Multi-dimensional Skyline to find shopping malls

Md “Amir” Amiruzzaman
Suphanut “Parn” Jamonnak
Zhengyong Ren
Introduction

In market research predicting customer movement is very important. While customers to decide which shopping mall to go to depends on many uncertain or probabilistic factors, so, it is not easy to compute their movement ahead of time. However, with the help of uncertain or probabilistic data management, it is possible to compute customer choices with some certainty.

Skyline query?
Skyline

- Minimize price (x-axis)
- Minimize distance to beach (y-axis)
- Points not dominated by other points
- Skyline contains everyone's favorite hotel regardless of preferences
Problem statement

Formal (mathematical) definitions of problems
Let, $U = \{u_1, u_2, u_3, \ldots, u_n\}$ are users/customer who shops in different shopping malls, $S = \{s_1, s_2, s_3, \ldots, s_n\}$. However, users have preference which can be represented as keyword $K = \{k_1, k_2, k_3, \ldots, k_n\}$. 
The distance of each shopping malls are \( D = \{d_1, d_2, d_3, \ldots, d_n\} \), and price of products \( C = \{c_1, c_2, c_3, \ldots, c_n\} \), however, we may consider total cost or sum of products, i.e., \( \). Based on previous visits of each shop, probability to pick a shop can be denoted as, \( P = \{p_1, p_2, p_3, \ldots, p_n\} \).
Note that, different shops may sale different types of goods, $G = \{g_1, g_2, g_3, \ldots, g_n\}$, and facilities (e.g., restaurant, kids zone, bar, etc.) in each shopping mall may vary as well, $F = \{f_1, f_2, f_3, \ldots, f_n\}$.
If we are interested to know about a particular user (i.e., query user, $u_q$), then this problem can be represented as a multi-dimensional skyline problem. As such, shorter distance, lower cost, more facilities, higher variety of goods are desirable. Also, for the simplicity of the problem, we will consider higher probability or mostly visited shopping malls first (see Figure 1).
Objective

We would like to let the query user ($u_q$) to use a tool to find the shopping mall based on all the parameters we mentioned earlier. The tool will find a shopping mall using the multidimensional skyline query.

We also want to perform a user study to find how to improve user experience and usability of this proposed tool.
Literature Review

1. Probabilistic Skylines on Uncertain Data
   Jian Pei, Bin Jiang, Xuemin Lin, Yidong Yuan

2. Computing All Skyline Probabilities for Uncertain Data
   Mikhail J. Atallah, Yinian Qi

3. Skyline Query Processing for Uncertain Data
   Mohamed E. Khalefa, Mohamed F. Mokbel, Justin J. Levandoski
Probabilistic Skyline on Uncertain Data

*Very Large Data Bases (VLDB), 2007*
Example of Calculating Skyline Probability

- The probability $Pr(D)$ that $D$ is not dominated by other objects is given by:

$$\frac{1}{3} \times \left( (1 - \frac{1}{4}) + (1 - \frac{1}{4}) \times (1 - \frac{2}{3}) + (1 - \frac{1}{4}) \right)$$

$$= \frac{7}{12}$$
Main Contributions:

• The First Sub-Quadratic Algorithm for Computing All Skyline Probabilities
• New Probabilistic Skyline Analysis
• More General Uncertain Data Model
Probabilistic Skyline

• The probability for an instance to be a skyline point is called the instance’s skyline probability.
• The object’s skyline probability is the sum of the skyline probabilities over all its instances.
Table 1: Instance probabilities in Figure 3

For example, B has two instances b1 and b2. b2 is not dominated by any point, so its skyline probability is simply its own probability 0.4. For b1 to be a skyline point, none of the points that dominate b1 (i.e., a2, a4, b2, points in the rectangle) should exist. Hence its skyline probability is 0.2 * (1 - 0.2 - 0.1) = 0.14. The skyline probability of B is 0.54.
The Grid Method

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>number of all uncertain objects</td>
</tr>
<tr>
<td>$n$</td>
<td>number of all instances</td>
</tr>
<tr>
<td>$d$</td>
<td>number of dimensions</td>
</tr>
<tr>
<td>$O_i$</td>
<td>the $i$th uncertain object</td>
</tr>
<tr>
<td>$n_i$</td>
<td>number of instances of $O_i$</td>
</tr>
<tr>
<td>$S$</td>
<td>the set of all instances ($n =</td>
</tr>
<tr>
<td>$S_i$</td>
<td>the set of instances of $O_i$ ($n_i =</td>
</tr>
<tr>
<td>$p$</td>
<td>point(instance) in $S$</td>
</tr>
<tr>
<td>$Pr_{sky}(\cdot)$</td>
<td>skyline probability</td>
</tr>
<tr>
<td>$D_{S,i}(p)$</td>
<td>instances of $O_i$ in $S$ that dominate $p$</td>
</tr>
<tr>
<td>$\sigma_i(p)$</td>
<td>sum of probabilities of $O_i$’s instances that dominate $p$</td>
</tr>
<tr>
<td>$\beta(p)$</td>
<td>the probability that $p$ is not dominated by any instance of other object</td>
</tr>
</tbody>
</table>

\[
\beta(p) = \prod_{i=1, i \neq j}^{m} (1 - \sigma_i(p))
\]

\[
Pr_{sky}(p) = Pr(p) \cdot \beta(p)
\]

In Figure 3, instance b1 is dominated by instances a2, a4 and b2. Therefore, $\beta(b1) = 1 - (Pr(a2) + Pr(a4)) = 0.7$. 
$Pr_{sky}(b1) = Pr(b1) \cdot \beta(b1) = 0.14$
1. Process the horizontal grid lines:

\[ \sigma_i^*(p) = \sum_{p' \in S_i, p' < h_p} Pr(p') \]

**Example 7:** In Figure 4, \( p_1 \) is an instance of \( O_1 \) with probability 0.8, \( p_2 \) and \( p_4 \) are instances of \( O_2 \) with probability 0.5 each, \( p_3 \) and \( p_5 \) are instances of \( O_3 \) with respective probabilities 0.6 and 0.1. Then for \( p_4 \) on the horizontal line \( h_1 \), \( \sigma_1^*(p_4) = \sigma_2^*(p_4) = 0 \) while \( \sigma_3^*(p_4) = 0.6 \).
2. Process the vertical grid lines

\[
\sigma_i(p) = \sigma_i^*(p) + \sigma_i(p') + \begin{cases} 
Pr(p') & \text{if } p' \in S_i \\
0 & \text{otherwise}
\end{cases}
\]

Example 8: To compute \(\sigma_i(p_4)\)'s from \(\sigma_i^*(p_4)\)'s computed in Example 7, we follow Equation 5 (take \(i = 3\) for example):

\[
\sigma_3(p_4) = \sigma_3^*(p_4) + \sigma_3(v) + 0 \\
= Pr(p_3) + \sigma_3^*(v) + \sigma_3(p_5) + Pr(p_5) \\
= 0.6 + 0 + \sigma_3^*(p_5) + 0.1 = 0.7
\]

Similarly, we compute \(\sigma_1(p_4) = 0.8\), \(\sigma_2(p_4) = 0.5\).
This paper proposes an efficient framework that supports skyline queries for uncertain data represented as a continuous range.
Uncertainty reduction

• An ordered pair of objects (Q, P) qualifies to uncertainty reduction only if the endpoint of Q dominates the endpoint of P.

• Reduce the upper bound probability for an object P by removing a portion of its uncertainty range which would have a zero probability being a skyline object.
Example: For object V, as the endpoint of the uncertainty range eV does not dominate any other endpoint, V doesn’t have any uncertainly reduction.

For object R, eR dominates eU, eT, and eS, so, the pairs (R,U), (R,T), and (R,S) qualify for uncertainty reduction. This results in reducing the uncertainty range of U to be [2-5] instead of [2-6]. Since the reduced range is one quarter of the original range, the upper bound probability of U is set to 75%.

Respectively. Figure 2b gives the result of all points after the uncertainty reduction with their upper probability bounds, pruning objects S, T and U.
Implementation
Data collection

1. Collect data from [http://www.shoppingcenters.com](http://www.shoppingcenters.com) with detail report for each shopping mall (focused in Cleveland/Akron areas)

2. Manually fill in the spreadsheet

3. Selecting attributes:
   a. Shopping Mall Name (Text)
   b. Shopping Mall Code (ID)
   c. Stores (Number)
   d. Parking Space (Number)
   e. Household Income (Number)
   f. Population (Number)
   g. Food Court (Yes/No)
   h. Facilities and Categories (Total Sum)

4. Total: 90 Shopping Malls (Remove missing data)
Data Preparation and Preprocessing

1. Generate geolocation (Latitude and Longitude) for each shopping mall
2. Apply indexes with 2-dimensional points for each shopping mall
3. Import spreadsheet to PostgreSQL Database using PgAdmin4
4. DB name = “Shopping Mall” with 1 table as global view
Methodologies and Design

1. Web-Interfaces
2. Front-End: HTML and JavaScript
3. Styles: Bootstrap
4. Back-End: PHP
5. External Library:
   a. Google Maps API
   b. Google Direction Services API
   c. Some pre computation libraries
6. Database: PostgreSQL
System Workflow

1. User-Interaction with marker on map
2. Precompute possible driving distance to each shopping mall based on user-selected location
3. Perform skyline-operator in term of regular SQL-Command
4. Draw top shopping mall markers with its content and visualize it with google map API
5. Visualization of top shopping mall by markers

Database: PostgreSQL
API: Google Map API
Results: Top Shopping Mall Results
Dynamic Location and Preference Inputs

1. User Location
   a. Interaction: Dragging marker over map

2. User Preferences
   a. Anchor: Walmart, Giant Eagles, Target
   b. Services: Chase Bank, NTB Car Repair
   c. Miscellaneous: Yankee Candle, Toy R Us
   d. Hi-Tech: At&t, Time Warner, Gamestop
   e. Foods and Restaurants
   f. ...

3. User input will **dynamically** change the SQL command of Sky-line Query
Skyline-Query Methods

No shopping mall better than another on every criteria.

While no one best shopping mall, we want to **eliminate shopping mall** which are worse on all criteria. In this case is “s2”

<table>
<thead>
<tr>
<th>Shopping Mall</th>
<th>Stores Number</th>
<th>Parking Space</th>
<th>Household Income</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>50</td>
<td>20</td>
<td>$20,000</td>
<td>10000</td>
</tr>
<tr>
<td>s2</td>
<td>20</td>
<td>0</td>
<td>$50,000</td>
<td>20000</td>
</tr>
<tr>
<td>s3</td>
<td>40</td>
<td>100</td>
<td>$30,000</td>
<td>7000</td>
</tr>
<tr>
<td>s4</td>
<td>60</td>
<td>40</td>
<td>$40,000</td>
<td>8000</td>
</tr>
<tr>
<td>s5</td>
<td>30</td>
<td>50</td>
<td>$45,000</td>
<td>5000</td>
</tr>
</tbody>
</table>
Skyline-operator

- **Skyline Operator**
  - SELECT * FROM global
    SKYLINE OF Distance MIN,
    Stores Number MAX,
    Parking Space MAX,
    Household Income MIN,
    Population MIN, ...

Can we write SQL query without using Skyline operator?
Skyline Implementation in N-dimension

- There are several Skyline-Algorithms presented
- In our project, we Implement a regular SQL query with Skyline Operator:
  - `SELECT *` FROM ShoppingMall S
    WHERE NOT EXISTS ( SELECT * FROM ShoppingMall S1 
      AND S1.Distance <= S.Distance 
      AND S1.StoresNum >= S.StoresNum 
      AND S1.ParkingSpace >= S.ParkingSpace 
      AND ( S1.Distance < S.Distance OR 
        S1.StoresNum > S.StoresNum OR 
        S1.ParkingSpace > S.ParkingSpace ) );
- This SQL query is equivalent to previous example but without skyline operator
- After generate the result, we sort it by distance in descending order
Block-nested Loop (BNL) Algorithms

- Block Nested Loop
- Compare each tuple with one another
- Window in main memory contain best tuple
- Write to temp file (if window has no space)
- Implement in javascript and compare it with SQL-Skyline result
- pick an optimal shopping malls.
BNL

Window (main memory)

<s1; 1 miles, 20, 200>
Drop s1
<s2; 0.5 miles, 30, 200>
<s2; 0.5 miles, 30, 200>
<s2; 0.5 miles, 30, 200>
<s3; 2 miles, 10, 100>
<s4; 1 miles, 50, 100>

Data Files (tuple)

<s1; 1 miles, 20, 200>
Write s1
<s2; 0.5 miles, 30, 200>
Write s2
<s2; 0.5 miles, 30, 200>
Drop s3
<s3; 2 miles, 10, 100>
Write s4
<s4; 1 miles, 50, 100>

Compare tuple write it when empty window
Evaluation Result

- Select 10 Participants
- Rating 1 to 5 based on our ranking result
- Collect comment and feedback
- Will include this part in final report
Conclusion

- There are several Skyline-Query algorithms out there, we found that our SQL command and BNL methods is cumbersome, expensive to evaluate, and huge result set.
- Both BNL and SQL command need to improve
  - E.g. Create Self-Organizing list for BNL algorithms
- Our system works with dynamic user input
- Future Work
  - Implement more Skyline algorithms (R-tree, Divide and Conquer, K-NN)
  - Evaluate and summarize which algorithm is the best to rank shopping mall from our dataset.
  - Perform user study with the domain experts