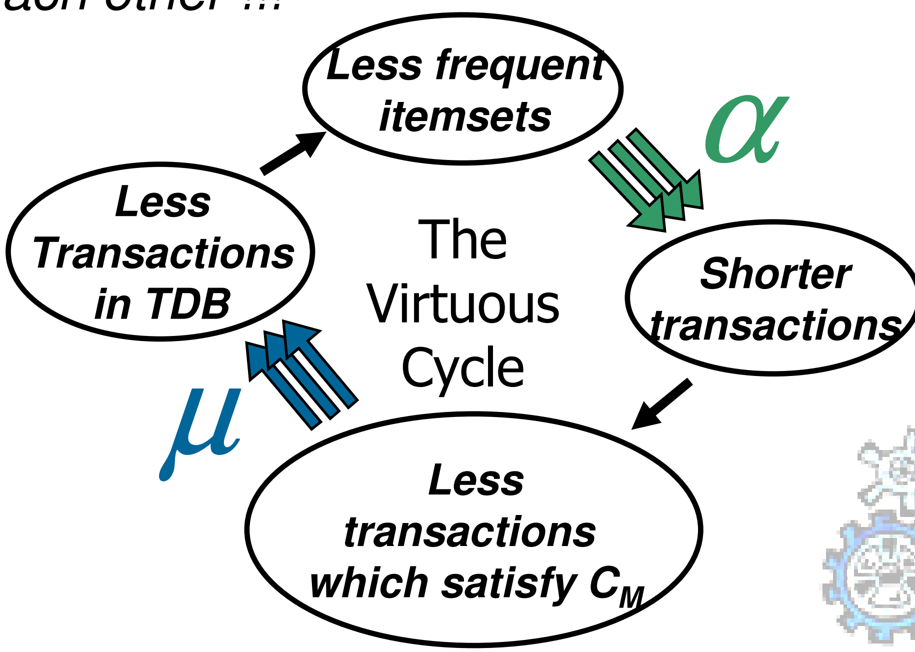
ExAnte Property (Monotone Data Reduction)

- ***ExAnte Property***: *a transaction which does not satisfy a M constraint can be pruned away from TDB, since it will never contribute to the support of any solution itemset.*
- *We call it Monotone Data Reduction and indicate it as **μ-reduction**.*
- ***Level 1 - Antimonotone Data Reduction of Items (α-reduction):*** *a singleton item which is not frequent can be pruned away from all transactions in TDB.*

- *The two components strengthen each other !!!*
- *ExAnte fixpoint computation.*



ExAMiner: key idea and basic data reductions

- *To exploit the real sinergy of AM and M pruning at all levels of a level-wise computation (generalizing Apriori algorithm with M constraints).*
- *Coupling **μ-reduction** with AM data reductions at all levels .*
- *At the generic level k:*

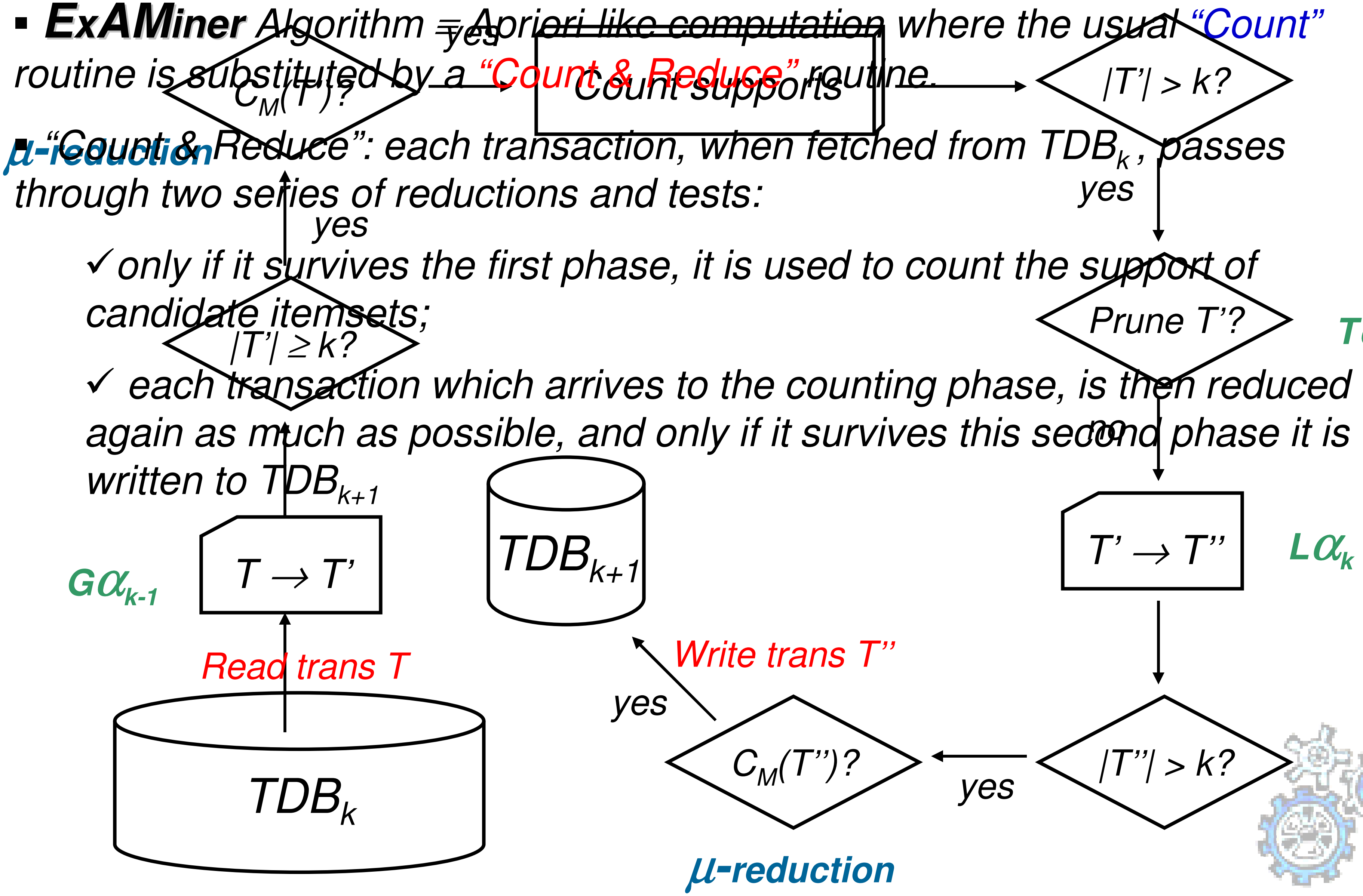
***[$G\alpha_k$] Global Antimonotone Data Reduction of Items**: a singleton item which is not subset of at least k frequent k-itemsets can be pruned away from all transactions in TDB.*

***[$T\alpha_k$] Antimonotone Data Reduction of Transactions**: a transaction which is not superset of at least k+1 candidate k-itemsets can be pruned away from TDB.*

***[$L\alpha_k$] Local Antimonotone Data Reduction of Items**: given an item i and a transaction X, if the number of candidate k-itemsets which are superset of i and subset of X is less than k, then i can be pruned away from transaction X.*

ExAMiner – Count & Reduce

- **ExAMiner** *Algorithm = Apriori-like computation where the usual "Count" routine is substituted by a "Count & Reduce" routine.*
- *"Count & Reduce": each transaction, when fetched from TDB_k, passes through two series of reductions and tests:*
  - ✓ *only if it survives the first phase, it is used to count the support of candidate itemsets;*
  - ✓ *each transaction which arrives to the counting phase, is then reduced again as much as possible, and only if it survives this second phase it is written to TDB_{k+1}*



Further Pruning Opportunities

- *When dealing with the Cardinality Monotone Constraint:* $\mathbf{C_M \equiv card(S) \geq n}$ *we can exploit stronger pruning at very low computational price.*
- *At the generic level k:*

  ***- Enhanced Data Reduction of Items**: a singleton item which is not subset of at least* $\binom{n-1}{k-1}$ *frequent k-itemsets can be pruned away from all transactions in TDB.*

  ***- Generators Pruning**: let* L_k *be the set of frequent k-itemsets, and let* S_k *be the set of itemsets in* L_k *which contain at least a singleton item which does not appear in at least* $\binom{n-1}{k-1}$ *frequent k-itemsets.*

  In order to generate the set of candidates for the next iteration C_{k+1} *do not use the whole set of generators* L_k*; use* $L_k \setminus S_k$ *instead.*

- *This is the first proposal of pruning of the generators ...*

Further Pruning Opportunities

- **Enhanced Local Antimonotone Data Reduction of Items**: *given an item i and a transaction X, if the number of candidate k-itemsets which are superset of i and subset of X is less than $\binom{n-1}{k-1}$ then i can be pruned away from transaction X.*

- *Similar pruning enhancement can be obtained also for all other monotone constraints, inducing weaker conditions from the cardinality based condition.*

- *Example:* $\mathbf{C_M \equiv sum(S.price) \geq m}$

For each item i:

1. *Compute the maximum value of n for which the number of frequent k-itemsets containing i is greater than* $\binom{n-1}{k-1}$

(this value is an upper bound for the maximum size of a frequent itemset containing i)

1. *From this value induce the maximum sum of price for a frequent itemset containing i*
2. *If this sum is less than m, prune away i from all transactions.*

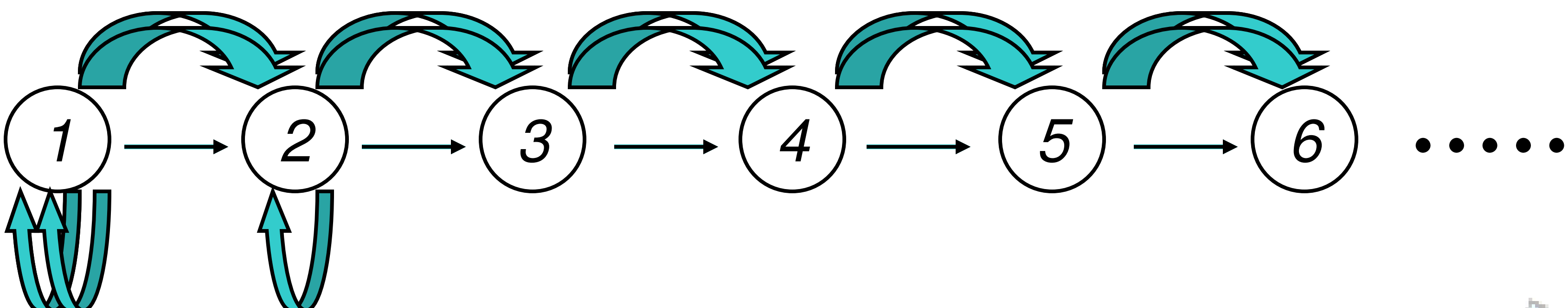
ExAMiner implementations

Count:

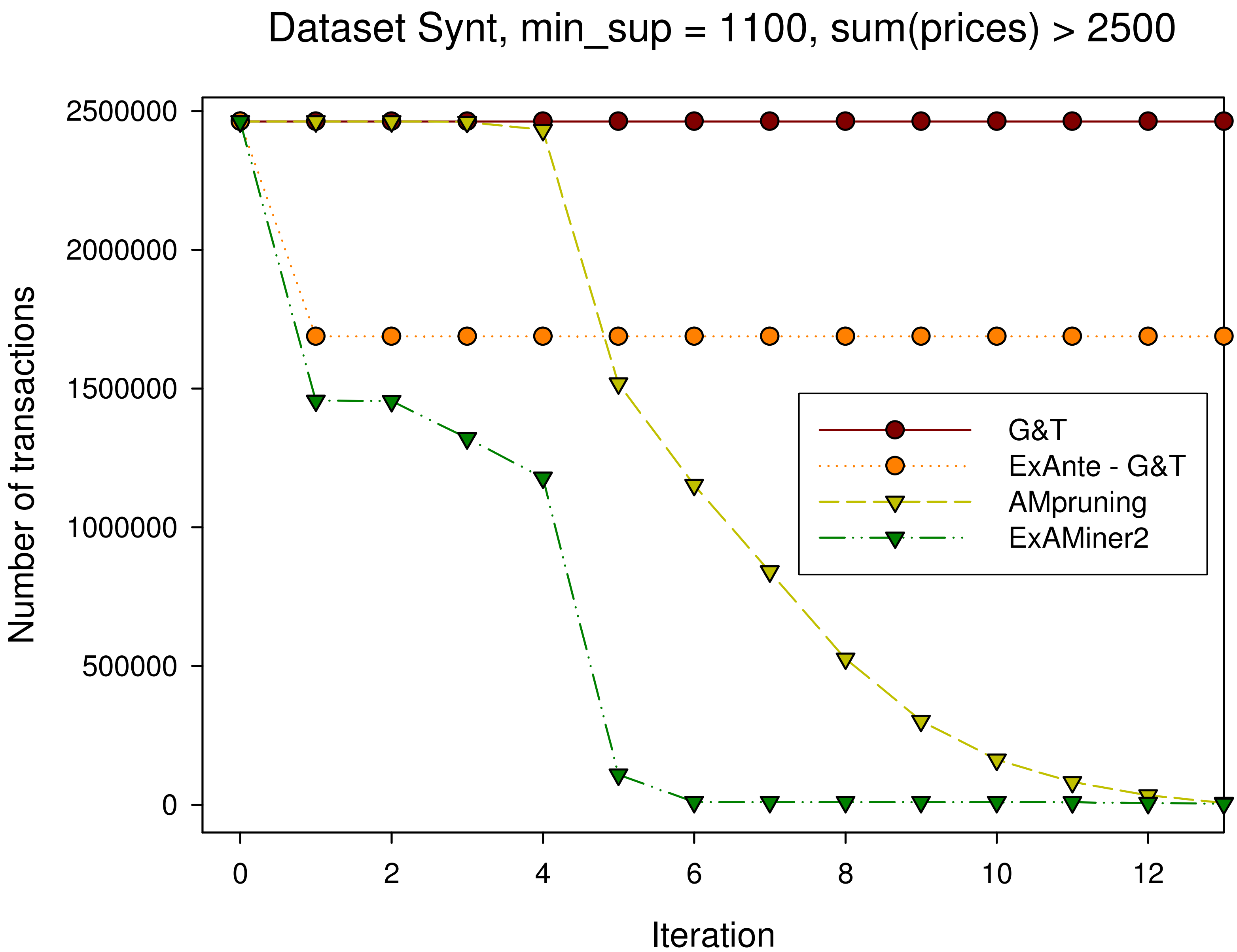
Count and AM reduce:

Count, AM and M reduce:

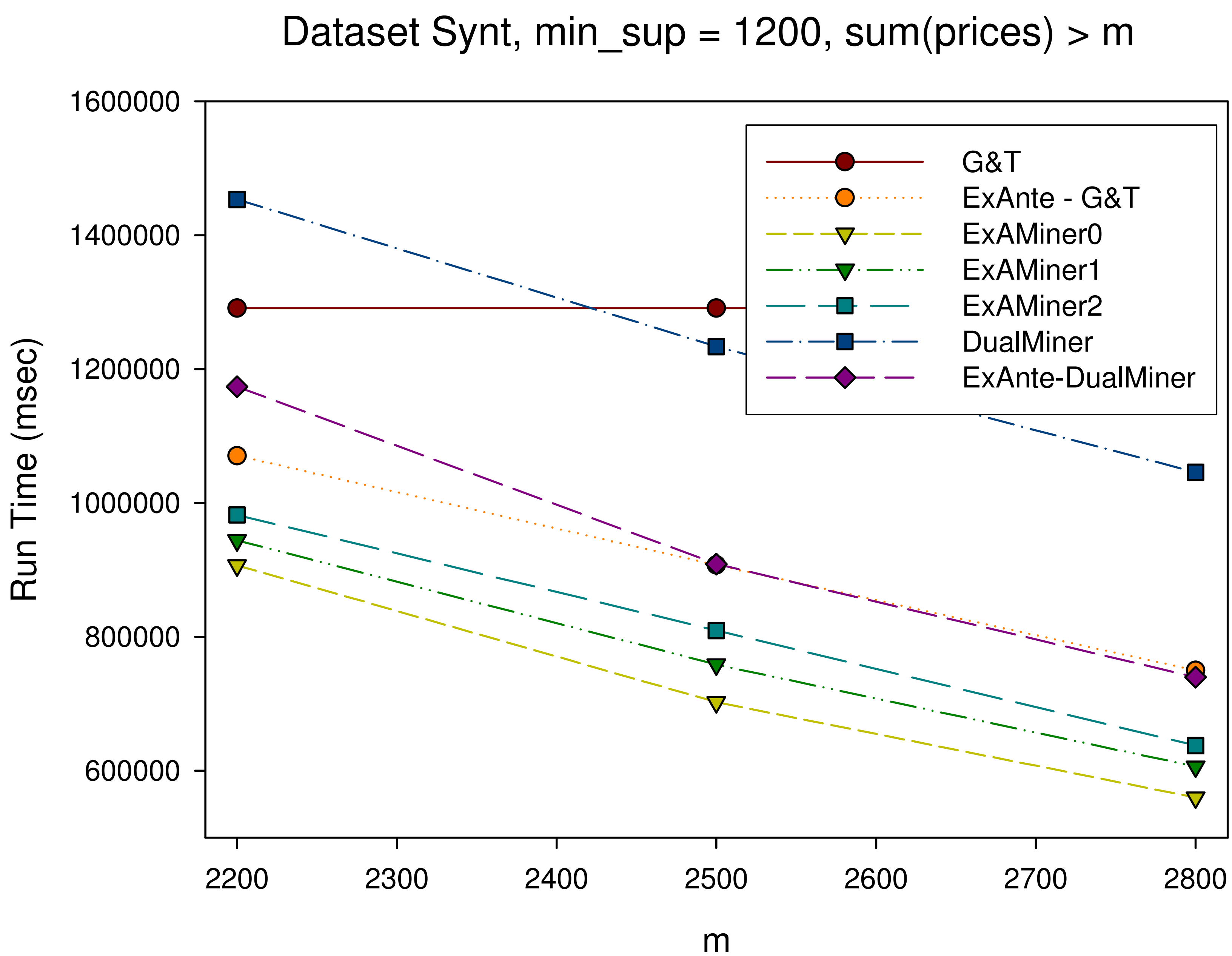
Count, AM and M reduce (fixpoint):



ExAMiner$_2$

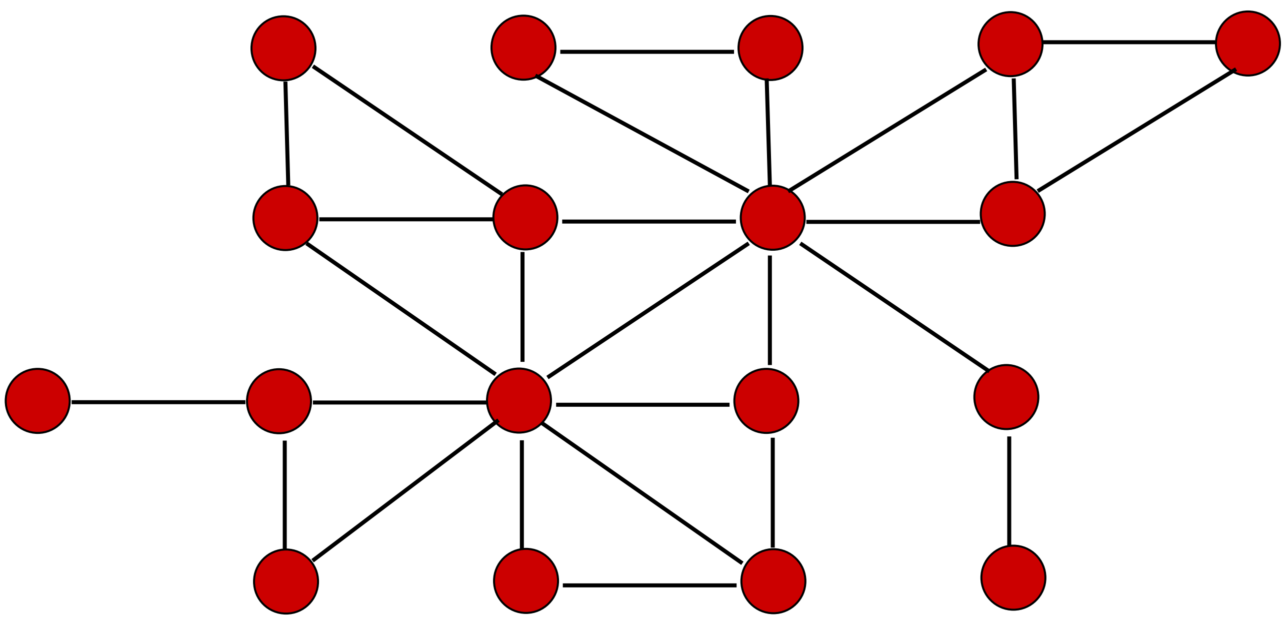


Dataset Synt, min_sup = 1100, sum(prices) > 2500
Number of transactions
0
500000
1000000
1500000
2000000
2500000
Iteration
0
2
4
6
8
10
12
G&T
ExAnte - G&T
AMpruning
ExAMiner2


Dataset Synt, min_sup = 1200, sum(prices) > m
Run Time (msec)
1600000
1400000
1200000
1000000
800000
600000
2200
2300
2400
2500
2600
2700
2800
m
G&T
ExAnte - G&T
ExAMiner0
ExAMiner1
ExAMiner2
DualMiner
ExAnte-DualMiner

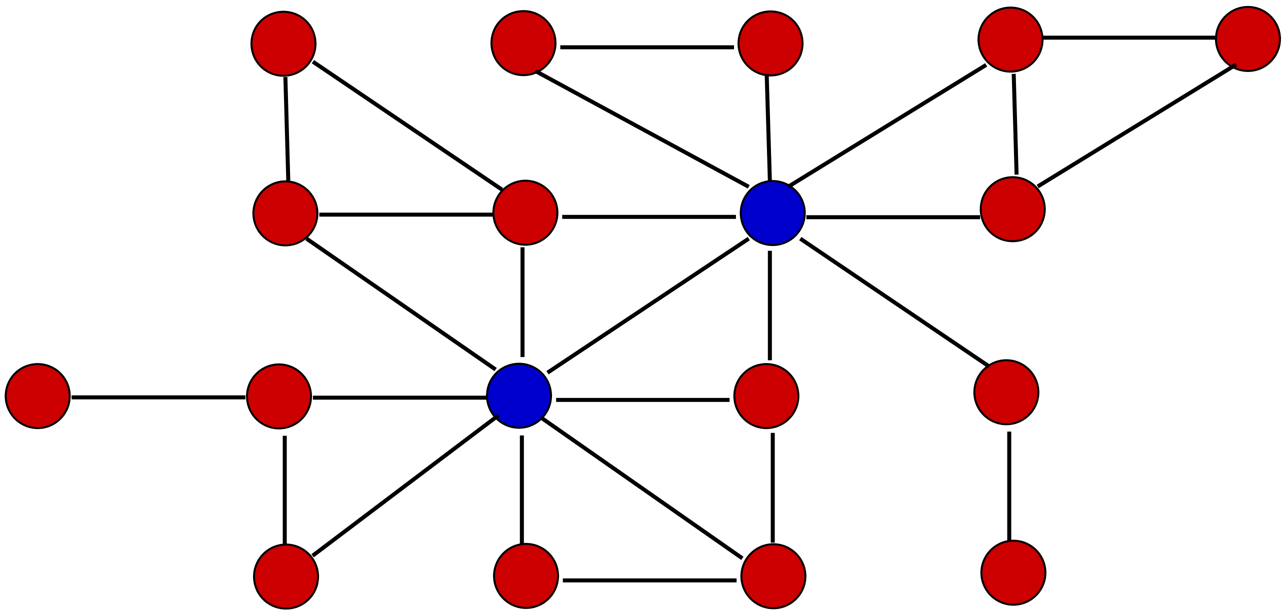
A very general idea

- *Mine frequent connected subgraphs*
- *Containing at least 4 nodes*



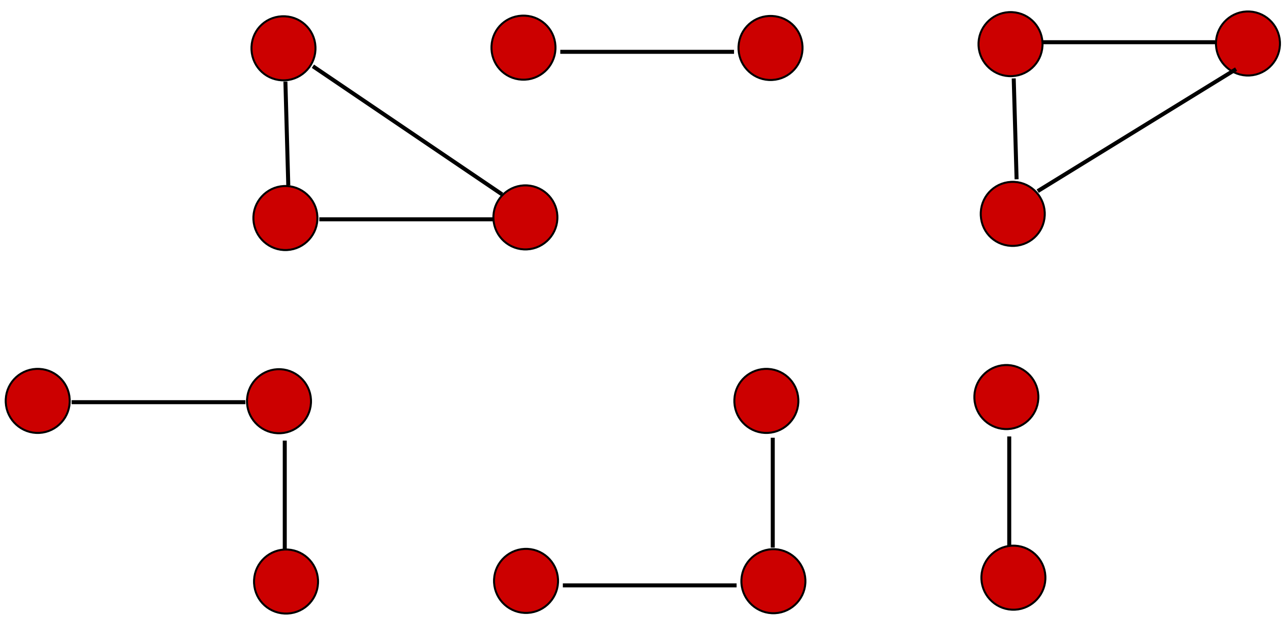
A very general idea

- *Mine frequent connected subgraphs*
- *Containing at least 4 nodes*



A very general idea

- *Mine frequent connected subgraphs*
- *Containing at least 4 nodes*



A New Class of Constraints
(on-going work)

Loose Anti-monotone Constraints

- *Motivations:*
  1. *There are interesting constraints which are not convertible (e.g. variance, standard deviation etc…): can we push them in the frequant pattern computation?*
  2. *For convertible constraints FIC^A and FIC^M solutions not really satisfactory*
  3. *Is it really true that we can not push tough (e.g. convertible) constraints in an Ariori-like frequent pattern computation?*

- *A new class of constraints …*

Anti-monotonicity:

When an intemset S satisfies the constraint, so does any of its subset …

Loose Anti-monotonicity:

When an (k+1)-intemset S, satisfies the constraint, so does at least one of its k-subset…

Class Characterization

- *Convertibe Anti-monotone constraints are Loose Anti-monotone constraints.*

- *There are many interesting constraints which are not Convertible but are Loose Anti-monotone*

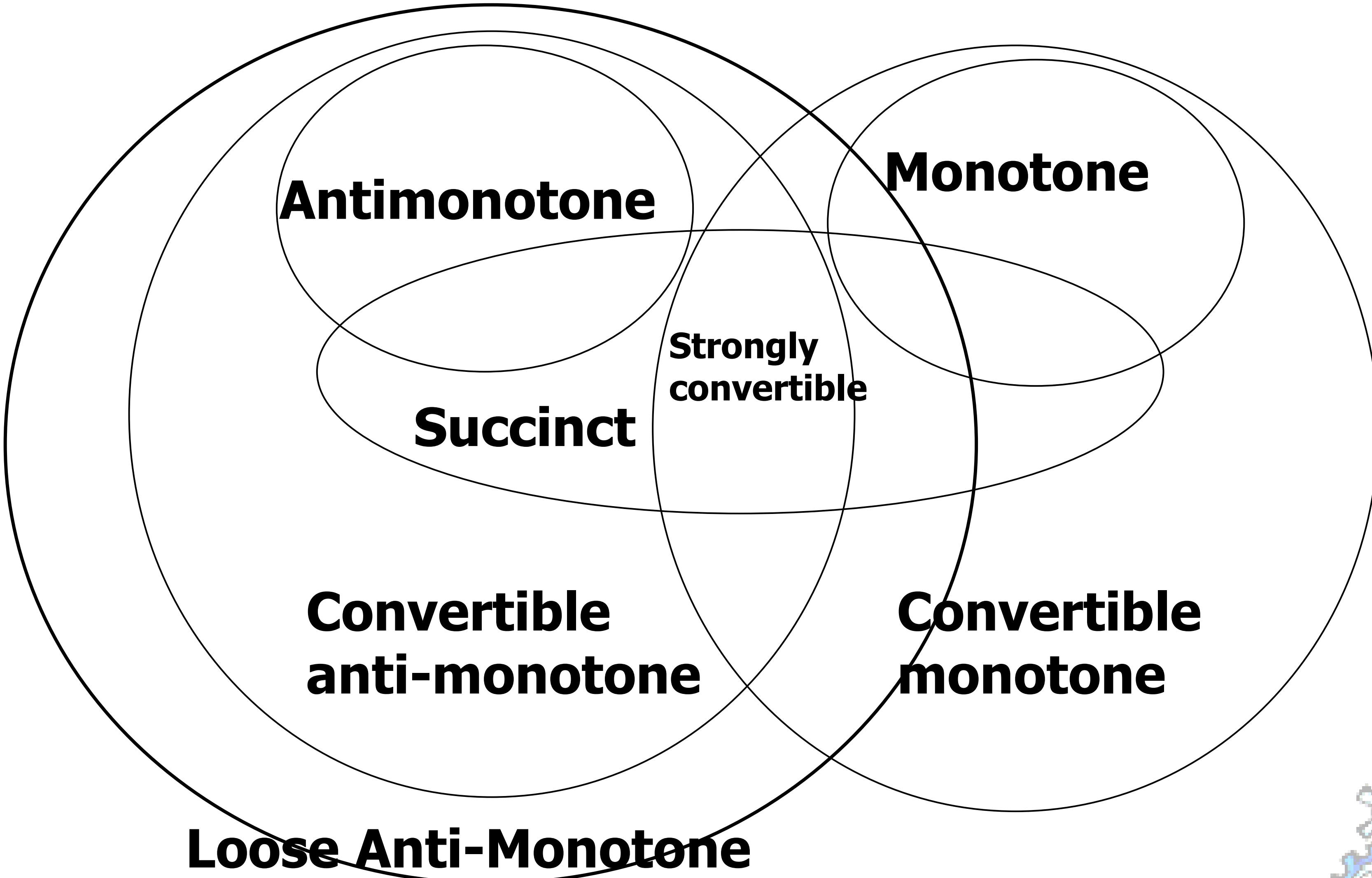
- *Example:* $var(X.profit) \leq n$

  Not Convertible …

  Loose Anti-monotone:

  given an itemset X which satisfies the constraint, let $i \in X$ *be the element of X with larger distance for the avg(X), then the itemset* $X \setminus \{i\}$ *has a variance which smaller than var(X), thus it satisfies the constraint.*

Classification of Constraints



Classification of Constraints

Constraint	Anti-monotone	Monotone	Succinct	Convertible	Loose-$\mathcal{A}$
$min(S.A) \geq v$	yes	no	yes	strongly	yes
$min(S.A) \leq v$	no	yes	yes	strongly	yes
$max(S.A) \geq v$	no	yes	yes	strongly	yes
$max(S.A) \leq v$	yes	no	yes	strongly	yes
$count(S) \leq v$	yes	no	weakly	$\mathcal{A}$	yes
$count(S) \geq v$	no	yes	weakly	$\mathcal{M}$	$k > v$
$sum(S.A) \leq v \ (\forall i \in S, i.A \geq 0)$	yes	no	no	$\mathcal{A}$	yes
$sum(S.A) \geq v \ (\forall i \in S, i.A \geq 0)$	no	yes	no	$\mathcal{M}$	no
$sum(S.A) \leq v \ (v \geq 0, \forall i \in S, i.A\theta 0)$	no	no	no	$\mathcal{A}$	yes
$sum(S.A) \geq v \ (v \geq 0, \forall i \in S, i.A\theta 0)$	no	no	no	$\mathcal{M}$	no
$sum(S.A) \leq v \ (v \leq 0, \forall i \in S, i.A\theta 0)$	no	no	no	$\mathcal{M}$	no
$sum(S.A) \geq v \ (v \leq 0, \forall i \in S, i.A\theta 0)$	no	no	no	$\mathcal{A}$	yes
$range(S.A) \leq v$	yes	no	no	strongly	yes
$range(S.A) \geq v$	no	yes	no	strongly	$k > 2$
$avg(S.A)\theta v$	no	no	no	strongly	yes
$median(S.A)\theta v$	no	no	no	strongly	yes
$var(S.A)\theta v$	no	no	no	no	yes
$std(S.A)\theta v$	no	no	no	no	yes
$md(S.A)\theta v$	no	no	no	no	yes

Table 1. Classification of commonly used constraints (where $\theta \in \{\geq, \leq\}$ and k denotes itemsets cardinality).

A First Interesting Property

Given the conjunction of frequency with a Loose Anti-monotone constraint.

At iteration k:

Loose Antimonotone Data Reduction of Transactions*: a transaction which is not superset of at least one solution k-itemsets can be pruned away from TDB.*

Example: *avg(X.profit) ≥ 15*

t = < a,b,c,d,e,f>

avg(t) = 20

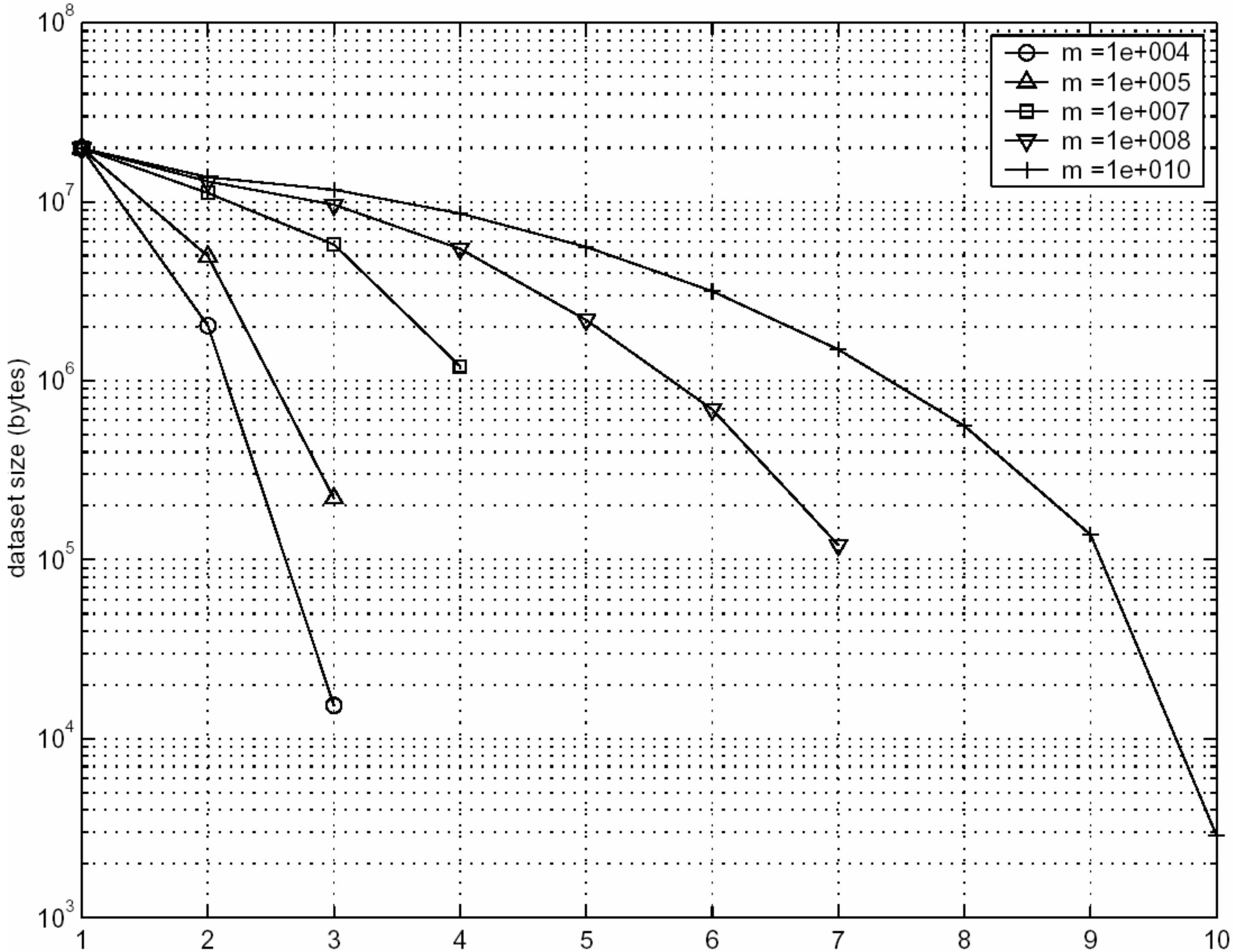
k= 3

t covers 3 frequent itemsets: <b,c,d>, <b,d,e>, <c,d,e>

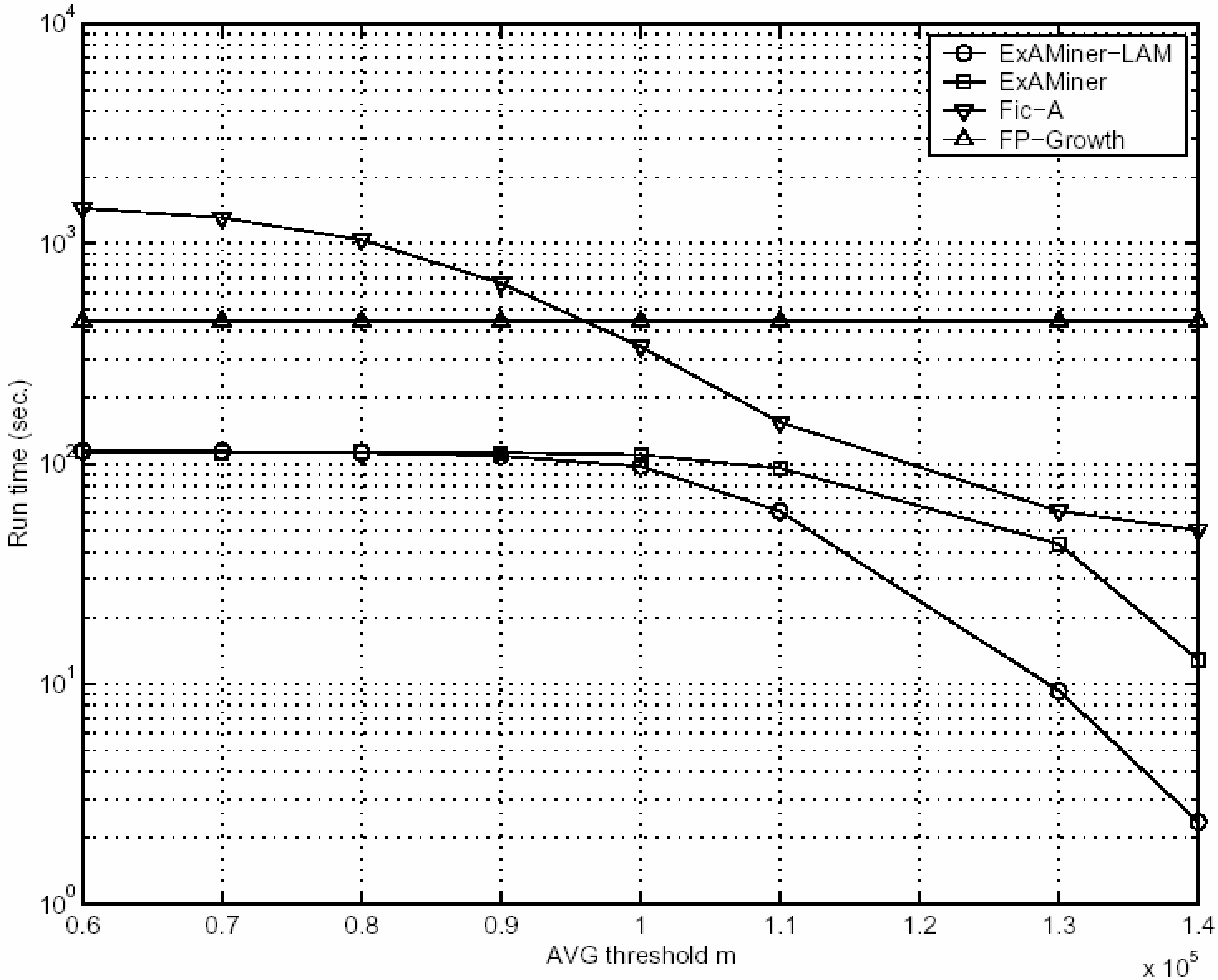
t can be pruned away from TDB

Item	Profit
a	40
b	5
c	20
d	5
e	15
f	35
g	20
h	10

$$\textbf{Dataset BMS-POS, } \sigma = \mathbf{400}, \ \mathcal{C}_{LAM} \equiv \mathbf{var}(\mathbf{X.S}) \leq \mathbf{m}$$



Dataset BMS-POS, $\sigma = 300$, $\mathcal{C}_{CAM} \equiv \mathbf{avg(X.S)} \geq \mathbf{m}$



“On Interactive Pattern Mining from Relational Databases”

ConQueSt

Constraint-based Querying System

KDD Laboratory
HPC Laboratory
ISTI – C.N.R.
Italy

Francesco Bonchi, Fosca Giannotti, Claudio Lucchese,

Salvatore Orlando, Raffaele Perego, Roberto Trasarti

ConQueSt Timeline

Data Mining systems, DMQL, Costraints

2002

Efficient Frequent (and Closed) Itemsets Mining

ExAnte

DCI

Bonchi's Ph.D. thesis

ExAMiner

2003

k-DCI

P3D Project

2004

ExAMiner*lam*

Mining engine — Lucchese, Bonchi

2005

Pre-processor

2006

GUI — Trasarti, Lucchese, Bonchi

2007

Soft Constraints

Graphs, ST Data

Visualization

Berlingerio,Trasarti, Lucchese, Bonchi

ConQueSt Tour 2006

- Demo given at:
  - Black Forest Workshop 06 (Germany)
  - Discussione Tesi Trasarti (Pisa)
  - ICDE'06 (USA)
  - SEBD'06 (Italy)
  - KDID'06 (ECML/PKDD) (Germany)
  - University of Helsinki (Finland)
- Most recent features:
  - Discretization tool
  - On the fly strenghtening/relaxing of contraints
  - Soft constraints (see the talk after the coffee break)

- Plan of the talk:
  - ConQueSt in a nutshell
  - Constraint-based Frequent Pattern Discovery
  - Language, architecture, mining engine
  - Demo
  - Soft Constraints
  - Future developments

ConQueSt in a nutshell

- A Constraint-based Querying System aimed at supporting Frequent Patterns Discovery.
- Follows the *Inductive Database* vision:
  - mining as a querying process
  - closure principle: patterns are first class citizens
  - mining engine amalgamated with commercial DBMS
- Focus on *constraint-based* frequent patterns:
  - large variety of constraints handled
  - very efficient and robust mining engine
- *SPQL: "simple pattern query language"*
  - superset of SQL
  - uses SQL to define the input data sources
  - plus some syntactic sugar to specify data prep-processing
  - plus some syntactic sugar to specify mining parameters

ConQueSt in a nutshell

- Knowledge Discovery is an intrinsically *exploratory* process*:*
  - human-guided
  - interactive
  - Iterative
  - … efficiency is a issue!

- Constraints can be used to drive the discovery process toward potentially interesting patterns.

- Constraints can also be used to reduce the cost of pattern mining computation.

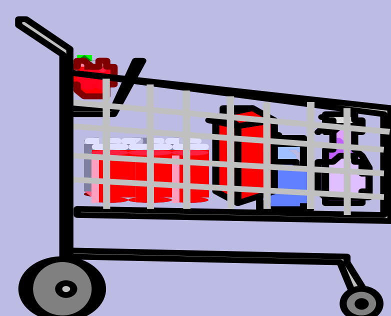
Frequent Pattern Discovery

- *Frequent Pattern Discovery*, i.e. mining patterns which satisfy a user-defined constraint of minimum frequency.
- Basic step of “*Association Rules*” mining
- *Market Basket Analysis*

Milk, eggs, sugar, bread

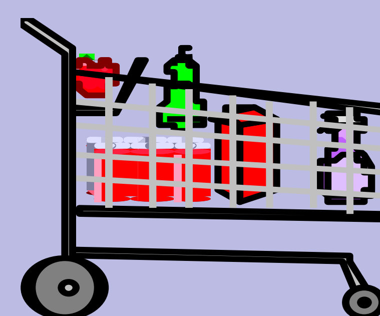
Customer1

Milk, eggs, cereal, bread



Customer2

Eggs, sugar



Customer3

Constraint-based Frequent Patterns

- $I = \{x_1,\ldots,x_n\}$
- Constraint: $C: 2^I \rightarrow \{True, False\}$
- Frequency constraint:
  - D a bag of transactions $t \subseteq I$
  - $sup_D(x) = |\{t \in D \mid X \subseteq t\}|$
  - minimum support σ
  - $sup_D(x) \geq \sigma$
- *Other constraints:*
  - *defined on the items belonging to an itemset*
  - *defined on some attributes of the items*

Constraint-based Frequent Patterns

Transaction ID	Items Bought
1	beer,milk
2	meat,fruit, vegetable
3	beer, fruit
4	fruit, cereals, meat

Item	price
beer	4
milk	2
meat	20
fruit	3
vegetables	15
cereals	6

- *Q:* $sup_D(x) \geq 2 \wedge sum(x.price) \geq 20$
- *Solution set:*
  - *{meat}*
  - *{fruit,meat}*

Constraint-based Frequent Patterns

Transaction ID	Items Bought
1	beer,milk
2	meat,fruit, vegetable
3	beer, fruit
4	fruit, cereals, meat

Item	price
beer	4
milk	2
meat	20
fruit	3
vegetables	15
cereals	6

- This is an ideal situation…

... when you come to real data:

  - No transactions but relations
  - Functional dipendency item→attribute hardly held (e.g. prices change along time)

ConQueSt provides:

- easy way to define the *"mining view"*
  - *just indicate which features are items*
  - *which features are transactions*
  - *which features are items attributes*
  - *it handles both inter-attribute and intra-attribute frequent patterns mining*
- *easy way to solve items-attribute conflicts*
  - *e.g. different prices for item "beer"*
  - *possible solutions: take-first, take-avg, take-min etc…*

ConQueSt

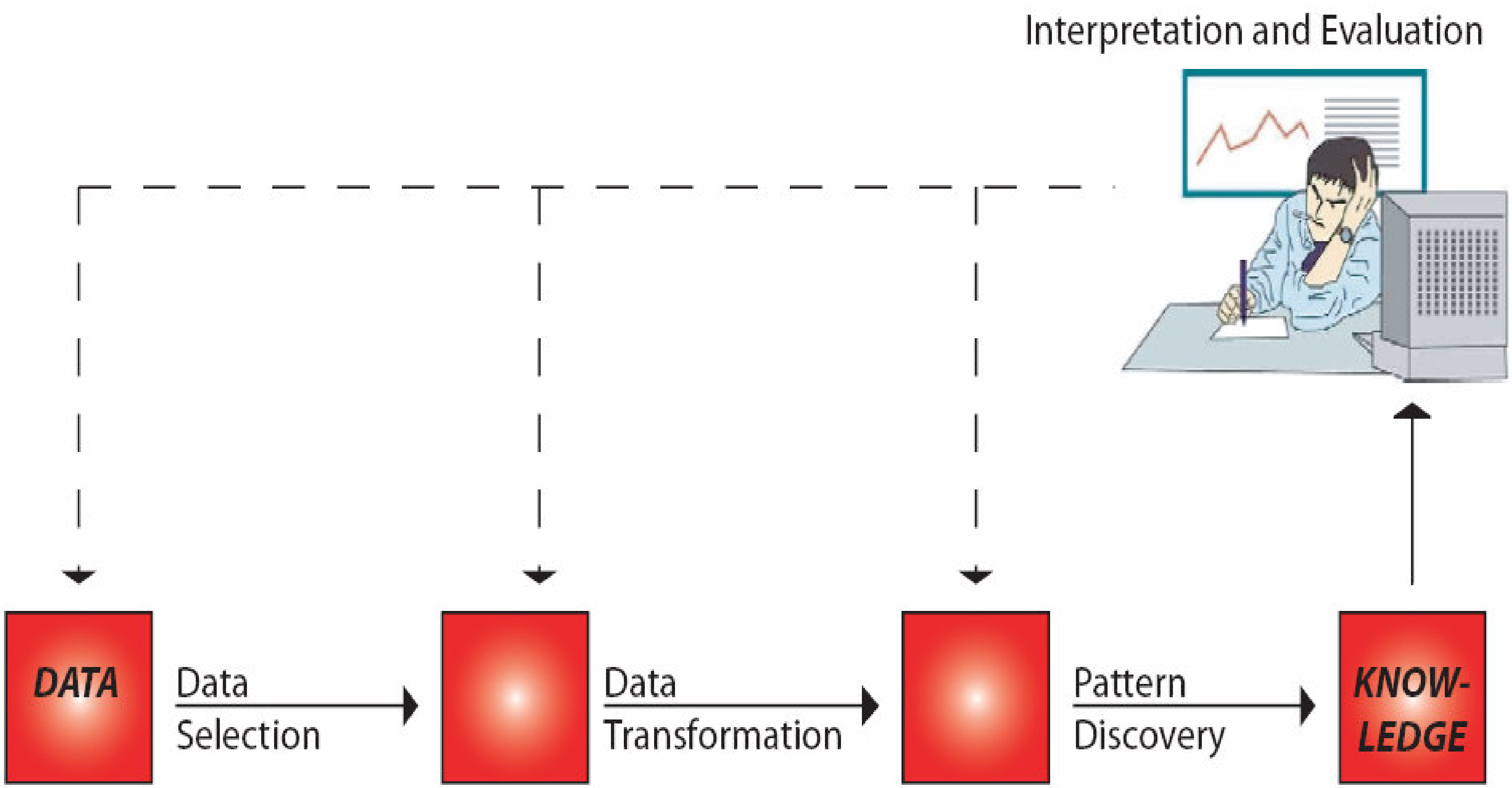
SPQL

(*Simple Pattern Query Language)*

```
MINE PATTERNS WITH SUPP>= 5 IN
SELECT product.product_name, product.gross_weight,
    sales.time_id, sales.customer_id, sales.store_id
FROM [product], [sales_fact_1998]
WHERE sales_fact_1998.product_id=product.product_id
TRANSACTION sales.time_id, sales.customer_id,
    sales.store_id
ITEM product.product_name
ATTRIBUTE product.gross_weight
CONSTRAINED BY Average(product.gross_weight)<=15
```

ConQueSt
SPQL

- Remember the Knowledge Discovery Process?



ConQueSt

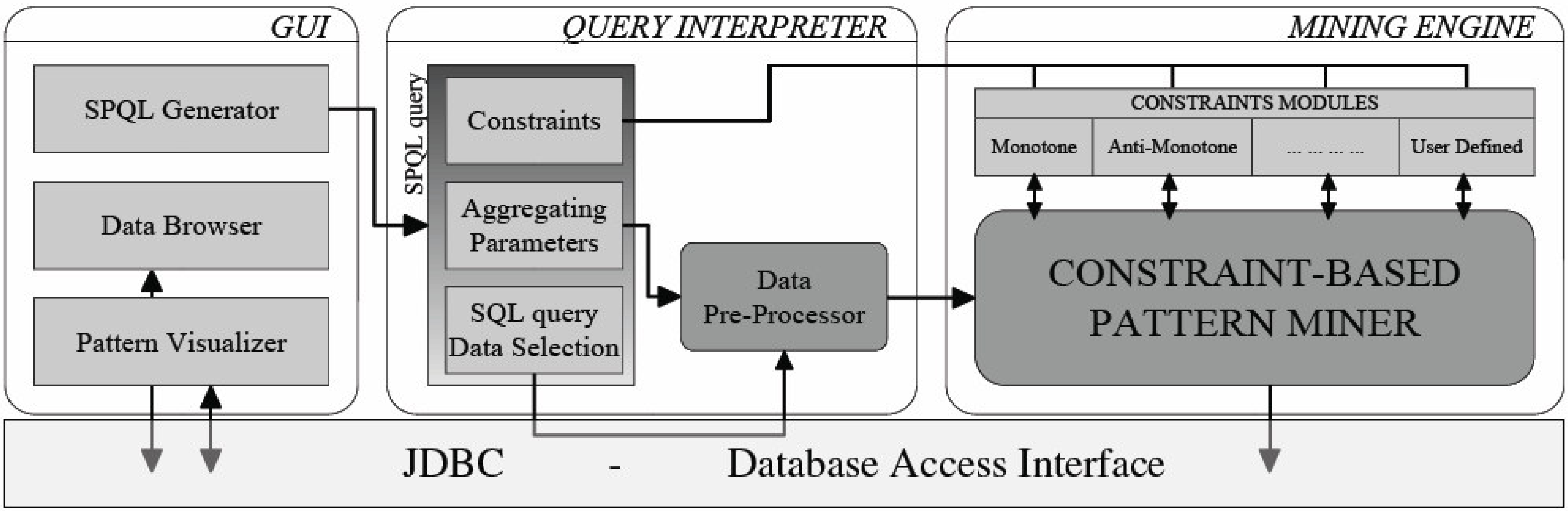
SPQL

(*Simple Pattern Query Language)*

```
MINE PATTERNS WITH SUPP>= 5 IN
SELECT product.product_name, product.gross_weight,
    sales.time_id, sales.customer_id, sales.store_id
FROM [product], [sales_fact_1998]
WHERE sales_fact_1998.product_id=product.product_id
TRANSACTION sales.time_id, sales.customer_id,
    sales.store_id
ITEM product.product_name
ATTRIBUTE product.gross_weight
CONSTRAINED BY Average(product.gross_weight)<=15
```

ConQueSt

Architecture



ConQueSt's mining engine

- Level-wise apriori-like algorithm
- **DCI** + **ExAMiner** + **ExAMiner**^{lam} + …
- Able to push a large variety of constraints
  subset, supset, lenght, min, max, sum, range, avg, var, med, md, std, etc…
- Efficient and robust
- Modular
- Data aware
- Resource aware

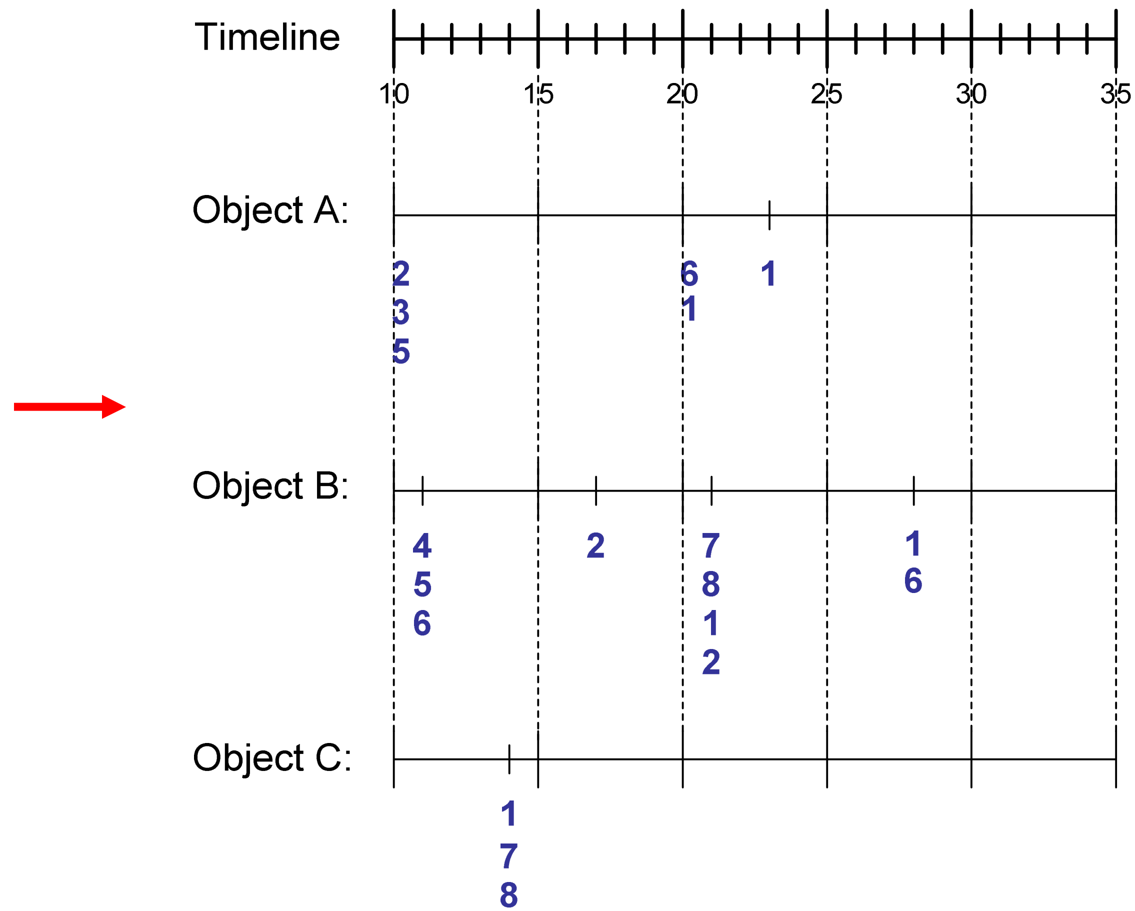
ConQueSt

Demo

Sequence Data

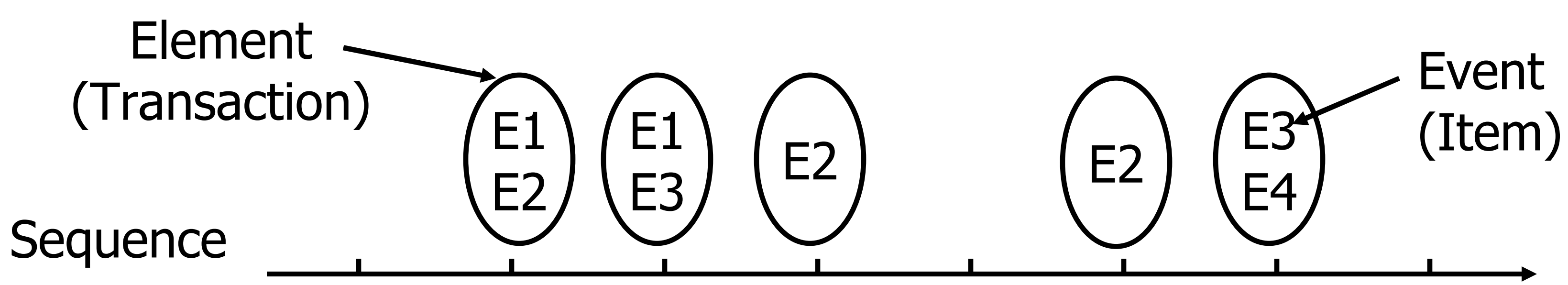
Sequence Database:

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7



Examples of Sequence Data

Sequence Database	Sequence	Element (Transaction)	Event (Item)
Customer	Purchase history of a given customer	A set of items bought by a customer at time t	Books, diary products, CDs, etc
Web Data	Browsing activity of a particular Web visitor	A collection of files viewed by a Web visitor after a single mouse click	Home page, index page, contact info, etc
Event data	History of events generated by a given sensor	Events triggered by a sensor at time t	Types of alarms generated by sensors
Genome sequences	DNA sequence of a particular species	An element of the DNA sequence	Bases A,T,G,C



Formal Definition of a Sequence

- A sequence is an ordered list of elements (transactions)

$$s = \langle e_1 \, e_2 \, e_3 \, \ldots \rangle$$

  – Each element contains a collection of events (items)

$$e_i = \{i_1, i_2, \ldots, i_k\}$$

  – Each element is attributed to a specific time or location

- Length of a sequence, $|s|$, is given by the number of elements of the sequence
- A k-sequence is a sequence that contains k events (items)

Examples of Sequence

- Web sequence:

  < {Homepage} {Electronics} {Digital Cameras} {Canon Digital Camera} {Shopping Cart} {Order Confirmation} {Return to Shopping} >

- Sequence of initiating events causing the nuclear accident at 3-mile Island:
  (http://stellar-one.com/nuclear/staff_reports/summary_SOE_the_initiating_event.htm)

  < {clogged resin} {outlet valve closure} {loss of feedwater} {condenser polisher outlet valve shut} {booster pumps trip} {main waterpump trips} {main turbine trips} {reactor pressure increases}>

- Sequence of books checked out at a library:

  <{Fellowship of the Ring} {The Two Towers} {Return of the King}>

Formal Definition of a Subsequence

- A sequence $<a_1\ a_2 \dots a_n>$ is contained in another sequence $<b_1\ b_2 \dots b_m>$ ($m \geq n$) if there exist integers $i_1 < i_2 < \dots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i1}$, …, $a_n \subseteq b_{in}$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {8} >	< {2} {3,5} >	Yes
< {1,2} {3,4} >	< {1} {2} >	No
< {2,4} {2,4} {2,5} >	< {2} {4} >	Yes

- The support of a subsequence w is defined as the fraction of data sequences that contain w
- A *sequential pattern* is a frequent subsequence (i.e., a subsequence whose support is ≥ *minsup*)

Sequential Pattern Mining: Definition

- Given:
  - a database of sequences
  - a user-specified minimum support threshold, *minsup*

- Task:
  - Find all subsequences with support ≥ *minsup*

Sequential Pattern Mining: Challenge

- Given a sequence: <{a b} {c d e} {f} {g h i}>
  - Examples of subsequences:

    <{a} {c d} {f} {g} >, < {c d e} >, < {b} {g} >, etc.

- How many k-subsequences can be extracted from a given n-sequence?

<{a b} {c d e} {f} {g h i}> n = 9

k=4: Y _ _ Y Y _ _ _ Y

<{a} {d e} {i}>

Answer :

$$\binom{n}{k} = \binom{9}{4} = 126$$

Sequential Pattern Mining: Example

Object	Timestamp	Events
A	1	1,2,4
A	2	2,3
A	3	5
B	1	1,2
B	2	2,3,4
C	1	1, 2
C	2	2,3,4
C	3	2,4,5
D	1	2
D	2	3, 4
D	3	4, 5
E	1	1, 3
E	2	2, 4, 5

Minsup = 50%

Examples of Frequent Subsequences:

< {1,2} >	s=60%
< {2,3} >	s=60%
< {2,4}>	s=80%
< {3} {5}>	s=80%
< {1} {2} >	s=80%
< {2} {2} >	s=60%
< {1} {2,3} >	s=60%
< {2} {2,3} >	s=60%
< {1,2} {2,3} >	s=60%

Extracting Sequential Patterns

- Given n events: $i_1, i_2, i_3, \ldots, i_n$
- Candidate 1-subsequences:

  $<\{i_1\}>, <\{i_2\}>, <\{i_3\}>, \ldots, <\{i_n\}>$
- Candidate 2-subsequences:

  $<\{i_1, i_2\}>, <\{i_1, i_3\}>, \ldots, <\{i_1\} \{i_1\}>, <\{i_1\} \{i_2\}>, \ldots, <\{i_{n-1}\} \{i_n\}>$
- Candidate 3-subsequences:

  $<\{i_1, i_2, i_3\}>, <\{i_1, i_2, i_4\}>, \ldots, <\{i_1, i_2\} \{i_1\}>, <\{i_1, i_2\} \{i_2\}>, \ldots,$

  $<\{i_1\} \{i_1, i_2\}>, <\{i_1\} \{i_1, i_3\}>, \ldots, <\{i_1\} \{i_1\} \{i_1\}>, <\{i_1\} \{i_1\} \{i_2\}>, \ldots$

Generalized Sequential Pattern (GSP)

- **Step 1**:
  - Make the first pass over the sequence database D to yield all the 1-element frequent sequences
- **Step 2**:

  Repeat until no new frequent sequences are found
  - **Candidate Generation**:
    - Merge pairs of frequent subsequences found in the (k-1)*th* pass to generate candidate sequences that contain k items
  - **Candidate Pruning**:
    - Prune candidate *k*-sequences that contain infrequent *(k-1)*-subsequences
  - **Support Counting**:
    - Make a new pass over the sequence database D to find the support for these candidate sequences
  - **Candidate Elimination**:
    - Eliminate candidate *k*-sequences whose actual support is less than *minsup*

Candidate Generation

- Base case (k=2):
  - Merging two frequent 1-sequences <$\{i_1\}$> and <$\{i_2\}$> will produce two candidate 2-sequences: <$\{i_1\}$ $\{i_2\}$> and <$\{i_1\ i_2\}$>
- General case (k>2):
  - A frequent *(k-1)*-sequence w_1 is merged with another frequent *(k-1)*-sequence w_2 to produce a candidate *k*-sequence if the subsequence obtained by removing the first event in w_1 is the same as the subsequence obtained by removing the last event in w_2
    - The resulting candidate after merging is given by the sequence w_1 extended with the last event of w_2.
      - If the last two events in w_2 belong to the same element, then the last event in w_2 becomes part of the last element in w_1
      - Otherwise, the last event in w_2 becomes a separate element appended to the end of w_1

Candidate Generation Examples

- Merging the sequences
  w_1=<{1} {2 3} {4}> and w_2 =<{2 3} {4 5}>
  will produce the candidate sequence < {1} {2 3} {4 5}> because the last two events in w_2 (4 and 5) belong to the same element

- Merging the sequences
  w_1=<{1} {2 3} {4}> and w_2 =<{2 3} {4} {5}>
  will produce the candidate sequence < {1} {2 3} {4} {5}> because the last two events in w_2 (4 and 5) do not belong to the same element

- We do not have to merge the sequences
  w_1 =<{1} {2 6} {4}> and w_2 =<{1} {2} {4 5}>
  to produce the candidate < {1} {2 6} {4 5}> because if the latter is a viable candidate, then it can be obtained by merging w_1 with
  < {1} {2 6} {5}>

GSP Example

Frequent 3-sequences

< {1} {2} {3} >
< {1} {2 5} >
< {1} {5} {3} >
< {2} {3} {4} >
< {2 5} {3} >
< {3} {4} {5} >
< {5} {3 4} >

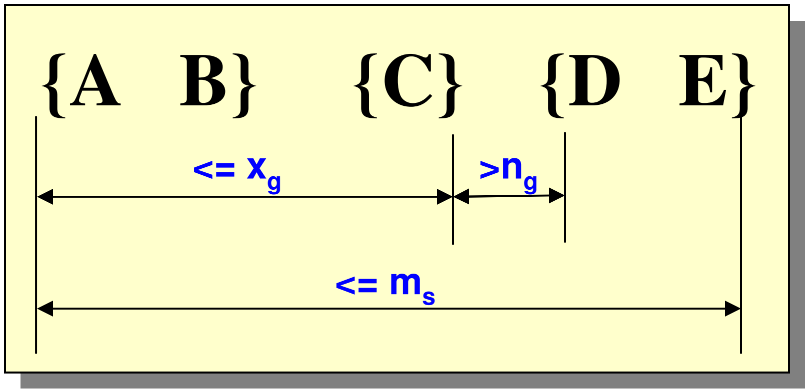
Candidate Generation

< {1} {2} {3} {4} >
< {1} {2 5} {3} >
< {1} {5} {3 4} >
< {2} {3} {4} {5} >
< {2 5} {3 4} >

Candidate Pruning

< {1} {2 5} {3} >

Timing Constraints (I)



x_g: max-gap

n_g: min-gap

m_s: maximum span

$x_g = 2$, $n_g = 0$, $m_s = 4$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {4,7} {4,5} {8} >	< {6} {5} >	Yes
< {1} {2} {3} {4} {5}>	< {1} {4} >	No
< {1} {2,3} {3,4} {4,5}>	< {2} {3} {5} >	Yes
< {1,2} {3} {2,3} {3,4} {2,4} {4,5}>	< {1,2} {5} >	No

Mining Sequential Patterns with Timing Constraints

- Approach 1:
  - Mine sequential patterns without timing constraints
  - Postprocess the discovered patterns
- Approach 2:
  - Modify GSP to directly prune candidates that violate timing constraints
  - Question:
    - Does Apriori principle still hold?

Apriori Principle for Sequence Data

Object	Timestamp	Events
A	1	1,2,4
A	2	2,3
A	3	5
B	1	1,2
B	2	2,3,4
C	1	1, 2
C	2	2,3,4
C	3	2,4,5
D	1	2
D	2	3, 4
D	3	4, 5
E	1	1, 3
E	2	2, 4, 5

Suppose:

$x_g = 1$ (max-gap)

$n_g = 0$ (min-gap)

$m_s = 5$ (maximum span)

minsup = 60%

<{2} {5}> support = 40%

but

<{2} {3} {5}> support = 60%

Problem exists because of max-gap constraint

No such problem if max-gap is infinite

Contiguous Subsequences

- s is a contiguous subsequence of

  $w = <e_1>< e_2>\ldots< e_k>$

  if any of the following conditions hold:

  1. s is obtained from w by deleting an item from either e_1 or e_k
  2. s is obtained from w by deleting an item from any element e_i that contains more than 2 items
  3. s is a contiguous subsequence of s' and s' is a contiguous subsequence of w (recursive definition)

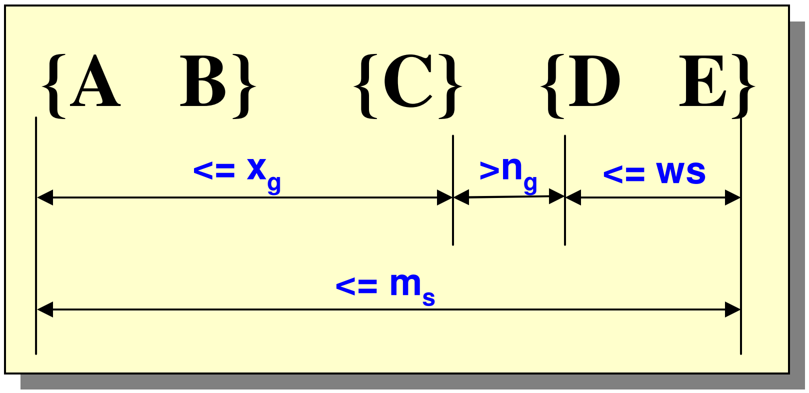
- Examples: s = < {1} {2} >
  - is a contiguous subsequence of
    < {1} {2 3}>, < {1 2} {2} {3}>, and < {3 4} {1 2} {2 3} {4} >
  - is not a contiguous subsequence of
    < {1} {3} {2}> and < {2} {1} {3} {2}>

Modified Candidate Pruning Step

- Without maxgap constraint:
  – A candidate k-sequence is pruned if at least one of its (k-1)-subsequences is infrequent

- With maxgap constraint:
  – A candidate *k*-sequence is pruned if at least one of its **contiguous** (*k-1*)-subsequences is infrequent

Timing Constraints (II)



x_g: max-gap

n_g: min-gap

ws: window size

m_s: maximum span

$x_g = 2$, $n_g = 0$, ws = 1, $m_s = 5$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {4,7} {4,6} {8} >	< {3} {5} >	No
< {1} {2} {3} {4} {5}>	< {1,2} {3} >	Yes
< {1,2} {2,3} {3,4} {4,5}>	< {1,2} {3,4} >	Yes

Modified Support Counting Step

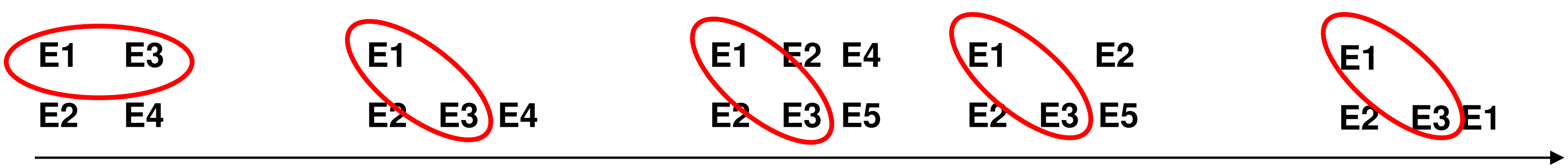
- Given a candidate pattern: <{a, c}>
  - Any data sequences that contain

    <… {a c} … >,
    <… {a} … {c}…> (where $time(\{c\}) - time(\{a\}) \leq ws$)
    <…{c} … {a} …> (where $time(\{a\}) - time(\{c\}) \leq ws$)

    will contribute to the support count of candidate pattern

Other Formulation

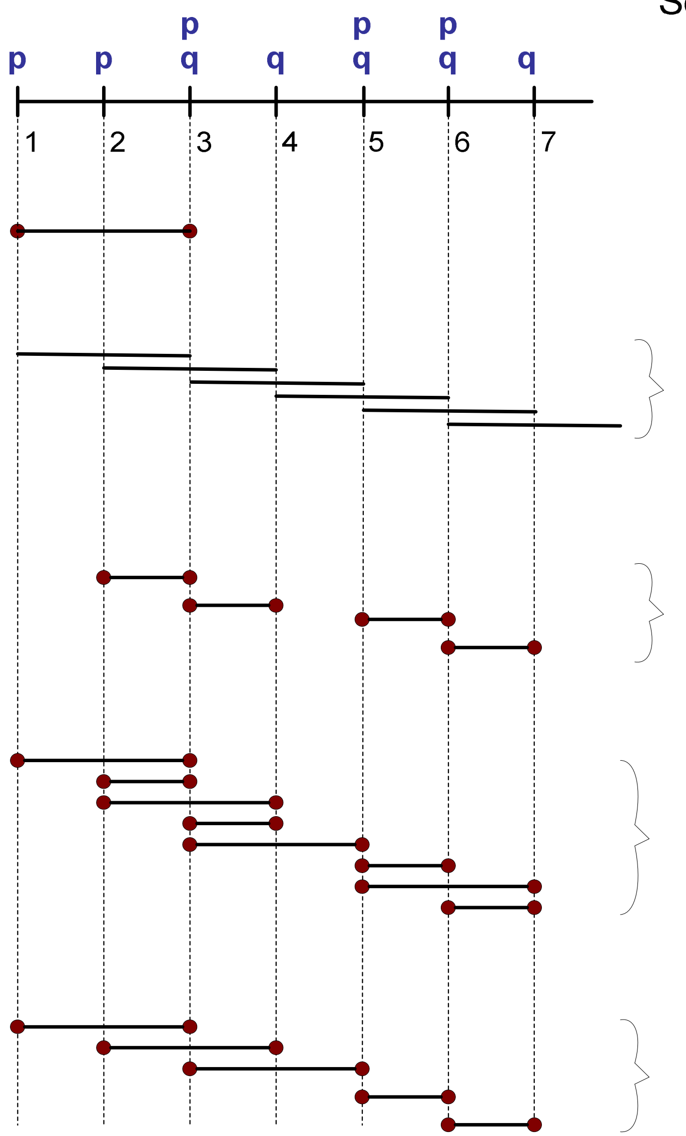
- In some domains, we may have only one very long time series
  – Example:
    - monitoring network traffic events for attacks
    - monitoring telecommunication alarm signals
- Goal is to find frequent sequences of events in the time series
  – This problem is also known as frequent episode mining



Pattern: <E1> <E3>

General Support Counting Schemes

Object's Timeline



Sequence: (p) (q)

Method	Support Count
COBJ	1
CWIN	6
CMINWIN	4
CDIST_O	8
CDIST	5

Assume:

x_g = 2 (max-gap)

n_g = 0 (min-gap)

ws = 0 (window size)

m_s = 2 (maximum span)