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

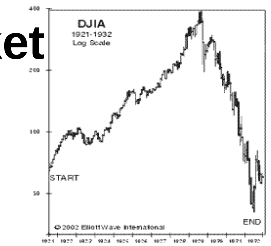
April 2011

This work is supported under NSF grants CCF-0811510, IIS-0119276, IIS-0414380.

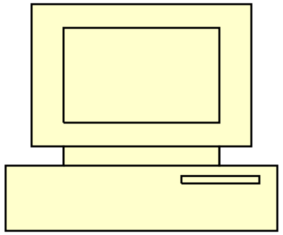
Motivation: data streams are everywhere

Are there any patterns in transactions over past hour?

Stock Market



transaction info



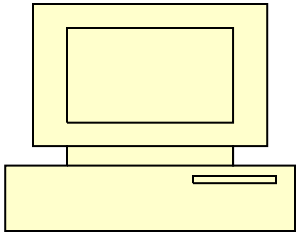
patterns

Stock Analysts

Where are the main clusters formed by enemy warcraft?

Battlefield

position info



patterns

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 / mi^2

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.

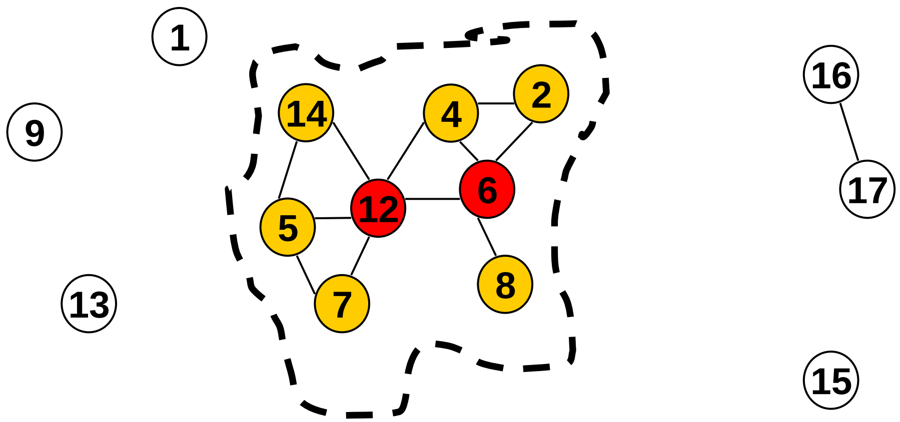
Definition of Density-Based Clustering

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

Core Object: has no less than θ^{cnt} neighbors in θ^{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.



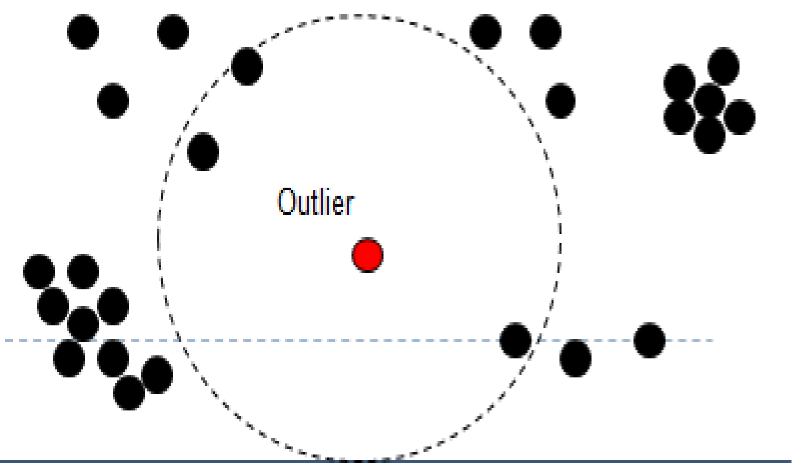
A Density-Based Cluster (DB-Cluster) is a maximum group of connected core objects and the edge objects attached to them

- **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: θ^{range}, θ^{fra})

Outlier: has no more than N* θ^{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.**

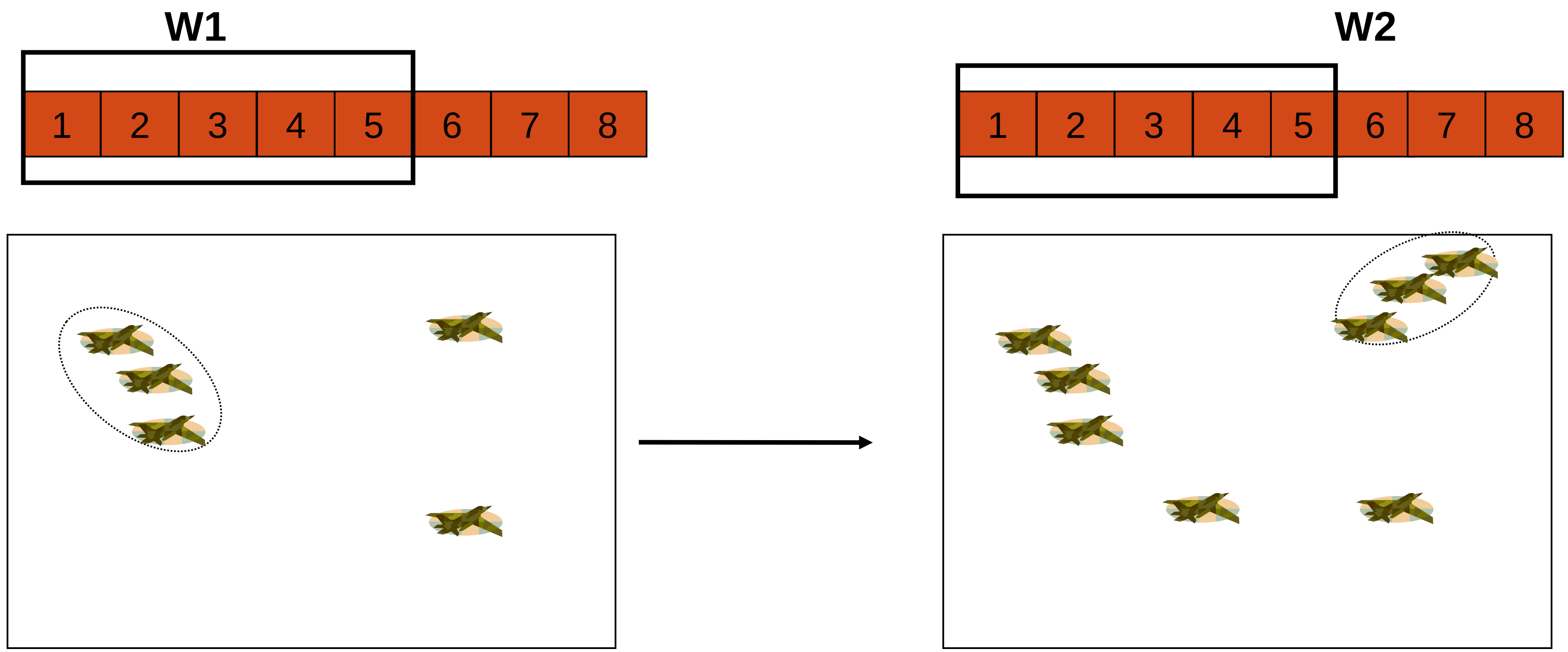
RANKINGS

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

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

Pattern Mining in Sliding Windows Over Streams



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.

$$Q_i: \textbf{\textit{DETECT Density-Based Clusters FROM}}\ stream$$
$$\textbf{\textit{USING}}\ \underline{\theta^{range}} = r\ \textbf{\textit{and}}\ \underline{\theta^{cnt}} = c$$
$$\textbf{\textit{IN Windows WITH}}\ \underline{win} = w\ \textbf{\textit{and}}\ \underline{slide} = s$$

Pattern-specific

Window-specific

Template Density-Based Clustering Query Over Sliding Windows

- 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

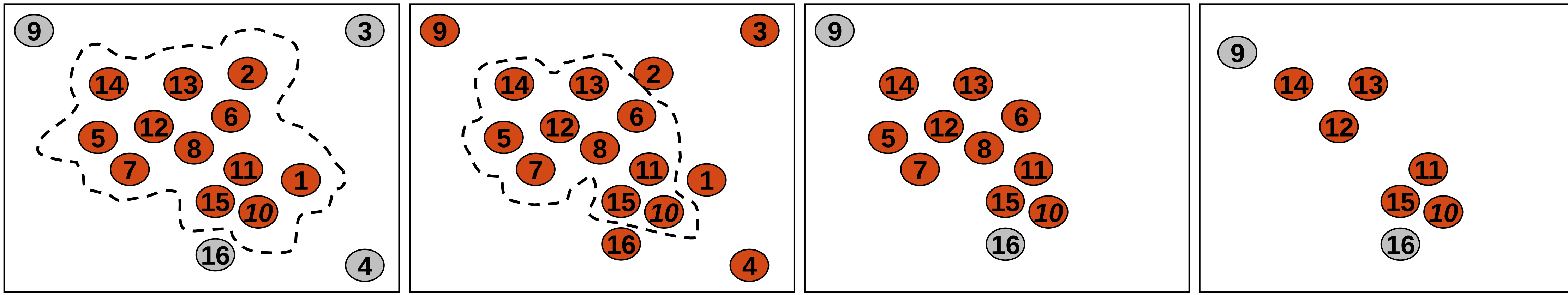
2. Integrated Pattern Capture Principle
   - for shared pattern storage and maintenance across multiple queries with **varying pattern** parameter settings.

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

View Prediction Technique

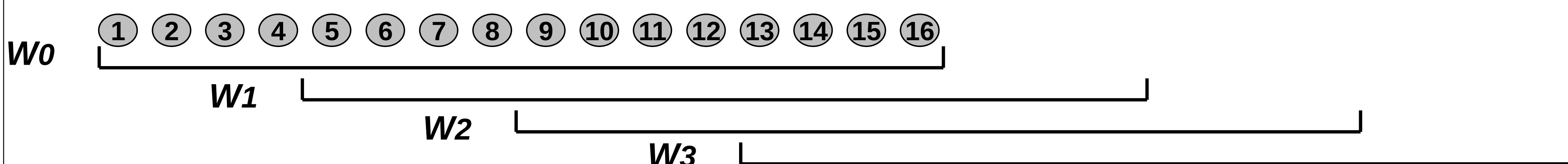
- Why?
  1. From-scratch computation at each window is too expensive
     1. thus incremental pattern maintenance method is critical
  2. Object expiration usually causes complex pattern structure changes
     1. thus makes incremental computation computationally expensive
- How?
  1. Analyze the life 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



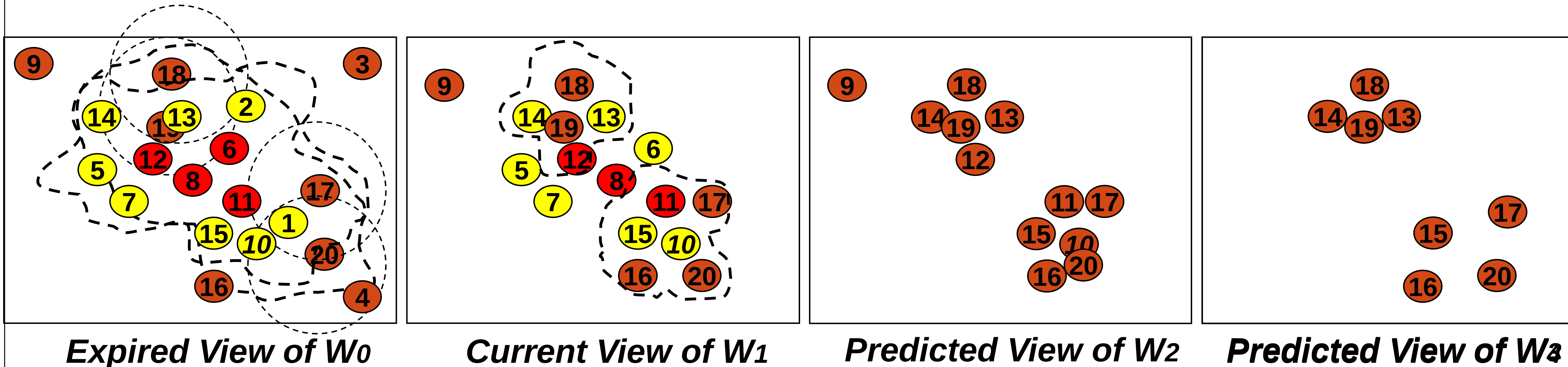
Current View of W_0　*Predicted View of W_1*　*Predicted View of W_2*　*Predicted View of W_3*

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

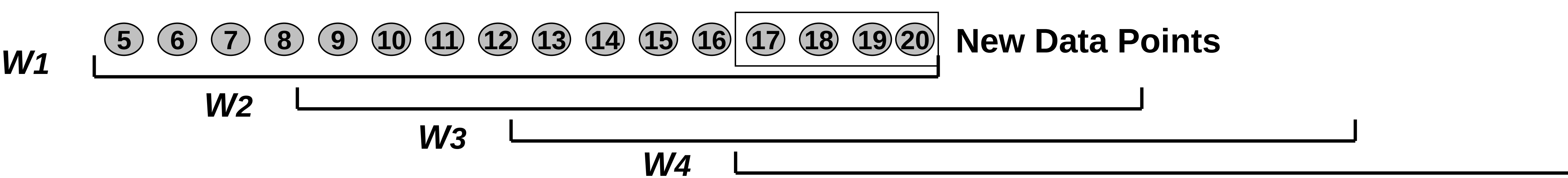


Based on Di Yang, et al., VLDB'2009

Update Predicted Views



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



View Prediction for Top-k Requests

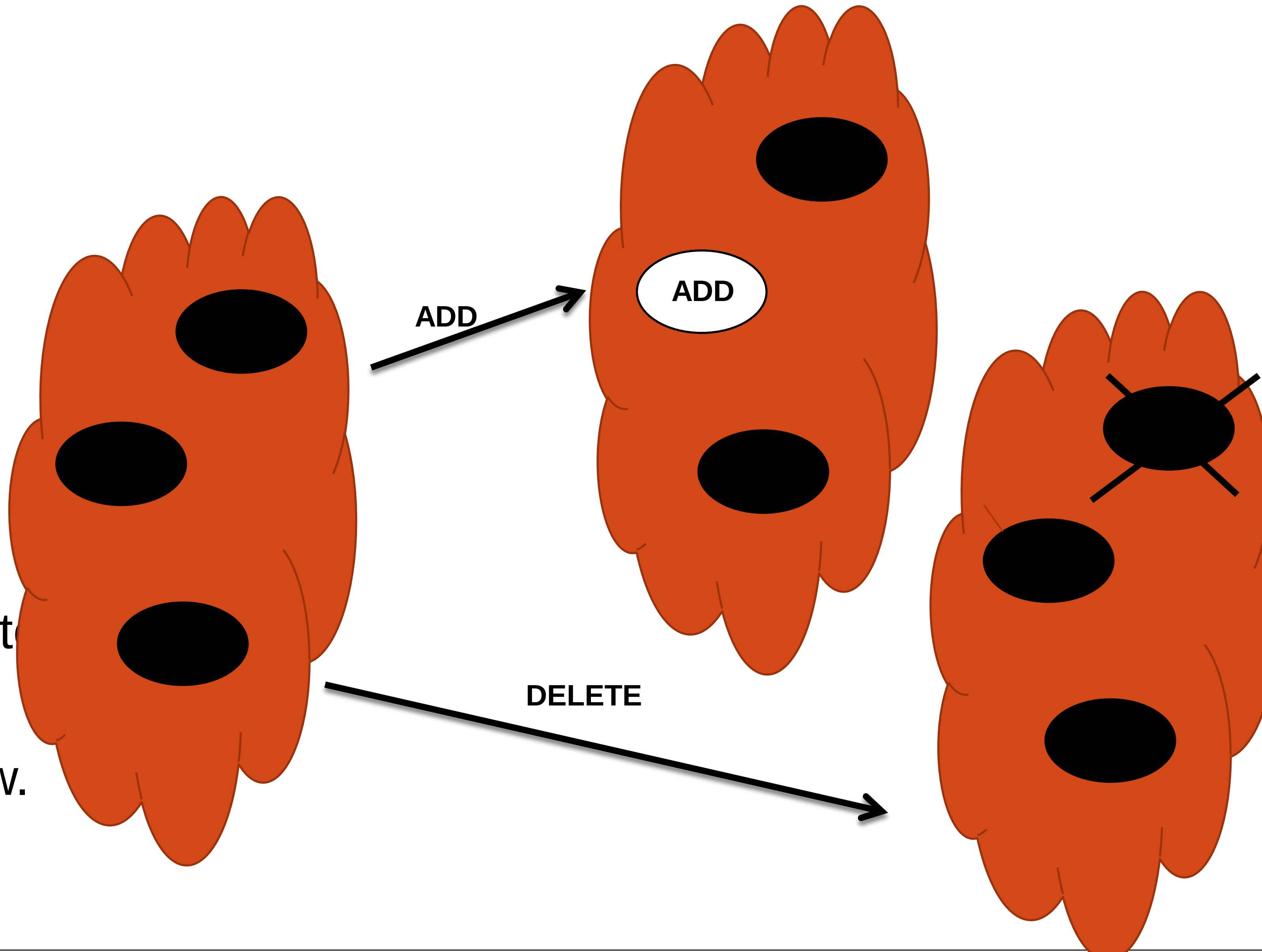
Why ?

<u>Insertion:</u>

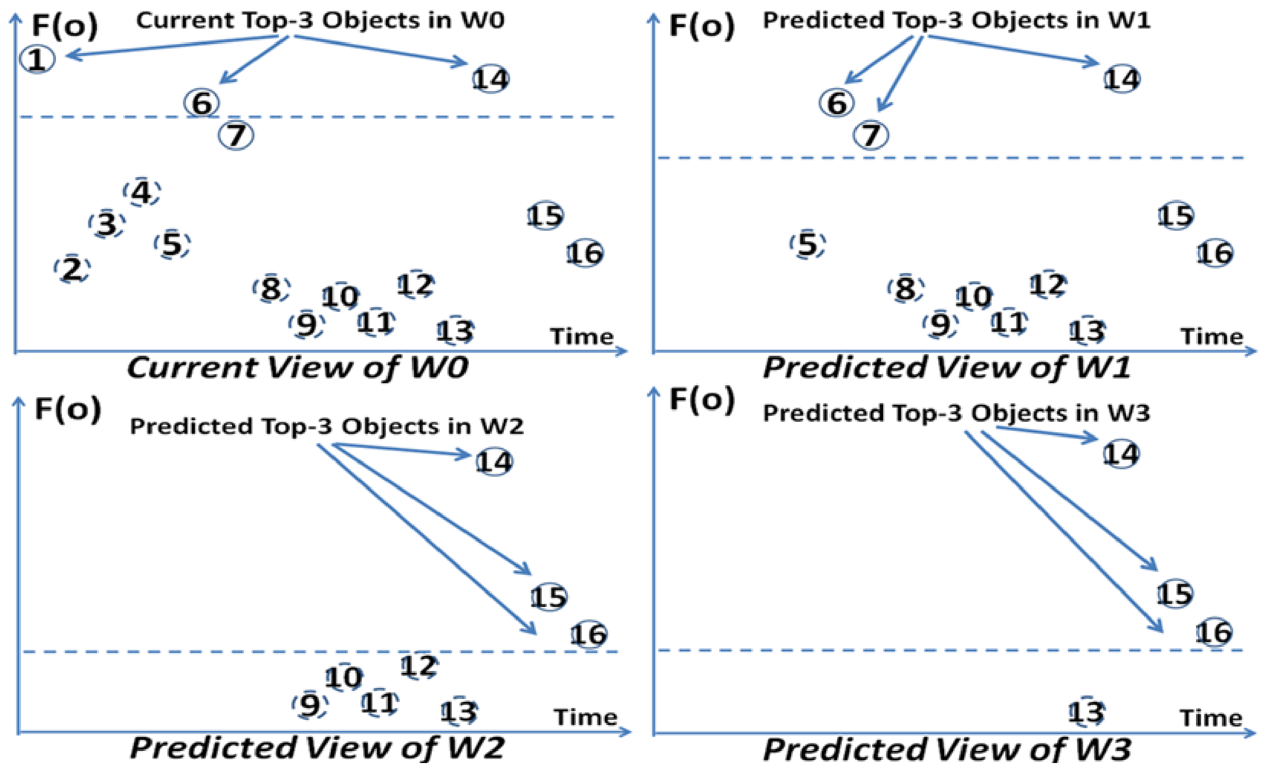
* cheap.

<u>Deletion:</u>

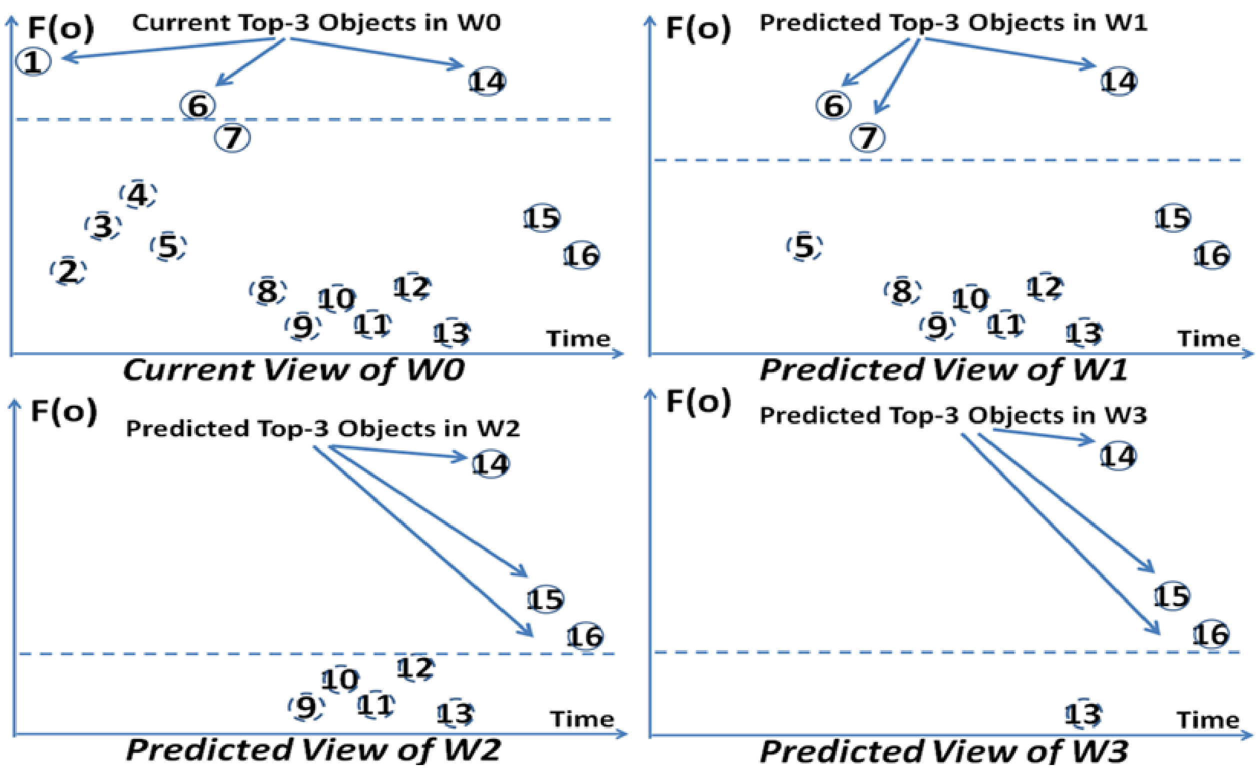
* expensive
* may need t
  examine
  full window.



View Prediction for Top-k Requests



View Prediction for Top-k Request



Major "Side" Bonus:

State-of-the-art :
[Mouratidis:SIGMOD06] requires to store the whole window,

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

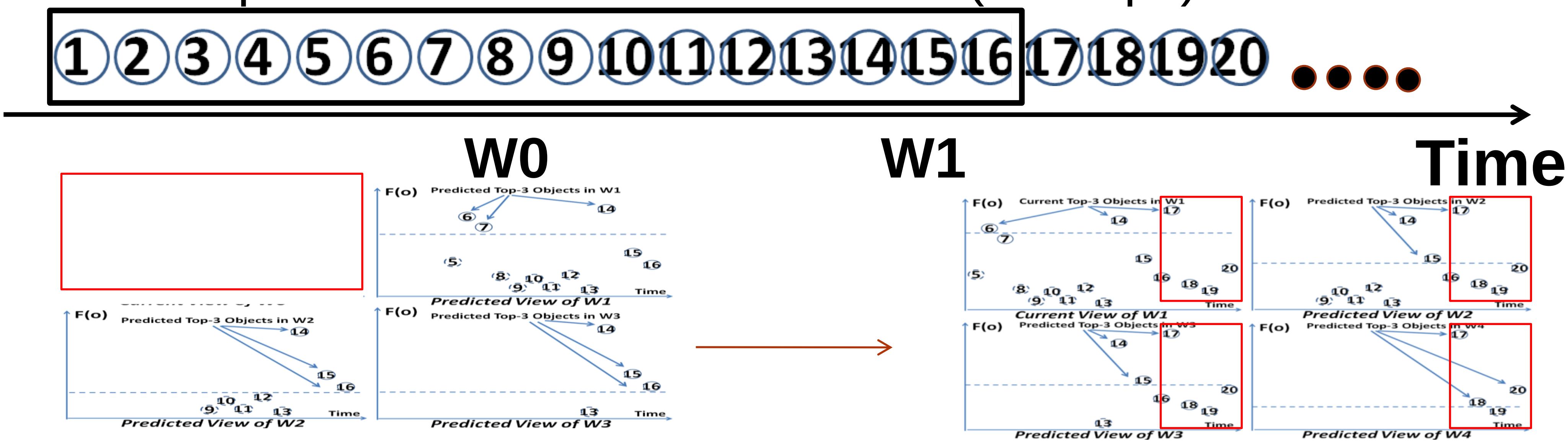
Proof:
1. Necessity & sufficiency of set
2. Optimality of solution

[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)



before window slide
at time of W0

W0	W1	W2	W3
1	6	14	14
6	14	15	15
14	7	16	16

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.

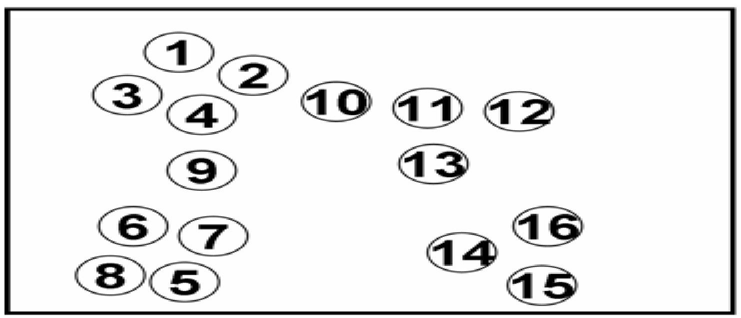
3. Meta-Query Principle
   - for shared pattern storage and maintenance across multiple queries with **varying window** parameters.

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

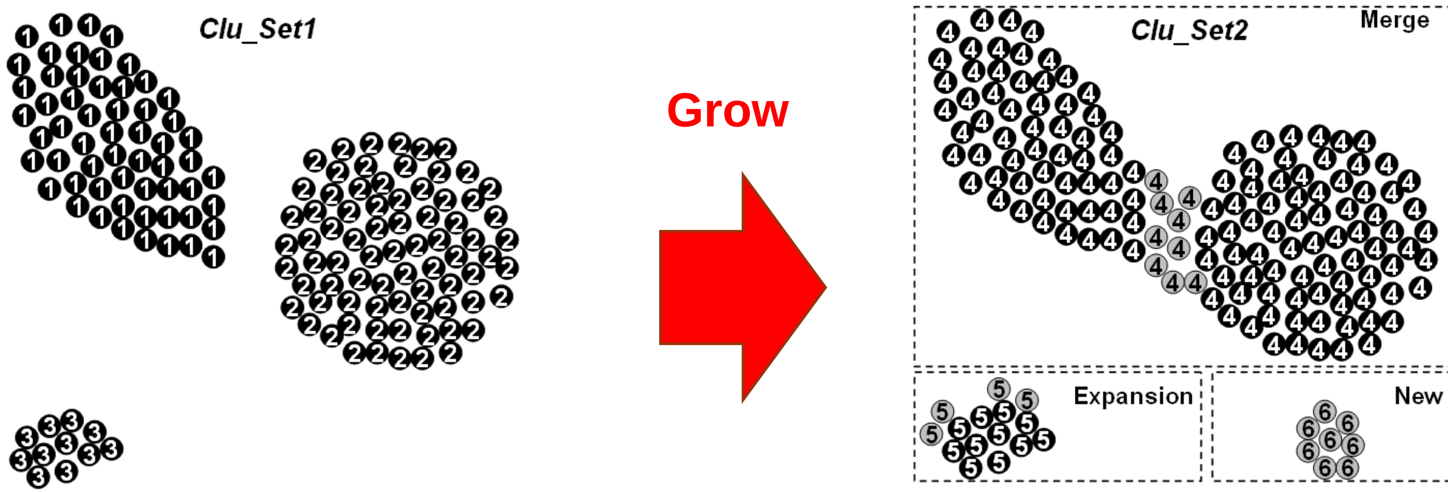
Towards an Integrated Representation for Clusters

- Any relationship between the cluster sets identified by them?

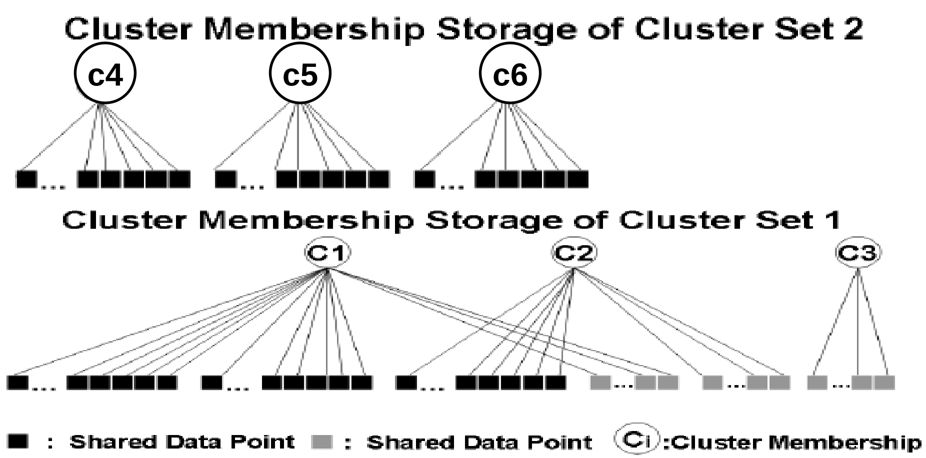


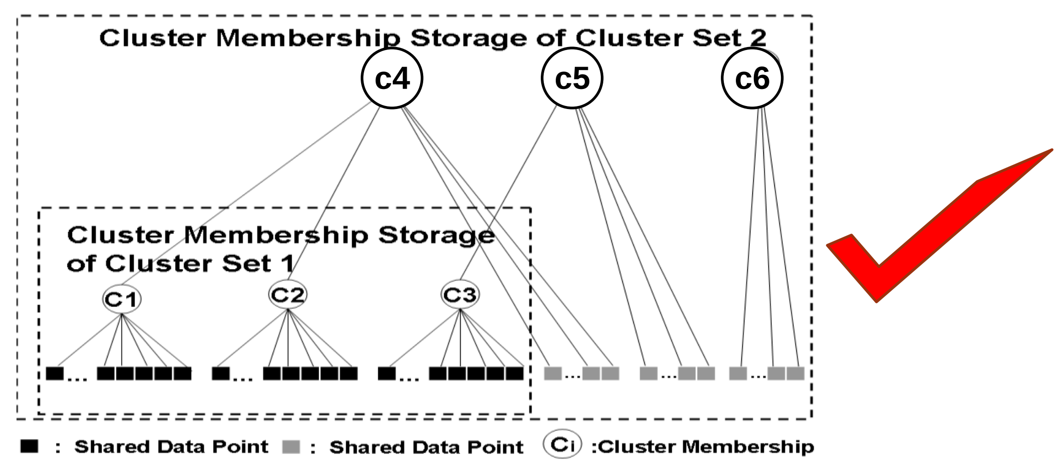
Original Data Set

“Growth Property” among Cluster Sets



If each cluster Ci in Clu_Set1 is **“contained”** by one cluster in Clu_Set2, Clu_Set2 is a **“Growth”** of Clu_Set1 .





Independent Cluster Structure Storage **Hierarchical Cluster Structure Storage**

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*|.

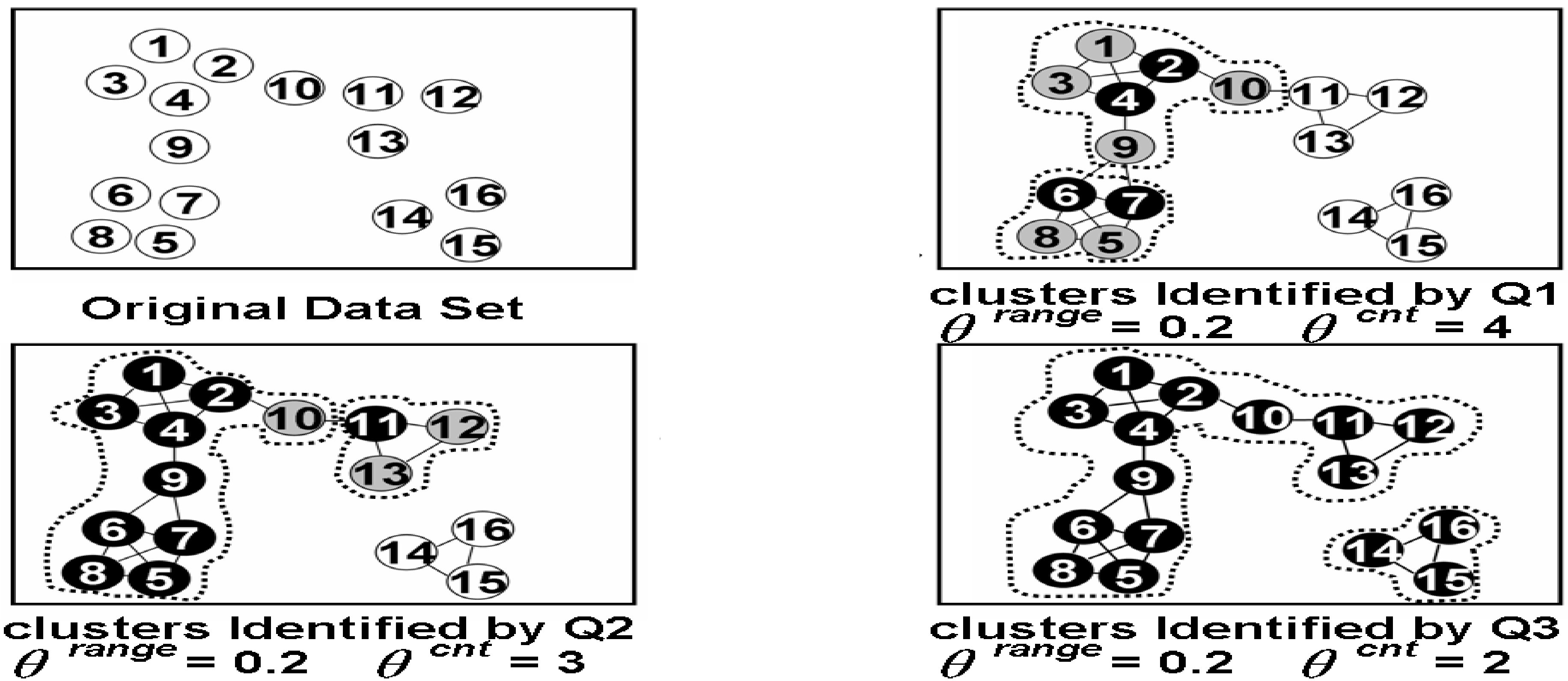
- Benefits for Computational Resources:

  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 θ^{cnt} , fixed θ^{range}

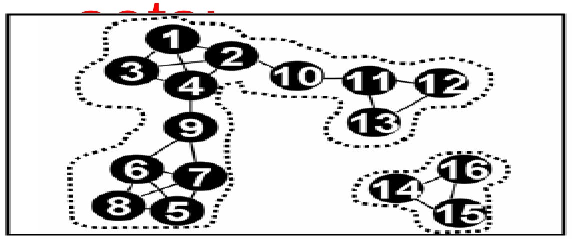
- Growth property transitively holds among the cluster sets identified by multiple queries with arbitrary θ^{cnt} and same θ^{range}.



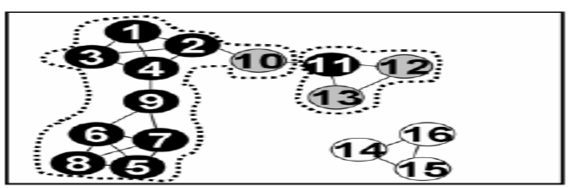
Arbitrary Pattern-Specific Parameter Case

-- arbitrary θ^{cnt} , fixed θ^{range}

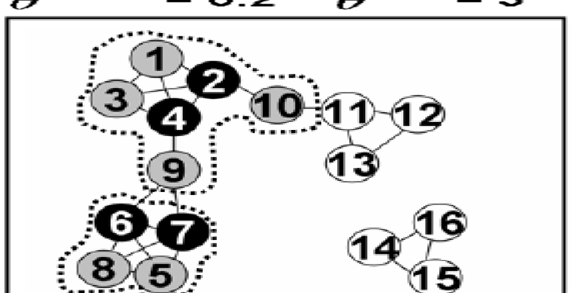
- Idea: Growth property transitively holds.
- Solution: A single integrated representation of multiple cluster sets:



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 θ^{range} , arbitrary θ^{cnt}

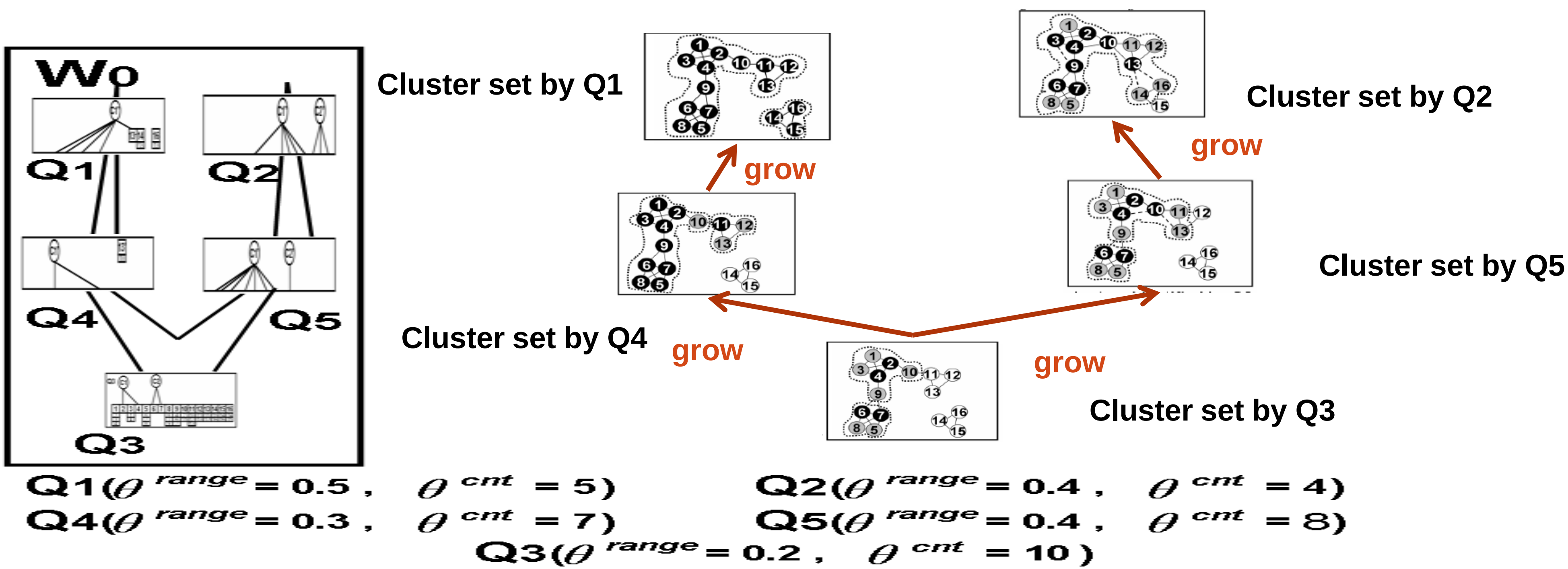
- Growth property holds, if Qi. θ^{range} ≥ Qj. θ^{range} and Qi. θ^{cnt} ≤ Qj. θ^{cnt} .

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

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

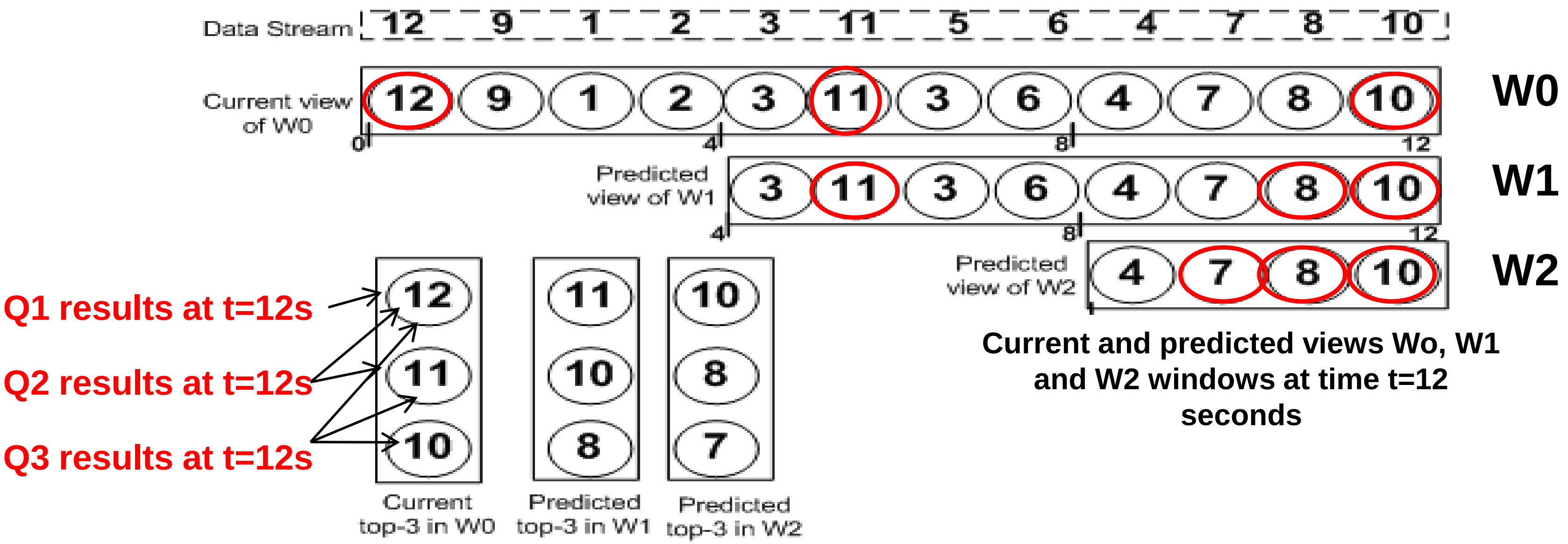
General Case for Pattern Parameters

- Growth property holds if $Qi.\theta^{range} \geq Qj.\theta^{range}$ and $Qi.\theta^{cnt} \leq Qj.\theta^{cnt}$



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]



Current and predicted views Wo, W1 and W2 windows at time t=12 seconds

Top-3 results for windows W0, W1 and W2

Avani S., Di Yang, et al, in progress.

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 **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.

Meta Query Strategy

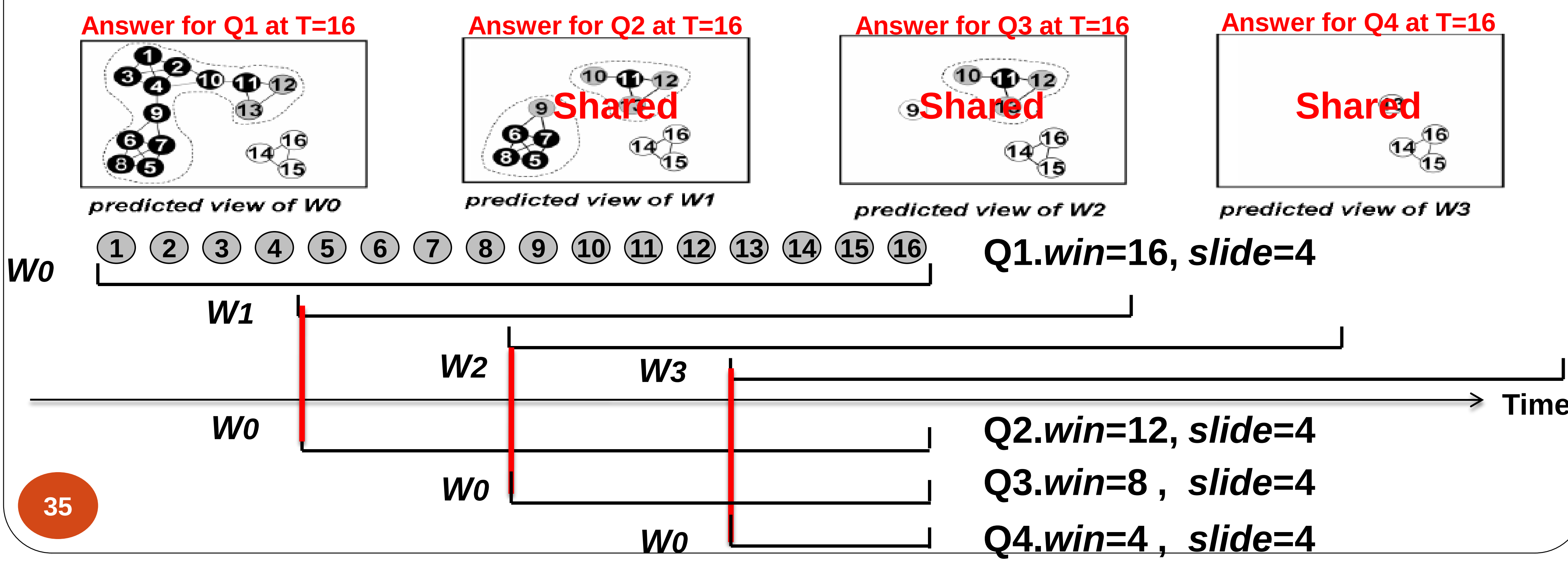
- Why?
  1. Predicted views (windows) maintained by multiple queries may overlap.
  2. The integrated pattern storage and maintenance has to be applied to these predicted views of different queries.

- 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

Parameters

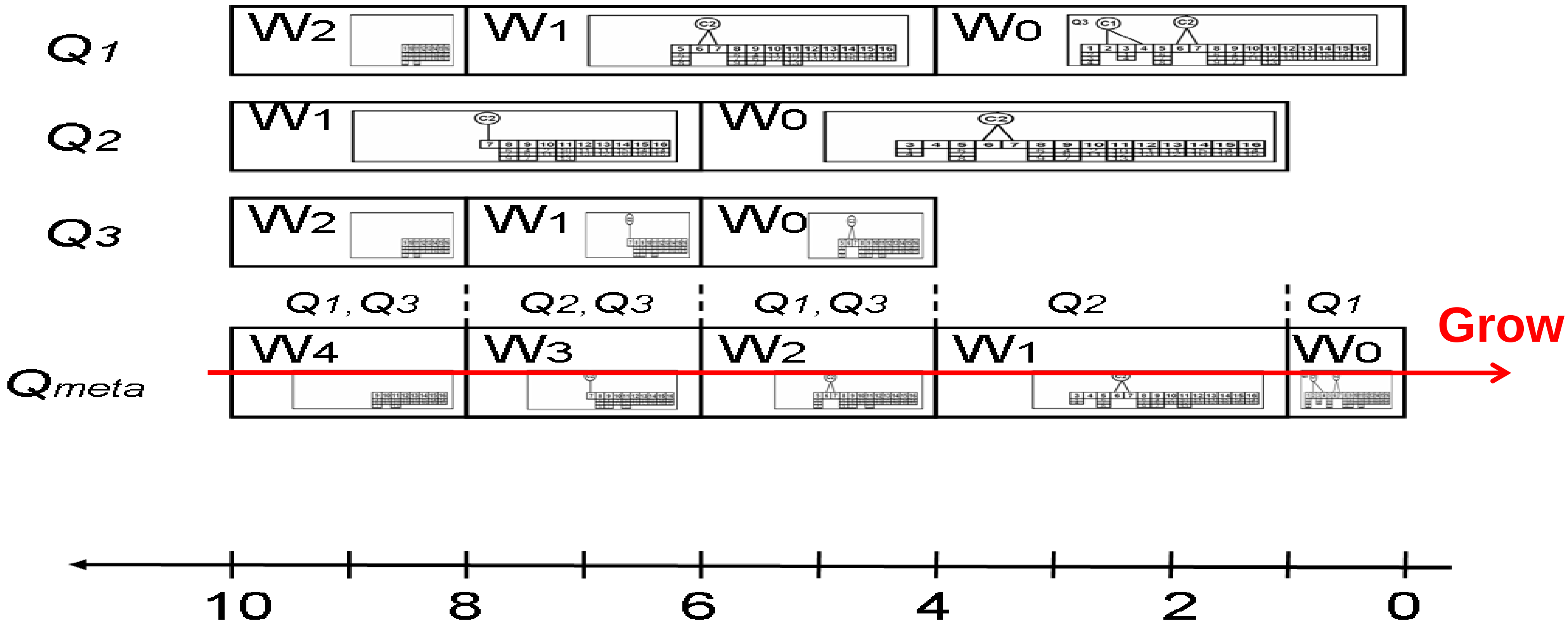
-- arbitrary ***win***, fixed ***slide***

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



Cluster Detection: Vary Window Parameters -- arbitrary *slide*, arbitrary *win*

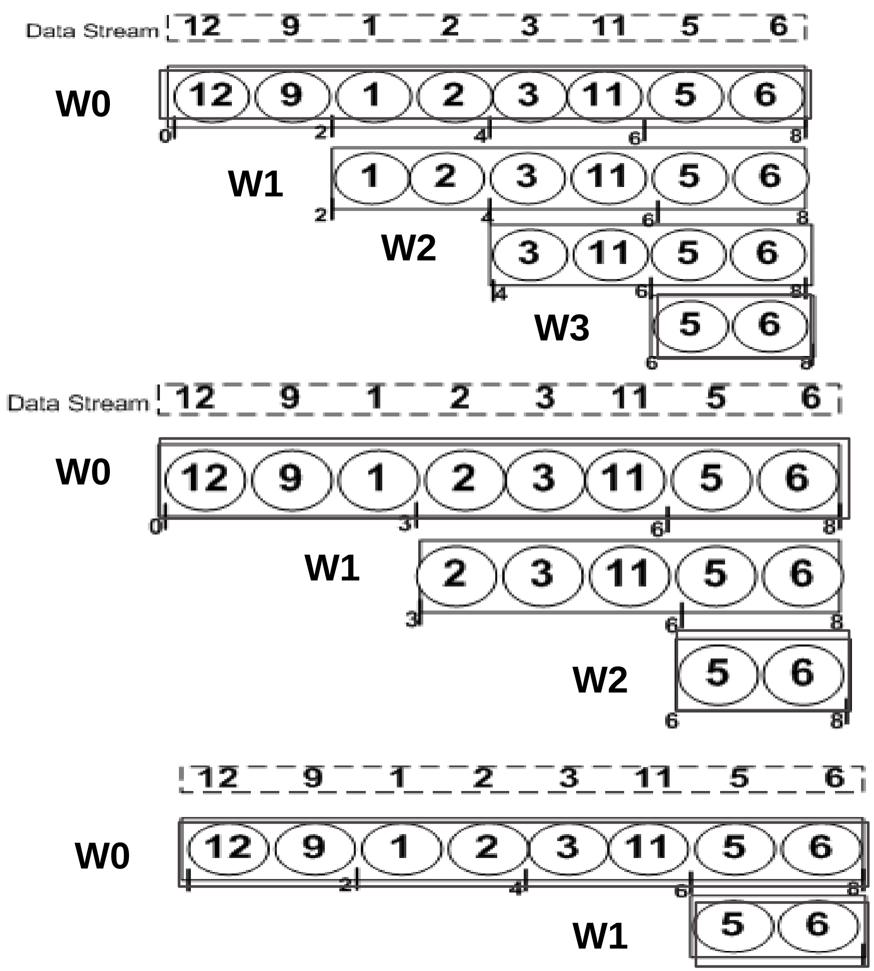
- Use a single *meta query* with largest *window size* and adaptive *slide size* to represent queries.



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

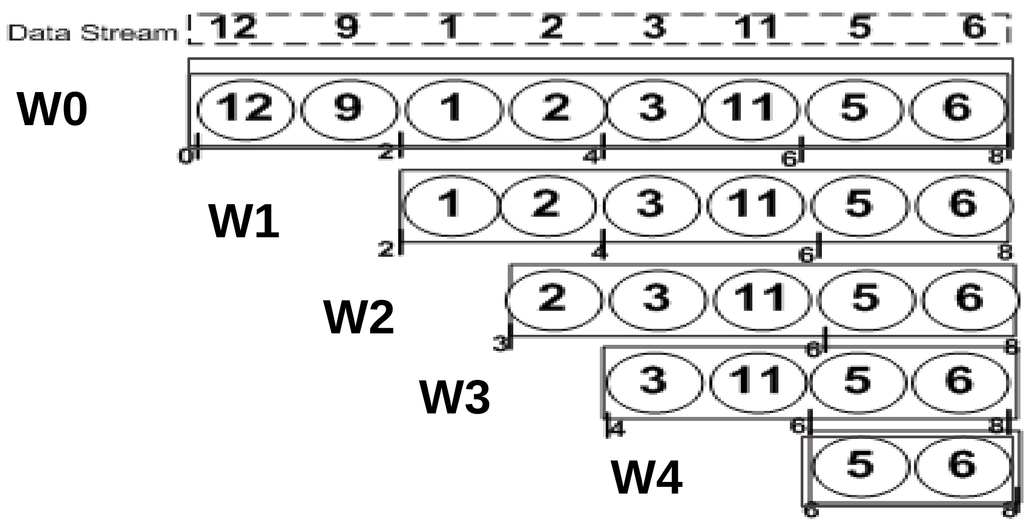
Query Group -
Q1: .. [WIN = 8s SLIDE=2s K=1]
Q2: .. [WIN =8s SLIDE=3s K=3]
Q3: .. [WIN =8s SLIDE=6s K=2]

=

Q meta: … [WIN=8s,SLIDE = ADAPTIVE, K = ADAPTIVE]

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

t= 8s - SLIDE = 2
t=10s - SLIDE = 1
t=11s – SLIDE = 1
t=12s – SLIDE = 2



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

Conclusion for Meta-Query Strategy

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

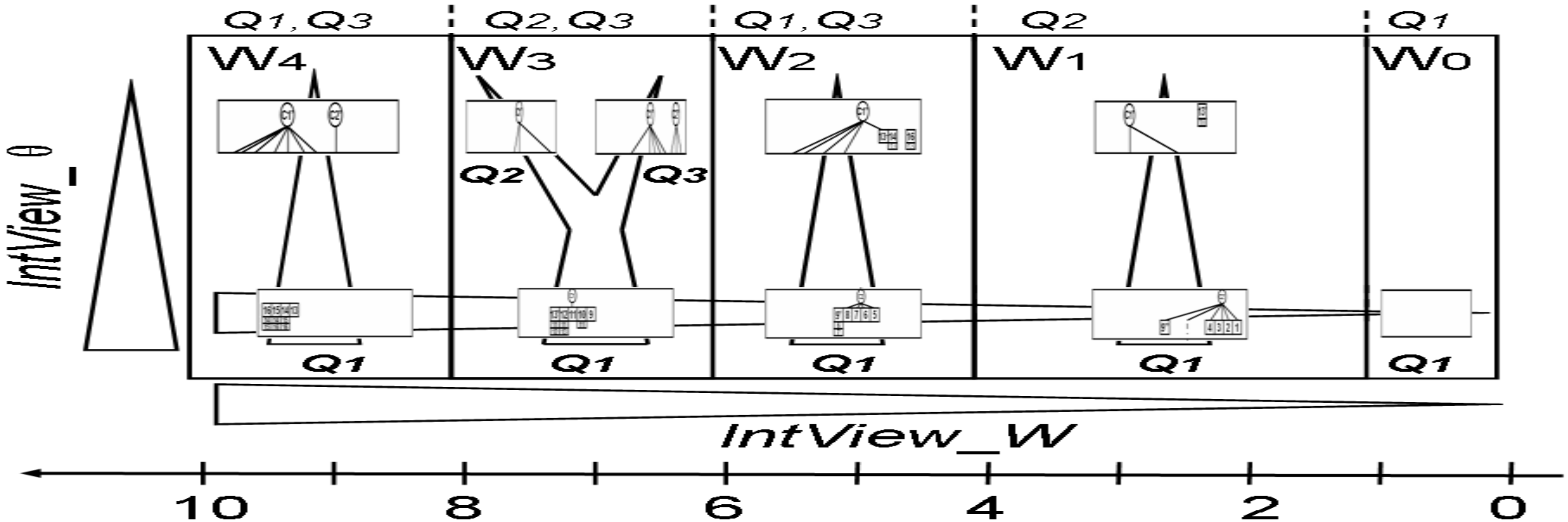
- When can be applied:

  1. Whenever overlapped predicted views exist (little to no overhead even if no overlapped predicted view exist).

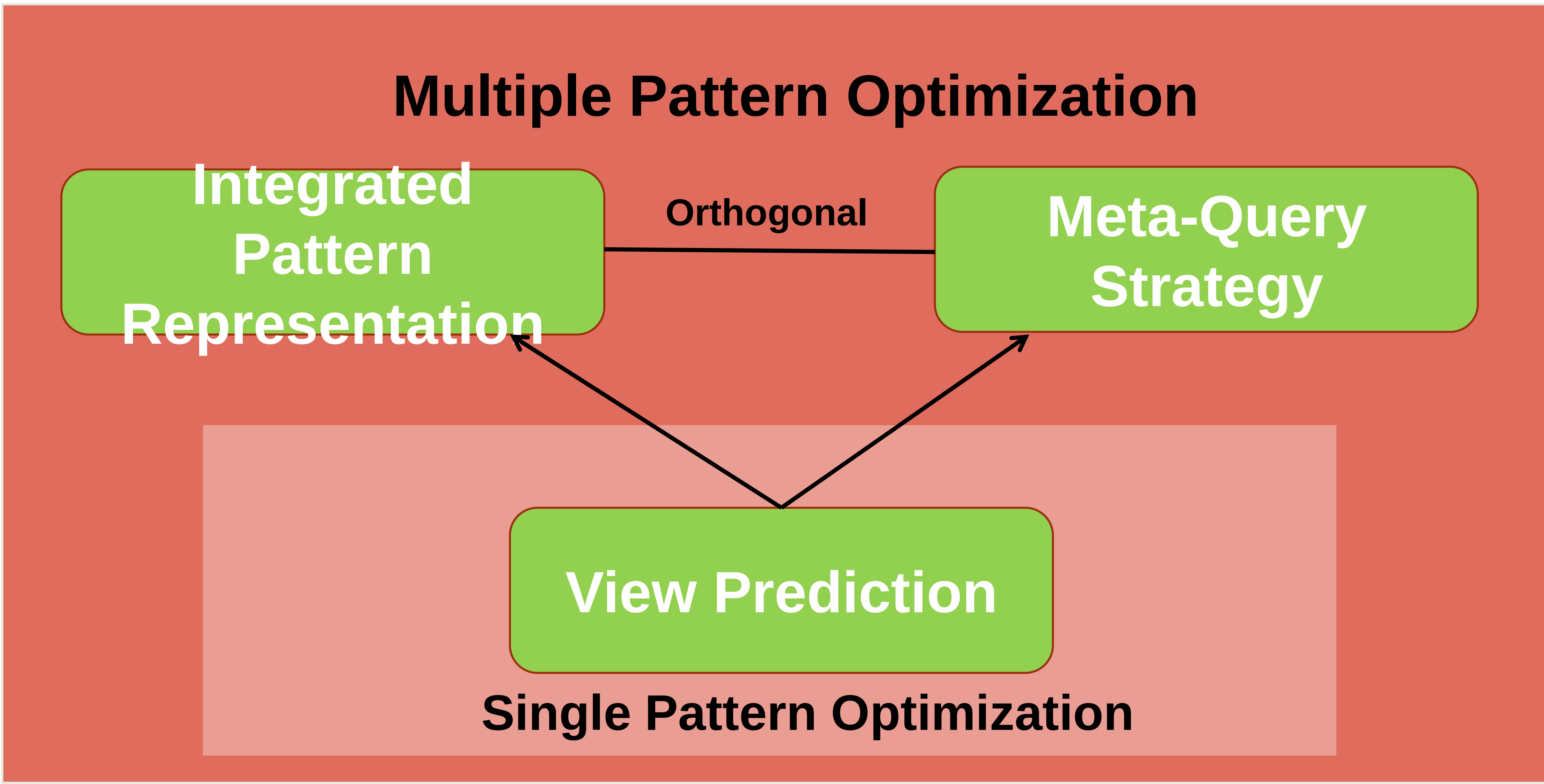
Put it all together
-- arbitrary ***all four parameters (clustering)***

- 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 **IntView** for *QG.*



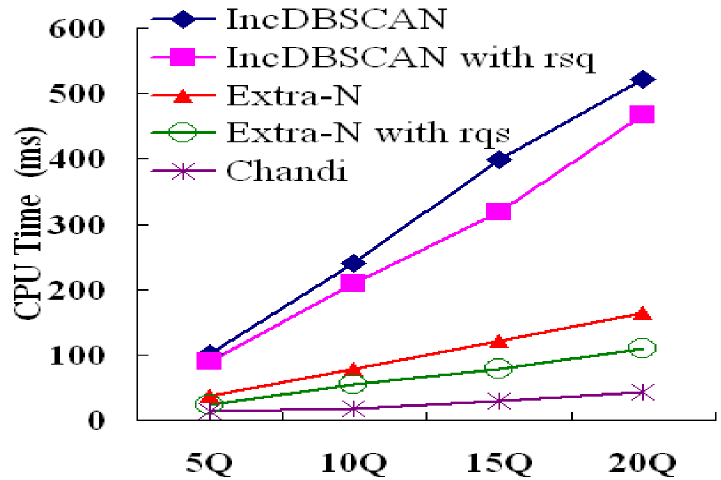
Relationship among optimization principles



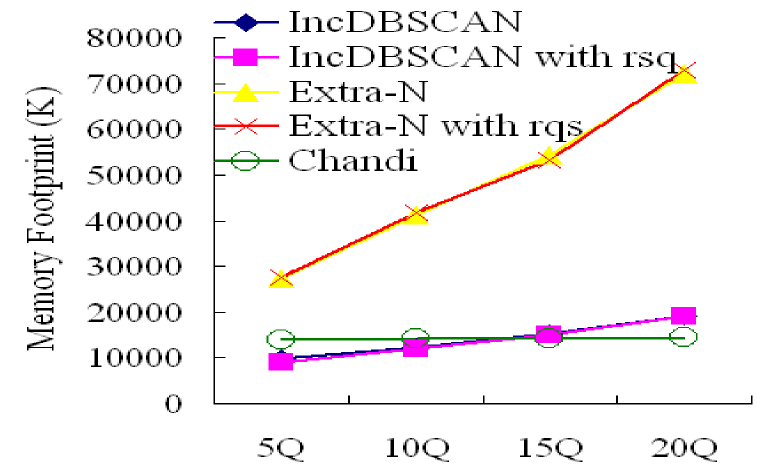
Experimental Study for Clustering

- 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



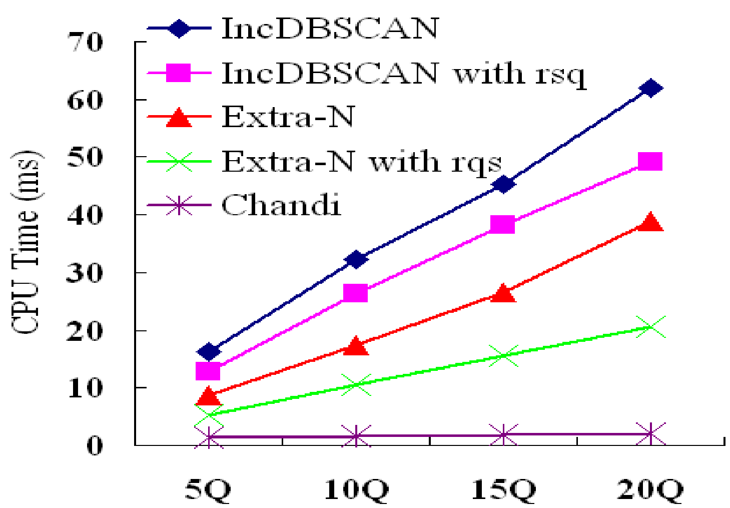
CPU for Arbitrary Pattern Parameter Case



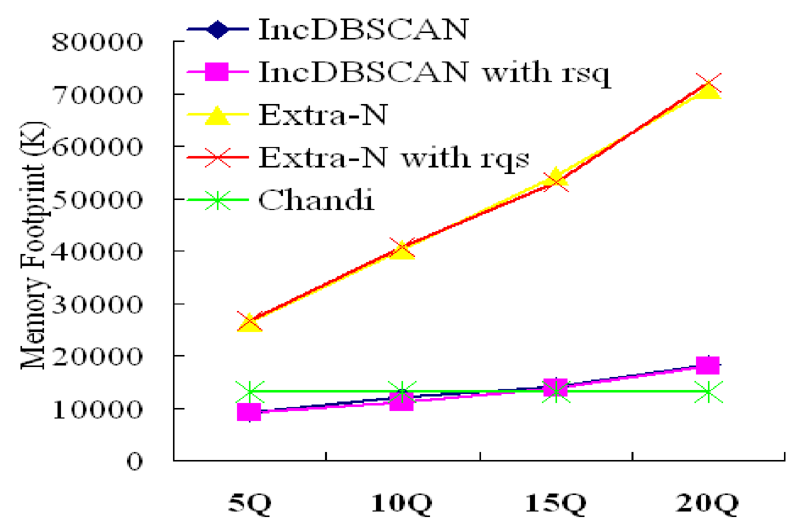
Memory for Arbitrary Pattern Parameter Case

Count parameter vary in 2-20 and range parameter vary in [0.01 – 0.1]

Cluster Performance Evaluation for Queries with Varying Window Parameters



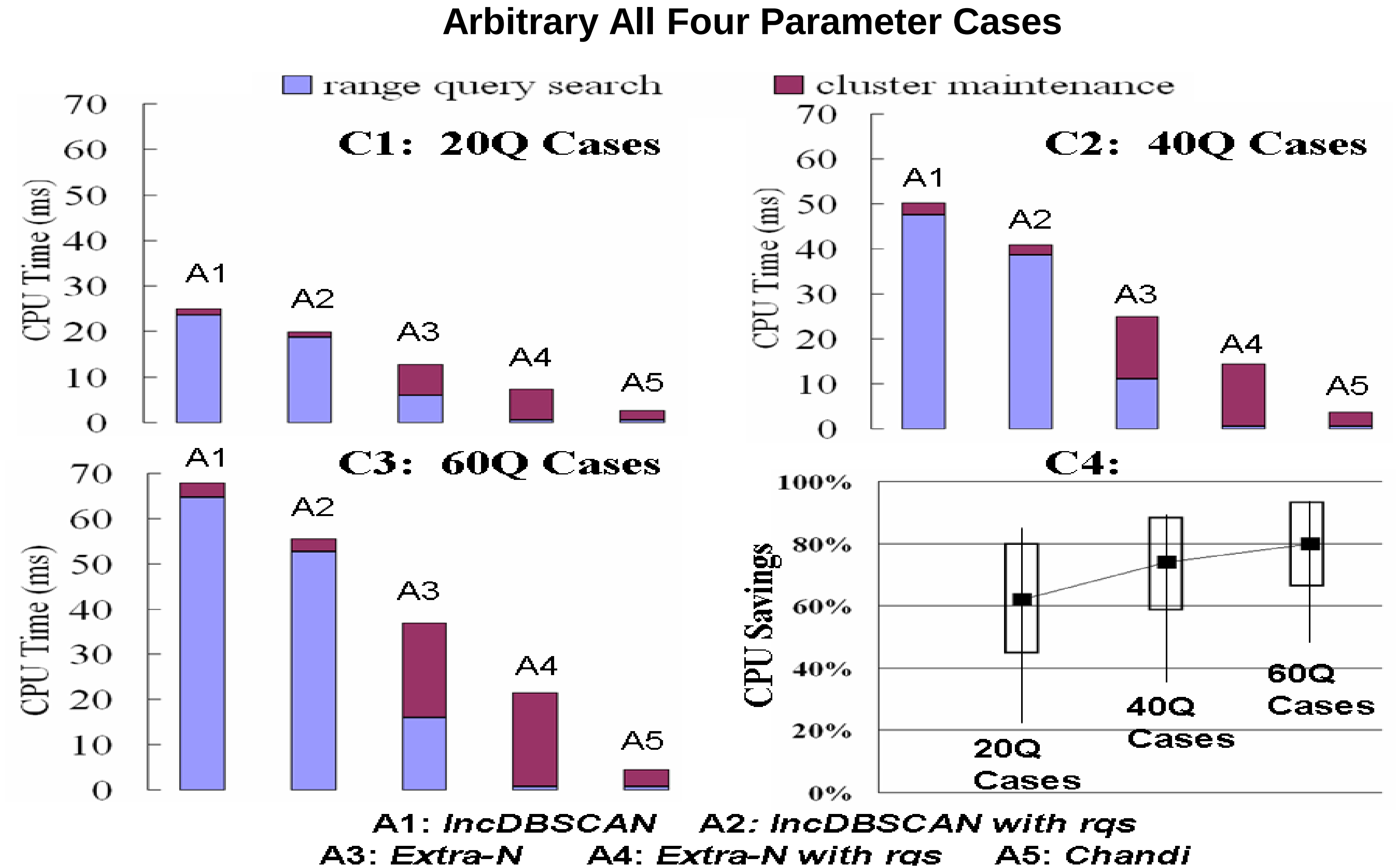
CPU in Arbitrary Window Parameter Case



Memory in Arbitrary Window Parameter Case

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

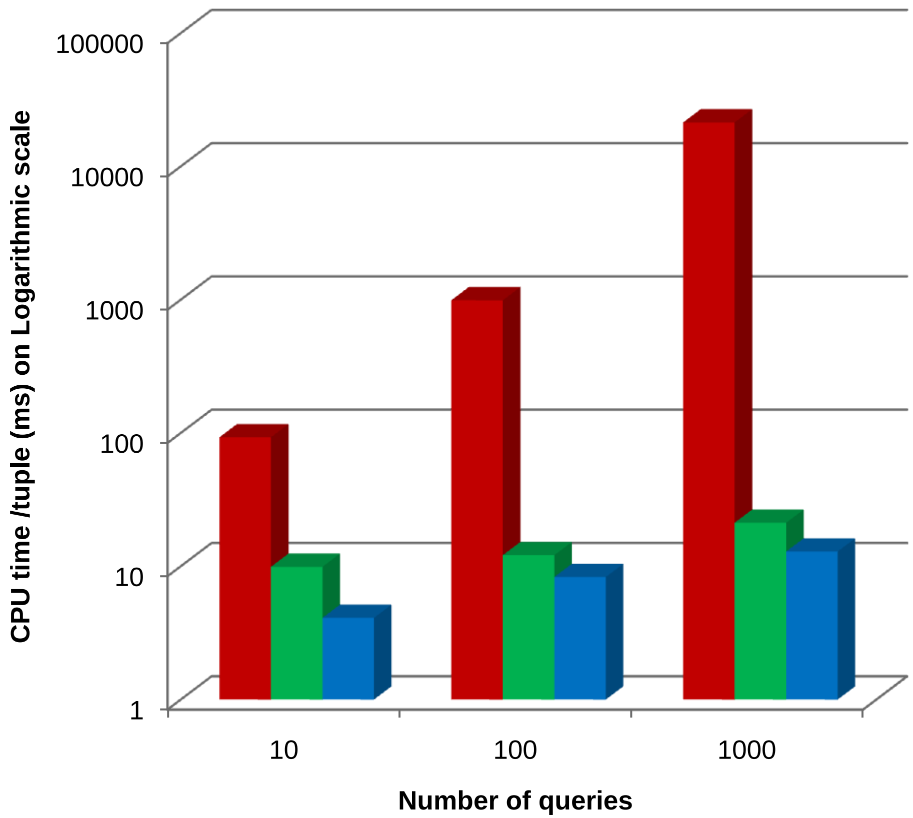
Evaluation for Performance



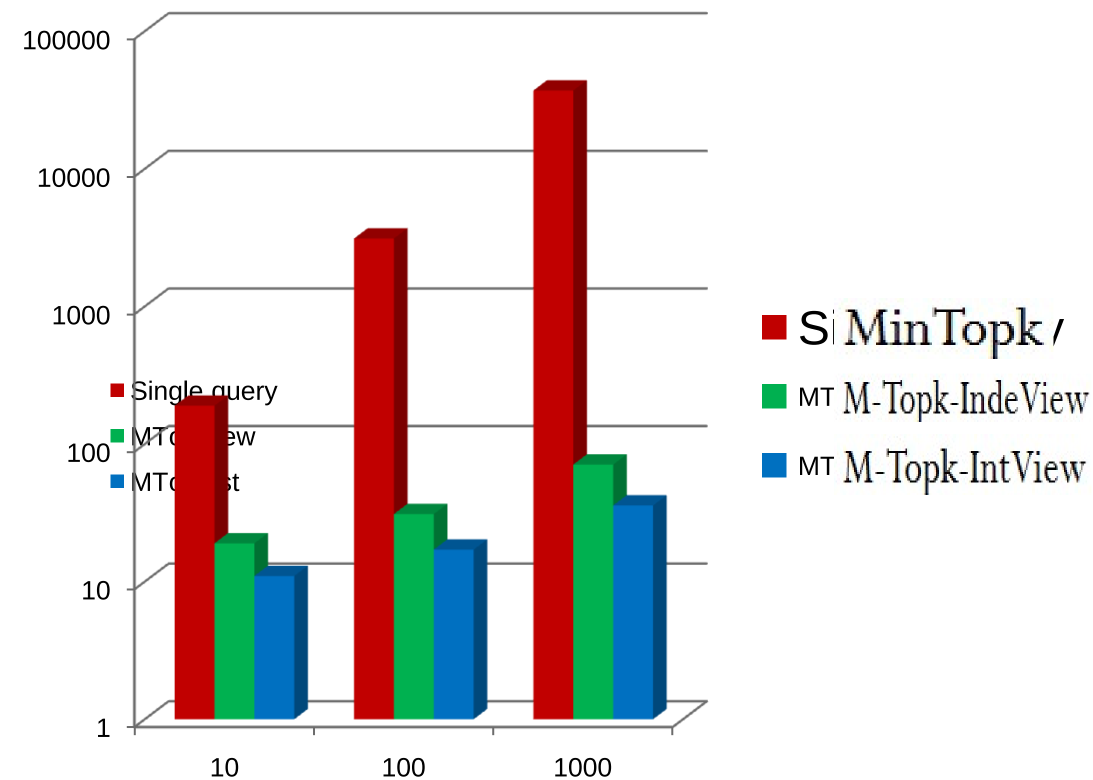
Experimental Study for TopK Requests

- 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.
3. Support interactive pattern mining with visualization.
4. Work collaboratively with domain experts to apply technologies.
5. Explore Extraction and Compaction of Significant Patterns into a Nugget Store

The End

Thanks