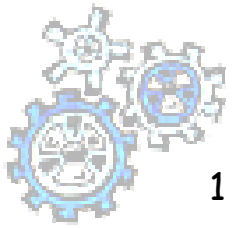
Adding Constraints to Frequent Itemset Mining

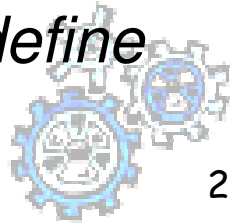


Why Constraints?

- *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.

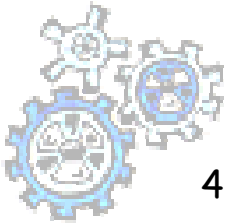
- We indicate the frequency constraint with C_{freq}
- Given a constraint C , let $Th(C) = \{X| C(X)\}$ denote the set of all itemsets X that satisfy C.
- The frequent itemsets mining problem requires to compute $Th(C_{freq})$
- The constrained frequent itemsets mining problem requires to compute: $Th(C_{freq})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

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.*

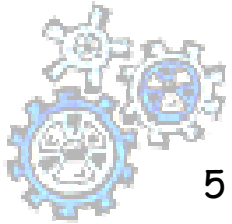


Anti-Monotonicity and Succinctness

- *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*

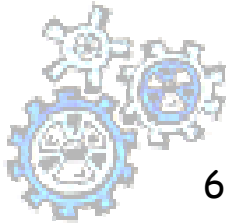
- *4 classes of constraints + associated computational strategy*
  1. *Constraints that are anti-monotone but not succinct*
  2. *Constraints that are both anti-monotone and succinct*
  3. *Constraints that are succinct but not anti-monotone*
  4. *Constraints that are neither*



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}.*

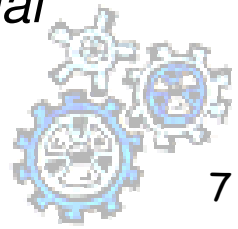
  - *sum(S.Price) ≤ v is anti-monotone*
  - *Very easy to push in the frequent itemset computation*



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

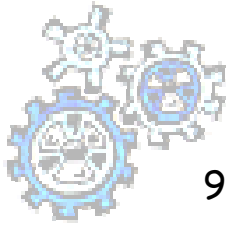
- *4 classes of constraints + associated computational strategy*
  1. *Constraints that are anti-monotone but not succinct*
     - *Check them in conjunction with frequency as a unique anti-monotone constraint*
  2. *Constraints that are both anti-monotone and succinct*
     - *Can be pushed at preprocessing 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*

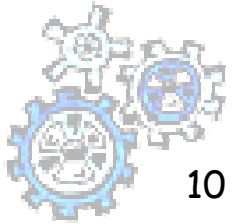


Converting "Tough" Constraints

- *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*

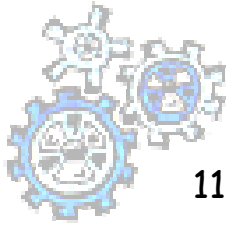


Strongly Convertible Constraints

- *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

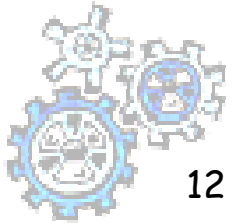
- *Thus, avg(X) ≥ 25 is strongly convertible*



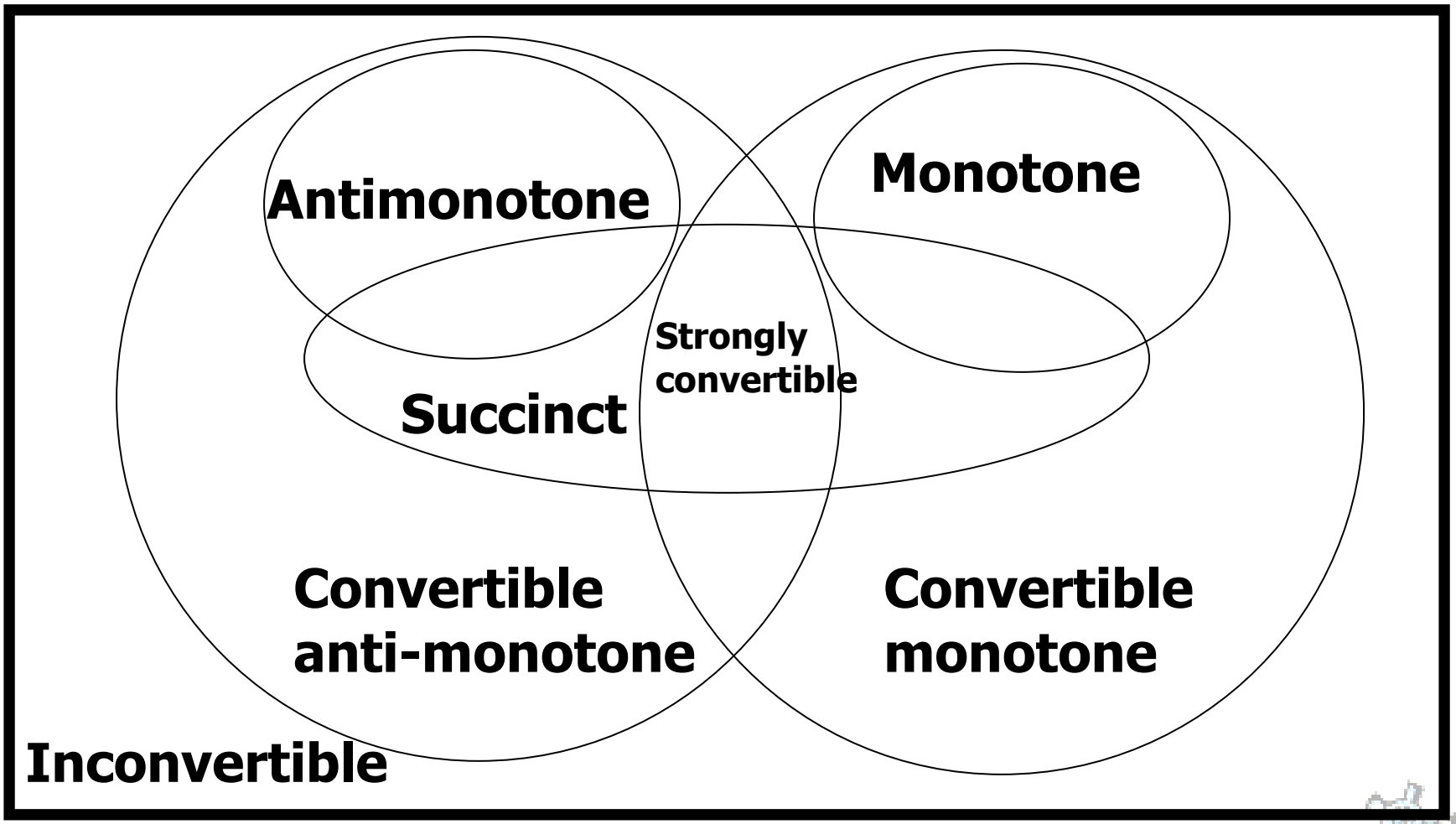
Monotonicity in Constraint-Based Mining

- *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*

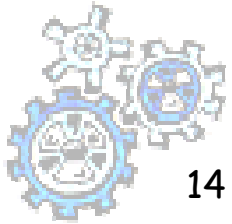
- *They behave exactly the opposite of frequency …*
- *How to push them in the Apriori computation?*



Classification of Constraints



ExAnte
ExAMiner



Our Problem …

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

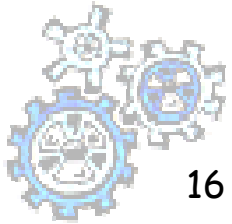
$$Th(\mathcal{C}_{freq}) \cap Th(\mathcal{C}_M)$$

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 …*

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]:*

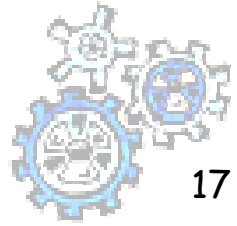
  Given a transaction database TDB and a monotone constraint C_M, we define the μ-reduction of TDB as the dataset resulting from pruning the transactions that do not satisfy C_M.

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

- *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

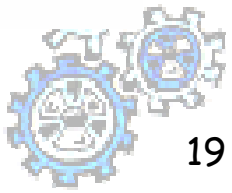
THEOREM 1 (μ-REDUCTION CORRECTNESS). *Given a transaction database TDB, a monotone constraint $\mathcal{C}_M$, and a frequency constraint $\mathcal{C}_{freq}$, we have that:*

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

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

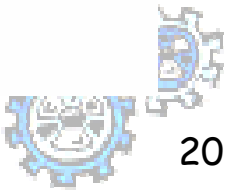
THEOREM 2 (α-REDUCTION CORRECTNESS). *Given a transaction database TDB, a monotone constraint $\mathcal{C}_M$, and a frequency constraint $\mathcal{C}_{freq}$, we have that:*

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

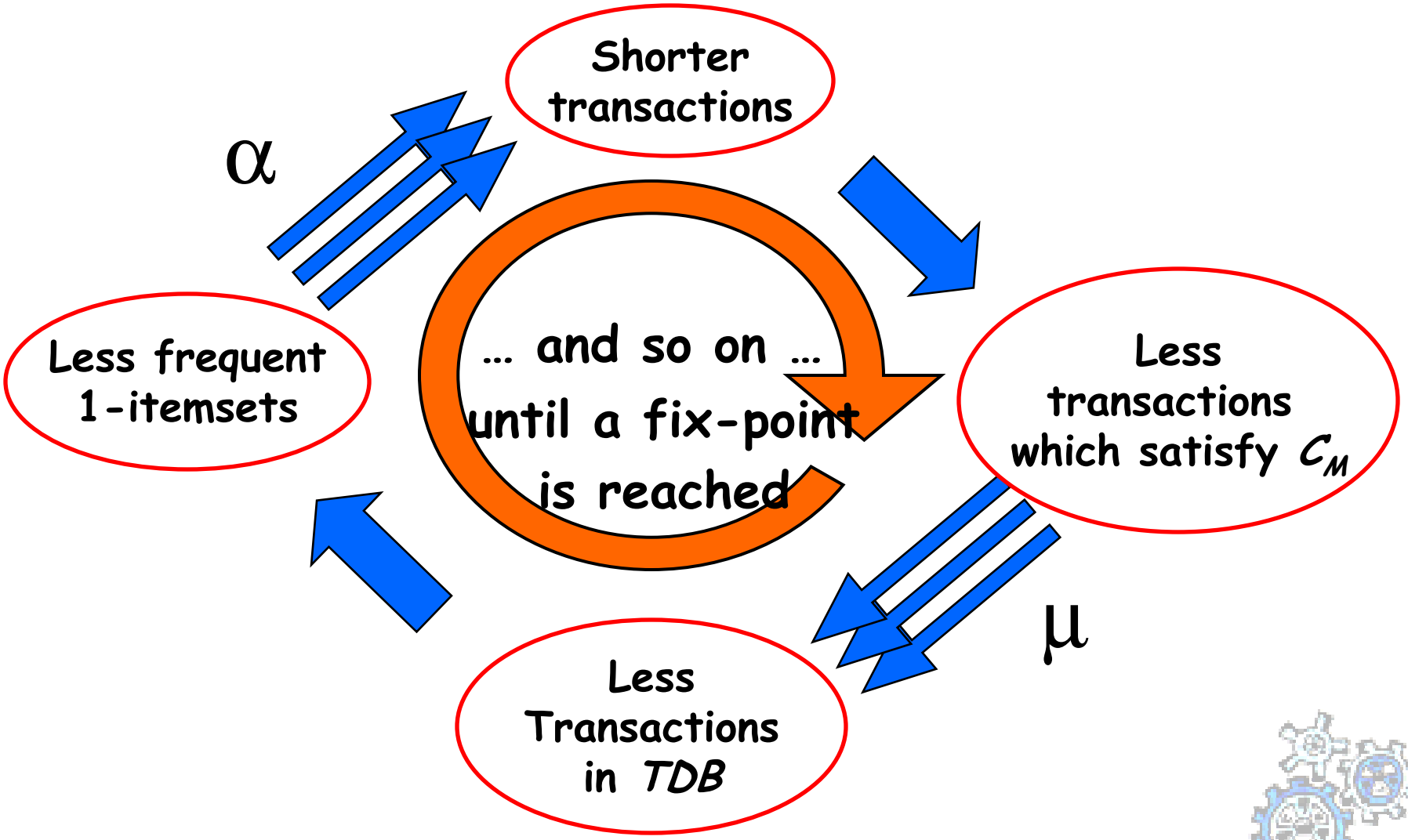
PROOF. Since $X \in Th(\mathcal{C}_{freq})$, all subsets of X will be frequent (by the anti-monotonicity of frequency). Therefore no subset of X will be α-pruned (in particular, no 1-itemsets in X). This implies that:

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

□



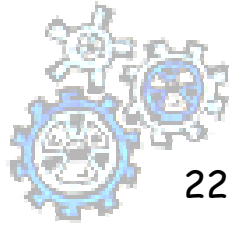
A Fix-Point Computation



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

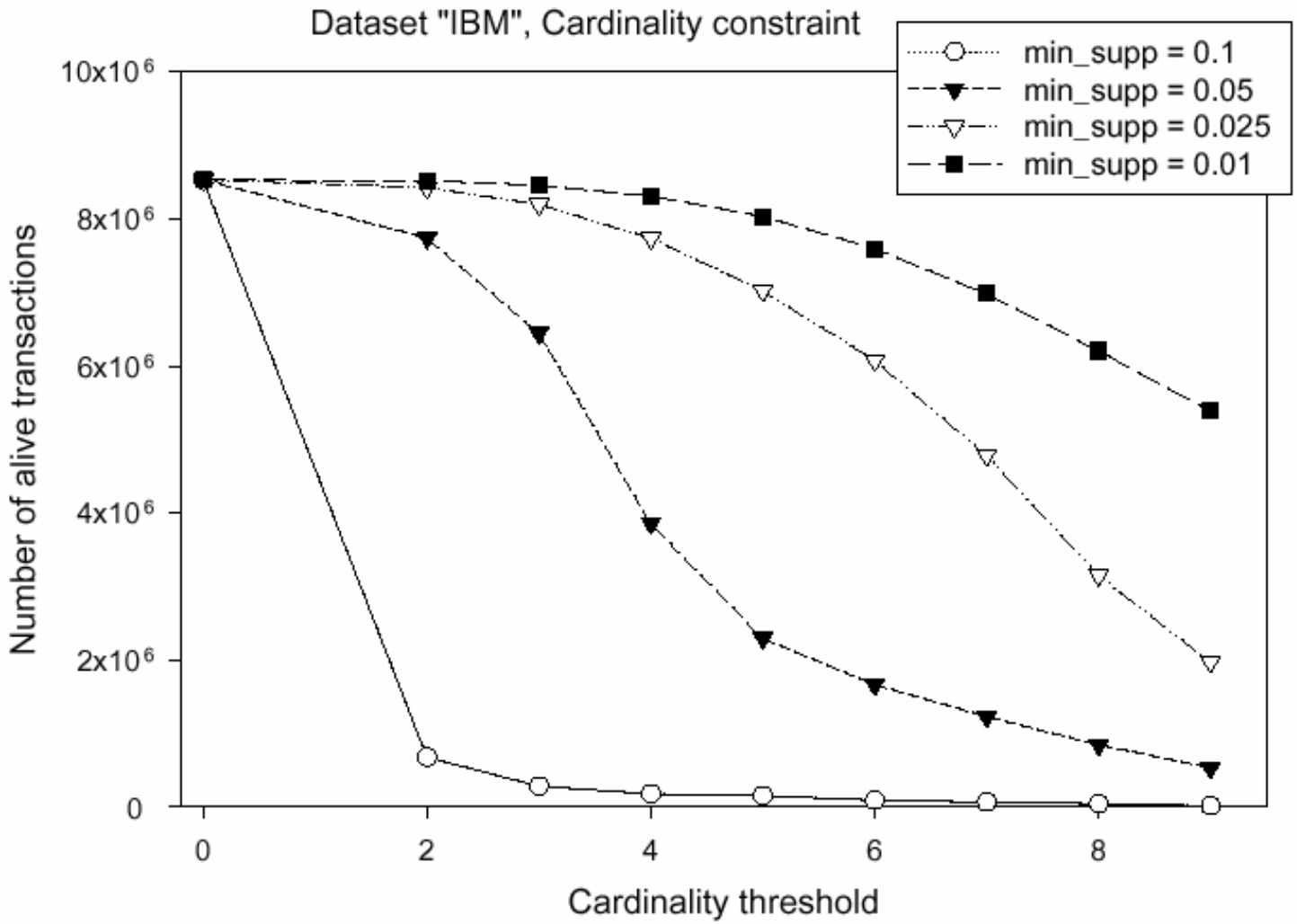
52

44

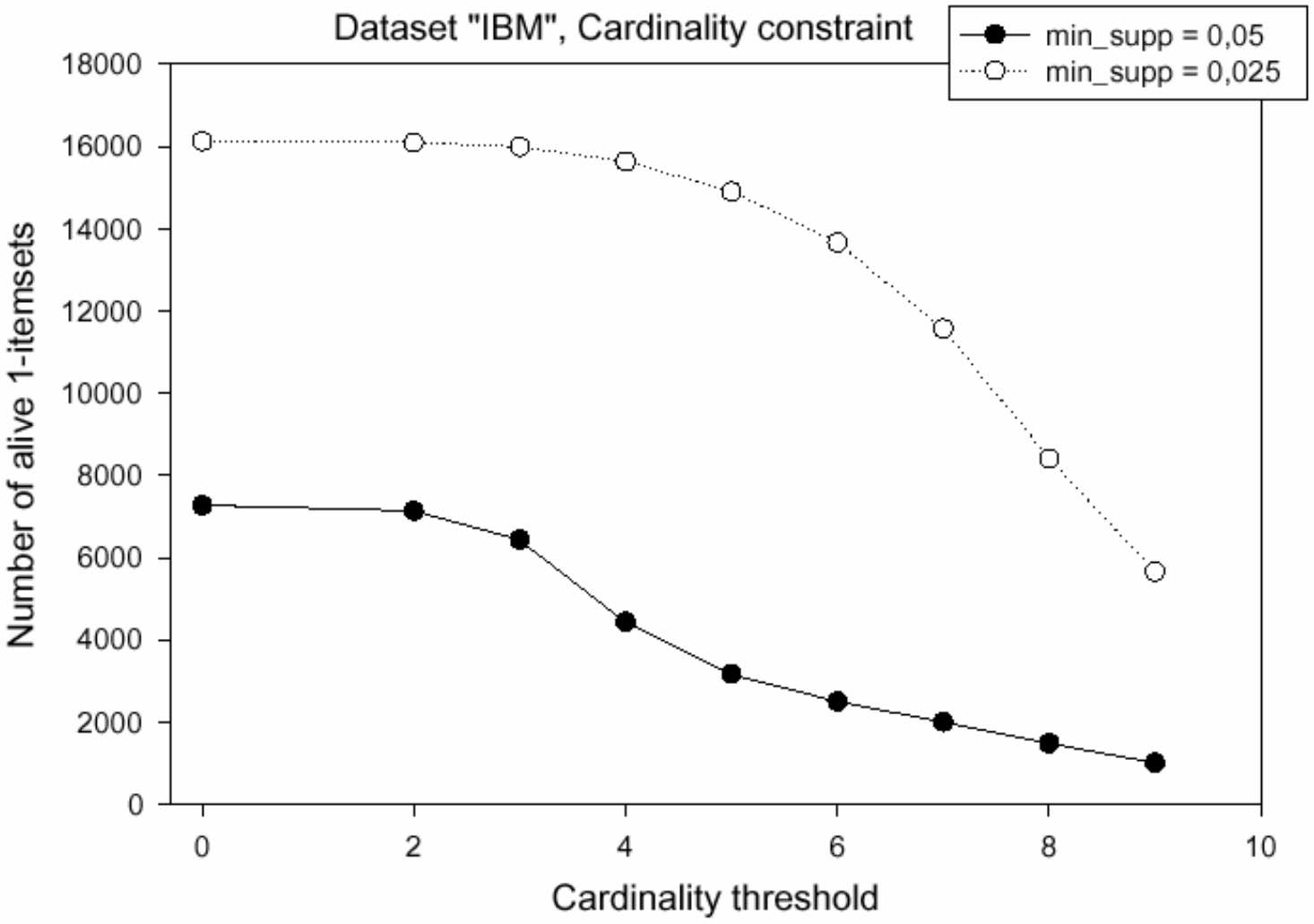
- *Min_sup = 4*
- $C_M \equiv$ *sum(X.price)* ≥ 45

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	†	†

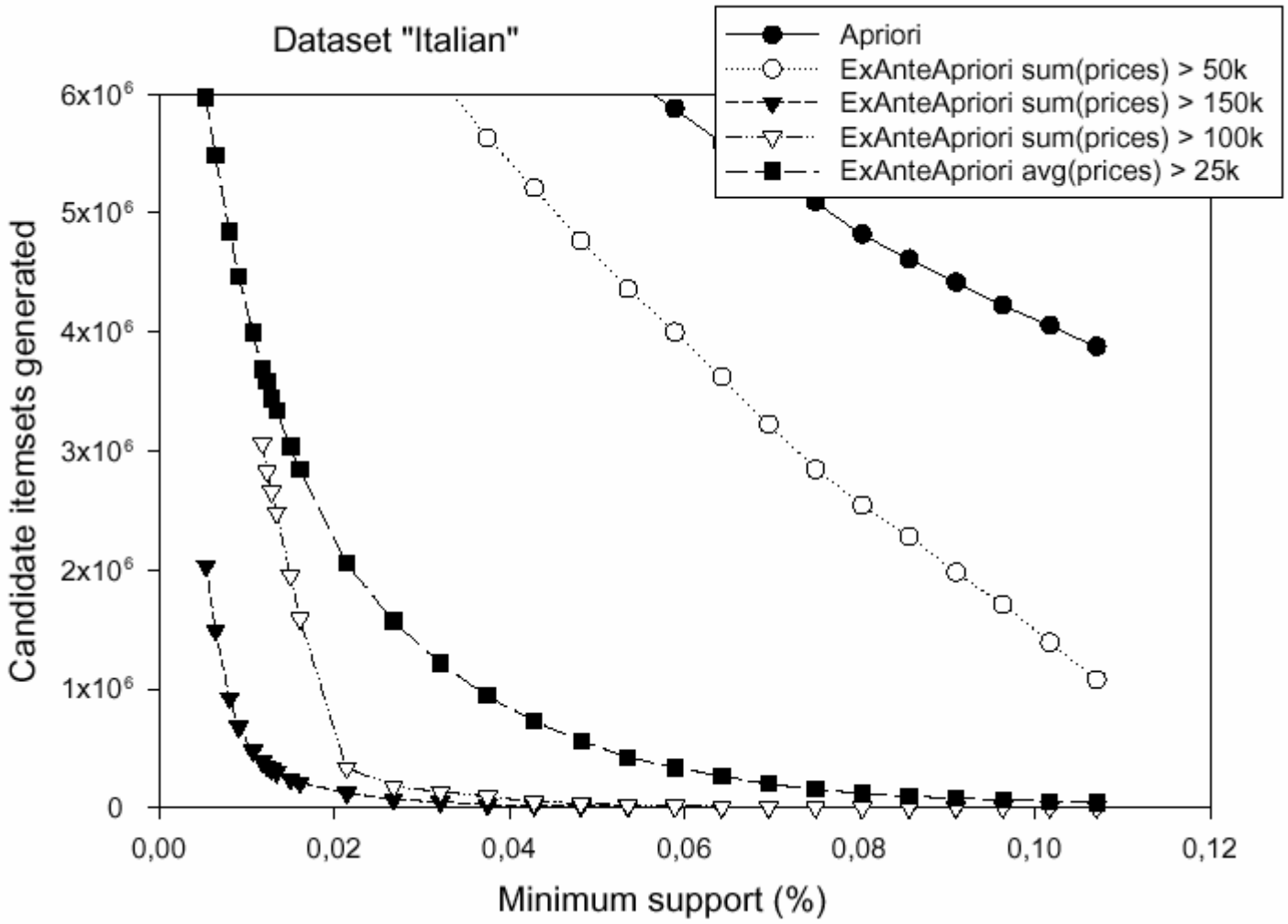
Experimental Results



Experimental Results



Experimental Results



Experimental Results