Pisa KDD Laboratory
http://www-kdd.isti.cnr.it

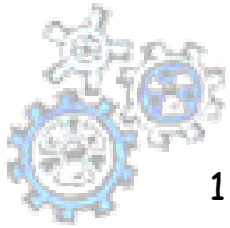
FP-growth Mining of Frequent Itemsets + Constraint-based Mining

Francesco Bonchi

e-mail: francesco.bonchi@isti.cnr.it

homepage: http://www-kdd.isti.cnr.it/~bonchi/

TDM – 11Maggio 06



Algorithm 1 Apriori

Input: $\mathcal{D}, \sigma$

Output: $Th(\mathcal{C}_{freq[\mathcal{D},\sigma]})$

1: $C_1 \leftarrow \{\{i\} \mid i \in \mathcal{I}\}$; $k \leftarrow 1$

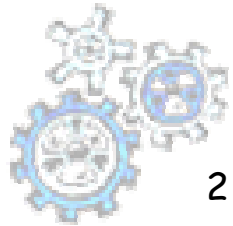
2: **while** $C_k \neq \emptyset$ **do**

3: $\quad L_k \leftarrow count(\mathcal{D}, C_k)$

4: $\quad C_{k+1} \leftarrow generate\_apriori(L_k)$

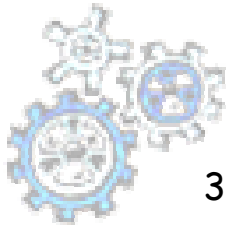
5: $\quad k++$

6: $Th(\mathcal{C}_{freq[\mathcal{D},\sigma]}) \leftarrow \bigcup_k L_k$



Is Apriori Fast Enough — Any Performance Bottlenecks?

- *The core of the Apriori algorithm:*
  - *Use frequent (k – 1)-itemsets to generate candidate frequent k-itemsets*
  - *Use database scan and pattern matching to collect counts for the candidate itemsets*
- *The bottleneck of Apriori: candidate generation*
  - *Huge candidate sets:*
    - *10^4 frequent 1-itemset will generate 10^7 candidate 2-itemsets*
    - *To discover a frequent pattern of size 100, e.g., $\{a_1, a_2, \ldots, a_{100}\}$, one needs to generate $2^{100} \approx 10^{30}$ candidates.*
  - *Multiple scans of database:*
    - *Needs (n +1) scans, n is the length of the longest pattern*



Mining Frequent Patterns Without Candidate Generation

- *Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure*
  - *highly condensed, but complete for frequent pattern mining*
  - *avoid costly database scans*
- *Develop an efficient, FP-tree-based frequent pattern mining method*
  - *A divide-and-conquer methodology: decompose mining tasks into smaller ones*
  - *Avoid candidate generation: sub-database test only!*

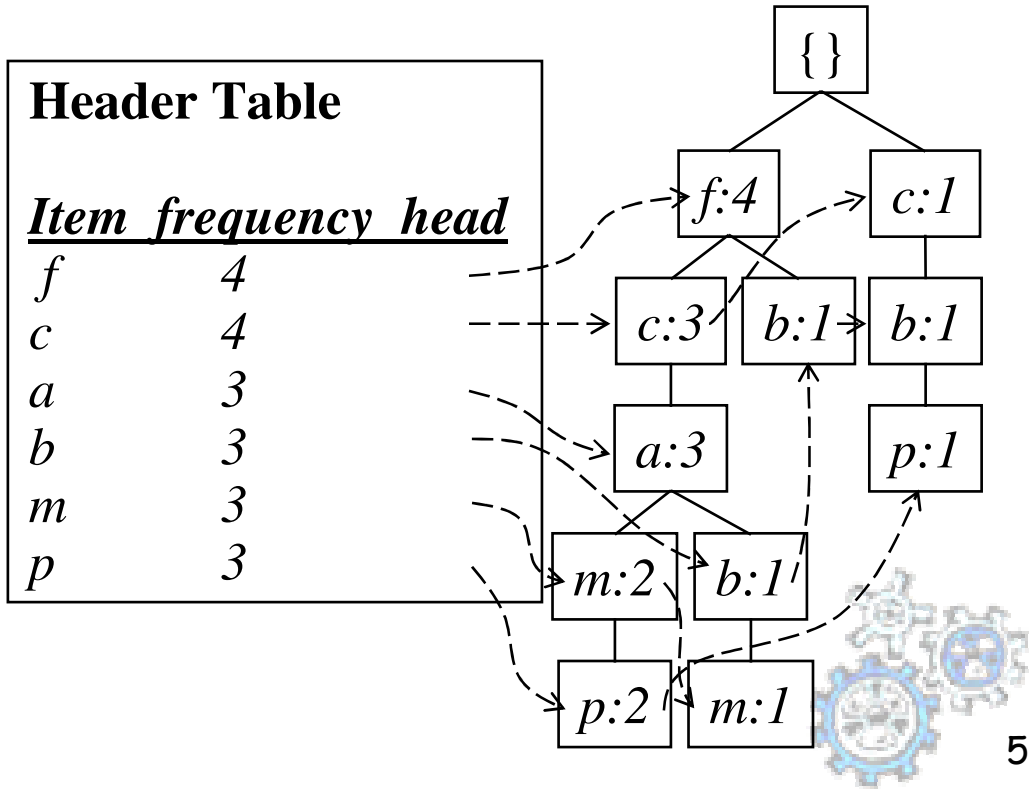
How to Construct FP-tree from a Transactional Database?

TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

min_support = 3

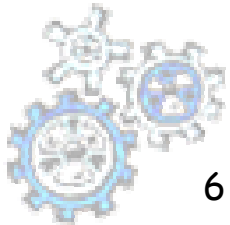
Steps:

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Order frequent items in frequency descending order
3. Scan DB again, construct FP-tree



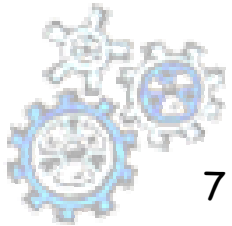
Benefits of the FP-tree Structure

- *Completeness:*
  - *never breaks a long pattern of any transaction*
  - *preserves complete information for frequent pattern mining*
- *Compactness*
  - *reduce irrelevant information—infrequent items are gone*
  - *frequency descending ordering: more frequent items are more likely to be shared*
  - *never be larger than the original database (if not count node-links and counts)*



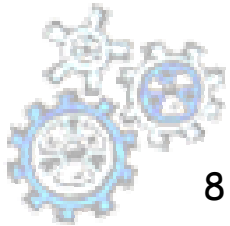
Mining Frequent Patterns Using FP-tree

- *General idea (divide-and-conquer)*
  - *Recursively grow frequent pattern path using the FP-tree*
- *Method*
  - *For each item, construct its conditional pattern-base, and then its conditional FP-tree*
  - *Repeat the process on each newly created conditional FP-tree*
  - *Until the resulting FP-tree is empty, or it contains only one path (single path will generate all the combinations of its sub-paths, each of which is a frequent pattern)*



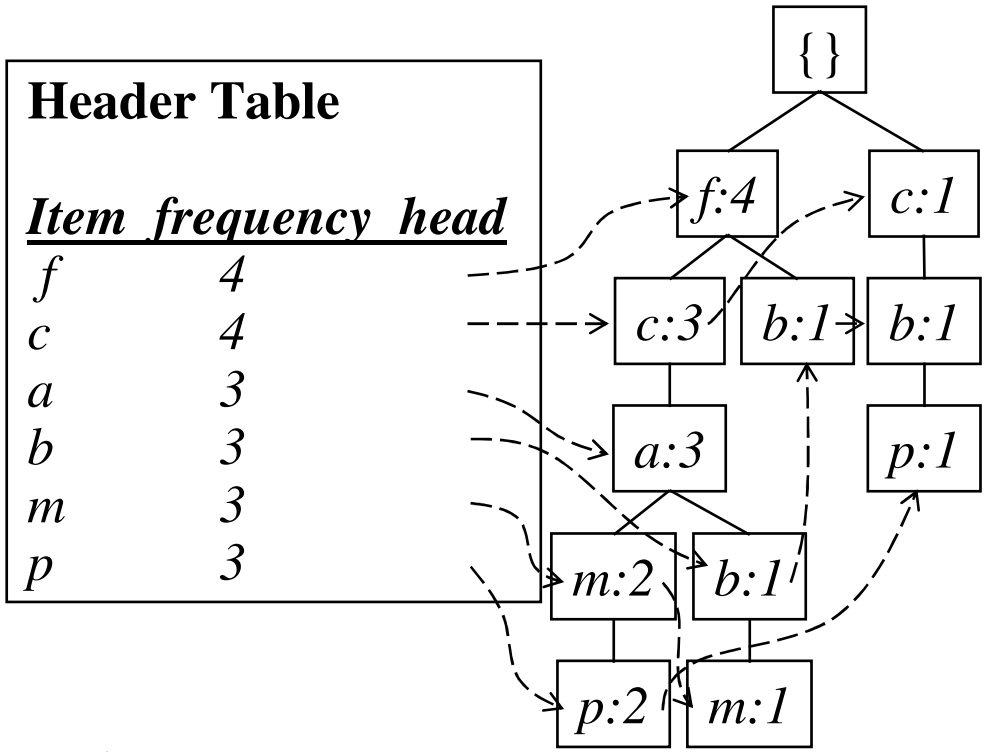
Major Steps to Mine FP-tree

1) *Construct conditional pattern base for each node in the FP-tree*
2) *Construct conditional FP-tree from each conditional pattern-base*
3) *Recursively mine conditional FP-trees and grow frequent patterns obtained so far*
4) *If the conditional FP-tree contains a single path, simply enumerate all the patterns*



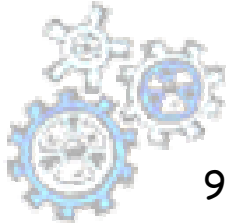
Step 1: From FP-tree to Conditional Pattern Base

- *Starting at the frequent header table in the FP-tree*
- *Traverse the FP-tree by following the link of each frequent item*
- *Accumulate all of transformed prefix paths of that item to form a conditional pattern base*



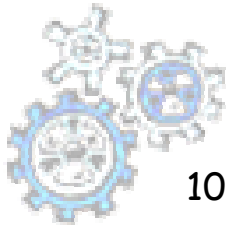
***Conditional* pattern bases**

item	*cond. pattern base*
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***



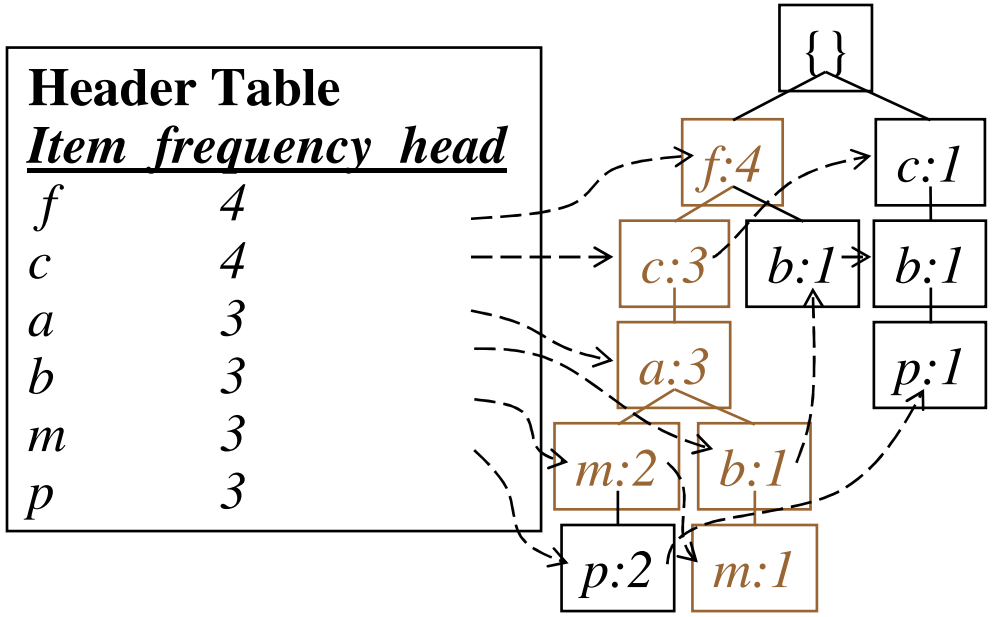
Properties of FP-tree for Conditional Pattern Base Construction

- *Node-link property*
  - *For any frequent item a_i, all the possible frequent patterns that contain a_i can be obtained by following a_i's node-links, starting from a_i's head in the FP-tree header*
- *Prefix path property*
  - *To calculate the frequent patterns for a node a_i in a path P, only the prefix sub-path of a_i in P need to be accumulated, and its frequency count should carry the same count as node a_i.*



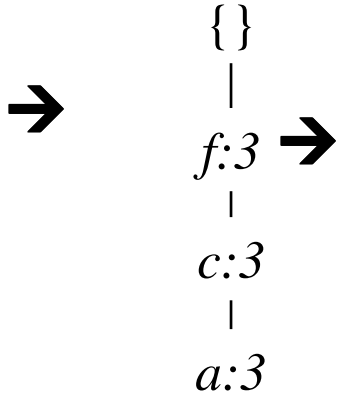
Step 2: Construct Conditional FP-tree

- *For each pattern-base*
  - *Accumulate the count for each item in the base*
  - *Construct the FP-tree for the frequent items of the pattern base*



***m-conditional* pattern base:**

fca:2, fcab:1



***m-conditional* FP-tree**

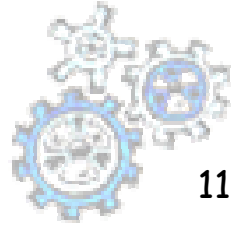
All frequent patterns concerning *m*

m,

fm, cm, am,

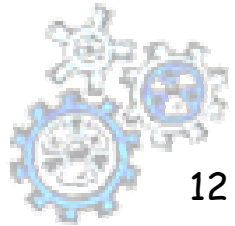
fcm, fam, cam,

fcam

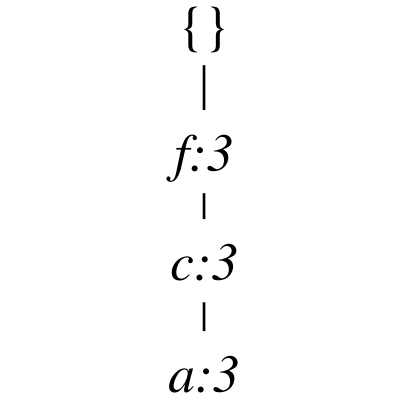


Mining Frequent Patterns by Creating Conditional Pattern Bases

Item	Conditional pattern-base	Conditional FP-tree
p	{(fcam:2), (cb:1)}	{(c:3)}|p
m	{(fca:2), (fcab:1)}	{(f:3, c:3, a:3)}|m
b	{(fca:1), (f:1), (c:1)}	Empty
a	{(fc:3)}	{(f:3, c:3)}|a
c	{(f:3)}	{(f:3)}|c
f	Empty	Empty

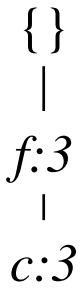


Step 3: recursively mine the conditional FP-tree



***m-conditional* FP-tree**

Cond. pattern base of "am": (fc:3)



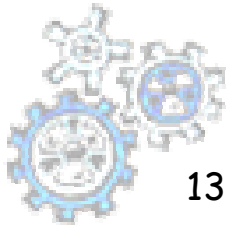
***am-conditional* FP-tree**

Cond. pattern base of "cm": (f:3)

***cm-conditional* FP-tree**

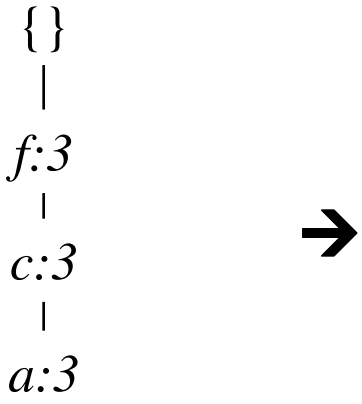
Cond. pattern base of "cam": (f:3)

***cam-conditional* FP-tree**



Single FP-tree Path Generation

- *Suppose an FP-tree T has a single path P*
- *The complete set of frequent pattern of T can be generated by enumeration of all the combinations of the sub-paths of* P



→

All frequent patterns concerning *m*

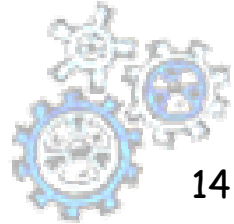
m,

fm, cm, am,

fcm, fam, cam,

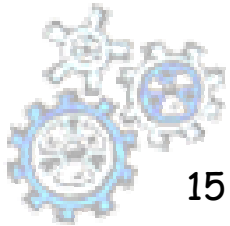
fcam

***m-conditional* FP-tree**

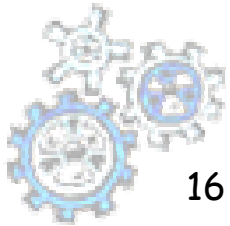


Principles of Frequent Pattern Growth

- *Pattern growth property*
  - *Let α be a frequent itemset in DB, B be α's conditional pattern base, and β be an itemset in B. Then $\alpha \cup \beta$ is a frequent itemset in DB iff β is frequent in B.*
- *"abcdef " is a frequent pattern, if and only if*
  - *"abcde " is a frequent pattern, and*
  - *"f " is frequent in the set of transactions containing "abcde "*



Adding Constraints to Frequent Itemset Mining

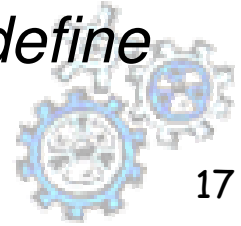


Why Constraints?

- *Frequent pattern mining usually produces too many solution patterns. This situation is harmful for two reasons:*
  1. *Performance: mining is usually inefficient or, often, simply unfeasible*
  2. *Identification of fragments of interesting knowledge blurred within a huge quantity of small, mostly useless patterns, is an hard task.*

- *Constraints are the solution to both these problems:*
  1. *they can be pushed in the frequent pattern computation exploiting them in pruning the search space, thus reducing time and resources requirements;*
  2. *they provide to the user guidance over the mining process and a way of focussing on the interesting knowledge.*

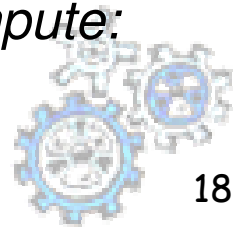
- *With constraints we obtain less patterns which are more interesting. Indeed constraints are the way we use to define what is "interesting".*



Problem Definition

- $I=\{x_1, ..., x_n\}$ set of distinct literals (called items)
- $X \subseteq I$, $X \neq \emptyset$, $|X| = k$, X is called k-itemset
- A transaction is a couple $\langle tID, X \rangle$ where X is an itemset
- A transaction database TDB is a set of transactions
- An itemset X is contained in a transaction $\langle tID, Y \rangle$ if $X \subseteq Y$
- Given a TDB the subset of transactions of TDB in which X is contained is named TDB[X].
- The support of an itemset X , written $supp_{TDB}(X)$ is the cardinality of TDB[X].
- Given a user-defined min_sup an itemset X is frequent in TDB if its support is no less than min_sup.

- We indicate the frequency constraint with C_{freq}
- Given a constraint C , let $Th(C) = \{X| C(X)\}$ denote the set of all itemsets X that satisfy C.
- The frequent itemsets mining problem requires to compute $Th(C_{freq})$
- The constrained frequent itemsets mining problem requires to compute: $Th(C_{freq}) \cap Th(C)$.

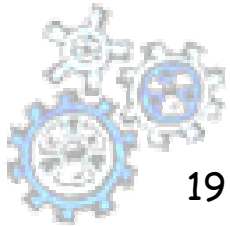


Constrained Frequent Pattern Mining: A Mining Query Optimization Problem

- *Given a frequent pattern mining query with a set of constraints C, the algorithm should be*
  - *sound: it only finds frequent sets that satisfy the given constraints C*
  - *complete: all frequent sets satisfying the given constraints C are found*

- *A naïve solution (generate&test)*
  - *Generate all frequent sets, and then test them for constraint satisfaction*

- *More efficient approaches:*
  - *Analyze the properties of constraints comprehensively*
  - *Push them as deeply as possible inside the frequent pattern computation.*

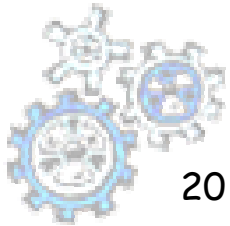


Anti-Monotonicity and Succinctness

- *A first work defining classes of constraints which exhibit nice properties [Ng et al. SIGMOD'98].*

- *Anti-monotonicity and Succinctness are introduced*
- *CAP, an Apriori-like algorithm which exploits anti-monotonicity and succinctness of constraints*

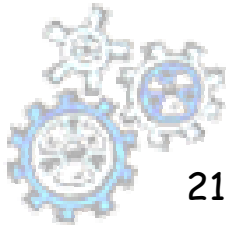
- *4 classes of constraints + associated computational strategy*
  1. *Constraints that are anti-monotone but not succinct*
  2. *Constraints that are both anti-monotone and succinct*
  3. *Constraints that are succinct but not anti-monotone*
  4. *Constraints that are neither*



Anti-Monotonicity in Constraint-Based Mining

- *Anti-monotonicity:*
  - *When an intemset S satisfies the constraint, so does any of its subset*
- *Frequency is an anti-monotone constraint.*
- *"Apriori property": if an itemset X does not satisfy C_{freq} then no superset of X can satisfy C_{freq}.*

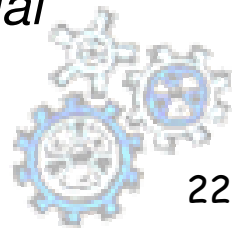
  - *sum(S.Price) ≤ v is anti-monotone*
  - *Very easy to push in the frequent itemset computation*



Succinctness in Constraint-Based Mining

- *Succinctness:*
  - *Given A_1, the set of items satisfying a succinct constraint C, then any set S satisfying C is based on A_1 , i.e., S contains a subset belonging to A_1*
  - *Idea: whether an itemset S satisfies constraint C can be determined based on the singleton items which are in S*
  - *min(S.Price) $\leq$ v is succinct*
  - *sum(S.Price) $\geq$ v is not succinct*

- *Optimization: If C is succinct, C is pre-counting pushable (can be satisfied at candidate-generation time).*
  - *Substitute the usual "Generate-Apriori" procedure with a special candidate generation procedure.*



CAP – computational strategies

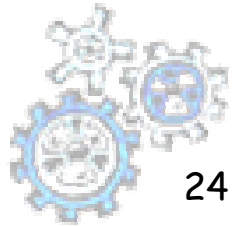
- *4 classes of constraints + associated computational strategy*
  1. *Constraints that are anti-monotone but not succinct*
     - *Check them in conjunction with frequency as a unique anti-monotone constraint*
  2. *Constraints that are both anti-monotone and succinct*
     - *Can be pushed at preprocessing time:* ***min(S.Price)*** $\geq$ ***v*** *just start the computation with candidates all singleton items having price* $\geq v$
  3. *Constraints that are succinct but not anti-monotone*
     - *Use the special candidate-generation function*
  4. *Constraints that are neither*
     - *Induce a weaker constraint which is either anti-monotone and/or succinct*

Converting "Tough" Constraints

- *Introduced in [Pei and Han KDD'00, ICDE'01]*

- *Let R be an order of items*
- *Convertible anti-monotone*
  - *If an itemset S violates a constraint C, so does every itemset having S as a prefix w.r.t. R*
  - *Ex. avg(S) ≤ v w.r.t. item value descending order*

- *Convertible monotone*
  - *If an itemset S satisfies constraint C, so does every itemset having S as a prefix w.r.t. R*
  - *Ex. avg(S) ≥ v w.r.t. item value descending order*

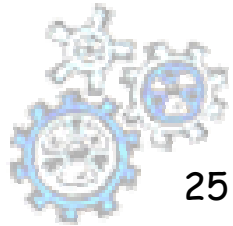


Converting "Tough" Constraints

- *Examine C: avg(S.profit) ≥ 25*
  - *Order items in value-descending order*
    - *<a, f, g, d, b, h, c, e>*
  - *If an itemset afb violates C*
    - *So does afbh, afb**
    - *It becomes anti-monotone!*

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

- *Authors state that convertible constraints can not be pushed in Apriori but they can be handled by FP-Growth approach.*
- *Two FP-Growth-based algorithms:*
  - *FIC^A and FIC^M*

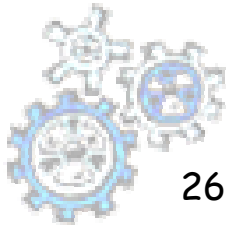


Strongly Convertible Constraints

- *avg(X) ≥ 25 is convertible anti-monotone w.r.t. item value descending order R: <a, f, g, d, b, h, c, e>*
  - *If an itemset af violates a constraint C, so does every itemset with af as prefix, such as afd*
- *avg(X) ≥ 25 is convertible monotone w.r.t. item value ascending order R^{-1}: <e, c, h, b, d, g, f, a>*
  - *If an itemset d satisfies a constraint C, so does itemsets df and dfa, which having d as a prefix*

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

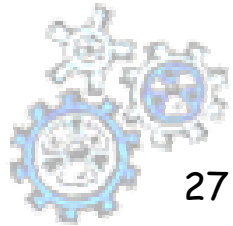
- *Thus, avg(X) ≥ 25 is strongly convertible*



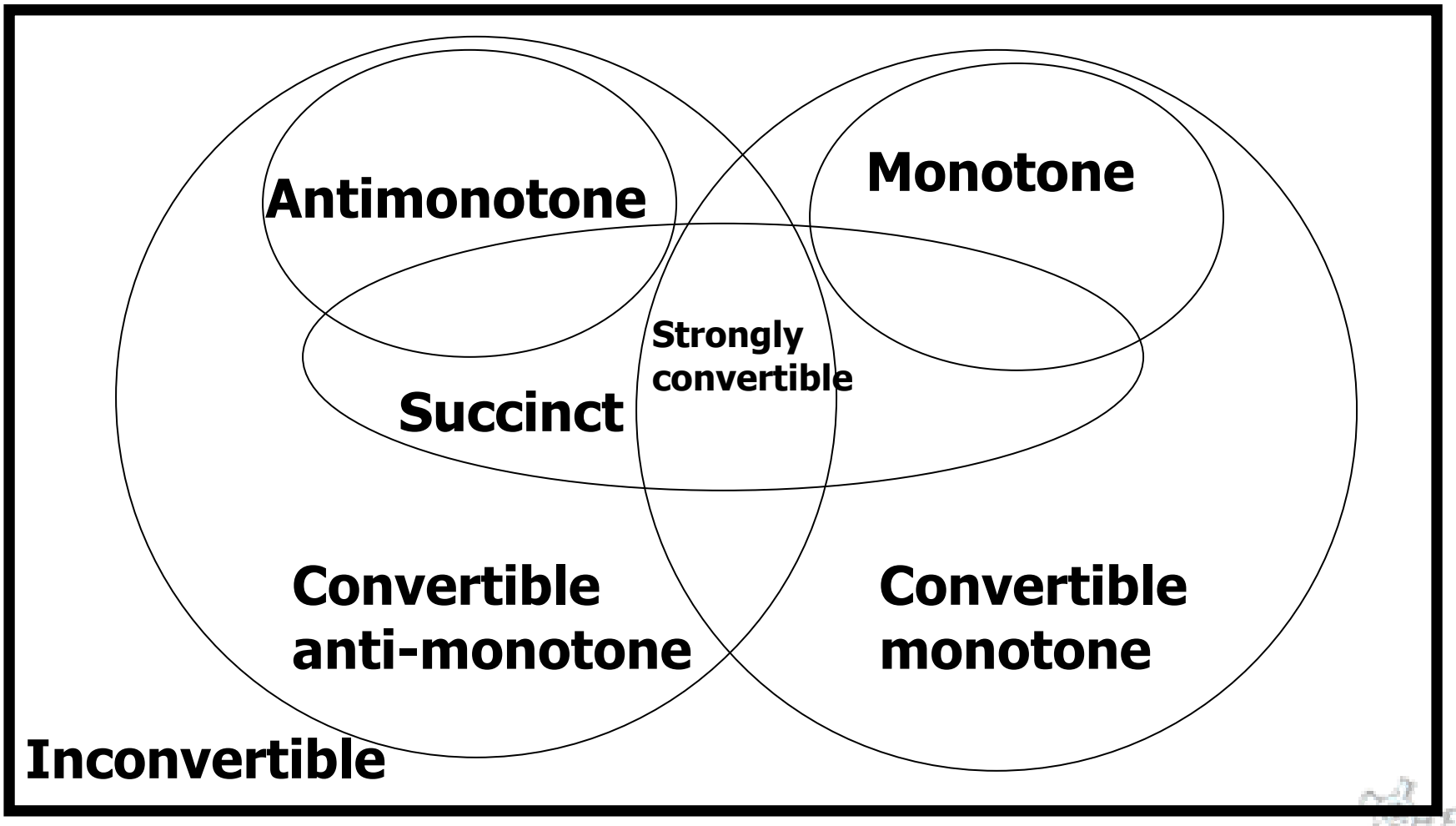
Monotonicity in Constraint-Based Mining

- *Monotonicity*
  - *When an itemset S satisfies the constraint, so does any of its superset*
  - *sum(S.Price) ≥ v is monotone*
  - *min(S.Price) ≤ v is monotone*

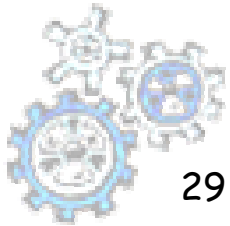
- *They behave exactly the opposite of frequency …*
- *How to push them in the Apriori computation?*



Classification of Constraints



ExAnte
ExAMiner



Our Problem …

… to compute itemsets which satisfy a conjunction of anti-monotone and monotone constraints.

$$Th(\mathcal{C}_{freq}) \cap Th(\mathcal{C}_M)$$

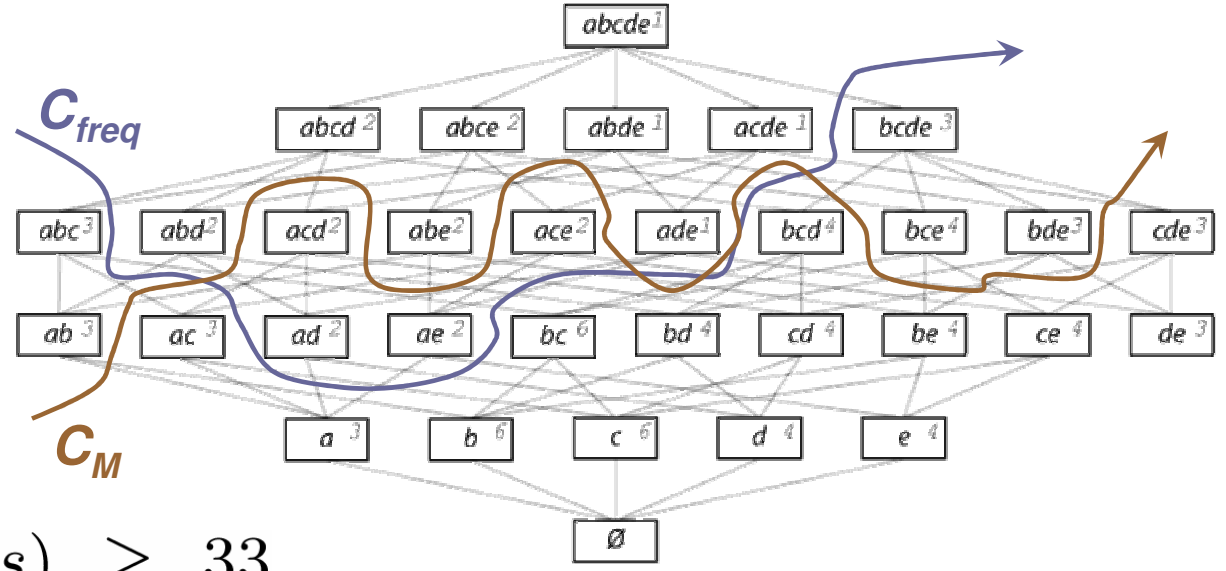
Why Monotone Constraints?

1. *They're the most useful in order to discover local high-value patterns (for instance very expansive or very large itemsets which can be found only with a very small min-sup)*
2. *We know how to exploit the other kinds of constraints (antimonotone, succinct) since '98 [Ng et al. SIGMOD'98], while for monotone constraints the situation is more complex …*

Characterizing the search space

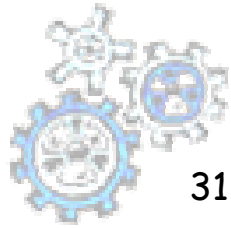
$\mathcal{D}$
a,b,c,d,e
b,c
b,c,d,e
a,b,c,d
a,b,c,e
b,c,d,e

item	price
a	15
b	18
c	2
d	4
e	14



$$\mathcal{C}_M \equiv sum(X.prices) \geq 33$$

$$FTh_{\mathcal{D}}(\mathcal{C}_{freq[\mathcal{D},3]} \wedge \mathcal{C}_M) = \{\langle ab,3\rangle, \langle abc,3\rangle, \langle bcde,3\rangle, \langle bce,4\rangle, \langle bde,3\rangle\}$$



AM Vs. M

- ***State of art before ExAnte****: when dealing with a conjunction of AM and M constraints we face a tradeoff between AM and M pruning.*
- ***Tradeoff****: pushing M constraints into the computation can help pruning the search space, but at the same time can lead to a reduction of AM pruning opportunities.*
- ***Our observation****: this is true only if we focus exclusively on the search space of itemsets. Reasoning on both the search space and the input TDB together we can find the real sinergy of AM and M pruning.*
- ***The real sinergy****: do not exploit M constraints directly to prune the search space, but use them to prune the data, which in turn induces a much stronger pruning of the search space.*
- *The real sinergy of AM and M pruning lies in* ***Data Reduction*** *…*

ExAnte μ-reduction

- *Definition [μ-reduction]:*

  Given a transaction database TDB and a monotone constraint C_M, we define the μ-reduction of TDB as the dataset resulting from pruning the transactions that do not satisfy C_M.

$$\mu[TDB]_{C_M} = Th(\mathcal{C}_M) \cap TDB$$

- *Example: $C_M \equiv$ sum(X.price) $\geq$ 55*

item	price
a	5
b	8
c	14
d	30
e	20
f	15
g	6
h	12

tID	Itemset	Total price
1	b,c,d,g	58
2	a,b,d,e	63
3	b,c,d,g,h	70
~~4~~	~~a,e,g~~	~~31~~
5	c,d,f,g	65
6	a,b,c,d,e	77
7	a,b,d,f,g,h	76
~~8~~	~~b,c,d~~	~~52~~
~~9~~	~~b,e,f,g~~	~~49~~

ExAnte α-reduction

- *Definition [α-reduction]:*

  Given a transaction database TDB, a transaction <tID,X> and a frequency constraint $C_{freq}[TDB]$, we define the α-reduction <tID,X> as the subset of items in X that satisfy $C_{freq}[TDB]$.

$$\alpha[\langle tID, X\rangle]_{\mathcal{C}_{freq}[TDB]} = F_1 \cap X$$

  Where:

$$F_1 = \{I \in Items | \{I\} \in Th(C_{freq}[TDB])\}$$

  We define the α-reduction of TDB as the dataset resulting from the α-reduction of all transactions in TDB.

- *Example:*

$$Items = \{a, b, c, d, e, f, g\} \quad X = \{a, c, d, f, g\}$$
$$Th(C_{freq}) = \{\{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$$
$$F_1 = \{a, b, c\}$$
$$\alpha[\langle tID, X\rangle]_{\mathcal{C}_{freq}} = F_1 \cap X = \{a, c\}$$

ExAnte Properties

THEOREM 1 (μ-REDUCTION CORRECTNESS). *Given a transaction database TDB, a monotone constraint $\mathcal{C}_M$, and a frequency constraint $\mathcal{C}_{freq}$, we have that:*

$$\forall X \in Th(\mathcal{C}_{freq}[TDB]) \cap Th(\mathcal{C}_M) :$$
$$supp_{TDB}(X) = supp_{\mu[TDB]_{C_M}}(X).$$

PROOF. Since $X \in Th(\mathcal{C}_M)$, all transactions containing X will also satisfy $\mathcal{C}_M$ for the monotonicity property. In other words: $TDB[X] \subseteq \mu[TDB]_{\mathcal{C}_M}$. This implies that:

$$supp_{TDB}(X) = supp_{\mu[TDB]_{C_M}}(X).$$

□

ExAnte Properties

THEOREM 2 (α-REDUCTION CORRECTNESS). *Given a transaction database TDB, a monotone constraint $\mathcal{C}_M$, and a frequency constraint $\mathcal{C}_{freq}$, we have that:*

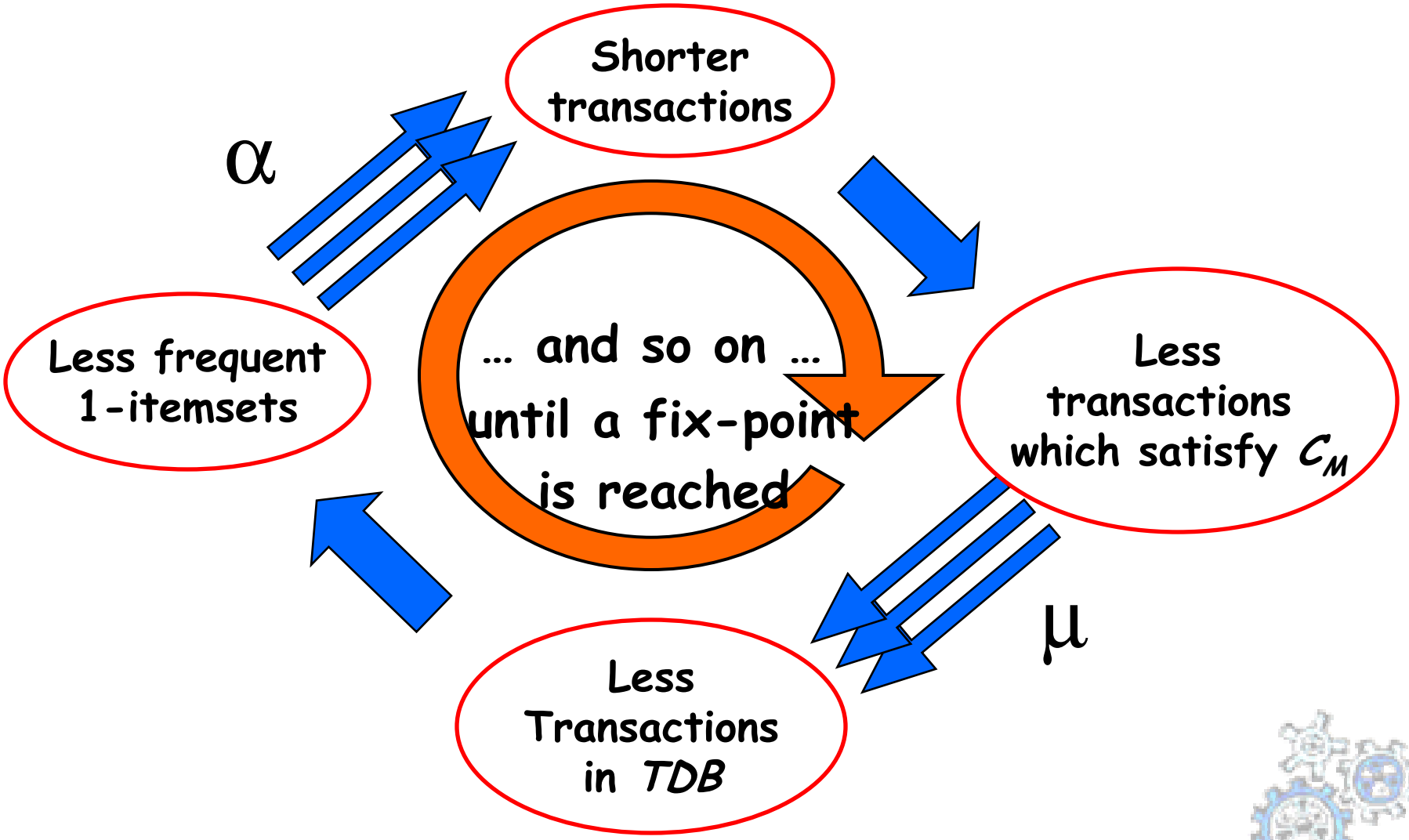
$$\forall X \in Th(\mathcal{C}_{freq}[TDB]) \cap Th(\mathcal{C}_M) :$$
$$supp_{TDB}(X) = supp_{\alpha[TDB]_{C_{freq}}}(X).$$

PROOF. Since $X \in Th(\mathcal{C}_{freq})$, all subsets of X will be frequent (by the anti-monotonicity of frequency). Therefore no subset of X will be α-pruned (in particular, no 1-itemsets in X). This implies that:

$$supp_{TDB}(X) = supp_{\alpha[TDB]_{C_{freq}}}(X).$$

□

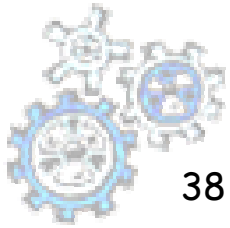
A Fix-Point Computation



ExAnte Algorithm

Procedure: **ExAnte**$(TDB, \mathcal{C}_M, min\_supp)$

1. $I := \emptyset$;
2. **forall** tuples t in TDB **do**
3. **if** $\mathcal{C}_M(t)$ **then forall** items i in t **do**
4. $i.count{+}{+}$; **if** $i.count \geq min\_supp$ **then** $I := I \cup i$;
5. $old\_number\_interesting\_items := |Items|$;
6. **while** $|I| < old\_number\_interesting\_items$ **do**
7. $TDB := \alpha[TDB]_{C_{freq}}$;
8. $TDB := \mu[TDB]_{C_M}$;
9. $old\_number\_interesting\_items := |I|$;
10. $I := \emptyset$;
11. **forall** tuples t in TDB **do**
12. **forall** items i in t **do**
13. $i.count + +$;
14. **if** $i.count \geq min\_supp$ **then** $I := I \cup i$;
15. **end while**



ExAnte Preprocessing Example

item	price
a	5
b	8
c	14
d	30
e	20
f	15
g	6
h	12

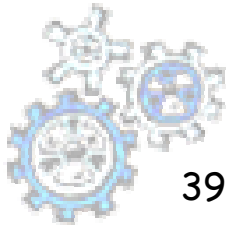
tID	Itemset	Total price
1	b,c,d,~~g~~	~~58~~ 52
2	~~a~~,b,d,~~e~~	~~63~~ 38
3	b,c,d,~~g~~,~~h~~	~~70~~ 58
4	a,e,g	31
5	c,d,~~f~~,~~g~~	~~65~~ 50
6	~~a~~,b,c,d,~~e~~	~~77~~ 52
7	~~a~~,b,d,~~f~~,g,~~h~~	~~76~~ 44
8	b,c,d	52
9	b,~~c~~,~~f~~,g	~~49~~ 14

52

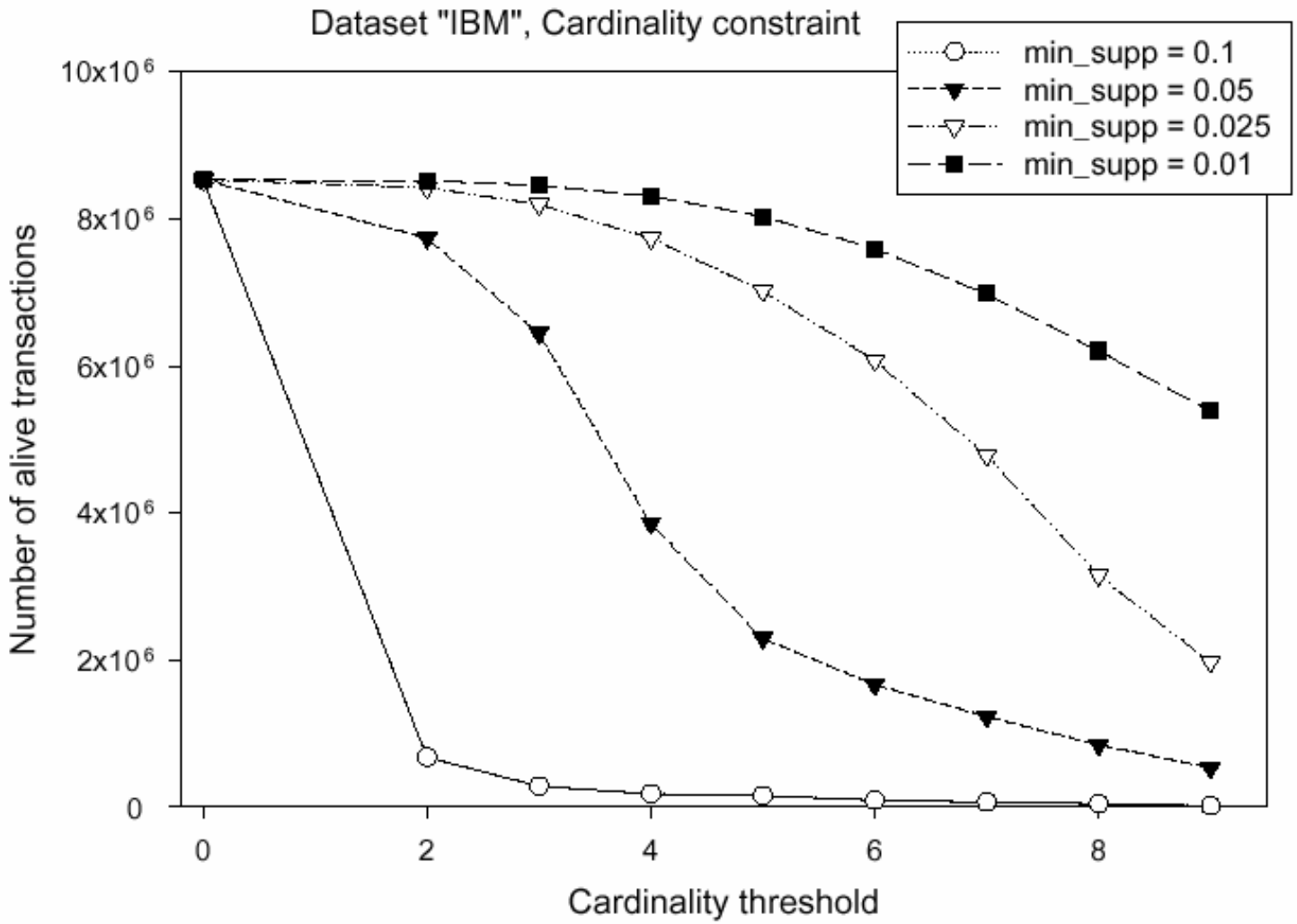
44

- *Min_sup = 4*
- $C_M \equiv$ *sum(X.price)* ≥ 45

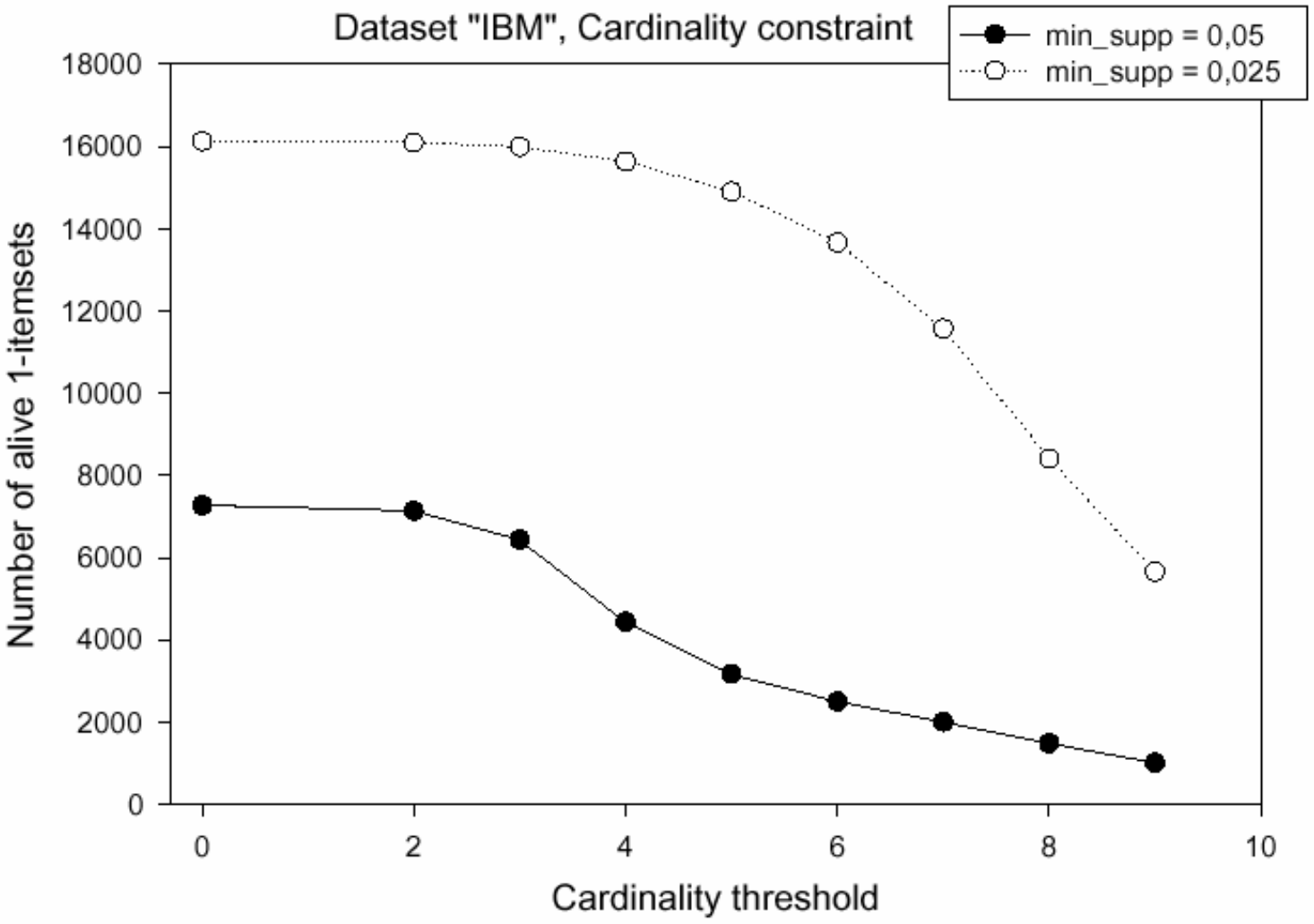
Item	Support			
~~a~~	~~4~~	3	†	†
b	7	~~7~~	4	4
c	5	5	5	4
d	7	~~7~~	5	4
~~e~~	~~4~~	3	†	†
~~f~~	3	3	†	†
~~g~~	~~6~~	~~5~~	3	†
~~h~~	2	2	†	†



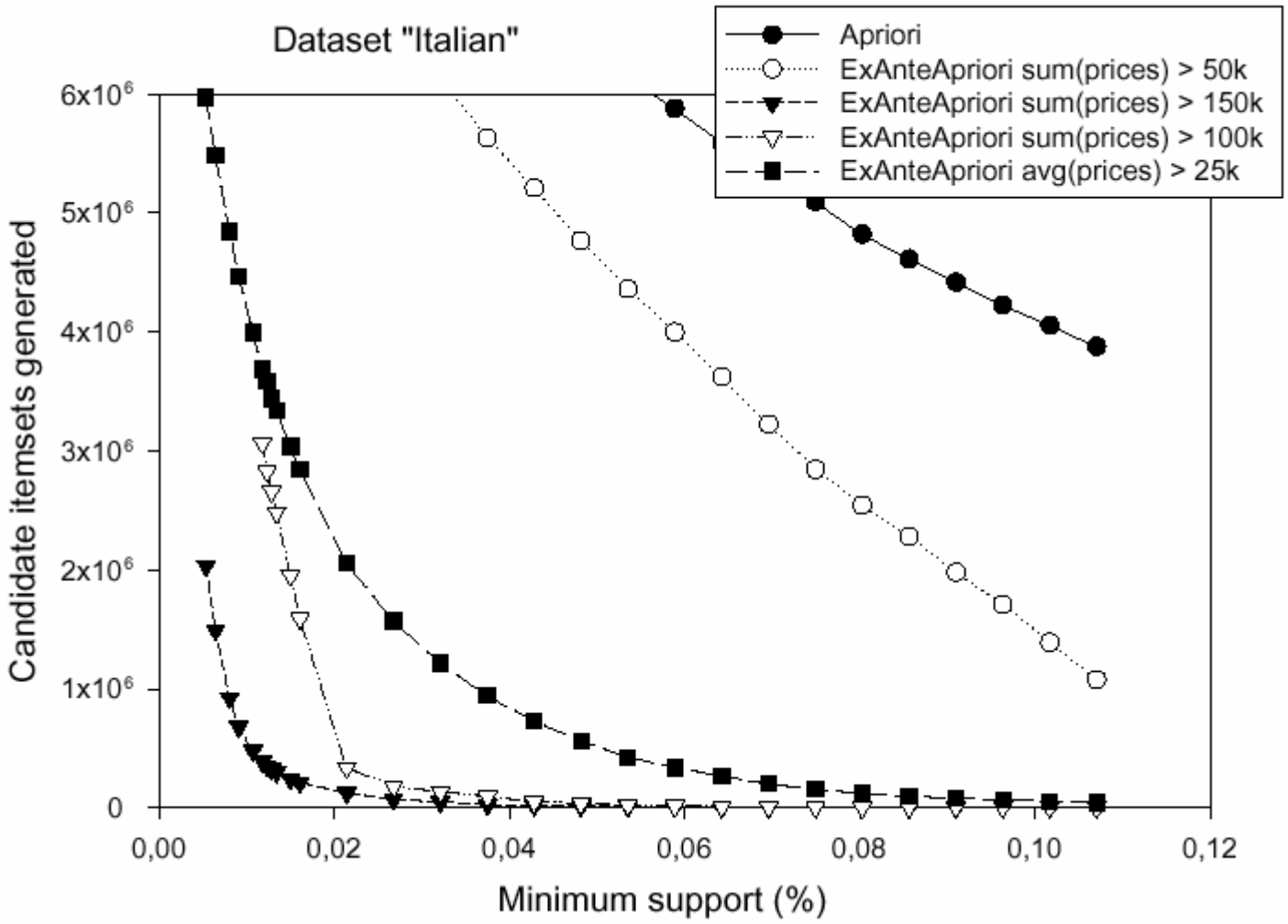
Experimental Results



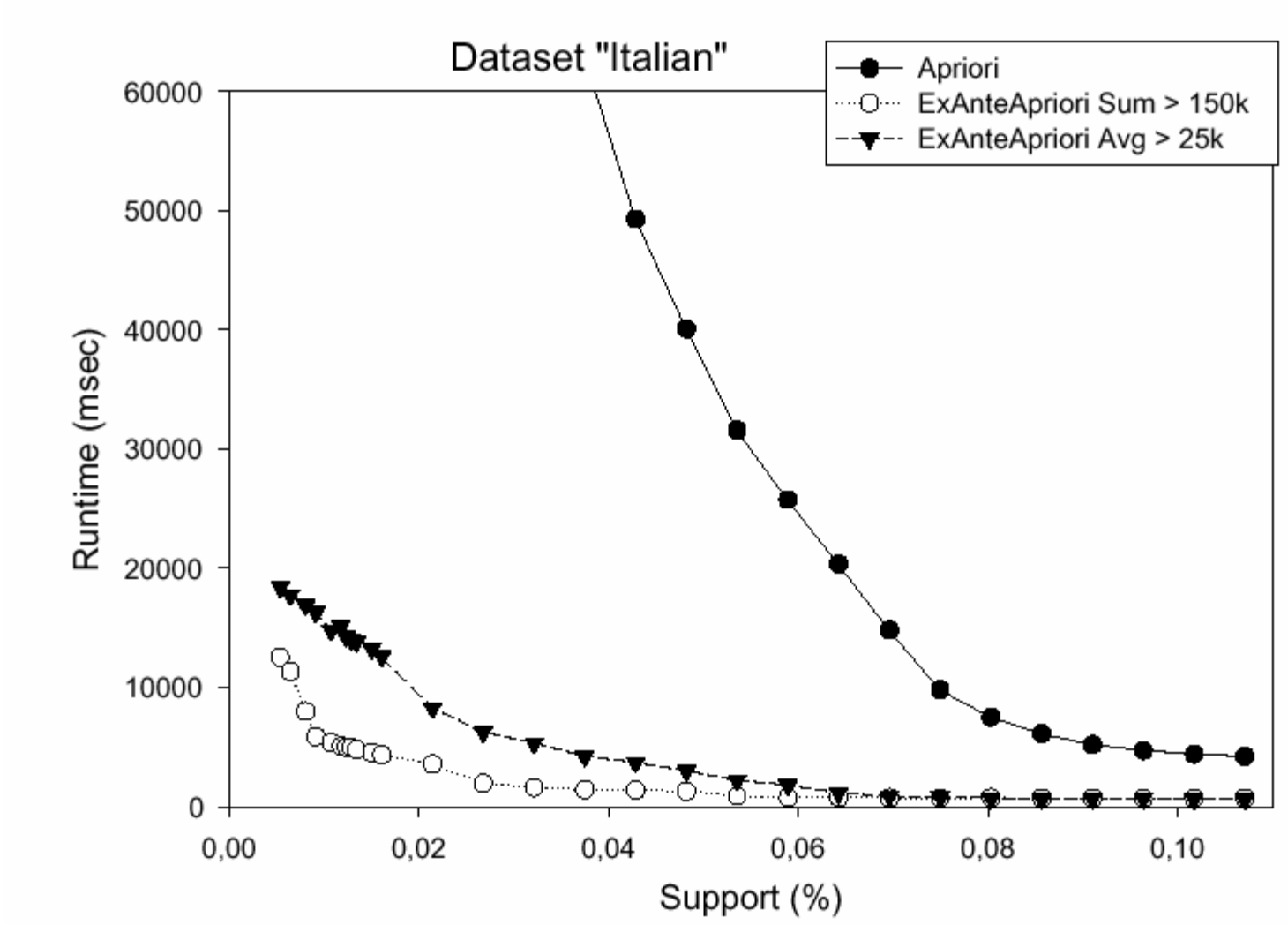
Experimental Results



Experimental Results



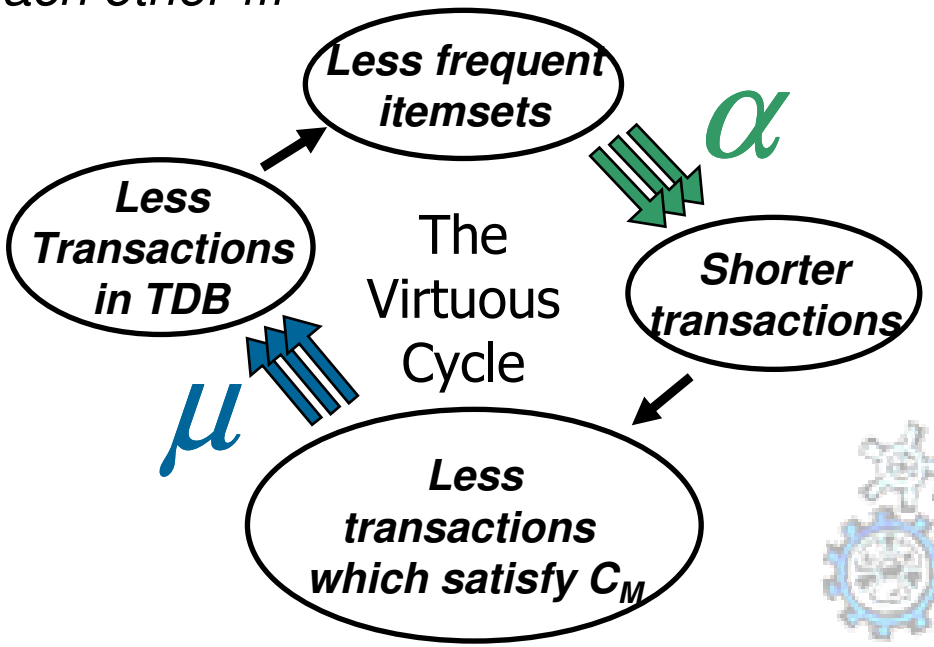
Experimental Results



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.*



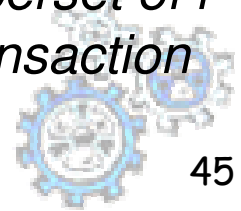
ExAMiner: key idea and basic data reductions

- *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:*

[Gα_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.*

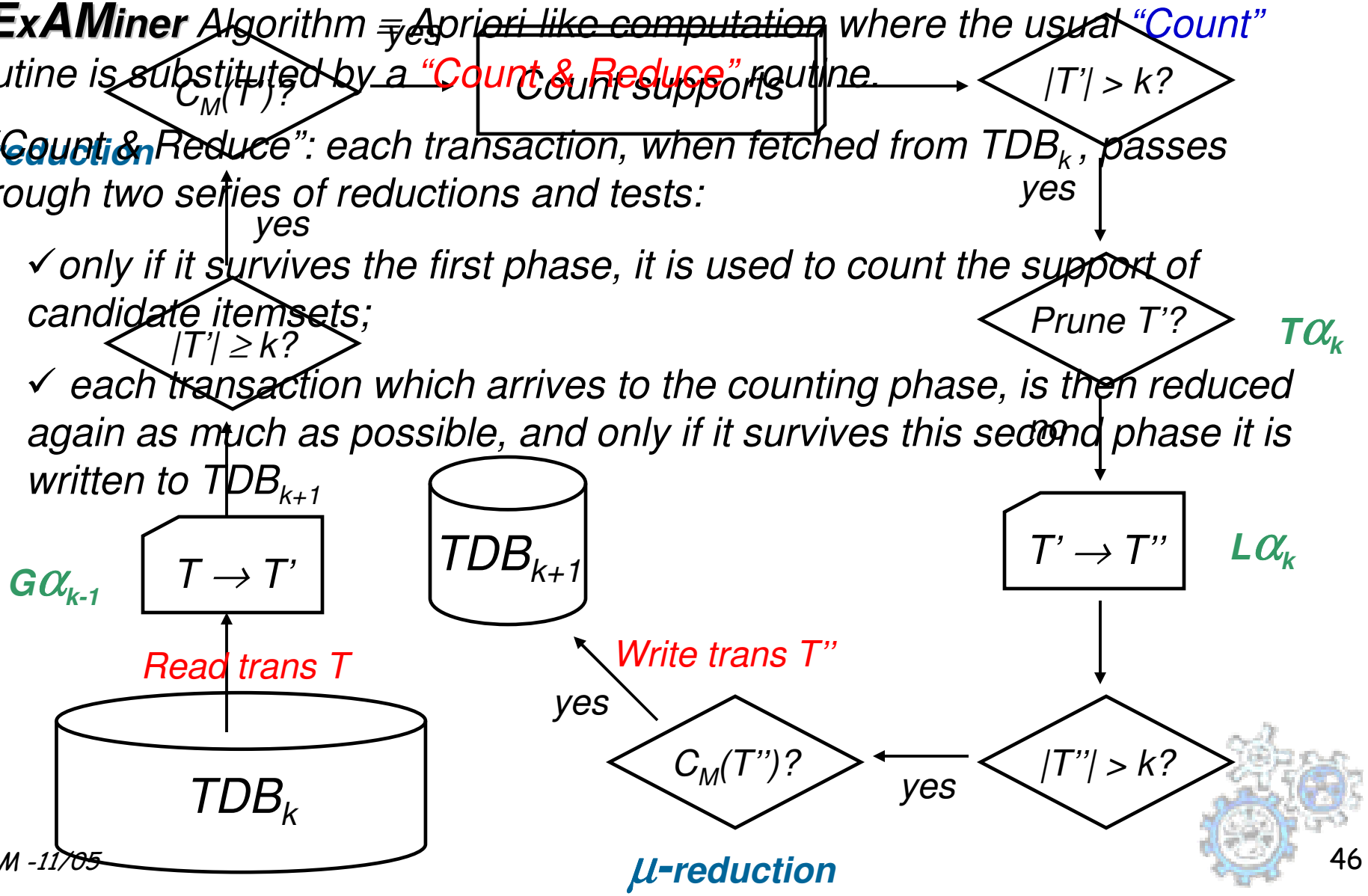
[Tα_k] Antimonotone Data Reduction of Transactions*: a transaction which is not superset of at least k+1 candidate k-itemsets can be pruned away from TDB.*

[Lα_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

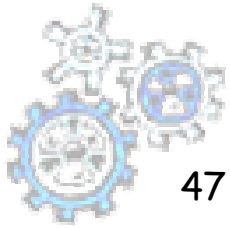
- *When dealing with the Cardinality Monotone Constraint:* $C_M \equiv$ **card(S) ≥ n** *we can exploit stronger pruning at very low computational price.*
- *At the generic level k:*

  - 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.*

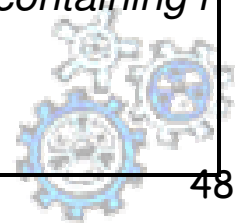
- *Example:* $C_M \equiv$ **sum(S.price)** $\geq m$

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.*



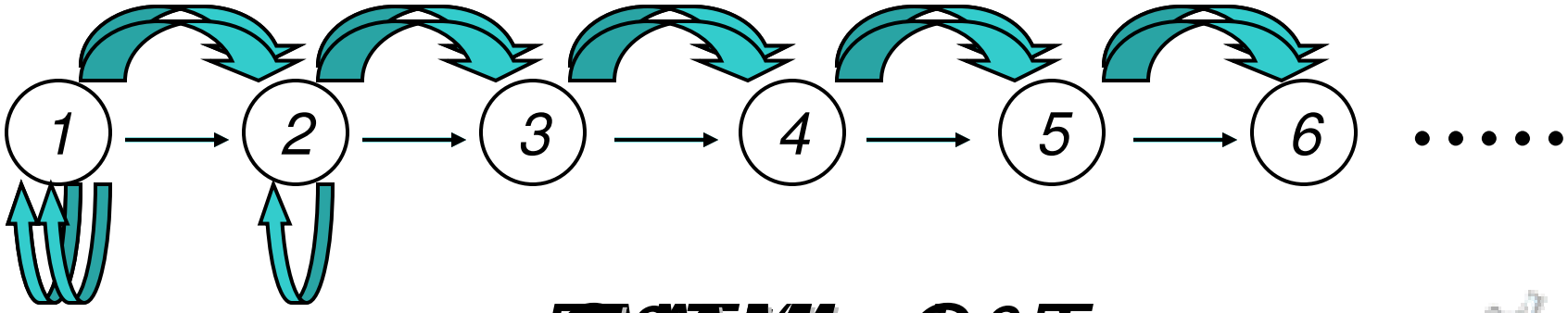
ExAMiner implementations

Count: ⟶

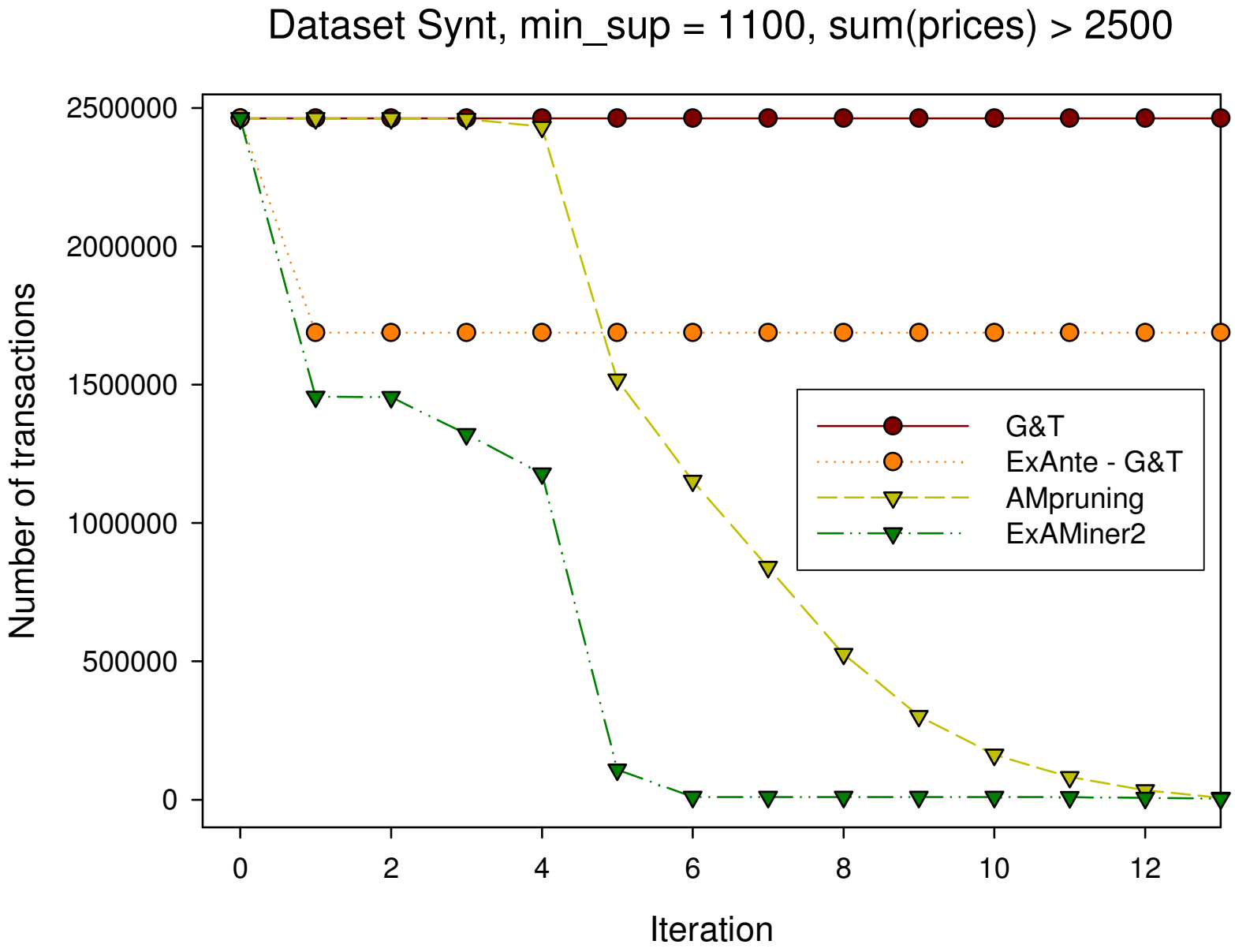
Count and AM reduce: ⟶

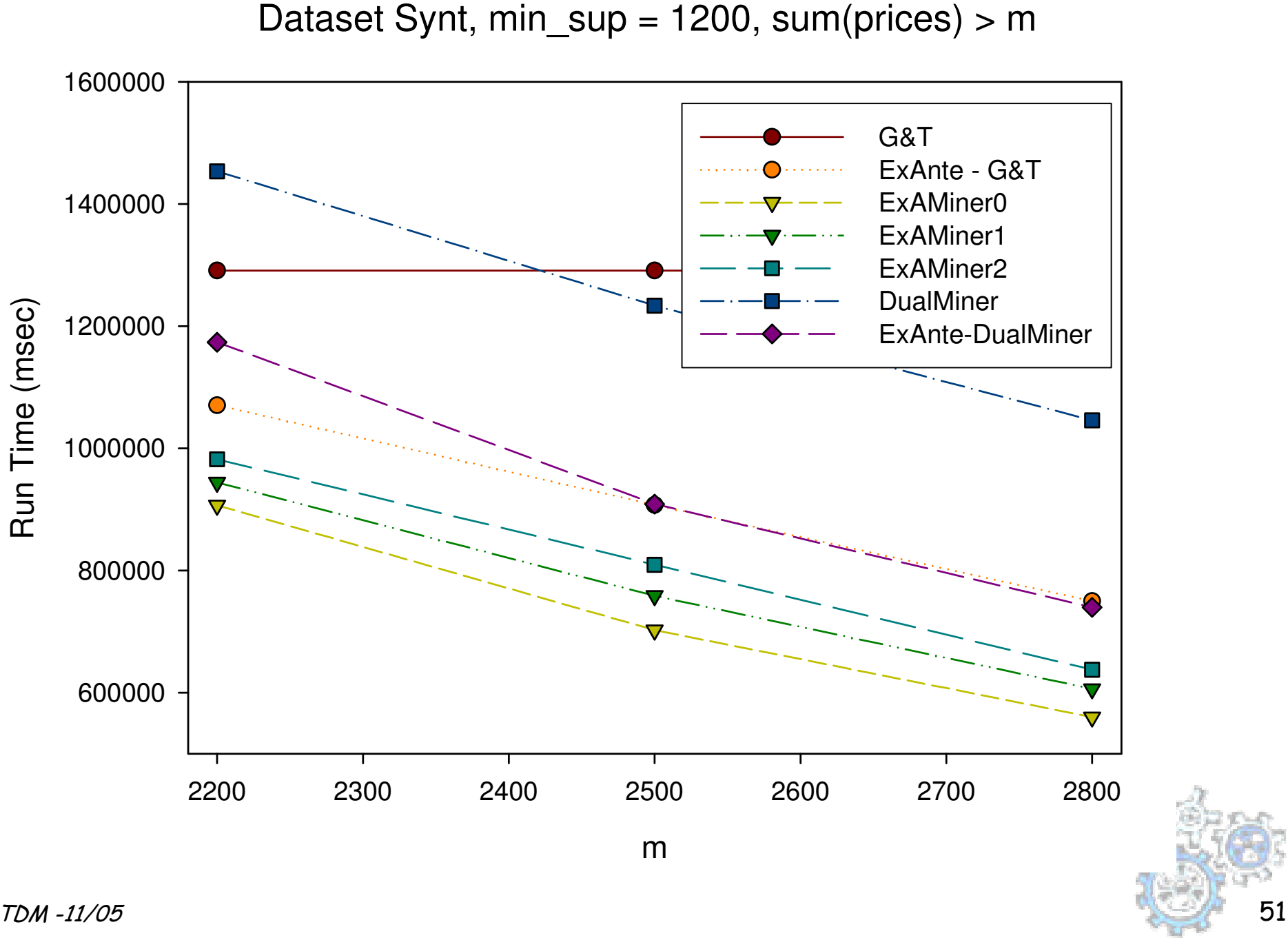
Count, AM and M reduce:

Count, AM and M reduce (fixpoint):



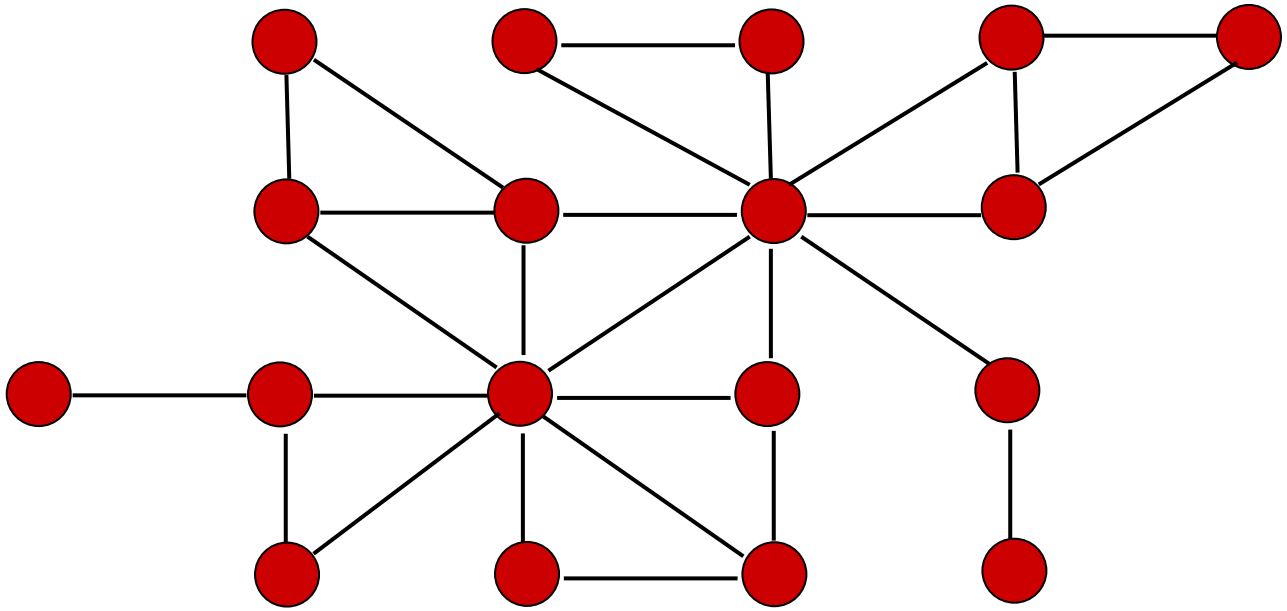
ExAMiner **ExAMiner$_2$**


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

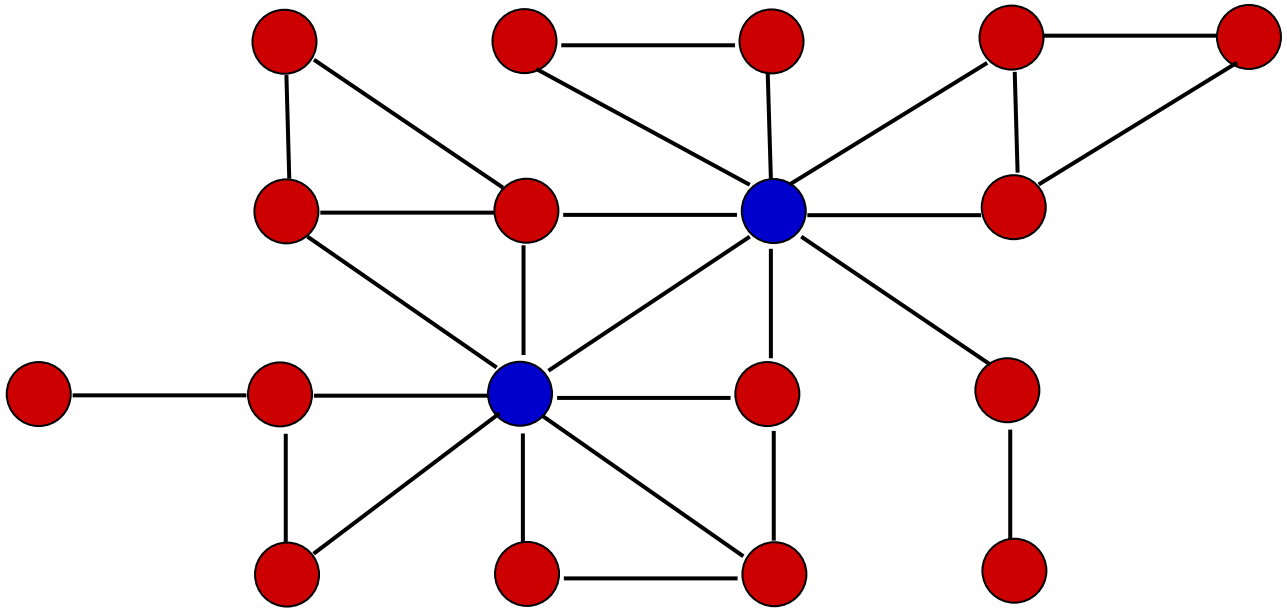
A very general idea

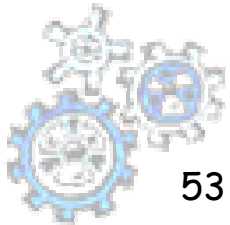
- *Mine frequent connected subgraphs*
- *Containing at least 4 nodes*



A very general idea

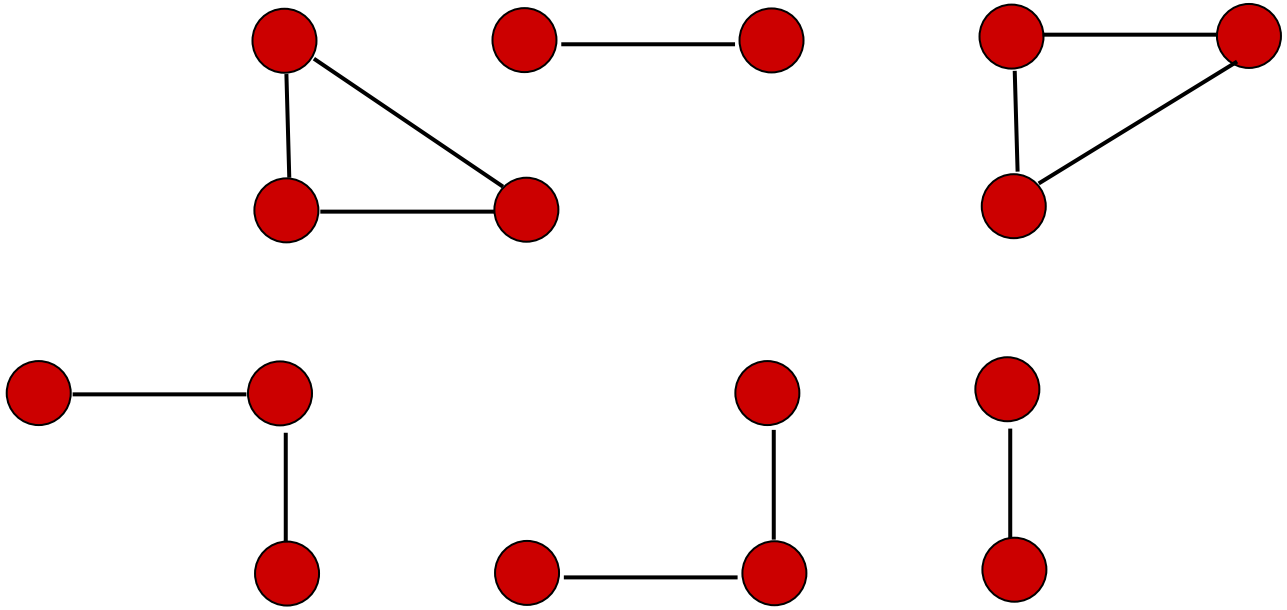
- *Mine frequent connected subgraphs*
- *Containing at least 4 nodes*

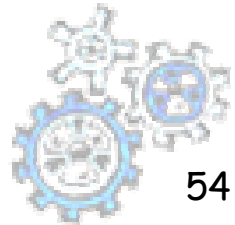




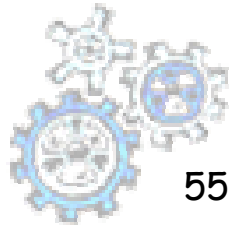
A very general idea

- *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.*

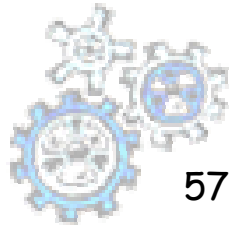
- *There are many interesting constraints which are not Convertible but are Loose Anti-monotone*

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

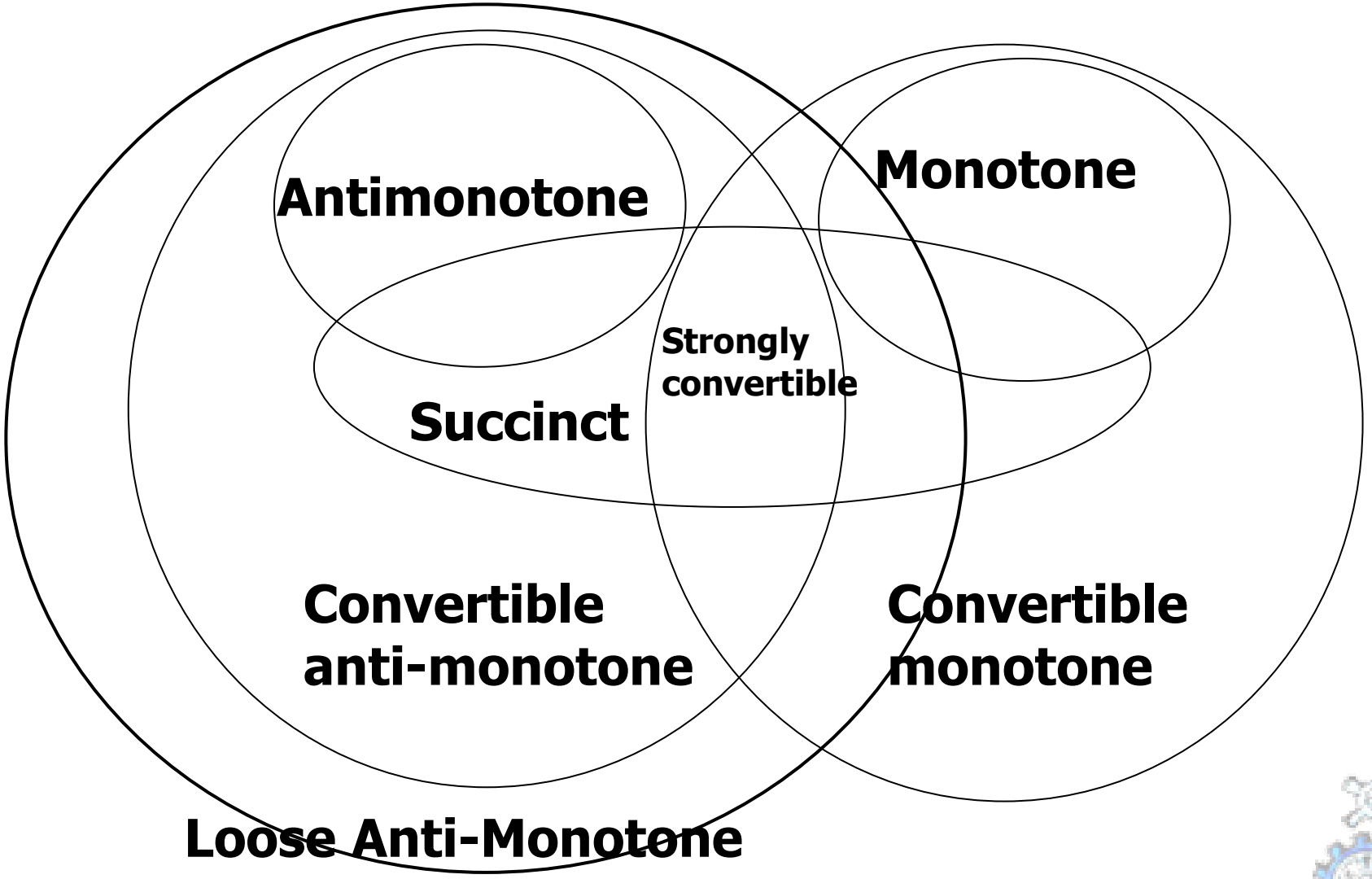
  Not Convertible …

  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: $avg(X.profit) \geq 15$

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

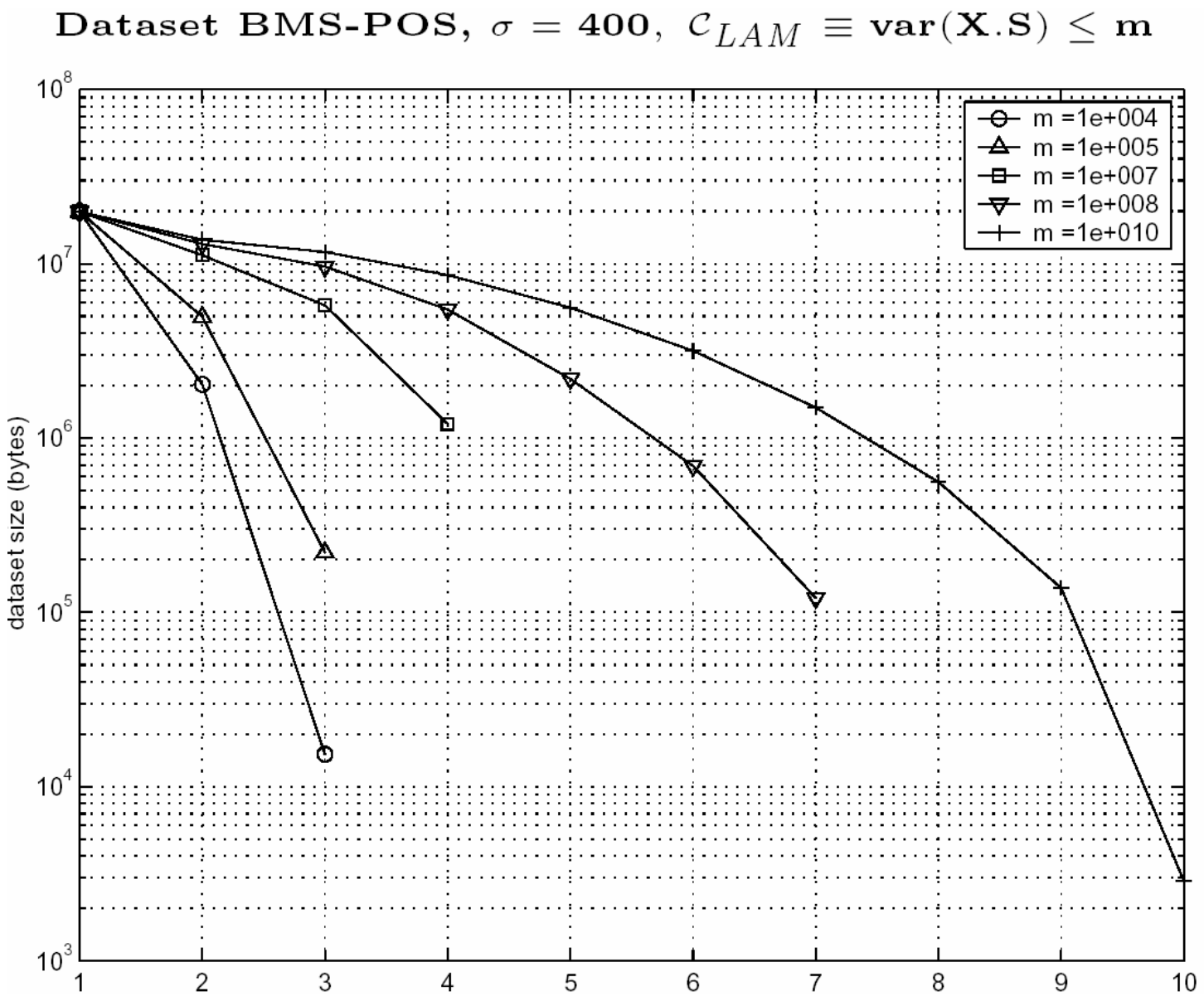
$avg(t) = 20$

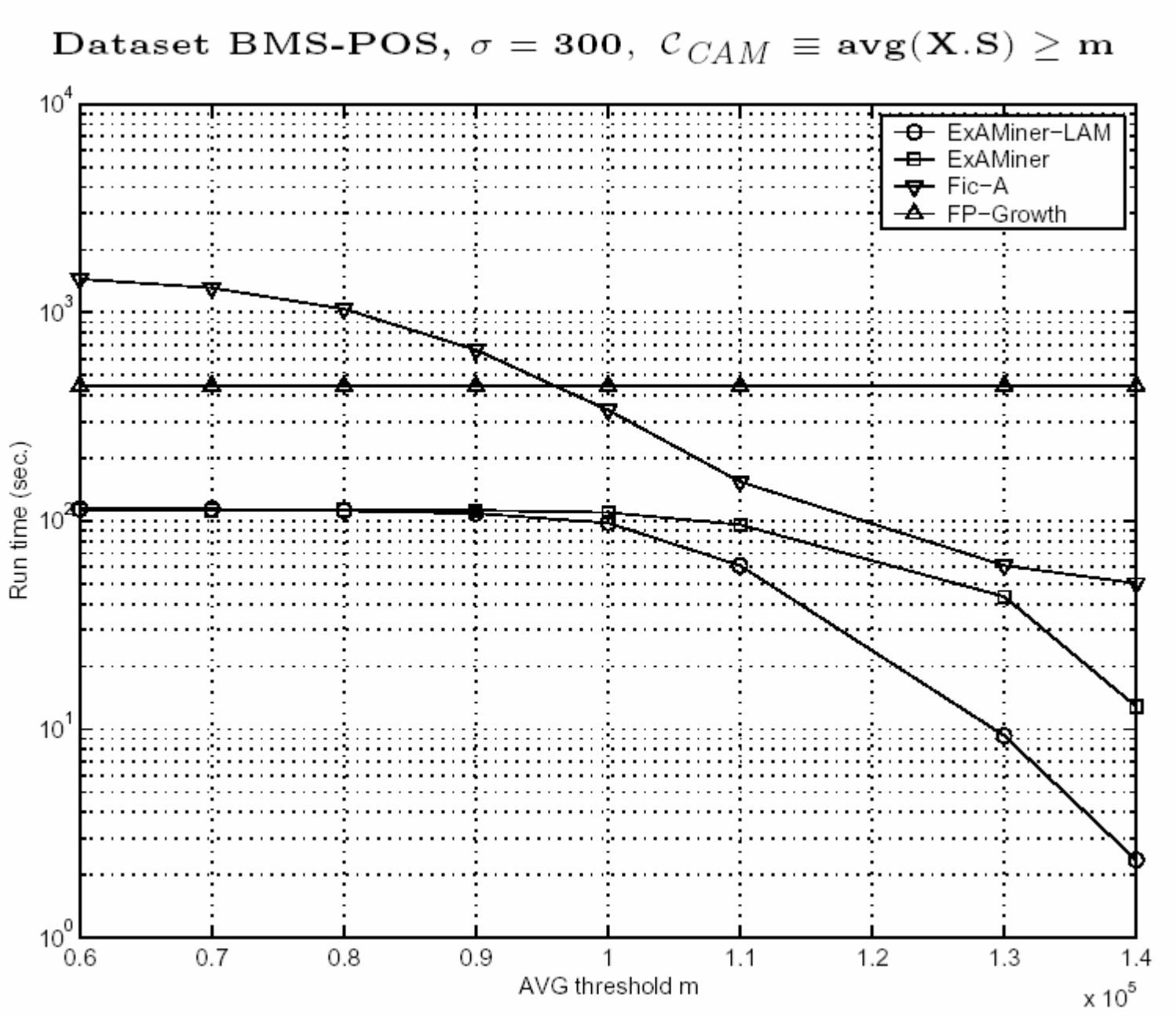
$k = 3$

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


Dataset BMS-POS, $\sigma = 400$, $\mathcal{C}_{LAM} \equiv \mathbf{var}(\mathbf{X}.\mathbf{S}) \leq \mathbf{m}$
dataset size (bytes)
10^8
10^7
10^6
10^5
10^4
10^3
1
2
3
4
5
6
7
8
9
10
m =1e+004
m =1e+005
m =1e+007
m =1e+008
m =1e+010


Dataset BMS-POS, $\sigma = 300$, $\mathcal{C}_{CAM} \equiv \mathbf{avg(X.S)} \geq \mathbf{m}$
ExAMiner-LAM
ExAMiner
Fic-A
FP-Growth
Run time (sec.)
10^4
10^3
10^2
10^1
10^0
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
1.4
AVG threshold m
x 10^5

References

- *Bonchi, Lucchese "Pushing Tougher Constraints" (PAKDD'05)*
- *Bonchi, Goethals "FP-Bonsai: the Art of Growing and Pruning Small FP-Trees" (PAKDD'04)*
- *Bonchi, Giannotti, Mazzanti, Pedreschi. "ExAnte: a Preprocessing Algorithm for Constrained Frequent Pattern Mining" (PKDD'03)*
- *Bonchi, Giannotti, Mazzanti, Pedreschi. "Adaptive Constraint Pushing in Frequent Pattern Mining" (PKDD03)*
- *Bonchi, Giannotti, Mazzanti, Pedreschi. "ExAMiner: Optimized Level-wise Frequent Pattern Mining with Monotone Constraints" (ICDM'03)*
- *Han, Pei, Yin: "Mining frequent patterns without candidate generation" (SIGMOD'00)*
- *Pei, Han "Can We Push More Constraints into Frequent Pattern Mining?" (KDD'00)*
- *Ng, Lakshmanan, Han, Pang "Exploratory Mining and Pruning Optimizations of Constrained Association Rules" (SIGMOD'98)*