  – features objects of interest
  – data objects of interest

• **Returns**
  – *k* best *data objects*
  – **rank**: highest score of *feature objects* inside the spatial area

• **Applications**
  – travel assistant
  – governmental planning

---

Example – *data objects of interest*: hotels
– *feature objects*: bars and cafés
– spatial neighborhood: range

- Hotel 1: *p*₁
  - Rank 1: *c₃* (0.2) – café
  - Rank 2: *c₁* (0.6) – café
  - Rank 3: *b₃* (0.0)

- Hotel 2: *p₂*
  - Rank 1: *c₁* (0.6) – café
  - Rank 2: *c₄* (0.8) – café
  - Rank 3: *b₂* (0.6)

- Hotel 3: *p₃*
  - Rank 1: *c₂* (0.4)
  - Rank 2: *b₄* (0.0)

---
Top-k spatial keyword

- **Query**
  - spatial location (lat, lng)
  - set of query keywords

- **Returns**
  - \( k \) best *spatio-textual objects*
  - **rank:** distance to the query location and relevance to the query keywords

- **Applications**
  - Twitter
  - Flickr
  - OpenStreetMap
Research questions

• How to improve the performance of
  – skyline queries in distributed environments?
  – top-k spatial preference queries?
  – top-k spatial keyword queries?
Outline

• Introduction and motivation
• Research questions
• Contributions
  – skyline queries in distributed systems
  – top-k spatial preference queries
  – top-k spatial keyword queries
• Conclusion
Efficient processing of skyline queries in distributed systems

- Two frameworks
  - AGiDS employs a grid-based data summary
  - SkyPlan employs efficient execution plans
Baseline approach\(^1\) (planning)

1) collects the MBRs from each server

2) defines an execution plan (e.g., sorting based on dominance)

---

\(^1\)Cui et al. “Parallel distributed processing of constrained skyline queries by filtering”. ICDE, 2008.
Baseline approach (executing)

Pros:
- reduces the communication cost
- reduces the processing cost

Cons:
- high latency, since opportunities for parallelism are not exploited
AGiDS

- A grid that captures the data distribution at each server
  - each grid is created uniformly
- Populated regions
  - relations: dominates, partially dominates, incomparable
- Region-skyline
- Good properties
  - regions are directly comparable
  - compact representation (bits)
AGiDS: Phase 1

1) computes the region-skyline

2) collects the region-skyline from each server

3) computes the global region-skyline
AGiDS: Phase 2

1) computes the skyline of the relevant regions

Query Originator

Server A

Server B

Server C
AGiDS: Phase 2

1) computes the skyline of the relevant regions
2) collects the relevant points
3) merges the related regions to obtain the global skyline

Pros:
- reduces the communication cost
- reduces the processing cost
- parallelism

Cons:
- bottleneck at the query originator
SkyPlan

• There must exist a *dependency* between consecutive servers in order to have a gain
  – the dependency is quantified in terms of *pruning power* (dominated volume)

• The dependences are complex and form a directed graph with cycles

• From the Graph, we produce a *plan* that maximizes the pruning power
  – supports also multi-objective plans

![Directed Graph Image]
SkyPlan: Framework (1)

1) collects the MBRs from each server

2) builds the graph incrementally

3) builds the execution plan
SkyPlan: Framework (2)

Query Originator

Server A

parallelism

efficient filtering

Server B

Server C
Result: AGiDS

- Efficient execution through parallelism
- Medium-scale distributed systems (50 servers)
Result: **SkyPlan**

- Efficient processing through *execution plans*
- Large-scale distributed systems (1000 servers)
Efficient processing of top-k spatial preference queries

- Novel materialization technique
- Algorithms to construct and search on the new indexes
• State-of-the-art$^{1,2}$
  – branch and bound and feature join algorithms

Motivation behind our idea...

• Few features objects are necessary to compute the score of a data object
  – feature not dominated by any other in terms of both distance and score

• Nice properties
  – small size in practice
  – sufficient to support the neighborhood conditions and query parameters
Mapping to distance-score space

![Diagram showing hotel, café, and pair (p1, c) and pair (p2, c) locations in distance-score space.](image-url)
Access to partial scores (range)
Query processing algorithm

• Queries with a single feature
  – only a single R-tree is accessed

• Queries with multiple features
  – require aggregating the partial scores of the objects retrieved from each feature R-tree

• Small modifications to support influence and nearest neighbor
Result

- Efficient processing through **materialization**
  - algorithms for efficient materialization/maintenance
Efficient processing of top-k spatial keyword queries

• Novel indexing technique
• Algorithms for efficient query processing
Current approaches

• Employ a modified R-tree\(^1,2\)
  – hybrid index combining spatial and textual indexes
  – each node keeps an abstract document representing all documents in the node sub-tree

• Problems
  – frequent and infrequent terms are stored in the same way
  – several nodes accessed due to text dimensionality

\(^1\)Cong et al. “Efficient retrieval of the top-k most relevant spatial web objects”. VLDB, 2009.
Distribution of terms (Zipf's law)

- The distribution of terms is very skewed
- Few hundred terms take up 50% of the text
Spatial Inverted Index (S2I)

permits accessing the objects in decreasing order of term relevance efficiently

permits efficient storage
Query processing algorithm

• Single keyword queries
  – only a single block or tree is accessed

• Multiple keyword queries
  – require aggregating the partial scores of the objects for each term of the query keywords
Results

- Efficient processing through **indexing**
  - experiments with real and synthetic datasets

![Graph showing time (milliseconds) vs. number of keywords for DIR-tree and S2I](image-url)
Efficient processing of top-k spatial keyword queries on road networks

- First time this query type is proposed
- Novel indexing and materialization techniques
Basic approach

Road network framework

Spatio-textual index

spatio-textual object
Basic approach

1) find the edge (road) of the query

Road network framework

2) find relevant objects on the edge

Spatio-textual index

2.1) spatial search
2.2) filtering objects
2.3) compute network distance
Enhanced approach

Advantages:
• efficient to find the relevant objects on the expanded edges
  - only relevant edges are evaluated
  - the network distance is already computed
  - no filtering is required

Problem:
• several edges are expanded
Overlay approach

Advantages:

- same good properties of the enhanced approach
- expands fewer edges
Results

- Efficient processing through **indexing** and **materialization**

![Graph showing the comparison of basic, enhanced, and overlay methods in terms of time (seconds) vs. number of keywords.](image-url)
Outline

• Introduction and motivation
• Research questions
• Contributions
  – skyline queries in distributed systems
  – top-k spatial preference queries
  – top-k spatial keyword queries
• Conclusion
Answering the research questions

How to improve the performance of

• skyline queries in distributed systems?
  – parallelism (AGiDS) ✓
  – efficient execution plans (SkyPlan) ✓

• top-k spatial preference queries?
  – indexing and materialization ✓

• top-k spatial keyword queries?
  – indexing (S2I) ✓

• top-k spatial keyword queries on road networks?
  – indexing and materialization ✓
Thanks!

João B. Rocha-Junior

joao@idi.ntnu.no