Semi-Skyline Optimization of Constrained Skyline Queries

Markus Endres and Werner Kießling
The 22nd Australasian Database Conference (ADC 2011)
1. Skyline Queries
2. Semi-Skylines
3. Constrained Skyline Optimization
4. Efficient Evaluation of Semi-Skylines
5. Performance benchmarks
6. Summary and Outlook
1. Skyline Queries
**Skyline Queries**

Skyline Queries and Pareto Preferences

---

**Beverages with lowest calories and lowest fat?**

---

**Skyline / Preference SQL query**

```sql
SELECT *
FROM Beverage B
PREFERING
    B.cal LOWEST AND B.fat LOWEST
```

---

**Literature:**

- On Finding the Maxima of a Set of Vectors (Kung et. al, 1975)
- The Skyline Operator (Börzsönyi et. al, 2001)
Constrained Skyline Query in Preference SQL

```
SELECT *
FROM  Soup S, Meat M, Beverage B
WHERE  S.Cal + M.Cal + B.Cal <= 1100

PREFERRING
  S.Name IN ('Chicken', 'Noodle') AND
  M.Name IN ('Beef') AND
  B.VitaminC HIGHEST
```

- hard constraints on 3 relations
- „full“ Cartesian product
- about 120,000,000 combinations, \(|S| = 350, |M| = 500, |B| = 680\)
- no optimization possible so far
- very high memory and computation costs
Algebraic Query Optimization

Heuristics

- Reduce size of intermediate results
- Eliminate tuples before building the costly join
- Apply Skyline operator $\sigma[P]$ before selection operator
- Push Skyline operator through a join
2. Semi-Skylines
Semi-Skylines
Preference Background

- **Preference**: strict partial order \( <_P \) on \( \text{dom}(A) \)
  \[ x <_P y \] means: I like \( y \) more than \( x \)

- Preference selection of a preference \( P \)
  \[ \sigma[P](R) := \{ t \in R \mid \neg \exists t' \in R : t[A] <_P t'[A] \} \]

- Base preference constructors
  LOWEST, HIGHEST, ANTICHAIN, POS, NEG, ...

- Complex preference constructors, e.g.
  - Pareto / traditional Skyline
    \[ P_1 \otimes P_2 = (A_1 \times A_2, <_P) \]
    \[ (x_1, x_2) <_P (y_1, y_2) \iff \]
    \[
    \begin{align*}
    (x_1 &<_{P_1} y_1 \land (x_2 <_{P_2} y_2 \lor x_2 = y_2)) \lor \\
    (x_2 &<_{P_2} y_2 \land (x_1 <_{P_1} y_1 \lor x_1 = y_1))
    \end{align*}
    \]
Semi-Skyline

Semi-Pareto:

- Left-Semi-Pareto \( P_1 \preccurlyeq P_2 = (A_1 \times A_2, \prec_{P_1 \preccurlyeq P_2}) \)

\[
(x_1, x_2) \prec_{P_1 \preccurlyeq P_2} (y_1, y_2) \iff x_1 \prec_{P_1} y_1 \land (x_2 \prec_{P_2} y_2 \lor x_2 = y_2)
\]

- Right-Semi-Pareto \( P_1 \succcurlyeq P_2 = (A_1 \times A_2, \prec_{P_1 \succcurlyeq P_2}) \)

\[
(x_1, x_2) \prec_{P_1 \succcurlyeq P_2} (y_1, y_2) \iff x_2 \prec_{P_2} y_2 \land (x_1 \prec_{P_1} y_1 \lor x_1 = y_1)
\]
Preferences on data set for Beverages

- $P_1 = \text{POS}(B.\text{Name}, 'Red Wine')$
- $P_2 = \text{HIGHEST}(B.\text{VitaminC})$

<table>
<thead>
<tr>
<th>B</th>
<th>ID</th>
<th>Name</th>
<th>Cal</th>
<th>Vc</th>
<th>Fat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Red Wine</td>
<td>85</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Red Wine</td>
<td>181</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Coke</td>
<td>220</td>
<td>21</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Lemonade</td>
<td>281</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>Red Wine</td>
<td>400</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

$\sigma[P_1 \bowtie P_2](B) = \{B1, B2, B3, B4, B5\}$
$\sigma[P_1 \bowtie P_2](B) = \{B2, B3\}$
3. Constrained Skyline Optimization
Known Theorem (Hafenrichter & Kießling (ADC 2005), Chomicki 2003)

**Push Preference over Hard Selection**

\[ \sigma_H(\sigma[P](R)) = \sigma[P](\sigma_H(R)) \iff \forall w \in R : H(w) \land \exists v \in R : w[A] <_P v[A] \rightarrow H(v) \]

**Example:** \( P_2 = \text{HIGHEST}(B.\text{VitaminC}) \)

\[ \sigma[P](\sigma_{B.\text{VitaminC} \geq 15}(B)) = \sigma_{B.\text{VitaminC} \geq 15}(\sigma[P](B)) \]

**Problem:**
- Complex condition, hard to check
- Idea: derive *induced preference* which can be used as *prefilter*
\( Q \) is **prefilter preference** for \( P \) w.r.t. a relation \( R \)

\[
\sigma[P](R) = \sigma[P](\sigma[Q](R))
\]

**Induced preference** for a hard constraint

\[
H := h(b_1, \ldots, b_k) \Theta c, \quad \Theta \in \{\leq, <, >, \geq, =, \neq\}
\]

\[
\leftarrow H_{B_i} := \begin{cases} 
\text{LOWEST}(B_i) & \text{if } \Theta \in \{<, \leq\} \\
\text{HIGHEST}(B_i) & \text{if } \Theta \in \{>, \geq\} \\
\text{ANTICHAIN}(B_i) & \text{if } \Theta \in \{=, \neq\}
\end{cases}
\]

**Overall induced preference**

\[
\leftarrow H := \leftarrow H_{B_1} \otimes \ldots \otimes \leftarrow H_{B_k}
\]
Q is **prefilter preference** for P w.r.t. a relation R

$$\sigma[P](R) = \sigma[P](\sigma[Q](R))$$

**Induced preference** for a hard constraint

$$H := h(b_1, \ldots, b_k) \Theta c, \quad \Theta \in \{\leq, <, >, \geq, =, \neq\}$$

$$\leftarrow H_{B_i} := \begin{cases} 
LOWEST(B_i) & \text{if } \Theta \in \{<, \leq\} \\
HIGHEST(B_i) & \text{if } \Theta \in \{>, \geq\} \\
ANTICHAIN(B_i) & \text{if } \Theta \in \{=, \neq\}
\end{cases}$$

**Overall induced preference**

$$\leftarrow H := \leftarrow H_{B_1} \otimes \ldots \otimes \leftarrow H_{B_k}$$
Constrained Skyline Optimization

Example

Query:

\[ \sigma[P] \]

\[ \sigma S.Cal + M.Cal + B.Cal \leq 1100 \]

Induced preferences:

- \( H_{S.Cal} := \text{LOWEST}(S.Cal) \)
- \( H_{M.Cal} := \text{LOWEST}(M.Cal) \)
- \( H_{B.Cal} := \text{LOWEST}(B.Cal) \)
Prefilter Preference and Hard Selection

\[ \sigma[P](\sigma_H(R)) = \sigma[P](\sigma[P \bowtie \bar{H}](\sigma_H(R))) \]

Insert and Push Semi-Pareto over Hard Constraint

\[ \sigma[P](\sigma_H(R)) = \sigma[P](\sigma_H(\sigma[P \bowtie \bar{H}](R))) \]
Push Semi-Pareto over Cartesian Product

1. \( \sigma[P_1](\sigma_H(R \times S)) = \sigma[P_1](\sigma_H(\sigma[P_1 \bowtie \leftarrow \hat{H}_{B_1}](R) \times S)) \)

2. \( \sigma[P_1 \otimes P_2](\sigma_H(R \times S)) = \sigma[P_1 \otimes P_2](\sigma_H(\sigma[P_1 \bowtie \leftarrow \hat{H}_{B_1}](R) \times \sigma[P_2 \bowtie \leftarrow \hat{H}_{B_2}](S))) \)

Push Semi-Pareto over Join

\[ \sigma[P](\sigma_H(R \bowtie_{R.X \Theta S.Y} S)) = \sigma[P](\sigma_H(\sigma[(P \bowtie \leftarrow \hat{H}_{B_1}) \text{ groupby } X](R) \bowtie_{R.X \Theta S.Y} (S))) \]
Constrained Skyline Optimization

Example

Push Semi-Pareto over Cartesian Product

SELECT *
FROM Soup S, Meat M, Beverage B
WHERE S.Cal + M.Cal + B.Cal <= 1100

PREFERRING

S.Name IN ('Chicken', 'Noodle') AND P_1
M.Name IN ('Beef') AND P_2
B.VitaminC HIGHEST AND P_3

Induced preferences

- $\hat{H}_{S.Cal} := \text{LOWEST}(S.Cal)$
- $\hat{H}_{M.Cal} := \text{LOWEST}(M.Cal)$
- $\hat{H}_{B.Cal} := \text{LOWEST}(B.Cal)$
Constrained Skyline Optimization

Example

Semi-Skyline Optimization

Unoptimized operator tree.

Optimized operator tree.
4. Efficient Evaluation of Semi-Skylines
Problem: Semi-Skyline Evaluation
- No specialized algorithm for Semi-Skylines
- Only BNL applicable
  - Worst-case $\mathcal{O}(n^2)$
  - Best-case $\mathcal{O}(n)$

Solution: The novel Staircase algorithm
- Worst-case $\mathcal{O}(n \log n)$
- Best-case $\mathcal{O}(n)$
Efficient Evaluation of Semi-Skylines

SCPR: StairCase Pruning Region
Efficient Evaluation of Semi-Skylines

Efficient data structure for maintaining SCPR: Skiplists (Pugh 1990)
5. Performance Benchmarks
Performance Benchmarks
BNL vs. Staircase

- Synthetic data sets (data set generator from Börzsönyi et al. (2001))
- Vary data cardinality, anti-correlated data
Performance Benchmarks
Constrained Skyline Optimization

- Real-world data set from *United States Department of Agriculture* (USDA)
- 3 database relations with 120,000,000 combinations
- Comparison *standard optimization vs. Semi-Skyline optimization*

Query:

```sql
SELECT * FROM Soup S, Meat M, Beverage B
WHERE S.Cal + M.Cal + B.Cal <= max_cal
PREFERRING
    S.Name IN ('Chicken', 'Noodle') AND
    M.Name IN ('Beef') AND
    B.Vc HIGHEST
```
Different hard constraints: \( S.Cal + M.Cal + B.Cal \leq \text{max\_cal} \)
Performance Benchmarks
Constrained Skyline Optimization

- Different relation sizes, \#combinations

![Graph showing runtime in seconds for different \#combinations with two methods: no-pref-prefilter and pref-prefilter. The x-axis represents \#combinations ranging from 1M to 120M, while the y-axis shows runtime in seconds from 0 to 600. The graph illustrates the performance comparison between the two methods for varying \#combinations.]
6. Summary and Outlook
Summary

‣ Semi-Skyline for Constrained Skyline Optimization
  ‣ Operator tree optimization
  ‣ Staircase evaluation, worst-case: $O(n \log n)$
  ‣ No pre-computed index structure necessary
‣ Integration: Preference SQL

http://www.trial.preferencesql.com/

Outlook

‣ Further optimization rules
‣ Further techniques for very fast retrieval (online service)