Optimization Strategies for Processing Multiple Pattern Mining Requests Over Streaming Data

Elke A. Rundensteiner
Computer Science Dept., Worcester Polytechnic Institute
rundenst@cs.wpi.edu

In collaboration with Di Yang, Avani Shastri, Matt Ward, and others from the XMDV research group
April 2011

This work is supported under NSF grants CCF-0811510, IIS-0119276, IIS-0414380.
Motivation: data streams are everywhere

Stock Market

Are there any patterns in transactions over past hour?

Battlefield

Where are the main clusters formed by enemy warcraft?

Commander
Motivation: pattern mining requests tend to be parameterized

- Example 1: give me the stocks that dropped significantly in the most recent transactions.
  
  *with in last 10,30, or 60 minutes.*

  *10%, 30% or 50% to the original price*

- Example 2: give me the major clusters formed by enemy warcraft.

  *size: n war-crafts*
  *density: m war-crafts / m²*
Motivation: best parameters settings are hard to determine

- Clusters formed by boom carriers need to be updated every 10 seconds
- I only care about the clusters that are formed by more than 20 warcraft
- Clusters formed by fighter planes need to be updated every 5 seconds
- I need info for any cluster sized 5 or higher

Problem: A lot of similar queries, yet with different parameter settings, how to answer them efficiently.

Multiple analysts may raise multiple queries with different parameter settings.

Parameter settings? I probably know. But, can I try different combinations of them?

A single analyst may raise multiple queries with different parameter settings.
State of the Art

- Efficient pattern mining strategies are designed for mining static data [Han09], [Marin03], [Hirji99].

- More recently, pattern mining algorithms are designed to mining streaming data; however mainly for executing single mining queries [Aggarwal 10][Han09] [Yu08].

- Multiple query optimization is a core principle studied by database community [Arasu06] [Hammad04][Krishnamurthy03], while barely being applied for complex pattern mining yet [Yang09].
Research Goal

• Shared execution of large numbers of pattern mining queries over data streams:

1. Focus on popular pattern mining algorithms, including clustering, outlier detection, and top-k requests.

2. Consider sliding window scenario, one of the most widely used query semantics for stream processing.
Definition of Density-Based Clustering

- **Density-Based Clustering** [Ester96] [Cao06] (input parameters: $\theta_{\text{range}}, \theta_{\text{cnt}}$)

- **Core Object**: has no less than $\theta_{\text{cnt}}$ neighbors in $\theta_{\text{range}}$ distance from it.

- **Edge Object**: not core object but a neighbor of a core object.

- **Noise**: not core object and not a neighbor of any core object.

A **Density-Based Cluster (DB-Cluster)** is a maximum group of connected core objects and the edge objects attached to them.

- **Why**: popular and well known, arbitrary shapes, allow unclassified mining, handles noise, deterministic process, customizable by parameter settings.
Definition of Distance-Based Outlier Detection

- Distance-based Outliers [Knorr98] (input parameters: $\theta^{\text{range}}$, $\theta^{\text{fra}}$)

  - Outlier: has no more than $N^{\theta^{\text{fra}}}$ neighbors, with $N$ the number of data points in the data set.
Definition of Top-k Requests

- Given a dataset $D$ and a preference function $F()$, return $k$ objects in $D$ with highest preference function score.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Previous Rank</th>
<th>Brand</th>
<th>Country of Origin</th>
<th>Sector</th>
<th>Brand Value ($M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Coca-Cola</td>
<td>United States</td>
<td>Beverages</td>
<td>70,452</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>IBM</td>
<td>United States</td>
<td>Business Services</td>
<td>64,727</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Microsoft</td>
<td>United States</td>
<td>Computer Software</td>
<td>60,895</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>Google</td>
<td>United States</td>
<td>Internet Services</td>
<td>43,557</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>GE</td>
<td>United States</td>
<td>Diversified</td>
<td>42,808</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>McDonalds</td>
<td>United States</td>
<td>Restaurants</td>
<td>33,578</td>
</tr>
</tbody>
</table>

Example Query:
Find Top-3 Brands for Year 2010

$D =$ all major companies in the world
$F =$ company’s brand value in 2010
$k =$ 3
Applications include:
- monitoring congestion (cluster) in traffic
- looking for intensive transaction areas (cluster) in stock trades
- identifying malicious attacks (cluster) in network
Problem Definition (clustering as example)

- Input: a query group $QG$ with multiple density-based clustering queries querying on the same input stream but with arbitrary parameter settings.

$$Q_i: \text{DETECT Density-Based Clusters FROM } \text{stream}$$

$$\text{USING } \theta^{\text{range}} = r \text{ and } \theta^{\text{cnt}} = c$$

$$\text{IN Windows WITH } \text{win} = w \text{ and } \text{slide} = s$$

- Goal: to minimize both the average processing time and the peak memory space needed by the system to process the full workload.
General Optimization Principles

1. View Prediction Principle
   - for **incremental** pattern maintenance across windows

2. Integrated Pattern Capture Principle
   - for shared pattern storage and maintenance across multiple queries with **varying pattern** parameter settings.

3. Meta-Query Principle
   - for shared pattern storage and maintenance across multiple queries with **varying window** parameters.
View Prediction Technique

• Why?
  1. From-scratch computation at each window is too expensive
  2. Object expiration usually causes complex pattern structure changes

• How?
  1. Analyze the life span of objects and relationship to future windows
  2. Determine their contribution to patterns being monitored.
  3. Prehandle the impact of objects’ expiration upon their arrival
Concept of Predicted Views

Current View of $W_0$

Predicted View of $W_1$

Predicted View of $W_2$

Predicted View of $W_3$

Window size=16, slide size=4, time=1

Based on Di Yang, et al., VLDB’2009
Update Predicted Views

Expired View of W₁

Current View of W₁

Predicted View of W₂

Predicted View of W₃

Window size=16, slide size=4, time=1

W₁

W₂

W₃

W₄

New Data Points
View Prediction for Top-k Requests

Why?

Insertion:
* cheap.

Deletion:
* expensive
* may need to examine full window.
View Prediction for Top-k Requests
View Prediction for Top-k Request

Major “Side” Bonus:

State-of-the-art:
[MouratidisSIGMOD06] requires to store the whole window,

Now [YangEDBT’2011]:
We succeed to only have to store small object set ever: \{o1, o6,o7,o14,o15, o16\}

Proof:
1. Necessity & sufficiency of set
2. Optimality of solution

[Mouratidis06]: Continuous monitoring of top-k queries over sliding windows SIGMOD 2006
[Yang et al.2011]: Optimal Solution for TopK Monitoring EDBT 2011
Predicted Top-k Maintenance

- Independent Window Maintenance (PreTopk)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

W0 W1

before window slide at time of W0

1 6 14 14
6 14 15 15
14 7 16 16

W0 W1 W2 W3
Conclusion for View Prediction Principle

• **Key Idea**
  - pre-prepare pattern detection results for future windows

• **Benefits:**
  - eliminate the need to deal with (expensive) object expiration.
  - realize efficient incremental pattern maintenance (save resources)

• **When can be applied:**
  - when object expiration constitutes key bottleneck for incremental pattern maintenance
  - has been found to be the case for clustering, outlier detection and top-k queries
  - other data mining algorithms likely also applicable: “low-hanging fruit”
General Optimization Principles

1. View Prediction Principle
   - for **incremental** pattern maintenance across windows

2. Integrated Pattern Capture Principle
   - for shared pattern storage and maintenance across multiple queries with **varying pattern** parameter settings.

3. Meta-Query Principle
   - for shared pattern storage and maintenance across multiple queries with **varying window** parameters.
From Independent to Integrated Pattern Capture

• Why?
  1. **Independent** Pattern Representation for each query prevents us from sharing storage space
  2. Plus, independent computation for pattern queries and thus prevents us from sharing maintenance costs
  3. Solution will not scalable for large number of queries

• How?
  1. Analyze interrelationships between patterns detected by queries with different parameters.
  2. Incrementally represent related patterns in a single structure.
  3. Devise maintenance algorithms that conduct mining for related patterns in one shot
Towards an Integrated Representation for Clusters

- Any relationship between the cluster sets identified by them?

![Original Data Set Diagram]
"Growth Property" among Cluster Sets

If each cluster \( C_i \) in Clu_Set1 is "contained" by one cluster in Clu_Set2, Clu_Set2 is a "Growth" of Clu_Set1.
Benefits of Hierarchical Cluster Structure

• Benefits for Memory Resources:
  Memory space needed by storing cluster sets identified by multiple queries in $QG$ is independent from $|QG|$.

• Benefits for Computational Resources:
  Multiple cluster sets stored in the hierarchical cluster structure (which are usually similar) can be maintained incrementally in one shot, rather than independently.
Arbitrary Pattern-Specific Parameter Case

--- arbitrary $\theta^{\text{cnt}}$, fixed $\theta^{\text{range}}$

- Growth property transitively holds among the cluster sets identified by multiple queries with arbitrary $\theta^{\text{cnt}}$ and same $\theta^{\text{range}}$. 

---

clusters Identified by Q1
$\theta^{\text{range}} = 0.2 \quad \theta^{\text{cnt}} = 4$

clusters Identified by Q2
$\theta^{\text{range}} = 0.2 \quad \theta^{\text{cnt}} = 3$

clusters Identified by Q3
$\theta^{\text{range}} = 0.2 \quad \theta^{\text{cnt}} = 2$
Arbitrary Pattern-Specific Parameter Case

-- arbitrary $\theta^{cnt}$, fixed $\theta^{range}$

- Idea: Growth property transitively holds.
- Solution: A single integrated representation of multiple clusters

```
clusters identified by Q3
$\theta^{range} = 0.2$ $\theta^{cnt} = 2$
```

```
clusters identified by Q2
$\theta^{range} = 0.2$ $\theta^{cnt} = 3$
```

```
clusters identified by Q1
$\theta^{range} = 0.2$ $\theta^{cnt} = 4$
```
General Case for Pattern Parameters

-- arbitrary $\theta^{\text{range}}$, arbitrary $\theta^{\text{cnt}}$

- Growth property holds $Q_i. \theta^{\text{range}} \leq Q_j. \theta^{\text{range}}$ $Q_i. \theta^{\text{cnt}}$ and $Q_j. \theta^{\text{cnt}}$

- That is, if query $Q_i$ is more “relaxed” (bigger) than $Q_j$, then $Q_i$’s clusters are a growth of $Q_j$’s clusters

- Propose: A single tree structure organizing for all queries based on this “growth” relationship.
General Case for Pattern Parameters

- Growth property holds if $\theta_{range}^{Qi} \geq \theta_{cnt}^{Qi}$ and $\theta_{range}^{Qj} \leq \theta_{cnt}^{Qj}$.
Integrated Representation for Top-k Queries

Query Group -

Q1: .. [WIN = 12s SLIDE=4s K=1]
Q2: .. [WIN =12s SLIDE=4s K=2]
Q3: .. [WIN =12s SLIDE=4s K=3]

Conclusion for Integrated Pattern Representation

• Benefits:
  1. Save memory space compared to independent pattern storage.
  2. Share computation for pattern detection maintenance.

• When can be applied?
  1. Queries are querying on the same portion of the stream (as common in window-based stream processing)
  2. The concept of “strictness” exists among queries (again common for queries with different parameter settings).
General Optimization Principles

1. View Prediction Principle
   - for \textit{incremental} pattern maintenance across windows

2. Integrated Pattern Capture Principle
   - for shared pattern storage and maintenance across multiple queries with \textit{varying pattern} parameter settings.

3. Meta-Query Principle
   - for shared pattern storage and maintenance across multiple queries with \textit{varying window} parameters.
Meta Query Strategy

• Why?
  1. Predicted views (windows) maintained by multiple queries may overlap.
  2. The integrated pattern storage and maintenance has to be applied to these predicted views of different queries.

• How?
  1. Analyze the predicted views of queries with different window parameter settings.
  2. Share maintenance process for overlapped predicted views driven by a scheduling process.
Cluster Detection: Varying Window

Parameters

-- arbitrary \textit{win}, fixed \textit{slide}

- Claim: maintaining a single query will be sufficient to answer all queries.
- Key: The predicted views for Qi with largest \textit{win} cover all needs.

\begin{itemize}
  \item Q1. \textit{win}=16, \textit{slide}=4
  \begin{itemize}
    \item Answer for Q1 at T=16
  \end{itemize}
  \item Q2. \textit{win}=12, \textit{slide}=4
  \begin{itemize}
    \item Answer for Q2 at T=16
  \end{itemize}
  \item Q3. \textit{win}=8, \textit{slide}=4
  \begin{itemize}
    \item Answer for Q3 at T=16
  \end{itemize}
  \item Q4. \textit{win}=4, \textit{slide}=4
  \begin{itemize}
    \item Answer for Q4 at T=16
  \end{itemize}
\end{itemize}
Cluster Detection: Vary Window Parameters
-- arbitrary *slide*, arbitrary *win*

- Use a single *meta query* with largest *window size* and adaptive *slide size* to represent queries.

![Diagram showing cluster detection and window parameters](image-url)
Meta-Query Strategy for Top-k Queries

Naïve method – execute queries one by one

Query Group -
Q1: .. [WIN = 8s SLIDE=2s K=1]
Q2: .. [WIN =8s SLIDE=3s K=3]
Q3: .. [WIN =8s SLIDE=6s K=2]

Q1 - 4 predicted views to output every 2s
Q2 - 3 predicted views to output every 3s
Q3 - 2 predicted views to output every 6s

Need to maintain 9 predicted view windows for answering each of the queries
**Meta-Query Strategy for Top-k Queries**

Meta query strategy - Slide size is NOT fixed but adaptive during execution.

**Query Group** -
- Q1: .. [WIN = 8s SLIDE=2s K=1]
- Q2: .. [WIN =8s SLIDE=3s K=3]
- Q3: .. [WIN =8s SLIDE=6s K=2]

Q1 needs output at 8, 10, 12, 14
Q2 needs output at 8, 11, 14,
Q3 needs output at 8, 14

Data Stream:
- W0
- W1
- W2
- W3
- W4

**Multiple Queries share windows:**
- W0 - serves Q1, Q2, and Q3,
  - k=max(Q1, Q2, Q3) is saved
- W1 - serves Q1, k=1 is saved
- W2 - serves Q2, K=2 is saved
- W3 - serves Q2 and Q3, k=max(Q2, Q3)
  - W4 - serves Q1, Q2, and Q3

**Significant saving in number of views - maintains only 5 (instead of 9) predicted views for all 3 queries in the workload**

Q meta: … [WIN=8s, SLIDE = ADAPTIVE, K = ADAPTIVE]
Conclusion for Meta-Query Strategy

• Benefits:
  1. Identify and share overlapped predicted views across multiple queries.
  2. Maximize the opportunities for Integrated Pattern Representation.

• When can be applied:
  1. Whenever overlapped predicted views exist (little to no overhead even if no overlapped predicted view exist).
Put it all together

-- arbitrary all four parameters (clustering)

- Our proposed techniques
  - for arbitrary pattern parameter cases (intra-window-optimization)
  - for arbitrary window parameter cases (inter-window-optimization) are orthogonal to each other.

- Final integrated structure for QG.
Relationship among optimization principles

Multiple Pattern Optimization

- Integrated Pattern Representation
- Orthogonal
- Meta-Query Strategy

View Prediction

Single Pattern Optimization
Experimental Study for Clustering

- **Alternative Methods:**
  1. Incremental DBSCAN [Ester98]
  2. Incremental DBSCAN with rqs (range query search sharing)
  3. Extra-N [Yang09]
  4. Extra-N with rqs (range query search sharing)
  5. Chandi [VLDB’2009 with all 3 principles applied]

- **Real Streaming Data:**
  1. GMTI data recording information about moving vehicles [Mitre08].
  2. STT data recording stock transactions from NYSE [INETATS08].

- **Measurements:**
  1. Average processing time for each tuple.
  2. Memory footprint to measure peak memory utilization.
Cluster Performance Evaluation for Varying Parameters

For arbitrary pattern parameter case, count parameter vary in 2-20 and range parameter vary in [0.01 – 0.1].
Cluster Performance Evaluation for Queries with Varying Window Parameters

Window parameter in [1000,5000] and slide parameter in [500:5000]
Evaluation for Performance

Arbitrary All Four Parameter Cases

C1: 20Q Cases
- A1: incDBSCAN
- A2: incDBSCAN with rqs
- A3: Extra-N
- A4: Extra-N with rqs
- A5: Chandi

C2: 40Q Cases
- A1
- A2
- A3
- A4
- A5

C3: 60Q Cases
- A1
- A2
- A3
- A4
- A5

C4: CPU Savings
- 20Q Cases
- 40Q Cases
- 60Q Cases

CPU Time (ms)
Experimental Study for TopK Requests

- **Alternative Methods:**
  1. MinTopk [Yang11] (optimal for single queries; uses prediction)
  2. M-Topk-IndeView (Independent window maintenance)
  3. M-Topk-IntView (Integrated window maintenance)

- **Real Streaming Data:**
  1. GMTI data recording information about moving vehicles [Mitre08].
  2. STT data recording stock transactions from NYSE [INETATS08].

- **Measurements:**
  1. Average processing time for each tuple.
  2. Memory footprint to capture peak utilization.
Some Experimental Findings for top-k Queries

CASE 1 - Fixed WIN & SLIDE, Arbitrary K

CASE 2 – Fixed WIN, Arbitrary SLIDE & K
Conclusions

1. Proposed three general principles for optimizing multi-pattern workloads.

2. Applied proposed principles to several popular parameterized pattern mining types (case studies)

3. Analytically and experimentally demonstrated the superiority of our methods to art-of-the-art solutions.
Future Work

1. Apply proposed principles to more pattern types.
2. Study other (multi-query) optimization principles.
4. Work collaboratively with domain experts to apply technologies.
5. Explore Extraction and Compaction of Significant Patterns into a Nugget Store.
The End

Thanks