The Spatial Skyline Queries

Instructor: Cyrus Shahabi
Outline

• Motivation
• Problem Definition
• Related Work
• Geometric Properties
• Our Algorithms: VS$^2$ and B$^2$S$^2$
• Performance Evaluation
• Conclusion and Future Work
• **Problem:** Finding Hotels close to Airport, Beach, and Conference
• **Query:** What are the candidate *interesting* hotels?
  – A skyline query with dynamic spatial attributes …
  – **Criteria for an interesting hotel:** No hotel is *closer* than a candidate hotel to A, B, and C
    • No hotel is better than a candidate hotel in terms of all distances to A, B, and C (i.e., 3 query functions to be optimized together)

• **Applications:** Trip Planning, Crisis Management, Defense and Intelligence, Wireless Sensor Networks
Problem Definition

\textbf{Spatial Skyline Query (SSQ):} find the data points \( p_i \) that are \textbf{not} spatially dominated by any other point \( p_j \) with respect to the given query points (here, \( p_2 \) and \( p_4 \)).

\begin{itemize}
  \item Data \( P = \{p_1, p_2, p_3, p_4\} \)
  \item Query \( Q = \{q_1, q_2\} \)
  \item Distance \( D() = \text{Euclidean} \)
  \item \( p_2 \text{ spatially dominates } p_1 \) with respect to \( \{q_1, q_2\} \)
  \item Dominator Region of \( p_1 \)
  \item \( p_1 \text{ spatially dominates } p_3 \)
  \item Dominance Region of \( p_1 \)
  \item No dominance relation between \( p_1 \) and \( p_4 \)
\end{itemize}

\( p_1 \text{ spatially dominates } p_2 \) with respect to \( Q \) iff

\[ D(p_1, q_i) \leq D(p_2, q_i) \quad \text{for all } q_i \text{ in } Q \text{ and} \]

\[ D(p_1, q_j) < D(p_2, q_j) \quad \text{for at least one } q_j \]
Related Work

- **General Skyline Query**
  - BNL and D&C, Börzsönyi et al., ICDE’01
  - Bitmap and Index, Tan et al., VLDB’01
  - NN, Kossmann et al., VLDB’02
  - SFS, Chomicki et al., ICDE’03
  - BBS, Papadias et al., SIGMOD’03
    - *Static attributes vs. dynamic spatial attributes in SSQ*
    - *SSQ is a dynamic skyline query*

- **Nearest Neighbor Search**
  - ANN, Papadias et al., TODS 2005, 30(2)
    - *Looks for subsets of spatial skyline points*

- **NN and Skyline**
  - Huang and Jensen, W2GIS’04
    - *Each point-of-interest has 2 dimensions: minimum distance to query point and minimum detour to pre-defined route ➔ dynamic skyline*
    - *Limited setting*
    - *Uses naïve in-memory skyline computation*
**Naïve Solution**

- Data $P = \{p_1, p_2, p_3, p_4\}$
- Query $Q = \{q_1, q_2\}$
- Distance $D() = $ Euclidean

For each point $p_i$
iterate over points $p_j$
if no point spatially
dominates $p_i$ then add $p_i$ to
spatial skyline

**Dominance check?**

\[
D(p_2, q_1) \leq D(p_1, q_1) \quad \text{AND} \quad D(p_2, q_2) \leq D(p_1, q_2)
\]

**Time Complexity:** $O(|P|^2 \cdot |Q|)$

$|P|$: number of data points, $|Q|$: number of query points
Problem Definition

• Naïve approach
  – **Complexity:** $O(|P|^2 |Q|)$
    
    $|P|$: number of data points, $|Q|$: number of query points

• **Why a new algorithm is needed:**
  – Complexity of Naïve approach is high
    • Each dominance check involves $2|Q|$ distance computation operations: increases with more query points
  – General skyline algorithms are either inapplicable or inefficient
    • Due to **dynamic spatial** attributes
  – Optimization opportunity
    • The geometric properties of space can be exploited
Geometric Properties

• Complexity of Naïve approach: $O(|P|^2 |Q|)$
  – $|P|$: number of data points
  – $|Q|$: number of query points

• We identify geometric properties to reduce this complexity by reducing the number of:
  – data points to be investigated
  – query points that has no effect on the result

• Less and cheaper dominance checks
• We identify three properties ...
Preliminaries: Voronoi Diagrams

• Given a set of spatial objects, a Voronoi diagram *uniquely* partitions the space into disjoint regions (cells).

• The region including object $p$ includes all locations which are closer to $p$ than to any other object $p'$.

**Ordinary Voronoi Diagram**

**Dataset:**
- Points

**Distance $D(.,.)$:**
- Euclidean ($L_2$)

Point $q$ inside the cell of $p$

$D(q, p) \leq D(q, p')$
Geometric Properties

\textbf{GP}_1: \textit{Any point }p\textit{ inside the convex hull of query points }Q\textit{ is a spatial skyline point.}

\textbf{Intuition:} circles defining the dominator region of }p\textit{ intersect only at }p\textit{.}
**GP_2:** The set of skyline points does not depend on any query point \( q \) inside the convex hull of query points \( Q \).

**Intuition:**
Circle corresponding to \( q_4 \) does not change the dominator region of \( p \).
**Geometric Properties**

**GP₃**: Any point \( p \) whose Voronoi cell **intersects** with the convex hull of \( Q \) is a spatial skyline point.

**Intuition:** any point inside \( \text{CH}(Q) \) (including parts of \( \text{VC}(p) \)) should be closer to \( p' \) that dominates \( p \) -> contradiction.
Algorithm: VS²

- VS²: Voronoi-based Spatial Skyline Algorithm
- Utilizes the geometric interpretation of the skyline
  - With **no dominance check**, adds any data point $p$ whose Voronoi cell intersects with the convex hull of $Q$
  - Performs **cheaper dominance check only on a small subset** of points (neighbors of skyline points $\sim O(S)$)
- Traverses the **Voronoi Diagram**\* of data points

\* Delaunay Graph
Algorithm: VS²

• We check the top of heap when all of its neighbors are already in the heap.
• No dominance check so far …
• Check with only the current spatial skyline points

Contents of the heap

Top of the heap
Algorithm: VS$^2$

- Traversal stops before reaching the dominance region of the current skyline set.
- We check only a small number of non-skyline points.
Algorithm: VS$^2$

- **Time Complexity:** $O(|S|^2 \cdot |CH_v(Q)| + \Phi(|P|))$
  - Naïve: $O(|P|^2 \cdot |Q|)$

- $|S|$: number of skyline points

- $|CH_v(Q)|$: number of vertices of the convex hull of $Q$ ($\leq |Q|$)

- $\Phi(|P|)$: complexity of finding the data point from which VS$^2$ starts traversing inside the convex hull of $Q$ ($O(\log(|P|))$ with point location or $O(|P|^{1/2})$)

- **Space Complexity:** $O(|P|)$
  - Space required for ordinary Voronoi Diagram is $O(|P|)$
Algorithms: $B^2S^2$

- **$B^2S^2$:** Branch-and-Bound Spatial Skyline Algorithm
- Customization of BBS [Papadias et al.] for SSQs
- Uses *some* of the geometric properties of the skyline ($GP_1$ and $GP_2$)
- Similar to BBS traverses an R-tree on data points
- *Traversal order:* specified by any monotone function (e.g., $\text{mindist}(p, CH_v(Q))$)
Performance Evaluation

• **Dataset:** USGS including one million locations
• R*-tree on data points for BBS and B²S²
• Pre-built Delaunay graph of data points for VS²
• Max $\text{MBR}(Q) = 0.3\%$

• The difference in improvement of $\text{VS}^2$ over BBS increases for larger query sets.
Variations of $B^2S^2$ require less dominance checks than BBS.
Note that each dominance check is cheaper in our $VS^2$ and $B^2S^2$ algorithms.
Performance Evaluation

\[ \text{CPU cost (sec)} \]

- Max \( |\text{MBR}(Q)| = 0.5\% \), \(|Q| = 6\)
- \( \text{VS}^2 \) is also scalable with respect to the density of data (i.e., number of skyline points)
Conclusion and Future Work

• We introduced the spatial skyline queries.
• We exploited the geometric properties of its solution space.
• We proposed two algorithms:
  – $B^2S^2$ that uses our properties to customize BBS for SSQs
  – $VS^2$ that utilizes a Voronoi diagram to minimize the number of dominance checks
• We proposed two variations of $VS^2$ for:
  – continuous spatial skyline query
  – handling non-spatial attributes
• $VS^2$ significantly outperforms its competitor approach BBS.

Future Work
• Addressing SSQ in other spaces
• Studying variations of SSQ
References