Efficient Algorithms for Constrained Subspace Skyline in Structured Peer-to-Peer Systems

Khaled M. Banafaa and Ruixuan Li
Huazhong University of Science and Technology
Outline

- Background
- Motivation and related work
- Constrained subspace skyline
  - Partitioning and distribution
  - Algorithms
- Experimental evaluation
- Conclusion
Skyline

- **A Skyline is the set of all non-dominated tuples**

**Formal Definition**

Skyline $S$ of $T$ tuples is

$$\{ s \in T : \neg \exists t \in T, \forall i \in [1..k] t_i \geq s_i \text{ and } \exists i \in [1..k] t_i > s_i \}$$

Where $t_i$ and $s_i$ are the values of the $i^{th}$ column of tuples $t$ & $s$ respectively.
Constrained Subspace Skyline

Definition

- Let
  - S={d1,d2,…,dn} be a d-dimensional space
  - PS be a set of points in S
  - p={p1,p2,…,pd} a point with d dimensions
  - S’⊆S, S’ is a subspace of S
  - C={c1,c2,…,ck} is a range constraints on S’, expressed by [ci,min, ci,max]

- A point p∈PS dominates q∈PS (p <s' q) on a subspace S’ if
  1) on every di∈S’ pi≤qi ; and 2) on at least one dimension dj∈S’, pj<qi

- The skyline of a subspace S’ is a set PS’⊆PS which are not dominated by any other point on the subspace S’

  PS’={p∈PS| ¬∃q∈PS:q <s’ p}

- A constrained subspace skyline for S’⊆S refers to a set of points

  PS’c = {p∈PSc| ¬∃q∈PSc:q <s’ p}, where PSc ⊆PS and PSc={p ∈PS| ∀di ∈S’:ci,min ≤pi ≤ci,max}
Motivation

• Various users may issue queries with different subsets of attributes due to their interests and constraints

• An example:
  ◦ A customer may be sensitive only on price and mileage regarding within some ranges while a car database may contain many attributes of cars, such as price, mileage, horsepower, age and fuel consumption
## Related Work

<table>
<thead>
<tr>
<th>Approach</th>
<th>Table: Different skyline query variants supported by distributed approaches (X: proposed for, O: also supports)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSL [18]</td>
<td>Structured overlay does not support both constrained and subspace skyline.</td>
</tr>
<tr>
<td>SSP/Skyframe [2, 3]</td>
<td>Unstructured overlay can support both constrained and subspace skyline but need contact all peers.</td>
</tr>
<tr>
<td>iSky [4, 5]</td>
<td></td>
</tr>
<tr>
<td>SSW [6]</td>
<td></td>
</tr>
<tr>
<td>SFP [7]</td>
<td></td>
</tr>
<tr>
<td>DDS [8, 9]</td>
<td></td>
</tr>
<tr>
<td>SKYPEER/SKYPEER+ [10, 11]</td>
<td></td>
</tr>
<tr>
<td>BITPEER [12]</td>
<td></td>
</tr>
<tr>
<td>PaDSkyline [13, 14]</td>
<td></td>
</tr>
<tr>
<td>AGiDS [15]</td>
<td></td>
</tr>
<tr>
<td>FDS [16]</td>
<td></td>
</tr>
<tr>
<td>SkyPlan [17]</td>
<td></td>
</tr>
</tbody>
</table>
Our Objectives

- Using a simple network overlay such as Chord or Baton
- Accomplish full/subspace skyline queries efficiently
  1. Decrease visited peers
  2. Increase bandwidth saving
  3. Preserve progressiveness
  4. Exploit parallelism (while preserving progressiveness)
Baseline

- We chose subsky [1] as a baseline for our work
  - It used Chord structure
  - It used a share-nothing structure
  - It did not support progressiveness
  - No parallelism
- It was modified to consider subspaces
I - The data space is partitioned and Cells are given z-order addresses

2 - Partitions assigned to peers of a Chord or a Baton
**Query Traversal and Example**

Peers’ Traversal depends on query subspace and constraints.

1. **Full space**
   - Peers are pruned when dominated.
   - Peers outside the range are not visited.

2. **Subspace query**
   - Peers are pruned when dominated.
   - Peers outside the range are not visited.

---

<table>
<thead>
<tr>
<th>Peers</th>
<th>CPU-performance</th>
<th>Traversal for subspace query</th>
<th>Price</th>
<th>Traversal for subspace query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z-address</td>
<td>Bits</td>
<td>CPU-performance (Z-address)</td>
<td>Bits</td>
<td>Price(Z-address)</td>
</tr>
<tr>
<td>0000</td>
<td>00</td>
<td>00 (0000)</td>
<td>00</td>
<td>00 (0000)</td>
</tr>
<tr>
<td>0000</td>
<td>00</td>
<td>00 (0001)</td>
<td>00</td>
<td>00 (0001)</td>
</tr>
<tr>
<td>0000</td>
<td>01</td>
<td>01 (0010)</td>
<td>00</td>
<td>00 (0100)</td>
</tr>
<tr>
<td>0000</td>
<td>01</td>
<td>01 (0111)</td>
<td>00</td>
<td>00 (0110)</td>
</tr>
<tr>
<td>0100</td>
<td>01</td>
<td>01 (0110)</td>
<td>01</td>
<td>01 (0011)</td>
</tr>
<tr>
<td>0100</td>
<td>01</td>
<td>01 (0011)</td>
<td>00</td>
<td>00 (1000)</td>
</tr>
<tr>
<td>0110</td>
<td>01</td>
<td>01 (1101)</td>
<td>01</td>
<td>01 (1010)</td>
</tr>
<tr>
<td>0110</td>
<td>01</td>
<td>01 (1111)</td>
<td>01</td>
<td>01 (1011)</td>
</tr>
<tr>
<td>1000</td>
<td>10</td>
<td>10 (1000)</td>
<td>10</td>
<td>10 (1000)</td>
</tr>
<tr>
<td>1000</td>
<td>11</td>
<td>11 (1010)</td>
<td>10</td>
<td>10 (1000)</td>
</tr>
<tr>
<td>1010</td>
<td>11</td>
<td>11 (1011)</td>
<td>10</td>
<td>10 (1010)</td>
</tr>
<tr>
<td>1010</td>
<td>11</td>
<td>11 (1101)</td>
<td>10</td>
<td>10 (1100)</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
<td>11 (1110)</td>
<td>10</td>
<td>10 (1110)</td>
</tr>
<tr>
<td>1100</td>
<td>11</td>
<td>11 (1111)</td>
<td>11</td>
<td>11 (1110)</td>
</tr>
</tbody>
</table>
Global Fullspace Skyline

Algorithm 1: Constrained Full Space Algorithm

Input:
- $RS$: Received Skyline

Output:
- $DS$: Discovered Skyline

BEGIN

$LS$: Constrained Local Skyline Points
$DS = \emptyset$

for all $P \in LS$ do

    if $\exists Q \in RS : Q < P$ then
        $DS = DS \cup P$
    end if

end for

Send $DS$ to querying peer as final skyline points

$RS = RS \cup DS$

Send $RS$ to next unpruned peer

END
Subspace Skyline

Algorithm 2 Constrained Subspace Skyline Algorithm

Input:
RFS: Rcvd Final Sky Pts
RGS: Rcvd Group Sky Pts
Output:
DS: Discovered Skyline
BEGIN
LS: Constrained Local Sky Pts
LS = findSkyline(\(LS \cup RGS\))
DS = \(\emptyset\)
for all \(P \in LS\) do
if \(\forall Q \in RFS: Q < P\) then
DS = DS \(\cup P\);
end if
end for
if (Last-peer-of-subspace-group) then
Send DS to query peer as a final sky
RFS = RFS \(\cup DS\)
Send RFS to first unpruned peer in SG
else
RGS = DS
Send RFS and RGS to next peer in SG
end if
END

Subspace group(SG) is treated as a peer of the full space
Parallelism

- As parallelism is used to minimize total processing time:
  - Each node/group can be sent to $d$ peers (for fullspace)/$|\text{subspace}|$ peers (for subspace) for queries
  - Due to un-comparability, progressiveness are preserved
For More Data Transfer Reduction

- In parallelism, notice:
  - Traversal is from lower to upper value in each dimension.

- Thus,
  - No need to send all so-far skyline points.
  - It is enough to only send the subspace skyline of all dimensions in consideration - the dimension we are sending through.
Results (I)

Queries vs. peers traversed

Traversed peers vs. network size
Results (II)

Data transferred vs network size

Saved Bandwidth

- ChordSky
- CSSA
- CSSPA
- CSSPA-DR

Network Size

% of data transferred

Percentage of saved bandwidth
Results (III)

Parallel effects on critical path

Cost

<table>
<thead>
<tr>
<th>Network size</th>
<th>% of Avg Critical path</th>
<th>Hops per peer visited</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>1000</td>
<td>100</td>
<td>0.5</td>
</tr>
<tr>
<td>1500</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>2000</td>
<td>100</td>
<td>1.5</td>
</tr>
<tr>
<td>2500</td>
<td>100</td>
<td>2</td>
</tr>
<tr>
<td>3000</td>
<td>100</td>
<td>2.5</td>
</tr>
<tr>
<td>3500</td>
<td>100</td>
<td>3</td>
</tr>
<tr>
<td>4000</td>
<td>100</td>
<td>3.5</td>
</tr>
</tbody>
</table>

Graphs show the comparison of ChordSky, CSSA, CSSPA, and CSSPA-DR across network sizes.
Results (IV)

Bird-eye view

Percentage of results to total

- Peers-Visited
- PointsTrans
- NoOfHops
- CriticalPath

ChordSky, CSSA, CSSPA, CSSA-DR
Thank you


Reference (II)


[13] Chen, L., Cui, B., Lu, H.: Constrained skyline query processing against distributed data sites. IEEE Transactions on Knowledge and Data Engineering (TKDE) 23(2), 204-217 (2011)


