CSE 5243 INTRO. TO DATA MINING

Advanced Frequent Pattern Mining

(Chapter 7)

Huan Sun, CSE@The Ohio State University

10/31/2017

Slides adapted from Prof. Jiawei Han @UIUC, Prof. Srinivasan Parthasarathy @OSU

Chapter 7 : Advanced Frequent Pattern Mining

- Mining Diverse Patterns
- Constraint-Based Frequent Pattern Mining
- Sequential Pattern Mining
- Graph Pattern Mining
- Pattern Mining Application: Mining Software Copy-and-Paste Bugs
- Summary

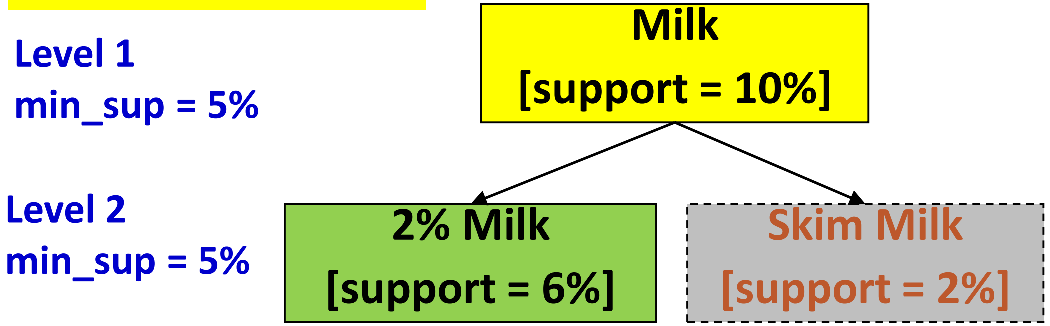
Mining Diverse Patterns

- Mining Multiple-Level Associations
- Mining Multi-Dimensional Associations
- Mining Negative Correlations
- Mining Compressed and Redundancy-Aware Patterns

Mining Multiple-Level Frequent Patterns

- Items often form hierarchies
  - Ex.: Dairyland 2% milk; Wonder wheat bread
- How to set min-support thresholds?

Uniform support



- Uniform min-support across multiple levels (reasonable?)

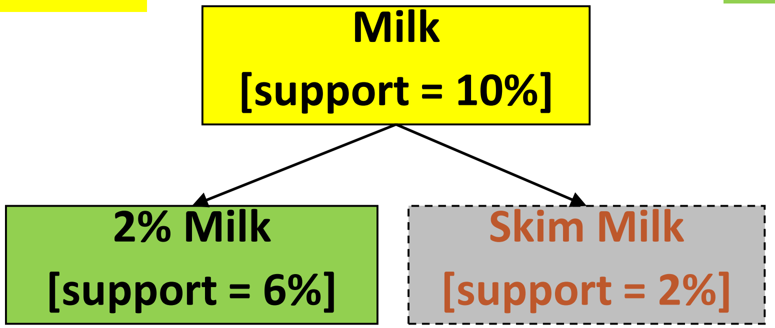
Mining Multiple-Level Frequent Patterns

- Items often form hierarchies
  - Ex.: Dairyland 2% milk; Wonder wheat bread
- How to set min-support thresholds?

Uniform support

Level 1
min_sup = 5%

Level 2
min_sup = 5%



Reduced support

Level 1
min_sup = 5%

Level 2
min_sup = 1%

- Uniform min-support across multiple levels (reasonable?)
- Level-reduced min-support: Items at the lower level are expected to have lower support

ML/MD Associations with Flexible Support Constraints

- Why flexible support constraints?
  - Real life occurrence frequencies vary greatly
    - Diamond, watch, pens in a shopping basket
  - Uniform support may not be an interesting model

- A flexible model
  - The lower-level, the more dimension combination, and the long pattern length, usually the smaller support
  - General rules should be easy to specify and understand
  - Special items and special group of items may be specified individually and have higher priority

Multi-level Association: Redundancy Filtering

- Some rules may be redundant due to "ancestor" relationships between items.
- Example
  - milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%]
  - 2% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%]
  - Suppose the 2% milk sold is about ¼ of milk sold
- We say the first rule is an ancestor of the second rule.
- A rule is redundant if its support is close to the "expected" value, based on the rule's ancestor.

Multi-Level Mining: Progressive Deepening

- A top-down, progressive deepening approach:
  - First mine high-level frequent items:
    milk (15%), bread (10%)
  - Then mine their lower-level "weaker" frequent itemsets:
    2% milk (5%), wheat bread (4%)

- Different min_support threshold across multi-levels lead to different algorithms:
  - If adopting the same *min_support* across multi-levels
    then toss *t* if any of *t*'s ancestors is infrequent.
  - If adopting reduced *min_support* at lower levels
    then examine only those descendents whose ancestor's support is frequent/non-negligible.

Multi-Level Mining: Progressive Deepening

- A top-down, progressive deepening approach:
  - First mine high-level frequent items:

    milk (15%), bread (10%)

  - Then mine their lower-level “weaker” frequent itemsets:

    2% milk (5%), wheat bread (4%)

- Different min_support threshold across multi-levels lead to different algorithms:
  - If adopting the same *min_support* across multi-levels

    then toss *t* if any of *t*’s ancestors is infrequent.
  - If adopting reduced *min_support* at lower levels

    then examine only those descendents whose ancestor’s support is frequent/non-negligible.

Mining Multi-Dimensional Associations

- Single-dimensional rules (e.g., items are all in "product" dimension)
  - buys(X, "milk") $\Rightarrow$ buys(X, "bread")

- Multi-dimensional rules (i.e., items in $\geq$ 2 dimensions or predicates)
  - Inter-dimension association rules (*no repeated predicates*)
    - age(X, "18-25") $\wedge$ occupation(X, "student") $\Rightarrow$ buys(X, "coke")
  - Hybrid-dimension association rules (*repeated predicates*)
    - age(X, "18-25") $\wedge$ buys(X, "popcorn") $\Rightarrow$ buys(X, "coke")

Mining Rare Patterns vs. Negative Patterns

- Rare patterns
  - Very low support but interesting (e.g., buying Rolex watches)
  - How to mine them? Setting individualized, group-based min-support thresholds for different groups of items

Mining Rare Patterns vs. Negative Patterns

- Rare patterns
  - Very low support but interesting (e.g., buying Rolex watches)
  - How to mine them? Setting individualized, group-based min-support thresholds for different groups of items
- Negative patterns
  - Negatively correlated: Unlikely to happen together
  - Ex.: Since it is unlikely that the same customer buys both a Ford Expedition (an SUV car) and a Ford Fusion (a hybrid car), buying a Ford Expedition and buying a Ford Fusion are likely negatively correlated patterns
  - How to define negative patterns?

Defining Negative Correlated Patterns

- A (relative) support-based definition
  - If itemsets A and B are both frequent but rarely occur together, i.e., $sup(A \cup B) << sup(A) \times sup(B)$
  - Then A and B are negatively correlated

Defining Negative Correlated Patterns

- A (relative) support-based definition
  - If itemsets A and B are both frequent but rarely occur together, i.e., $sup(A \cup B) \ll sup(A) \times sup(B)$
  - Then A and B are negatively correlated

Does this remind you the definition of *lift*?

- Is this a good definition for large transaction datasets?

Defining Negative Correlated Patterns

- A (relative) support-based definition
  - If itemsets A and B are both frequent but rarely occur together, i.e., $sup(A \cup B) << sup(A) \times sup(B)$
  - Then A and B are negatively correlated

Does this remind you the definition of *lift*?

- Is this a good definition for large transaction datasets?
- Ex.: Suppose a store sold two needle packages A and B 100 times each, but only one transaction contained both A and B
  - When there are in total 200 transactions, we have
    - $s(A \cup B) = 0.005$, $s(A) \times s(B) = 0.25$, $s(A \cup B) << s(A) \times s(B)$
  - But when there are 10^5 transactions, we have
    - $s(A \cup B) = 1/10^5$, $s(A) \times s(B) = 1/10^3 \times 1/10^3$, $s(A \cup B) > s(A) \times s(B)$

Defining Negative Correlated Patterns

- A (relative) support-based definition
  - If itemsets A and B are both frequent but rarely occur together, i.e., $sup(A \cup B) \ll sup(A) \times sup(B)$
  - Then A and B are negatively correlated

Does this remind you the definition of *lift?*

- Is this a good definition for large transaction datasets?
- Ex.: Suppose a store sold two needle packages A and B 100 times each, but only one transaction contained both A and B
  - When there are in total 200 transactions, we have
    - $s(A \cup B) = 0.005$, $s(A) \times s(B) = 0.25$, $s(A \cup B) \ll s(A) \times s(B)$
  - But when there are 10^5 transactions, we have
    - $s(A \cup B) = 1/10^5$, $s(A) \times s(B) = 1/10^3 \times 1/10^3$, $s(A \cup B) > s(A) \times s(B)$
  - What is the problem?—Null transactions: The support-based definition is not null-invariant!

Defining Negative Correlation: Need Null-Invariance in Definition

- A good definition on negative correlation should take care of the null-invariance problem
  - Whether two itemsets A and B are negatively correlated should not be influenced by the number of null-transactions

Which measure should we use? Recall last lectures….

Defining Negative Correlation: Need Null-Invariance in Definition

- A good definition on negative correlation should take care of the null-invariance problem
  - Whether two itemsets A and B are negatively correlated should not be influenced by the number of null-transactions

- A Kulczynski measure-based definition
  - If itemsets A and B are frequent but

    $(s(A \cup B)/s(A) + s(A \cup B)/s(B))/2 < \epsilon$,

    where ϵ is a negative pattern threshold, then A and B are negatively correlated

- For the same needle package problem:
  - No matter there are in total 200 or 10^5 transactions
    - If $\epsilon = 0.02$, we have

      $(s(A \cup B)/s(A) + s(A \cup B)/s(B))/2 = (0.01 + 0.01)/2 < \epsilon$

Chapter 7 : Advanced Frequent Pattern Mining

- Mining Diverse Patterns
- Constraint-Based Frequent Pattern Mining
- Sequential Pattern Mining
- Graph Pattern Mining
- Pattern Mining Application: Mining Software Copy-and-Paste Bugs
- Summary

Constraint-based Data Mining

- Finding all the patterns in a database autonomously? — unrealistic!
  - The patterns could be too many but not focused!

Constraint-based Data Mining

- Finding all the patterns in a database autonomously? — unrealistic!
  - The patterns could be too many but not focused!

- Data mining should be an interactive process
  - User directs what to be mined using a data mining query language (or a graphical user interface)

Constraint-based Data Mining

- Finding all the patterns in a database autonomously? — unrealistic!
  - The patterns could be too many but not focused!
- Data mining should be an interactive process
  - User directs what to be mined using a data mining query language (or a graphical user interface)
- Constraint-based mining
  - User flexibility: provides constraints on what to be mined
  - System optimization: explores such constraints for efficient mining—constraint-based mining

Categories of Constraints

CONSTRAINT 1 (ITEM CONSTRAINT). *An* item constraint *specifies what are the particular individual or groups of items that should or should not be present in the pattern.* □

For example, a dairy company may be interested in patterns containing only dairy products, when it mines transactions in a grocery store.

CONSTRAINT 2 (LENGTH CONSTRAINT). *A* length constraint *specifies the requirement on the length of the patterns, i.e., the number of items in the patterns.* □

For example, when mining classification rules for documents, a user may be interested in only frequent patterns with at least 5 keywords, a typical length constraint.

Categories of Constraints

Constraint 3 (Model-based constraint). *A* model-based constraint *looks for patterns which are sub- or super-patterns of some given patterns (models).* □

For example, a travel agent may be interested in what other cities that a visitor is likely to travel if s/he visits both Washington and New York city. That is, they want to find frequent patterns which are super-patterns of {Washington, New York city}.

Constraint 4 (Aggregate constraint). *An* aggregate constraint *is on an aggregate of items in a pattern, where the aggregate function can be* SUM, AVG, MAX, MIN, *etc.* □

For example, a marketing analyst may like to find frequent patterns where the average price of all items in each pattern is over $100.

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

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
  - First find all frequent sets, and then test them for constraint satisfaction

The Apriori Algorithm — Example

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
{2 3 5}	2

Naïve Algorithm: Apriori + Constraint (Naïve Solution)

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
~~{5}~~	~~3~~

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

Sum{S.price < 5}

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
  - First find 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 in Constraint-Based Mining

- Anti-monotonicity
  - *When an itemset S* ***violates*** *the constraint, so does any of its superset*
  - $sum(S.Price) \leq v$ is anti-monotone?
  - $sum(S.Price) \geq v$ is anti-monotone?

Anti-Monotonicity in Constraint-Based Mining

- Anti-monotonicity
  - *When an itemset S* ***violates*** *the constraint, so does any of its superset*
  - $sum(S.Price) \leq v$ is anti-monotone
  - $sum(S.Price) \geq v$ is not anti-monotone

Anti-Monotonicity in Constraint-Based Mining

- Anti-monotonicity
  - *When an itemset S **violates** the constraint, so does any of its superset*
  - $sum(S.Price) \leq v$ is anti-monotone
  - $sum(S.Price) \geq v$ is not anti-monotone
- Example. C: range(S.profit) $\leq$ 15 is anti-monotone
  - Itemset *ab* violates C
  - So does every superset of *ab*
  - *Define* $range(S.profit) = max(S.A) - min(S.A)$

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

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

Which Constraints Are Anti-Monotone?

Constraint	Antimonotone
$v \in S$	No
$S \supseteq V$	no
$S \subseteq V$	yes
$\min(S) \leq v$	no
$\min(S) \geq v$	yes
$\max(S) \leq v$	yes
$\max(S) \geq v$	no
$count(S) \leq v$	yes
$count(S) \geq v$	no
$sum(S) \leq v$ ($a \in S, a \geq 0$)	yes
$sum(S) \geq v$ ($a \in S, a \geq 0$)	no
$range(S) \leq v$	yes
$range(S) \geq v$	no
$avg(S)\ \theta\ v, \theta \in \{=, \leq, \geq\}$	convertible
$support(S) \geq \xi$	yes
$support(S) \leq \xi$	no

Monotonicity in Constraint-Based Mining

- Monotonicity
  - *When an intemset S* ***satisfies*** *the constraint, so does any of its superset*
  - *sum(S.Price)* $\geq$ *v* is ?
  - *min(S.Price)* $\leq$ *v* is ?

Monotonicity in Constraint-Based Mining

- Monotonicity
  - *When an intemset S **satisfies** the constraint, so does any of its superset*
  - $sum(S.Price) \geq v$ is monotone
  - $min(S.Price) \leq v$ is monotone

Monotonicity in Constraint-Based Mining

- Monotonicity
  - *When an intemset S **satisfies** the constraint, so does any of its superset*
  - *sum(S.Price)* $\geq v$ is monotone
  - *min(S.Price)* $\leq v$ is monotone
- Example. C: range(S.profit) ≥ 15
  - Itemset *ab* satisfies C
  - So does every superset of *ab*

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

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

Which Constraints Are Monotone?

Constraint	Monotone
$v \in S$	yes
$S \supseteq V$	yes
$S \subseteq V$	no
$\min(S) \leq v$	yes
$\min(S) \geq v$	no
$\max(S) \leq v$	no
$\max(S) \geq v$	yes
$count(S) \leq v$	no
$count(S) \geq v$	yes
$sum(S) \leq v$ ($a \in S, a \geq 0$)	no
$sum(S) \geq v$ ($a \in S, a \geq 0$)	yes
$range(S) \leq v$	no
$range(S) \geq v$	yes
$avg(S)\ \theta\ v, \theta \in \{=, \leq, \geq\}$	convertible
$support(S) \geq \xi$	no
$support(S) \leq \xi$	yes

The Apriori Algorithm — Example

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
{2 3 5}	2

Naïve Algorithm: Apriori + Constraint

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
~~{5}~~	~~3~~

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

Sum{S.price < 5}

Pushing the constraint deep into the process

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
~~{5}~~	~~3~~

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
~~{5}~~	~~3~~

C_2

itemset
{1 2}
{1 3}
~~{1 5}~~
{2 3}
~~{2 5}~~
~~{3 5}~~

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
~~{1 5}~~	~~1~~
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

Why?

C_3

itemset
~~{2 3 5}~~

Scan D

L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

Sum{S.price < 5}

Converting “Tough” Constraints

- Convert tough constraints into anti-monotone or monotone by properly ordering items

Converting "Tough" Constraints

- Convert tough constraints into anti-monotone or monotone by properly ordering items
- 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!

Converting "Tough" Constraints

- Convert tough constraints into anti-monotone or monotone by properly ordering items
- 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!

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

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

Convertible Constraints

- 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. $avg(S) \le v$ w.r.t. item value ascending order

  Why?

Convertible Constraints

- 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. $avg(S) \leq v$ w.r.t. item value ascending 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) $\geq v$ w.r.t. item value ascending order

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

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

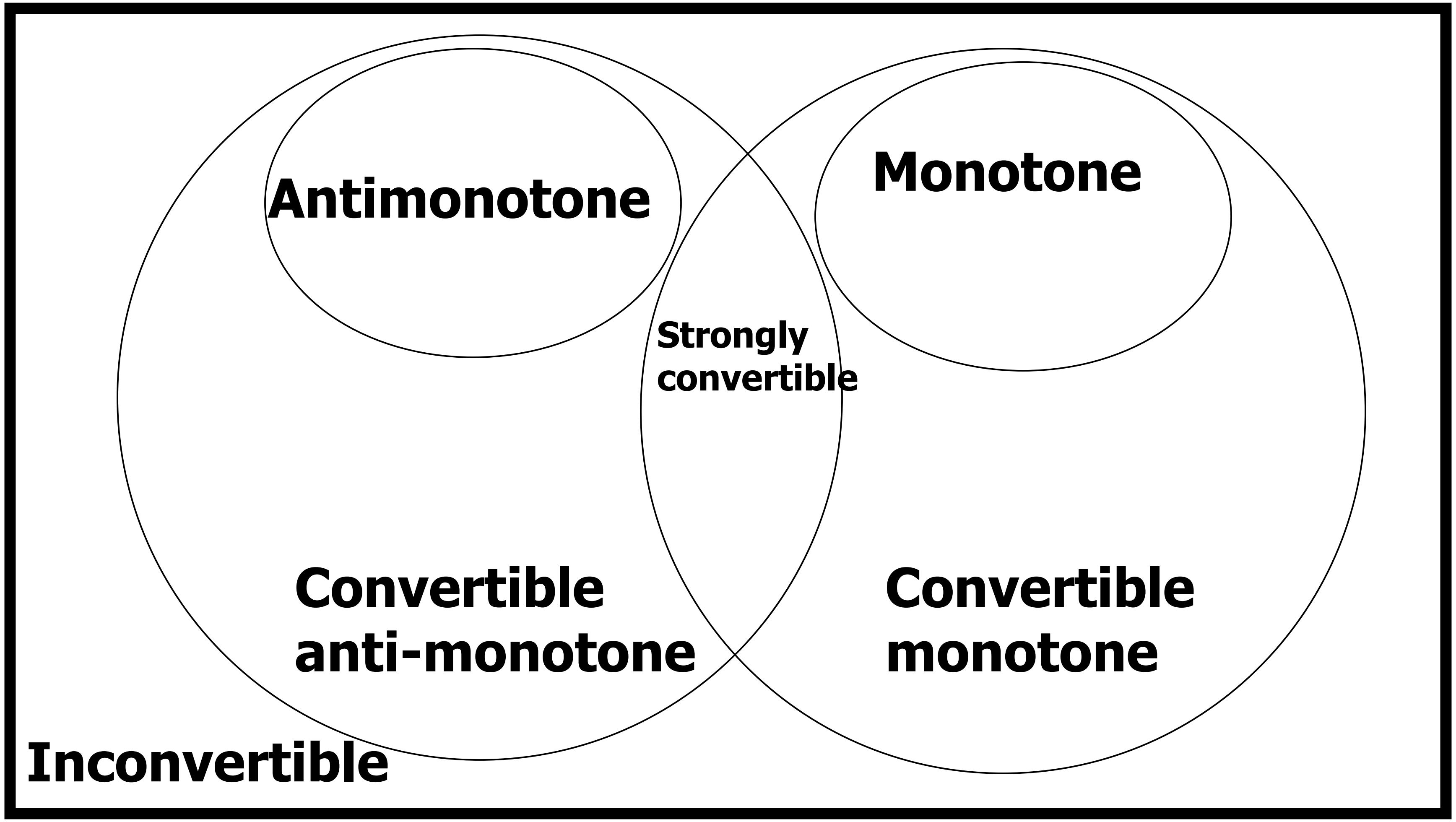
What Constraints Are Convertible?

Constraint	Convertible anti-monotone	Convertible monotone	Strongly convertible
avg(S) ≤ , ≥ v	Yes	Yes	Yes
median(S) ≤ , ≥ v	Yes	Yes	Yes
sum(S) ≤ v (items could be of any value, v ≥ 0)	Yes	No	No
sum(S) ≤ v (items could be of any value, v ≤ 0)	No	Yes	No
sum(S) ≥ v (items could be of any value, v ≥ 0)	No	Yes	No
sum(S) ≥ v (items could be of any value, v ≤ 0)	Yes	No	No
……			

Combing Them Together—A General Picture

Constraint	Antimonotone	Monotone
$v \in S$	no	yes
$S \supseteq V$	no	yes
$S \subseteq V$	yes	no
$\min(S) \le v$	no	yes
$\min(S) \ge v$	yes	no
$\max(S) \le v$	yes	no
$\max(S) \ge v$	no	yes
$\text{count}(S) \le v$	yes	no
$\text{count}(S) \ge v$	no	yes
$\text{sum}(S) \le v$ ($a \in S, a \ge 0$)	yes	no
$\text{sum}(S) \ge v$ ($a \in S, a \ge 0$)	no	yes
$\text{range}(S) \le v$	yes	no
$\text{range}(S) \ge v$	no	yes
$\text{avg}(S)\ \theta\ v, \theta \in \{=, \le, \ge\}$	convertible	convertible
$\text{support}(S) \ge \xi$	yes	no
$\text{support}(S) \le \xi$	no	yes

Classification of Constraints



Mining With Convertible Constraints

- C: avg(S.profit) ≥ 25

TDB (min_sup=2)

TID	Transaction
10	a, f, d, b, c
20	f, g, d, b, c
30	a, f, d, c, e
40	f, g, h, c, e

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

- Scan transaction DB once
  - remove infrequent items
    - Item *h* in transaction 40 is dropped
  - Itemsets *a* and *f* are good

Can Apriori Handle Convertible Constraint?

- A convertible, not monotone nor anti-monotone cannot be pushed deep into the an Apriori mining algorithm
  - Within the level wise framework, no direct pruning based on the constraint can be made
  - Itemset df violates constraint C: avg(X)>=25
  - **Can we prune df afterwards?**

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

Can Apriori Handle Convertible Constraint?

- A convertible, not monotone nor anti-monotone cannot be pushed deep into the an Apriori mining algorithm
  - Within the level wise framework, no direct pruning based on the constraint can be made
  - Itemset df violates constraint C: avg(X)>=25
  - **Since adf satisfies C, Apriori needs df to assemble adf, df cannot be pruned**
- But it can be pushed into frequent-pattern growth framework!

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

Mining With Convertible Constraints in FP-Growth Framework

- C: avg(X)>=25, min_sup=2
- List items in every transaction in value descending order R: <a, f, g, d, b, h, c, e>
  - C is convertible anti-monotone w.r.t. R
- Scan TDB once
  - remove infrequent items
    - Item h is dropped
  - Itemsets a and f are good, …
- Projection-based mining
  - Imposing an appropriate order on item projection
  - Many tough constraints can be converted into (anti)-monotone

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

TDB (min_sup=2)

TID	Transaction
10	a, f, d, b, c
20	f, g, d, b, c
30	a, f, d, c, e
40	f, g, h, c, e

Mining With Convertible Constraints in FP-Growth Framework

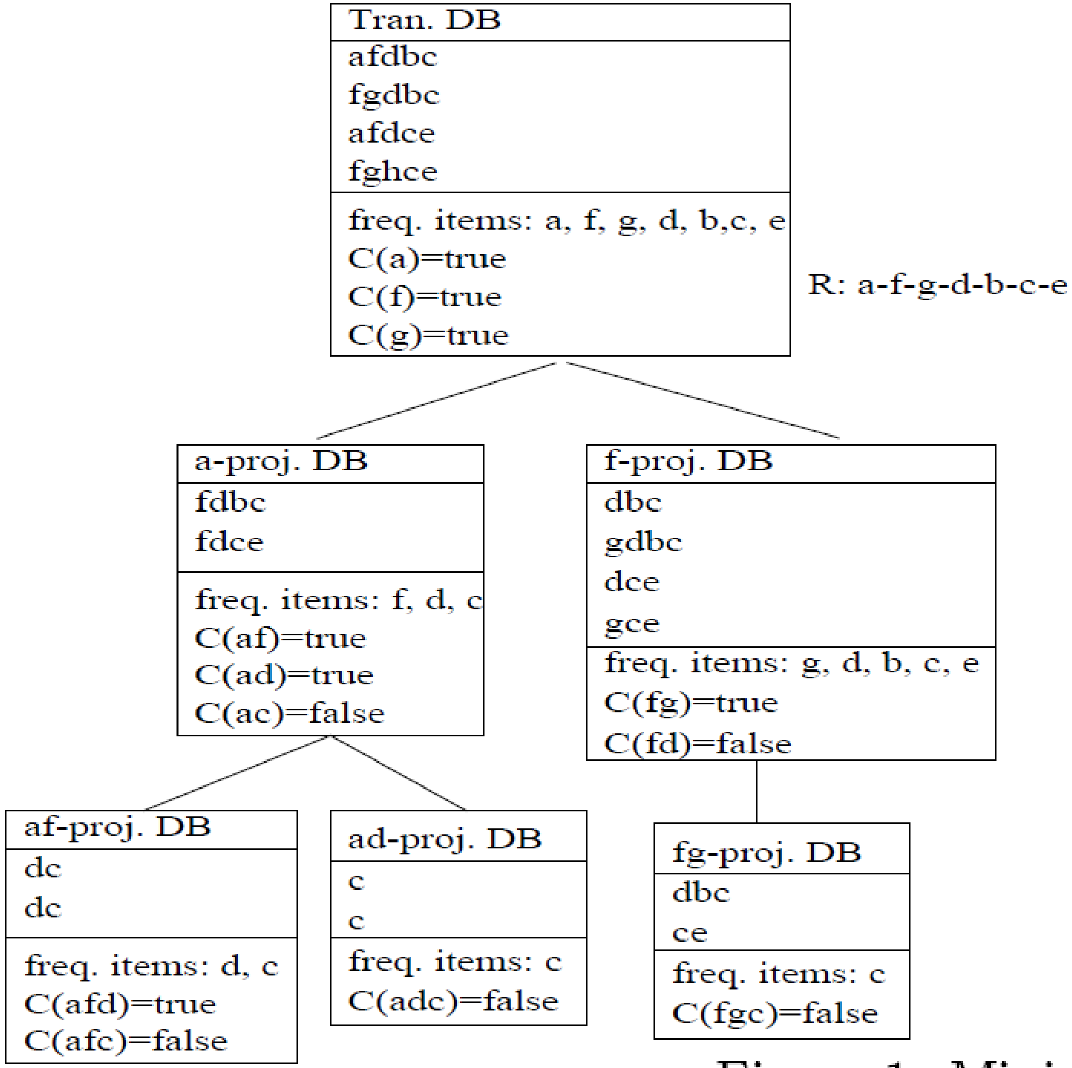


Figure 1: Mining frequent itemsets satisfying constraint $avg(S) \geq 25$.

Constrained Frequent Pattern Mining: A Pattern-Growth View

Jian Pei, Jiawei Han, SIGKDD 2002

Handling Multiple Constraints

- Different constraints may require different or even conflicting item-ordering
- If there exists an order *R* s.t. both C_1 and C_2 are convertible w.r.t. *R*, then there is no conflict between the two convertible constraints
- If there exists conflict on order of items
  - Try to satisfy one constraint first
  - Then using the order for the other constraint to mine frequent itemsets in the corresponding projected database

Chapter 7 : Advanced Frequent Pattern Mining

- Mining Diverse Patterns
- Constraint-Based Frequent Pattern Mining
- Sequential Pattern Mining
- Graph Pattern Mining
- Pattern Mining Application: Mining Software Copy-and-Paste Bugs
- Summary

Sequence Databases & Sequential Patterns

- Sequential pattern mining has broad applications
  - Customer shopping sequences
    - Purchase a laptop first, then a digital camera, and then a smartphone, within 6 months
  - Medical treatments, natural disasters (e.g., earthquakes), science & engineering processes, stocks and markets, ...
  - Weblog click streams, calling patterns, …
  - Software engineering: Program execution sequences, …
  - Biological sequences: DNA, protein, …
- Transaction DB, sequence DB vs. time-series DB
- Gapped vs. non-gapped sequential patterns
  - Shopping sequences, clicking streams vs. biological sequences

Sequence Mining: Description

- Input
  - A database ***D*** of sequences called *data-sequences,* in which:
    - $I=\{i_1, i_2,\ldots,i_n\}$ is the set of items
    - each sequence is a list of transactions ordered by transaction-time
    - each transaction consists of fields: sequence-id, transaction-id, transaction-time and a set of items.

- Problem
  - To discover all the sequential patterns with a user-specified minimum support

Input Database: example

Database $\mathcal{D}$

Sequence-Id	Transaction Time	Items
C1	1	Ringworld
C1	2	Foundation
C1	15	Ringworld Engineers, Second Foundation
C2	1	Foundation, Ringworld
C2	20	Foundation and Empire
C2	50	Ringworld Engineers

45% of customers who bought ***Foundation*** will buy ***Foundation and Empire*** within the next month.

Sequential Pattern and Sequential Pattern Mining

- <u>Sequential pattern mining</u>: Given a set of sequences, find the complete set of *frequent* subsequences (i.e., satisfying the min_sup threshold)

A ***sequence database***

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

A ***sequence:*** < (ef) (ab) (df) c b >

- An <u>element</u> may contain a set of *items* (also called *events*)
- Items within an element are unordered and we list them alphabetically

<a(bc)dc> is a ***subsequence*** of <a(abc)(ac)d(cf)>

- Given ***support threshold*** *min_sup* = 2, <(ab)c> is a ***sequential pattern***

A Basic Property of Sequential Patterns: Apriori

- A basic property: Apriori (Agrawal & Sirkant'94)
  - If a sequence S is not frequent
  - Then none of the super-sequences of S is frequent
  - E.g, <hb> is infrequent → so do <hab> and <(ah)b>

Seq. ID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

Given *support threshold*
min_sup =2

GSP: Apriori-Based Sequential Pattern Mining

- Initial candidates: All 8-singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Scan DB once, count support for each candidate
- Generate length-2 candidate sequences

SID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

min_sup = 2

Cand.	sup
<a>	3
<b>	5
<c>	4
<d>	3
<e>	3
<f>	2
~~<g>~~	1
~~<h>~~	1

	<a>	<b>	<c>	<d>	<e>	<f>
<a>	<aa>	<ab>	<ac>	<ad>	<ae>	<af>
<b>	<ba>	<bb>	<bc>	<bd>	<be>	<bf>
<c>	<ca>	<cb>	<cc>	<cd>	<ce>	<cf>
<d>	<da>	<db>	<dc>	<dd>	<de>	<df>
<e>	<ea>	<eb>	<ec>	<ed>	<ee>	<ef>
<f>	<fa>	<fb>	<fc>	<fd>	<fe>	<ff>

	<a>	<b>	<c>	<d>	<e>	<f>
<a>		<(ab)>	<(ac)>	<(ad)>	<(ae)>	<(af)>
<b>			<(bc)>	<(bd)>	<(be)>	<(bf)>
<c>				<(cd)>	<(ce)>	<(cf)>
<d>					<(de)>	<(df)>
<e>						<(ef)>
<f>						

- Without Apriori pruning:

  (8 singletons) 8*8+8*7/2 = 92 length-2 candidates
- With pruning, length-2 candidates: 36 + 15= 51

GSP (Generalized Sequential Patterns): Srikant & Agrawal @ EDBT'96

GSP Mining and Pruning

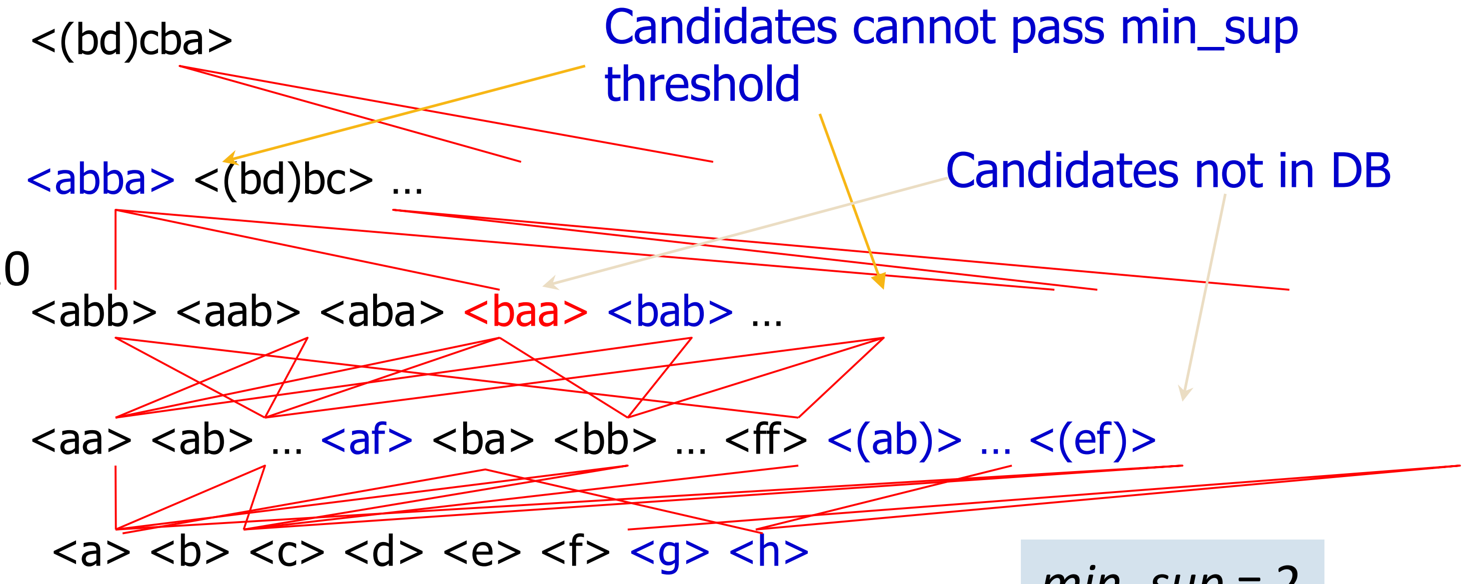
5th scan: 1 cand. 1 length-5 seq. pat.

4th scan: 8 cand. 7 length-4 seq. pat.

3rd scan: 46 cand. 20 length-3 seq. pat. 20 cand. not in DB at all

2nd scan: 51 cand. 19 length-2 seq. pat. 10 cand. not in DB at all

1st scan: 8 cand. 6 length-1 seq. pat.



min_sup = 2

SID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

- Repeat (for each level (i.e., length-k))
  - Scan DB to find length-k frequent sequences
  - Generate length-(k+1) candidate sequences from length-k frequent sequences using Apriori
  - set k = k+1
- Until no frequent sequence or no candidate can be found

GSP: Algorithm

- **Phase 1:**
  - Scan over the database to identify all the frequent items, i.e., 1-element sequences
- **Phase 2:**
  - Iteratively scan over the database to discover all frequent sequences. Each iteration discovers all the sequences with the same length.
  - In the iteration to generate all *k*-sequences
    - Generate the set of all candidate *k*-sequences, $\mathbf{C}_k$, by joining two (*k-1*)-sequences if only their first and last items are different
    - Prune the candidate sequence if any of its *k-1* contiguous subsequence is not frequent
    - Scan over the database to determine the support of the remaining candidate sequences
  - Terminate when no more frequent sequences can be found

GSP: Candidate Generation

Frequent 3-Sequences	Candidate 4-Sequences after join	Candidate 4-Sequences after pruning
$\langle (1, 2)\ (3) \rangle$	$\langle (1, 2)\ (3, 4) \rangle$	$\langle (1, 2)\ (3, 4) \rangle$
$\langle (1, 2)\ (4) \rangle$	$\langle (1, 2)\ (3)\ (5) \rangle$	
$\langle (1)\ (3, 4) \rangle$		
$\langle (1, 3)\ (5) \rangle$		
$\langle (2)\ (3, 4) \rangle$		
$\langle (2)\ (3)\ (5) \rangle$		

Figure 3: Candidate Generation: Example

The sequence **< (1,2) (3) (5) >** is dropped in the pruning phase, since its contiguous subsequence **< (1) (3) (5) >** is not frequent.

GSP: Optimization Techniques

- Applied to phase 2: computation-intensive
- Technique 1: the hash-tree data structure
  - Used for counting candidates to reduce the number of candidates that need to be checked
    - Leaf: a list of sequences
    - Interior node: a hash table
- Technique 2: data-representation transformation
  - From horizontal format to vertical format

Transaction-Time	Items
10	1, 2
25	4, 6
45	3
50	1, 2
65	3
90	2, 4
95	6

$\rightarrow$

Item	Times
1	$\rightarrow 10 \rightarrow 50 \rightarrow$ NULL
2	$\rightarrow 10 \rightarrow 50 \rightarrow 90 \rightarrow$ NULL
3	$\rightarrow 45 \rightarrow 65 \rightarrow$ NULL
4	$\rightarrow 25 \rightarrow 90 \rightarrow$ NULL
5	$\rightarrow$ NULL
6	$\rightarrow 25 \rightarrow 95 \rightarrow$ NULL
7	$\rightarrow$ NULL

Backup slides

Sequential Pattern Mining in Vertical Data Format: The SPADE Algorithm

- A sequence database is mapped to: <SID, EID>
- Grow the subsequences (patterns) one item at a time by Apriori candidate generation

SID	Sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

min_sup = 2

Ref: SPADE (Sequential PAttern Discovery using Equivalent Class) [M. Zaki 2001]

SID	EID	Items
1	1	a
1	2	abc
1	3	ac
1	4	d
1	5	cf
2	1	ad
2	2	c
2	3	bc
2	4	ae
3	1	ef
3	2	ab
3	3	df
3	4	c
3	5	b
4	1	e
4	2	g
4	3	af
4	4	c
4	5	b
4	6	c

a		b		. . .
SID	EID	SID	EID	. . .
1	1	1	2	
1	2	2	3	
1	3	3	2	
2	1	3	5	
2	4	4	5	
3	2			
4	3			

ab			ba			. . .
SID	EID (a)	EID(b)	SID	EID (b)	EID(a)	. . .
1	1	2	1	2	3	
2	1	3	2	3	4	
3	2	5				
4	3	5				

aba				. . .
SID	EID (a)	EID(b)	EID(a)	. . .
1	1	2	3	
2	1	3	4	

PrefixSpan: A Pattern-Growth Approach

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

min_sup = 2

Prefix	Suffix (Projection)
<a>	<(abc)(ac)d(cf)>
<aa>	<(_bc)(ac)d(cf)>
<ab>	<(_c)(ac)d(cf)>

- Prefix and suffix
  - Given <a(abc)(ac)d(cf)>
  - Prefixes: <a>, <aa>, <a(ab)>, <a(abc)>, …
  - Suffix: Prefixes-based projection

- PrefixSpan Mining: Prefix Projections
  - Step 1: Find length-1 sequential patterns
    - <a>, <b>, <c>, <d>, <e>, <f>
  - Step 2: Divide search space and mine each projected DB
    - <a>-projected DB,
    - <b>-projected DB,
    - …
    - <f>-projected DB, …

PrefixSpan (Prefix-projected Sequential pattern mining) Pei, et al. @TKDE'04

PrefixSpan: Mining Prefix-Projected DBs

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

min_sup = 2

Length-1 sequential patterns
<a>, <b>, <c>, <d>, <e>, <f>

prefix <a>

<a>-projected DB
<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>

Length-2 sequential patterns
<aa>, <ab>, <(ab)>,
<ac>, <ad>, <af>

prefix <b>

<b>-projected DB

prefix <c>, …, <f>

…

… …

prefix <aa>

<aa>-projected DB

..

prefix <af>

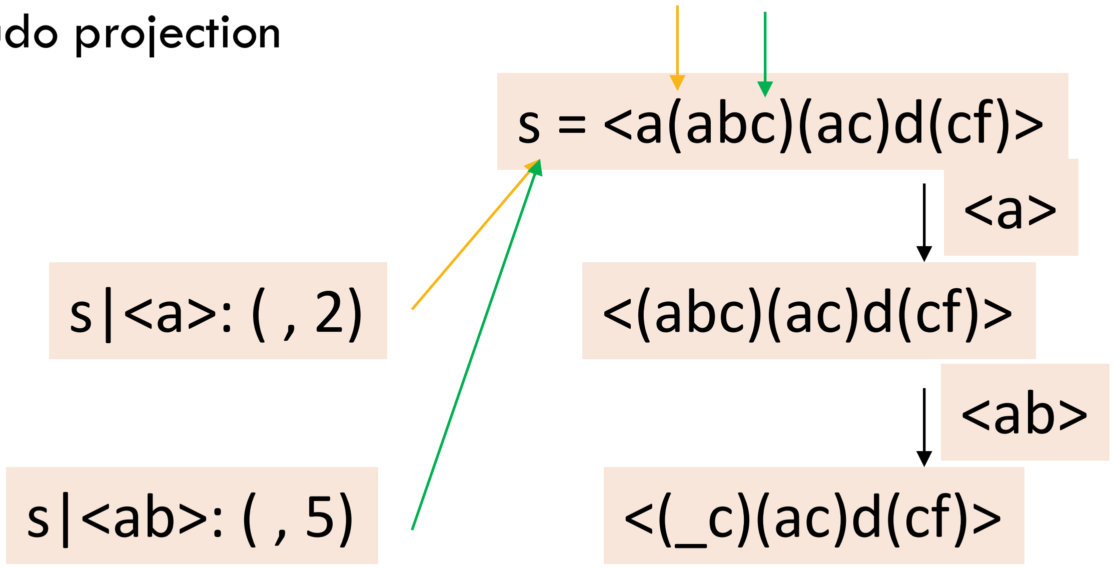
<af>-projected DB

Major strength of PrefixSpan:

- No candidate subseqs. to be generated
- Projected DBs keep shrinking

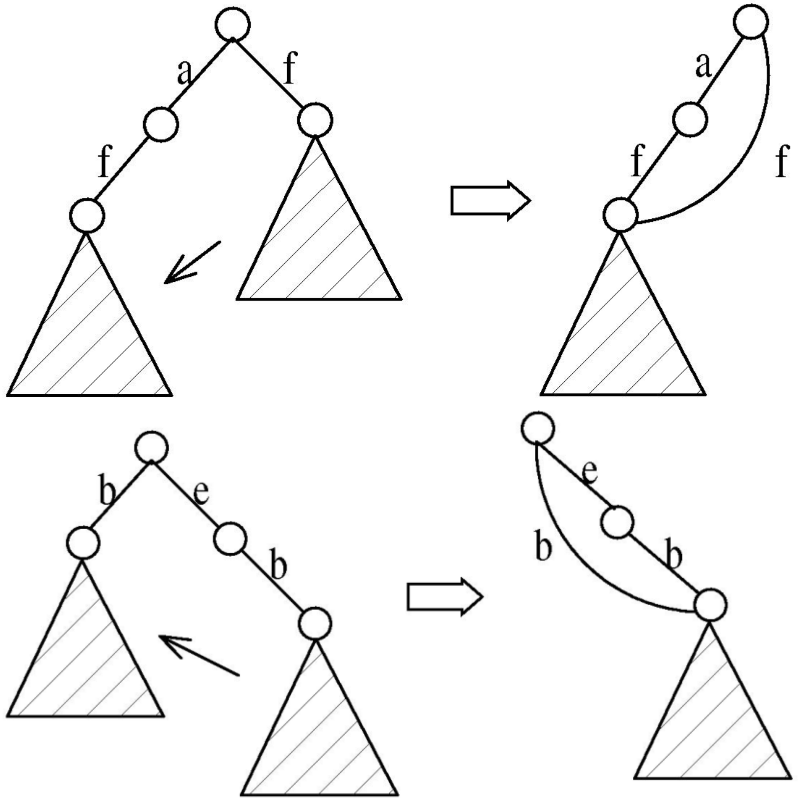
Consideration:
Pseudo-Projection vs. Physical PrImplementation ojection

- Major cost of PrefixSpan: Constructing projected DBs
  - Suffixes largely repeating in recursive projected DBs
- When DB can be held in main memory, use pseudo projection
  - No physically copying suffixes
  - Pointer to the sequence
  - Offset of the suffix
  - But if it does not fit in memory
    - Physical projection
  - Suggested approach:
    - Integration of physical and pseudo-projection
    - Swapping to pseudo-projection when the data fits in memory



CloSpan: Mining Closed Sequential Patterns

- A closed sequential pattern *s*: There exists no superpattern *s'* such that *s'* ⊃ *s*, and *s'* and *s* have the same support
- Which ones are closed? <abc>: 20, <abcd>:20, <abcde>: 15
- Why directly mine closed sequential patterns?
  - Reduce # of (redundant) patterns
  - Attain the same expressive power
- Property P_1: If $s \supset s_1$, s is closed iff two project DBs have the same size
- Explore *Backward Subpattern* and *Backward Superpattern* pruning to prune redundant search space
- Greatly enhances efficiency (Yan, et al., SDM'03)

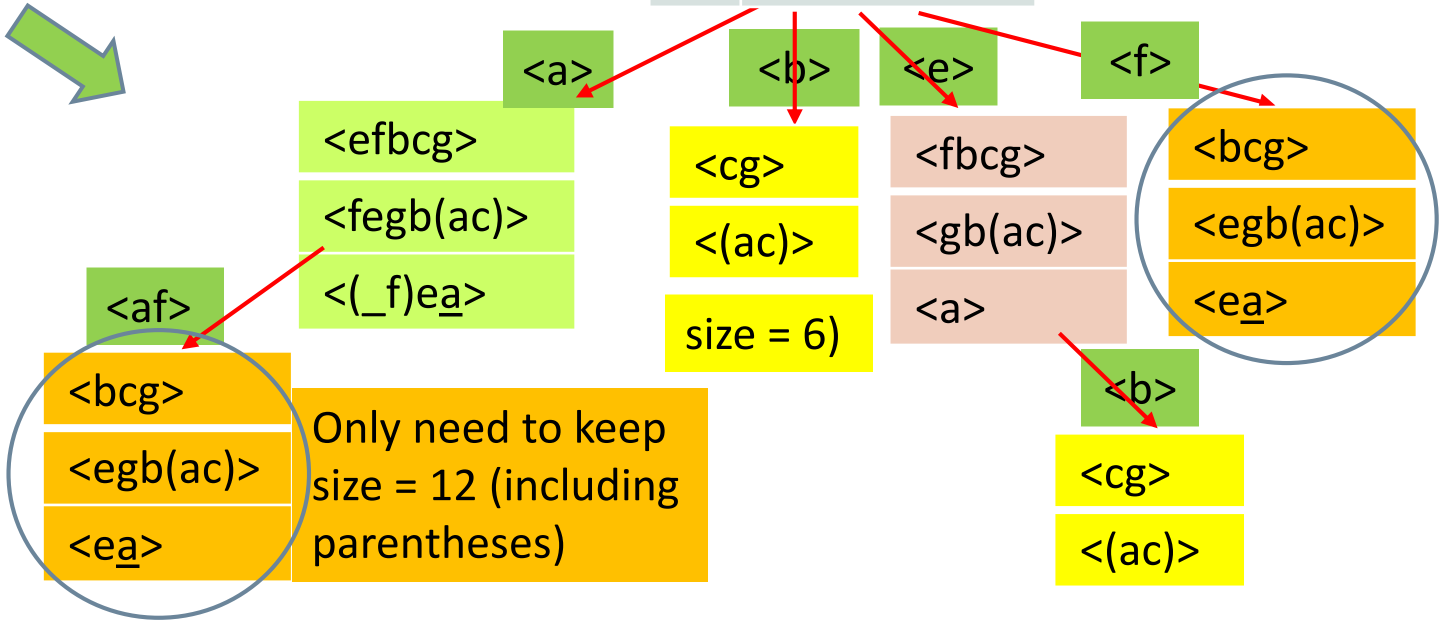


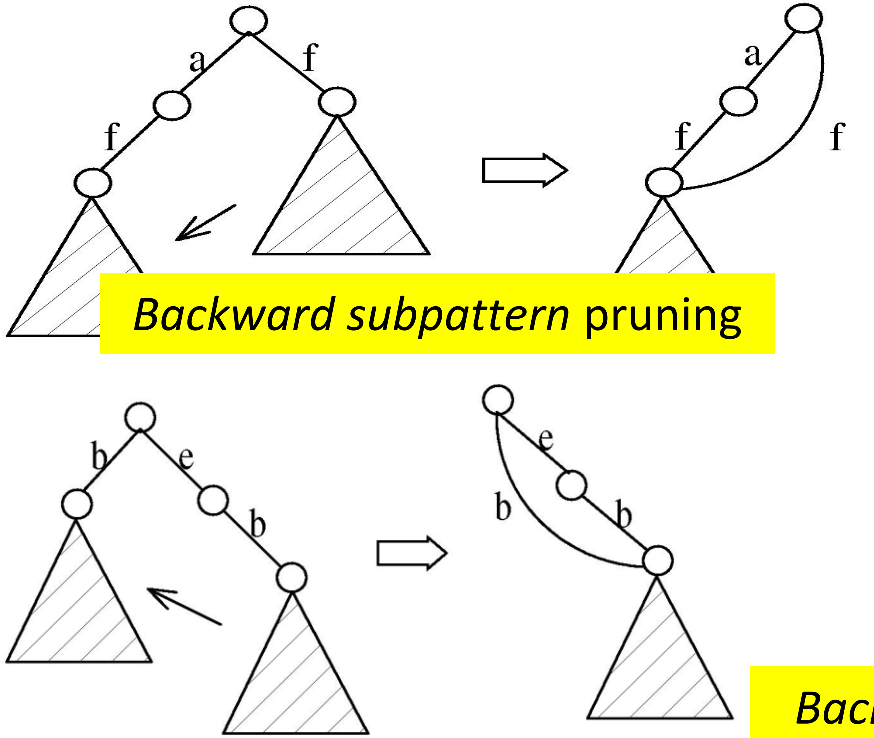
CloSpan: When Two Projected DBs Have the Same Size

- If $s \supset s_1$, s is closed iff two project DBs have the same size
  - When two projected sequence DBs have the same size?
    - Here is one example:

ID	Sequence
1	<aefbcg>
2	<afegb(ac)>
3	<(af)ea>

min_sup = 2





Backward subpattern pruning

Backward superpattern pruning

Chapter 7 : Advanced Frequent Pattern Mining

- Mining Diverse Patterns
- Sequential Pattern Mining
- Constraint-Based Frequent Pattern Mining
- Graph Pattern Mining
- Pattern Mining Application: Mining Software Copy-and-Paste Bugs
- Summary

Constraint-Based Pattern Mining

- Why Constraint-Based Mining?
- Different Kinds of Constraints: Different Pruning Strategies
- Constrained Mining with Pattern Anti-Monotonicity
- Constrained Mining with Pattern Monotonicity
- Constrained Mining with Data Anti-Monotonicity
- Constrained Mining with Succinct Constraints
- Constrained Mining with Convertible Constraints
- Handling Multiple Constraints
- Constraint-Based Sequential-Pattern Mining

Why Constraint-Based Mining?

- Finding all the patterns in a dataset autonomously?—unrealistic!
  - Too many patterns but not necessarily user-interested!
- Pattern mining in practice: Often a user-guided, interactive process
  - User directs what to be mined using a data mining query language (or a graphical user interface), specifying various kinds of constraints
- What is constraint-based mining?
  - Mine together with user-provided constraints
- Why constraint-based mining?
  - User flexibility: User provides constraints on what to be mined
  - Optimization: System explores such constraints for mining efficiency
    - E.g., Push constraints deeply into the mining process

Various Kinds of User-Specified Constraints in Data Mining

- Knowledge type constraint—Specifying what kinds of knowledge to mine
  - Ex.: Classification, association, clustering, outlier finding, …
- Data constraint—using SQL-like queries
  - Ex.: Find products sold together in NY stores this year
- Dimension/level constraint—similar to projection in relational database
  - Ex.: In relevance to region, price, brand, customer category
- Interestingness constraint—various kinds of thresholds
  - Ex.: Strong rules: min_sup $\geq$ 0.02, min_conf $\geq$ 0.6, min_correlation $\geq$ 0.7
- Rule (or pattern) constraint ← The focus of this study
  - Ex.: Small sales (price < $10) triggers big sales (sum > $200)

Pattern Space Pruning with Pattern Anti-Monotonicity

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

min_sup = 2

Item	Price	Profit
a	100	40
b	40	0
c	150	−20
d	35	−15
e	55	−30
f	45	−10
g	80	20
h	10	5

- A constraint *c* is ***anti-monotone***
  - If an itemset S **violates** constraint *c*, so does any of its superset
  - That is, mining on itemset S can be terminated
- Ex. 1: c_1: $sum(S.price) \leq v$ is anti-monotone
- Ex. 2: c_2: range(S.profit) $\leq$ 15 is anti-monotone
  - Itemset *ab* violates c_2 (range(ab) = 40)
  - So does every superset of *ab*
- Ex. 3. c_3: $sum(S.Price) \geq v$ is not anti-monotone
- Ex. 4. Is c_4: $support(S) \geq \sigma$ anti-monotone?
  - Yes! Apriori pruning is essentially pruning with an anti-monotonic constraint!

Note: item.price > 0
Profit can be negative

Pattern Monotonicity and Its Roles

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

min_sup = 2

Item	Price	Profit
a	100	40
b	40	0
c	150	−20
d	35	−15
e	55	−30
f	45	−10
g	80	20
h	10	5

- A constraint *c* is *monotone*: If an itemset S **satisfies** the constraint c, so does any of its superset
  - That is, we do not need to check *c* in subsequent mining
- Ex. 1: c_1: $sum(S.Price) \geq v$ is monotone
- Ex. 2: c_2: $min(S.Price) \leq v$ is monotone
- Ex. 3: c_3: range(S.profit) $\geq$ 15 is monotone
  - Itemset *ab* satisfies c_3
  - So does every superset of *ab*

Note: item.price > 0
Profit can be negative

Data Space Pruning with Data Anti-Monotonicity

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

min_sup = 2

Item	Price	Profit
a	100	40
b	40	0
c	150	−20
d	35	−15
e	55	−30
f	45	−10
g	80	20
h	10	5

- A constraint c is *data anti-monotone*: In the mining process, if a data entry *t* cannot satisfy a pattern *p* under c, *t* cannot satisfy *p*'s superset either
  - Data space pruning: Data entry *t* can be pruned
- Ex. 1: c_1: $sum(S.Profit) \geq v$ is data anti-monotone
  - Let constraint c_1 be: $sum(S.Profit) \geq 25$
    - T_{30}: {b, c, d, f, g} can be removed since none of their combinations can make an S whose sum of the profit is ≥ 25
- Ex. 2: c_2: $min(S.Price) \leq v$ is data anti-monotone
  - Consider $v = 5$ but every item in a transaction, say T_{50}, has a price higher than 10
- Ex. 3: c_3: $range(S.Profit) > 25$ is data anti-monotone

Note: item.price > 0
Profit can be negative

Expressing Patterns in Compressed Form: Closed Patterns

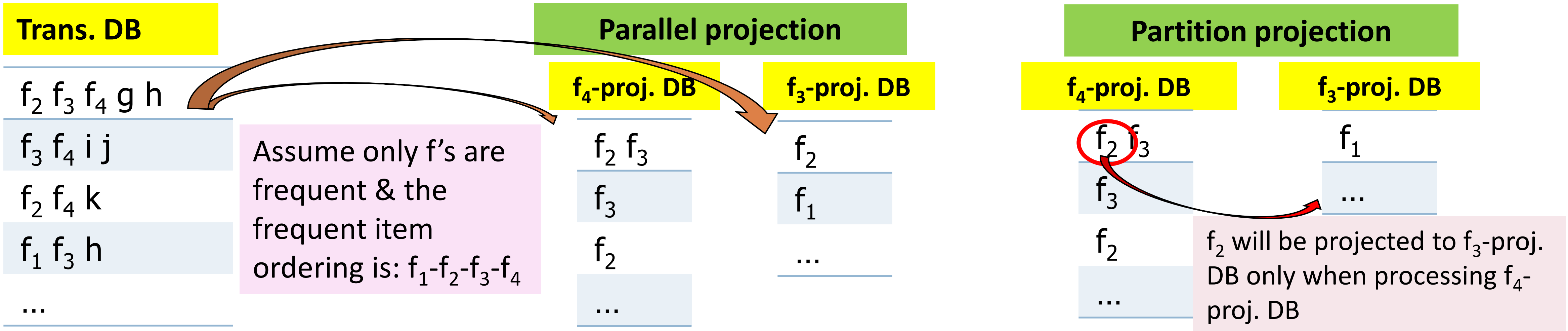
- How to handle such a challenge?
- Solution 1: **Closed patterns:** A pattern (itemset) X is closed if X is *frequent*, and there exists *no super-pattern* Y ↄ X, *with the same support* as X
  - Let Transaction DB TDB_1: T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - Suppose *minsup* = 1. How many closed patterns does TDB_1 contain?
    - Two: P_1: "$\{a_1, \ldots, a_{50}\}$: 2"; P_2: "$\{a_1, \ldots, a_{100}\}$: 1"
- Closed pattern is a lossless compression of frequent patterns
  - Reduces the # of patterns but does not lose the support information!
  - You will still be able to say: "$\{a_2, \ldots, a_{40}\}$: 2", "$\{a_5, a_{51}\}$: 1"

Expressing Patterns in Compressed Form: Max-Patterns

- Solution 2: **Max-patterns:** A pattern X is a maximal frequent pattern or max-pattern if X is frequent and there exists no frequent super-pattern Y ⊃ X
- Difference from close-patterns?
  - Do not care the real support of the sub-patterns of a max-pattern
  - Let Transaction DB TDB_1: T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - Suppose *minsup* = 1. How many max-patterns does TDB_1 contain?
    - One: P: "$\{a_1, \ldots, a_{100}\}$: 1"
- Max-pattern is a lossy compression!
  - We only know $\{a_1, \ldots, a_{40}\}$ is frequent
  - But we do not know the real support of $\{a_1, \ldots, a_{40}\}$, …, any more!
  - Thus in many applications, close-patterns are more desirable than max-patterns

Scaling FP-growth by Item-Based Data Projection

- What if FP-tree cannot fit in memory?—Do not construct FP-tree
  - "Project" the database based on frequent single items
  - Construct & mine FP-tree for each projected DB
- Parallel projection vs. partition projection
  - Parallel projection: Project the DB on each frequent item
    - Space costly, all partitions can be processed in parallel
  - Partition projection: Partition the DB in order
    - Passing the unprocessed parts to subsequent partitions



Analysis of DBLP Coauthor Relationships

- DBLP: Computer science research publication bibliographic database
  - > 3.8 million entries on authors, paper, venue, year, and other information

ID	Author A	Author B	$s(A \cup B)$	$s(A)$	$s(B)$	Jaccard	*Cosine*	*Kulc*
1	Hans-Peter Kriegel	Martin Ester	28	146	54	0.163 (2)	0.315 (7)	0.355 (9)
2	Michael Carey	Miron Livny	26	104	58	0.191 (1)	0.335 (4)	0.349 (10)
3	Hans-Peter Kriegel	Joerg Sander	24	146	36	0.152 (3)	0.331 (5)	0.416 (8)
4	Christos Faloutsos	Spiros Papadimitriou	20	162	26	0.119 (7)	0.308 (10)	0.446 (7)
5	Hans-Peter Kriegel	Martin Pfeifle	18	146	18	0.123 (6)	0.351 (2)	0.562 (2)
6	Hector Garcia-Molina	Wilburt Labio	16	144	18	0.110 (9)	0.314 (8)	0.500 (4)
7	Divyakant Agrawal	Wang Hsiung	16	120	16	0.133 (5)	0.365 (1)	0.567 (1)
8	Elke Rundensteiner	Murali Mani	16	104	20	0.148 (4)	0.351 (3)	0.477 (6)
9	Divyakant Agrawal	Oliver Po	12	120	12	0.100 (10)	0.316 (6)	0.550 (3)
10	Gerhard Weikum	Martin Theobald	12	106	14	0.111 (8)	0.312 (9)	0.485 (5)

Advisor-advisee relation: Kulc: high, Jaccard: low, cosine: middle

- Which pairs of authors are strongly related?
  - Use Kulc to find Advisor-advisee, close collaborators

Analysis of DBLP Coauthor Relationships

- DBLP: Computer science research publication bibliographic database
  - > 3.8 million entries on authors, paper, venue, year, and other information

ID	Author A	Author B	$s(A \cup B)$	$s(A)$	$s(B)$	Jaccard	*Cosine*	*Kulc*
1	Hans-Peter Kriegel	Martin Ester	28	146	54	0.163 (2)	0.315 (7)	0.355 (9)
2	Michael Carey	Miron Livny	26	104	58	0.191 (1)	0.335 (4)	0.349 (10)
3	Hans-Peter Kriegel	Joerg Sander	24	146	36	0.152 (3)	0.331 (5)	0.416 (8)
4	Christos Faloutsos	Spiros Papadimitriou	20	162	26	0.119 (7)	0.308 (10)	0.446 (7)
5	Hans-Peter Kriegel	Martin Pfeifle	18	146	18	0.123 (6)	0.351 (2)	0.562 (2)
6	Hector Garcia-Molina	Wilburt Labio	16	144	18	0.110 (9)	0.314 (8)	0.500 (4)
7	Divyakant Agrawal	Wang Hsiung	16	120	16	0.133 (5)	0.365 (1)	0.567 (1)
8	Elke Rundensteiner	Murali Mani	16	104	20	0.148 (4)	0.351 (3)	0.477 (6)
9	Divyakant Agrawal	Oliver Po	12	120	12	0.100 (10)	0.316 (6)	0.550 (3)
10	Gerhard Weikum	Martin Theobald	12	106	14	0.111 (8)	0.312 (9)	0.485 (5)

Advisor-advisee relation: Kulc: high, Jaccard: low, cosine: middle

- Which pairs of authors are strongly related?
  - Use Kulc to find Advisor-advisee, close collaborators

What Measures to Choose for Effective Pattern Evaluation?

- Null value cases are predominant in many large datasets
  - Neither milk nor coffee is in most of the baskets; neither Mike nor Jim is an author in most of the papers; ……
- *Null-invariance* is an important property
- Lift, χ^2 and cosine are good measures if null transactions are not predominant
  - Otherwise, *Kulczynski* + *Imbalance Ratio* should be used to judge the interestingness of a pattern
- Exercise: Mining research collaborations from research bibliographic data
  - Find a group of frequent collaborators from research bibliographic data (e.g., DBLP)
  - Can you find the likely advisor-advisee relationship and during which years such a relationship happened?
  - Ref.: C. Wang, J. Han, Y. Jia, J. Tang, D. Zhang, Y. Yu, and J. Guo, "Mining Advisor-Advisee Relationships from Research Publication Networks", KDD'10

Mining Compressed Patterns

Pat-ID	Item-Sets	Support
P1	{38,16,18,12}	205227
P2	{38,16,18,12,17}	205211
P3	{39,38,16,18,12,17}	101758
P4	{39,16,18,12,17}	161563
P5	{39,16,18,12}	161576

- Closed patterns
  - P1, P2, P3, P4, P5
  - Emphasizes too much on support
  - There is no compression
- Max-patterns
  - P3: information loss
- Desired output (a good balance):
  - P2, P3, P4

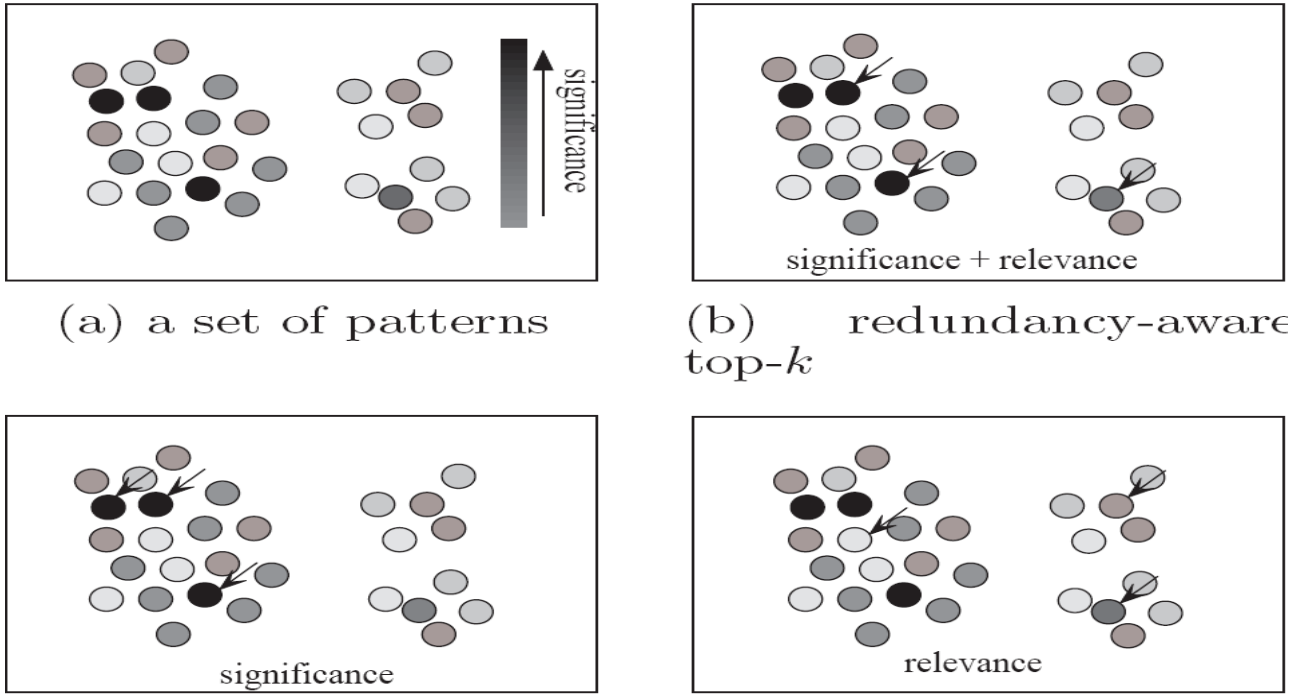
- Why mining compressed patterns?
  - Too many scattered patterns but not so meaningful
- Pattern distance measure

$$Dist(P_1, P_2) = 1 - \frac{|T(P_1) \cap T(P_2)|}{|T(P_1) \cup T(P_2)|}$$

- δ-clustering: For each pattern P, find all patterns which can be expressed by P and whose distance to P is within δ (δ-cover)
- All patterns in the cluster can be represented by P
- Method for efficient, direct mining of compressed frequent patterns (e.g., D. Xin, J. Han, X. Yan, H. Cheng, "On Compressing Frequent Patterns", Knowledge and Data Engineering, 60:5-29, 2007)

Redundancy-Aware Top-k Patterns

- Desired patterns: high significance & low redundancy



(a) a set of patterns (b) redundancy-aware top-k

(c) traditional top-k (d) summarization

- Method: Use MMS (Maximal Marginal Significance) for measuring the combined significance of a pattern set
- Xin et al., Extracting Redundancy-Aware Top-K Patterns, KDD'06

Redundancy Filtering at Mining Multi-Level Associations

- Multi-level association mining may generate many redundant rules
- Redundancy filtering: Some rules may be redundant due to "ancestor" relationships between items
  - milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%] (1)
  - 2% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%] (2)
    - Suppose the "2% milk" sold is about "¼" of milk sold
      - Does (2) provide any novel information?
- A rule is *redundant* if its support is close to the "expected" value, according to its "ancestor" rule, and it has a similar confidence as its "ancestor"
  - Rule (1) is an ancestor of rule (2), which one to prune?

Succinctness

- Succinctness:
  - Given $A_{1,}$ the set of items satisfying a succinctness constraint C, then any set S satisfying C is based on A_1 , i.e., S contains a subset belonging to A_1
  - Idea: Without looking at the transaction database, whether an itemset S satisfies constraint C can be determined based on the selection of items
  - $min(S.Price) \leq v$ is succinct
  - $sum(S.Price) \geq v$ is not succinct
- Optimization: If C is succinct, C is pre-counting pushable

Which Constraints Are Succinct?

Constraint	Succinct
$v \in S$	yes
$S \supseteq V$	yes
$S \subseteq V$	yes
$\min(S) \leq v$	yes
$\min(S) \geq v$	yes
$\max(S) \leq v$	yes
$\max(S) \geq v$	yes
$\text{sum}(S) \leq v$ ($a \in S, a \geq 0$)	no
$\text{sum}(S) \geq v$ ($a \in S, a \geq 0$)	no
$\text{range}(S) \leq v$	no
$\text{range}(S) \geq v$	no
$\text{avg}(S)\ \theta\ v, \theta \in \{=, \leq, \geq\}$	no
$\text{support}(S) \geq \xi$	no
$\text{support}(S) \leq \xi$	no

Push a Succinct Constraint Deep

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
~~{2 3}~~
~~{2 5}~~
~~{3 5}~~

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
~~{2 3 5}~~

Scan D

L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

min{S.price <= 1 }

Sequential Pattern Mining

- Sequential Pattern and Sequential Pattern Mining
- GSP: Apriori-Based Sequential Pattern Mining
- SPADE: Sequential Pattern Mining in Vertical Data Format
- PrefixSpan: Sequential Pattern Mining by Pattern-Growth
- CloSpan: Mining Closed Sequential Patterns