Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Pattern Mining in Multi-Level, Multi-Dimensional Space
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Mining Compressed or Approximate Patterns
- Sequential Pattern Mining
- Graph Pattern Mining
- Summary
Research on Pattern Mining: A Road Map

Kinds of patterns and rules

- Basic Patterns
  - frequent pattern
  - association rule
  - closed/max pattern
  - generator
- Multilevel & Multidimensional Patterns
  - multilevel (uniform, varied, or itemset-based support)
  - multidimensional pattern (incl. high-dimensional pattern)
  - continuous data (discretization-based, or statistical)
- Extended Patterns
  - approximate pattern
  - uncertain pattern
  - compressed pattern
  - rare pattern/negative pattern
  - high-dimensional and colossal patterns

Basic Mining Methods

- candidate generation (Apriori, partitioning, sampling, ...)
- Pattern growth (FPGrowth, HMine, FPMax, Closet+, ...)
- vertical format (EClat, CHARM, ...)

Mining Interesting Patterns

- interestingness (subjective vs. objective)
- constraint-based mining
- correlation rules
- exception rules

Distributed, parallel & incremental

- distributed/parallel mining
- incremental mining
- stream pattern

Extensions & Application

- sequential ad time-series patterns
- structural (e.g., tree, lattice, graph) patterns
- spatial (e.g., co-location) pattern
- temporal (evolutionary, periodic)
- image, video and multimedia patterns
- network patterns

Applications

- pattern-based classification
- pattern-based clustering
- pattern-based semantic annotation
- collaborative filtering
- privacy-preserving
Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Pattern Mining in Multi-Level, Multi-Dimensional Space
  - Mining Multi-Level Association
  - Mining Multi-Dimensional Association
  - Mining Quantitative Association Rules
  - Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Mining Compressed or Approximate Patterns
- Sequential Pattern Mining
- Graph Pattern Mining
- Summary
Mining Multiple-Level Association Rules

- Items often form hierarchies
- Flexible support settings
  - Items at the lower level are expected to have lower support
- Exploration of *shared* multi-level mining (Agrawal & Srikant@VLB’95, Han & Fu@VLDB’95)

**uniform support**

Level 1
min_sup = 5%

Milk
[support = 10%]

Level 2
min_sup = 5%

2% Milk
[support = 6%]

Skim Milk
[support = 4%]

**reduced support**

Level 1
min_sup = 5%

Level 2
min_sup = 3%
Multi-level Association: Flexible Support and Redundancy filtering

- Flexible min-support thresholds: Some items are more valuable but less frequent
  - Use non-uniform, group-based min-support
  - E.g., \{diamond, watch, camera\}: 0.05%; \{bread, milk\}: 5%; ...

- Redundancy Filtering: Some rules may be redundant due to “ancestor” relationships between items
  - milk → wheat bread [support = 8%, confidence = 70%]
  - 2% milk → wheat bread [support = 2%, confidence = 72%]

  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
Chapter 7 : Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Pattern Mining in Multi-Level, Multi-Dimensional Space
  - Mining Multi-Level Association
  - Mining Multi-Dimensional Association
  - Mining Quantitative Association Rules
  - Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Mining Compressed or Approximate Patterns
- Sequential Pattern Mining
- Graph Pattern Mining
- Summary
Mining Multi-Dimensional Association

- Single-dimensional rules:
  \[ \text{buys}(X, \text{“milk”}) \Rightarrow \text{buys}(X, \text{“bread”}) \]

- Multi-dimensional rules: \( \geq 2 \) dimensions or predicates
  - Inter-dimension assoc. rules (\textit{no repeated predicates})
    \[ \text{age}(X,\text{“19-25”}) \land \text{occupation}(X,\text{“student”}) \Rightarrow \text{buys}(X, \text{“coke”}) \]
  - hybrid-dimension assoc. rules (\textit{repeated predicates})
    \[ \text{age}(X,\text{“19-25”}) \land \text{buys}(X, \text{“popcorn”}) \Rightarrow \text{buys}(X, \text{“coke”}) \]

- Categorical Attributes: finite number of possible values, no ordering among values—data cube approach

- Quantitative Attributes: Numeric, implicit ordering among values—discretization, clustering, and gradient approaches
Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Pattern Mining in Multi-Level, Multi-Dimensional Space
  - Mining Multi-Level Association
  - Mining Multi-Dimensional Association
  - Mining Quantitative Association Rules
  - Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Mining Compressed or Approximate Patterns
- Sequential Pattern Mining
- Graph Pattern Mining
- Summary
Mining Quantitative Associations

Techniques can be categorized by how numerical attributes, such as age or salary are treated:

1. Static discretization based on predefined concept hierarchies (data cube methods)
2. Dynamic discretization based on data distribution (quantitative rules, e.g., Agrawal & Srikant@SIGMOD96)
3. Clustering: Distance-based association (e.g., Yang & Miller@SIGMOD97)
   □ One dimensional clustering then association
4. Deviation: (such as Aumann and Lindell@KDD99)
   Sex = female  =>  Wage: mean=$7/hr (overall mean = $9)
Static Discretization of Quantitative Attributes

- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges
- In relational database, finding all frequent k-predicate sets will require \( k \) or \( k+1 \) table scans
- Data cube is well suited for mining
- The cells of an n-dimensional cuboid correspond to the predicate sets
- Mining from data cubes can be much faster
Quantitative Association Rules Based on Statistical Inference Theory [Aumann and Lindell@DMKD’03]

- Finding extraordinary and therefore interesting phenomena, e.g.,
  \[(\text{Sex} = \text{female}) \Rightarrow \text{Wage: mean} = \$7/\text{hr} \ (\text{overall mean} = \$9)\]
  - LHS: a subset of the population
  - RHS: an extraordinary behavior of this subset
- The rule is accepted only if a statistical test (e.g., Z-test) confirms the inference with high confidence
- Subrule: highlights the extraordinary behavior of a subset of the pop. of the super rule
  - E.g., \((\text{Sex} = \text{female}) \land (\text{South} = \text{yes}) \Rightarrow \text{mean wage} = \$6.3/\text{hr}\)
- Two forms of rules
  - Categorical \Rightarrow\text{ quantitative rules, or Quantitative} \Rightarrow\text{ quantitative rules}
  - E.g., Education in \([14-18]\) (yrs) \Rightarrow \text{mean wage} = \$11.64/\text{hr}
- Open problem: Efficient methods for LHS containing two or more quantitative attributes
Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Pattern Mining in Multi-Level, Multi-Dimensional Space
  - Mining Multi-Level Association
  - Mining Multi-Dimensional Association
  - Mining Quantitative Association Rules
  - Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Mining Compressed or Approximate Patterns
- Sequential Pattern Mining
- Graph Pattern Mining
- Summary
Negative and Rare Patterns

- Rare patterns: Very low support but interesting
  - E.g., buying Rolex watches
  - Mining: Setting individual-based or special group-based support threshold for valuable items

- Negative patterns
  - Since it is unlikely that one buys Ford Expedition (an SUV car) and Toyota Prius (a hybrid car) together, Ford Expedition and Toyota Prius are likely negatively correlated patterns
  - Negatively correlated patterns that are infrequent tend to be more interesting than those that are frequent
Defining Negative Correlated Patterns (I)

- Definition 1 (support-based)
  - If itemsets X and Y are both frequent but rarely occur together, i.e.,
    \[ \text{sup}(X \cup Y) < \text{sup}(X) \times \text{sup}(Y) \]
  - Then X and Y are negatively correlated

- Problem: A store sold two needle 100 packages A and B, only one transaction containing both A and B.
  - When there are in total 200 transactions, we have
    \[ \text{s}(A \cup B) = 0.005, \text{s}(A) \times \text{s}(B) = 0.25, \text{s}(A \cup B) < \text{s}(A) \times \text{s}(B) \]
  - When there are \(10^5\) transactions, we have
    \[ \text{s}(A \cup B) = 1/10^5, \text{s}(A) \times \text{s}(B) = 1/10^3 \times 1/10^3, \text{s}(A \cup B) > \text{s}(A) \times \text{s}(B) \]
  - Where is the problem? —Null transactions, i.e., the support-based definition is not null-invariant!
Defining Negative Correlated Patterns (II)

- Definition 2 (negative itemset-based)
  - X is a *negative itemset* if (1) $X = \bar{A} \cup B$, where B is a set of positive items, and $\bar{A}$ is a set of negative items, $|\bar{A}| \geq 1$, and (2) $s(X) \geq \mu$
  - Itemsets X is negatively correlated, if
    
    $$s(X) < \prod_{i=1}^{k} s(x_i), \text{where } x_i \in X, \text{ and } s(x_i) \text{ is the support of } x_i$$

- This definition suffers a similar null-invariant problem

- Definition 3 (Kulzynski measure-based) If itemsets X and Y are frequent, but $(P(X|Y) + P(Y|X))/2 < \epsilon$, where $\epsilon$ is a negative pattern threshold, then X and Y are negatively correlated.

- Ex. For the same needle package problem, when no matter there are 200 or $10^5$ transactions, if $\epsilon = 0.05$, we have
  
  $$(P(A|B) + P(B|A))/2 = (0.01 + 0.01)/2 < \epsilon$$
Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Pattern Mining in Multi-Level, Multi-Dimensional Space
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Mining Compressed or Approximate Patterns
- Sequential Pattern Mining
- Graph Pattern Mining
- Summary
Constraint-based (Query-Directed) 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
  - Optimization: explores such constraints for efficient mining — constraint-based mining: constraint-pushing, similar to push selection first in DB query processing
  - Note: still find all the answers satisfying constraints, not finding some answers in “heuristic search”
Constraints in Data Mining

- Knowledge type constraint:
  - classification, association, etc.

- Data constraint — using SQL-like queries
  - find product pairs sold together in stores in Chicago this year

- Dimension/level constraint
  - in relevance to region, price, brand, customer category

- Rule (or pattern) constraint
  - small sales (price < $10) triggers big sales (sum > $200)

- Interestingness constraint
  - strong rules: min_support ≥ 3%, min_confidence ≥ 60%
Meta-Rule Guided Mining

- Meta-rule can be in the rule form with partially instantiated predicates and constants
  \[ P_1(X, Y) \land P_2(X, W) \Rightarrow \text{buys}(X, "iPad") \]

- The resulting rule derived can be
  \[ \text{age}(X, "15-25") \land \text{profession}(X, "student") \Rightarrow \text{buys}(X, "iPad") \]

- In general, it can be in the form of
  \[ P_1 \land P_2 \land \ldots \land P_l \Rightarrow Q_1 \land Q_2 \land \ldots \land Q_r \]

- Method to find meta-rules
  - Find frequent \((l+r)\) predicates (based on min-support threshold)
  - Push constants deeply when possible into the mining process (see the remaining discussions on constraint-push techniques)
  - Use confidence, correlation, and other measures when possible
Constraint-Based Frequent Pattern Mining

- Pattern space pruning constraints
  - **Anti-monotonic**: If constraint c is violated, its further mining can be terminated
  - **Monotonic**: If c is satisfied, no need to check c again
  - **Succinct**: c must be satisfied, so one can start with the data sets satisfying c
  - **Convertible**: c is not monotonic nor anti-monotonic, but it can be converted into it if items in the transaction can be properly ordered

- Data space pruning constraint
  - **Data succinct**: Data space can be pruned at the initial pattern mining process
  - **Data anti-monotonic**: If a transaction t does not satisfy c, t can be pruned from its further mining
Pattern Space Pruning with Anti-Monotonicity Constraints

- A constraint \( C \) is **anti-monotone** if the super pattern satisfies \( C \), all of its sub-patterns do so too.
- In other words, **anti-monotonicity**: If an itemset \( S \) violates the constraint, so does any of its superset.
- Ex. 1. \( \text{sum}(S\text{.price}) \leq v \) is anti-monotone
- Ex. 2. \( \text{range}(S\text{.profit}) \leq 15 \) is anti-monotone
  - Itemset \( ab \) violates \( C \)
  - So does every superset of \( ab \)
- Ex. 3. \( \text{sum}(S\text{.Price}) \geq v \) is **not** anti-monotone
- Ex. 4. **support count** is anti-monotone: core property used in Apriori

### TDB (min_sup=2)

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, b, c, d, f</td>
</tr>
<tr>
<td>20</td>
<td>b, c, d, f, g, h</td>
</tr>
<tr>
<td>30</td>
<td>a, c, d, e, f</td>
</tr>
<tr>
<td>40</td>
<td>c, e, f, g</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>-10</td>
</tr>
</tbody>
</table>
Pattern Space Pruning with Monotonicity Constraints

- A constraint C is *monotone* if the pattern satisfies C, we do not need to check C in subsequent mining.

- Alternatively, monotonicity: *If an itemset S satisfies the constraint, so does any of its superset*.

- Ex. 1. \( \text{sum}(S.Price) \geq v \) is monotone.

- Ex. 2. \( \text{min}(S.Price) \leq v \) is monotone.

- Ex. 3. C: range(S.profit) \( \geq 15 \)
  - Itemset \( ab \) satisfies C
  - So does every superset of \( ab \)

TDB (min_sup=2)

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, b, c, d, f</td>
</tr>
<tr>
<td>20</td>
<td>b, c, d, f, g, h</td>
</tr>
<tr>
<td>30</td>
<td>a, c, d, e, f</td>
</tr>
<tr>
<td>40</td>
<td>c, e, f, g</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>-10</td>
</tr>
</tbody>
</table>
Data Space Pruning with Data Anti-monotonicity

- A constraint $c$ is data anti-monotone if for a pattern $p$ cannot satisfy a transaction $t$ under $c$, $p$’s superset cannot satisfy $t$ under $c$ either.
- The key for data anti-monotone is recursive data reduction.
- Ex. 1. $\text{sum}(S.Price) \geq v$ is data anti-monotone.
- Ex. 2. $\text{min}(S.Price) \leq v$ is data anti-monotone.
- Ex. 3. C: $\text{range}(S.profit) \geq 25$ is data anti-monotone.
  - Itemset $\{b, c\}$’s projected DB:
    - $T10'$: $\{d, f, h\}$, $T20'$: $\{d, f, g, h\}$, $T30'$: $\{d, f, g\}$
    - since C cannot satisfy $T10'$, $T10'$ can be pruned.

TDB (min_sup=2)

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, b, c, d, f, h</td>
</tr>
<tr>
<td>20</td>
<td>b, c, d, f, g, h</td>
</tr>
<tr>
<td>30</td>
<td>b, c, d, f, g</td>
</tr>
<tr>
<td>40</td>
<td>c, e, f, g</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>-15</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>-10</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>-5</td>
</tr>
</tbody>
</table>
Pattern Space Pruning with 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.
  
- $\min(S.\text{Price}) \leq \nu$ is succinct
  
- $\sum(S.\text{Price}) \geq \nu$ is not succinct

- Optimization: If $C$ is succinct, $C$ is pre-counting pushable.
Apriori + Constraint

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
</tr>
</tbody>
</table>

Scan D

Scan D

Scan D

C1

<table>
<thead>
<tr>
<th>itemset</th>
<th>sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{4}</td>
<td>1</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

Scan D

Scan D

L1

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

C2

<table>
<thead>
<tr>
<th>itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 2}</td>
<td>1</td>
</tr>
<tr>
<td>{1 3}</td>
<td>2</td>
</tr>
<tr>
<td>{1 5}</td>
<td>1</td>
</tr>
<tr>
<td>{2 3}</td>
<td>2</td>
</tr>
<tr>
<td>{2 5}</td>
<td>3</td>
</tr>
<tr>
<td>{3 5}</td>
<td>2</td>
</tr>
</tbody>
</table>

C2

<table>
<thead>
<tr>
<th>itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 2}</td>
</tr>
<tr>
<td>{1 3}</td>
</tr>
<tr>
<td>{1 5}</td>
</tr>
<tr>
<td>{2 3}</td>
</tr>
<tr>
<td>{2 5}</td>
</tr>
<tr>
<td>{3 5}</td>
</tr>
</tbody>
</table>

C3

<table>
<thead>
<tr>
<th>itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{2 3 5}</td>
</tr>
</tbody>
</table>

L3

itemset | sup  |
--------|------|
{2 3 5} | 2    |

Constraint:
Sum{S.price} < 5
Constrained Apriori: Push a Succinct Constraint Deep

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
</tr>
</tbody>
</table>

Scan D

\[
C_1 \Rightarrow L_1
\]

\[
\begin{array}{|c|c|}
\hline
\text{itemset} & \text{sup.} \\
\hline
\{1\} & 2 \\
\{2\} & 3 \\
\{3\} & 3 \\
\{4\} & 1 \\
\{5\} & 3 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{itemset} & \text{sup.} \\
\hline
\{1\} & 2 \\
\{2\} & 3 \\
\{3\} & 3 \\
\{3\} & 3 \\
\{5\} & 3 \\
\hline
\end{array}
\]

Scan D

\[
L_2 \Rightarrow L_3
\]

\[
\begin{array}{|c|c|}
\hline
\text{itemset} & \text{sup.} \\
\hline
\{1 2\} & 1 \\
\{1 3\} & 2 \\
\{1 5\} & 1 \\
\{2 3\} & 2 \\
\{2 5\} & 3 \\
\{3 5\} & 2 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{itemset} & \text{sup.} \\
\hline
\{1 2\} & 1 \\
\{1 3\} & 2 \\
\{1 5\} & 3 \\
\{2 3\} & 2 \\
\{2 5\} & 3 \\
\{3 5\} & 2 \\
\hline
\end{array}
\]

Scan D

\[
C_2 \Rightarrow C_3
\]

Constraint:
\[\min\{S.\text{price}\} \leq 1\]
Constrained FP-Growth: Push a Succinct Constraint Deep

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
</tr>
</tbody>
</table>

Remove infrequent length 1

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
</tr>
</tbody>
</table>

1-Projected DB

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>3 4</td>
</tr>
<tr>
<td>300</td>
<td>2 3 5</td>
</tr>
</tbody>
</table>

No Need to project on 2, 3, or 5

Constraint:

\[
\text{min\{S.price\} } \leq 1
\]
Constrained FP-Growth: Push a Data Anti-monotonic Constraint Deep

Remove from data

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3</td>
</tr>
<tr>
<td>300</td>
<td>1 3</td>
</tr>
</tbody>
</table>

Constraint:
\[ \min\{S.\text{price} \} \leq 1 \]

Single branch, we are done
Constrained FP-Growth: Push a Data Anti-monotonic Constraint Deep

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, b, c, d, f, h</td>
</tr>
<tr>
<td>20</td>
<td>b, c, d, f, g, h</td>
</tr>
<tr>
<td>30</td>
<td>b, c, d, f, g</td>
</tr>
<tr>
<td>40</td>
<td>a, c, e, f, g</td>
</tr>
</tbody>
</table>

B-Projected DB

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, c, d, f, h</td>
</tr>
<tr>
<td>20</td>
<td>c, d, f, g, h</td>
</tr>
<tr>
<td>30</td>
<td>c, d, f, g</td>
</tr>
</tbody>
</table>

Item | Profit |
-----|--------|
| a   | 40     |
| b   | 0      |
| c   | -20    |
| d   | -15    |
| e   | -30    |
| f   | -10    |
| g   | 20     |
| h   | -5     |

Constraint:
range{S.price } > 25
min_sup >= 2
Convertible Constraints: Ordering Data in Transactions

- Convert tough constraints into anti-monotone or monotone by properly ordering items.
- Examine C: \( \text{avg}(S.\text{profit}) \geq 25 \)
  - 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!

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, b, c, d, f</td>
</tr>
<tr>
<td>20</td>
<td>b, c, d, f, g, h</td>
</tr>
<tr>
<td>30</td>
<td>a, c, d, e, f</td>
</tr>
<tr>
<td>40</td>
<td>c, e, f, g</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>-10</td>
</tr>
</tbody>
</table>
Strongly Convertible Constraints

- \( \text{avg}(X) \geq 25 \) 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 \( afd \)
- \( \text{avg}(X) \geq 25 \) 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, \( \text{avg}(X) \geq 25 \) is strongly convertible

<table>
<thead>
<tr>
<th>Item</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>-10</td>
</tr>
</tbody>
</table>
Can Apriori Handle Convertible Constraints?

- A convertible, not monotone nor anti-monotone nor succinct constraint 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: \( \text{avg}(X) \geq 25 \)
  - Since adf satisfies C, Apriori needs df to assemble adf, df cannot be pruned
- But it can be pushed into frequent-pattern growth framework!

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>-10</td>
</tr>
</tbody>
</table>
Pattern Space Pruning w. Convertible Constraints

- C: \( \text{avg}(X) \geq 25, \text{min\_sup}=2 \)
- List items in every transaction in value descending order \( R: \langle a, f, g, d, b, h, c, e \rangle \)
  - 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

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>f</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>h</td>
<td>-10</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
</tbody>
</table>

TDB (min\_sup=2)

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, f, d, b, c</td>
</tr>
<tr>
<td>20</td>
<td>f, g, d, b, c</td>
</tr>
<tr>
<td>30</td>
<td>a, f, d, c, e</td>
</tr>
<tr>
<td>40</td>
<td>f, g, h, c, e</td>
</tr>
</tbody>
</table>
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.
### Constraint-Based Mining — A General Picture

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Anti-monotone</th>
<th>Monotone</th>
<th>Succinct</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v \in S )</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>( S \supseteq V )</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>( S \subseteq V )</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>( \text{min}(S) \leq v )</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>( \text{min}(S) \geq v )</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>( \text{max}(S) \leq v )</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>( \text{max}(S) \geq v )</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>( \text{count}(S) \leq v )</td>
<td>yes</td>
<td>no</td>
<td>weakly</td>
</tr>
<tr>
<td>( \text{count}(S) \geq v )</td>
<td>no</td>
<td>yes</td>
<td>weakly</td>
</tr>
<tr>
<td>( \text{sum}(S) \leq v )</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>( \text{sum}(S) \geq v )</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>( \text{range}(S) \leq v )</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>( \text{range}(S) \geq v )</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>( \text{avg}(S) \theta v, \theta \in { =, \leq, \geq } )</td>
<td>convertible</td>
<td>convertible</td>
<td>no</td>
</tr>
<tr>
<td>( \text{support}(S) \geq \xi )</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>( \text{support}(S) \leq \xi )</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
### What Constraints Are Convertible?

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Convertible anti-monotone</th>
<th>Convertible monotone</th>
<th>Strongly convertible</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{avg}(S) \leq, \geq v )</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>( \text{median}(S) \leq, \geq v )</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>( \text{sum}(S) \leq v ) (items could be of any value, ( v \geq 0 ))</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>( \text{sum}(S) \leq v ) (items could be of any value, ( v \leq 0 ))</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>( \text{sum}(S) \geq v ) (items could be of any value, ( v \geq 0 ))</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>( \text{sum}(S) \geq v ) (items could be of any value, ( v \leq 0 ))</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>......</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Pattern Mining in Multi-Level, Multi-Dimensional Space
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Mining Compressed or Approximate Patterns
- Sequential Pattern Mining
- Graph Pattern Mining
- Summary
F. Zhu, X. Yan, J. Han, P. S. Yu, and H. Cheng, “Mining Colossal Frequent Patterns by Core Pattern Fusion”, ICDE'07.

We have many algorithms, but can we mine large (i.e., colossal) patterns? — such as just size around 50 to 100? Unfortunately, not!

Why not? — the curse of “downward closure” of frequent patterns

- The “downward closure” property
  - Any sub-pattern of a frequent pattern is frequent.
  - Example. If \((a_1, a_2, \ldots, a_{100})\) is frequent, then \(a_1, a_2, \ldots, a_{100}, (a_1, a_2), (a_1, a_3), \ldots, (a_1, a_{100}), (a_1, a_2, a_3), \ldots\) are all frequent! There are about \(2^{100}\) such frequent itemsets!

- No matter using breadth-first search (e.g., Apriori) or depth-first search (FPgrowth), we have to examine so many patterns

Thus the downward closure property leads to explosion!
Closed/maximal patterns may partially alleviate the problem but not really solve it: We often need to mine scattered large patterns!

Let the minimum support threshold \( \sigma = 20 \)

There are \( \binom{40}{20} \) frequent patterns of size 20

Each is closed and maximal

\[
\# \text{ patterns} = \binom{n}{n/2} \approx \sqrt{2/\pi} \frac{2^n}{\sqrt{n}}
\]

The size of the answer set is exponential to \( n \)
Colossal Pattern Set: Small but Interesting

- It is often the case that only a small number of patterns are colossal, i.e., of large size.

- Colossal patterns are usually attached with greater importance than those of small pattern sizes.
Motivation: Many real-world tasks need mining colossal patterns
- Micro-array analysis in bioinformatics (when support is low)
- Biological sequence patterns
- Biological/sociological/information graph pattern mining

*No hope for completeness*
- If the mining of mid-sized patterns is explosive in size, there is no hope to find colossal patterns efficiently by insisting "complete set" mining philosophy

*Jumping out of the swamp of the mid-sized results*
- What we may develop is a philosophy that may jump out of the swamp of mid-sized results that are explosive in size and jump to reach colossal patterns

*Striving for mining almost complete colossal patterns*
- The key is to develop a mechanism that may quickly reach colossal patterns and discover most of them
Let the min-support threshold $\sigma = 20$

Then there are \( \binom{40}{20} \) closed/maximal frequent patterns of size 20

However, there is only one with size greater than 20, (i.e., colossal):

\[
\alpha = \{41, 42, \ldots, 79\} \quad \text{of size 39}
\]

The existing fastest mining algorithms (e.g., FPClose, LCM) fail to complete running

Our algorithm outputs this colossal pattern in seconds
Methodology of Pattern-Fusion Strategy

- Pattern-Fusion traverses the tree in a bounded-breadth way
  - Always pushes down a frontier of a bounded-size candidate pool
  - Only a fixed number of patterns in the current candidate pool will be used as the starting nodes to go down in the pattern tree — thus avoids the exponential search space
- Pattern-Fusion identifies “shortcuts” whenever possible
  - Pattern growth is not performed by single-item addition but by leaps and bounded: agglomeration of multiple patterns in the pool
  - These shortcuts will direct the search down the tree much more rapidly towards the colossal patterns
Observation: Colossal Patterns and Core Patterns

Subpatterns $\alpha_1$ to $\alpha_k$ cluster tightly around the colossal pattern $\alpha$ by sharing a similar support. We call such subpatterns core patterns of $\alpha$. 
Robustness of Colossal Patterns

- **Core Patterns**
  
  Intuitively, for a frequent pattern $\alpha$, a subpattern $\beta$ is a $\tau$-core pattern of $\alpha$ if $\beta$ shares a similar support set with $\alpha$, i.e.,
  
  $$\frac{|D_\alpha|}{|D_\beta|} \geq \tau \quad 0 < \tau \leq 1$$

  where $\tau$ is called the core ratio

- **Robustness of Colossal Patterns**
  
  A colossal pattern is robust in the sense that it tends to have much more core patterns than small patterns
Example: Core Patterns

- A colossal pattern has far more core patterns than a small-sized pattern.
- A colossal pattern has far more core descendants of a smaller size $c$.
- A random draw from a complete set of pattern of size $c$ would more likely to pick a core descendant of a colossal pattern.
- A colossal pattern can be generated by merging a set of core patterns.

<table>
<thead>
<tr>
<th>Transaction (# of Ts)</th>
<th>Core Patterns ($\tau = 0.5$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(abe) (100)</td>
<td>(abe), (ab), (be), (ae), (e)</td>
</tr>
<tr>
<td>(bcf) (100)</td>
<td>(bcf), (bc), (bf)</td>
</tr>
<tr>
<td>(acf) (100)</td>
<td>(acf), (ac), (af)</td>
</tr>
<tr>
<td>(abcef) (100)</td>
<td>(ab), (ac), (af), (ae), (bc), (bf), (be) (ce), (fe), (e), (abc), (abf), (abe), (ace), (acf), (afe), (bcf), (bce), (bfe), (cfe), (abcf), (abce), (bcfe), (acfe), (abfe), (abcef)</td>
</tr>
</tbody>
</table>
Colossal Patterns Correspond to Dense Balls

- Due to their robustness, colossal patterns correspond to dense balls
  - $\Omega(2^d)$ in population
- A random draw in the pattern space will hit somewhere in the ball with high probability
Idea of Pattern-Fusion Algorithm

- Generate a complete set of frequent patterns up to a small size.
- Randomly pick a pattern $\beta$, and $\beta$ has a high probability to be a core-descendant of some colossal pattern $\alpha$.
- Identify all $\alpha$’s descendants in this complete set, and merge all of them — This would generate a much larger core-descendant of $\alpha$.
- In the same fashion, we select $K$ patterns. This set of larger core-descendants will be the candidate pool for the next iteration.
Pattern-Fusion: The Algorithm

- Initialization (Initial pool): Use an existing algorithm to mine all frequent patterns up to a small size, e.g., 3
- Iteration (Iterative Pattern Fusion):
  - At each iteration, k seed patterns are randomly picked from the current pattern pool
  - For each seed pattern thus picked, we find all the patterns within a bounding ball centered at the seed pattern
  - All these patterns found are fused together to generate a set of super-patterns. All the super-patterns thus generated form a new pool for the next iteration
- Termination: when the current pool contains no more than K patterns at the beginning of an iteration
Why Is Pattern-Fusion Efficient?

- A bounded-breadth pattern tree traversal
  - It avoids explosion in mining mid-sized ones
  - Randomness comes to help to stay on the right path
- Ability to identify “short-cuts” and take “leaps”
  - fuse small patterns together in one step to generate new patterns of significant sizes
- Efficiency
Pattern-Fusion Leads to Good Approximation

- Gearing toward colossal patterns
  - The larger the pattern, the greater the chance it will be generated
- Catching outliers
  - The more distinct the pattern, the greater the chance it will be generated
Experimental Setting

- Synthetic data set
  - $\text{Diag}_n$, an $n \times (n-1)$ table where $i^{\text{th}}$ row has integers from 1 to $n$ except $i$. Each row is taken as an itemset. min_support is $n/2$.

- Real data set
  - Replace: A program trace data set collected from the "replace" program, widely used in software engineering research
    - Each item is a column, representing the activity level of gene/protein in the same
    - Frequent pattern would reveal important correlation between gene expression patterns and disease outcomes
**Experiment Results on Diag\(_n\)**

- LCM run time increases exponentially with pattern size \(n\).
- Pattern-Fusion finishes efficiently.
- The approximation error of Pattern-Fusion (with min-sup 20) in comparison with the complete set) is rather close to uniform sampling (which randomly picks \(K\) patterns from the complete answer set).
**Experimental Results on ALL**

- **ALL**: A popular gene expression data set with 38 transactions, each with 866 columns
  - There are 1736 items in total
  - The table shows a high frequency threshold of 30
Experimental Results on REPLACE

- REPLACE
  - A program trace data set, recording 4395 calls and transitions
  - The data set contains 4395 transactions with 57 items in total
  - With support threshold of 0.03, the largest patterns are of size 44
  - They are all discovered by Pattern-Fusion with different settings of $K$ and $\tau$, when started with an initial pool of 20948 patterns of size $\leq 3$
Experimental Results on REPLACE

- Approximation error when compared with the complete mining result
- Example. Out of the total 98 patterns of size $\geq 42$, when $K=100$, Pattern-Fusion returns 80 of them
- A good approximation to the colossal patterns in the sense that any pattern in the complete set is on average at most 0.17 items away from one of these 80 patterns
Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Pattern Mining in Multi-Level, Multi-Dimensional Space
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Mining Compressed or Approximate Patterns
- Sequential Pattern Mining
- Graph Pattern Mining
- Summary
Mining Compressed Patterns: $\delta$-clustering

- Why compressed patterns?
  - too many, but less meaningful

- Pattern distance measure

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

- $\delta$-clustering: For each pattern $P$, find all patterns which can be expressed by $P$ and their distance to $P$ are within $\delta$ ($\delta$-cover)

- All patterns in the cluster can be represented by $P$

- Xin et al., “Mining Compressed Frequent-Pattern Sets”, VLDB’05

<table>
<thead>
<tr>
<th>ID</th>
<th>Item-Sets</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>{38,16,18,12}</td>
<td>205227</td>
</tr>
<tr>
<td>P2</td>
<td>{38,16,18,12,17}</td>
<td>205211</td>
</tr>
<tr>
<td>P3</td>
<td>{39,38,16,18,12,17}</td>
<td>101758</td>
</tr>
<tr>
<td>P4</td>
<td>{39,16,18,12,17}</td>
<td>161563</td>
</tr>
<tr>
<td>P5</td>
<td>{39,16,18,12}</td>
<td>161576</td>
</tr>
</tbody>
</table>

- Closed frequent pattern
  - Report P1, P2, P3, P4, P5
  - Emphasize too much on support
  - no compression

- Max-pattern, P3: info loss

- A desirable output: P2, P3, P4
Redundancy-Award Top-k Patterns

- Why redundancy-aware top-k patterns?
- Desired patterns: high significance & low redundancy
- Propose the MMS (Maximal Marginal Significance) for measuring the combined significance of a pattern set
- Xin et al., Extracting Redundancy-Aware Top-K Patterns, KDD’06

(a) a set of patterns

(b) redundancy-aware top-k

(c) traditional top-k

(d) summarization
Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Pattern Mining in Multi-Level, Multi-Dimensional Space
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Mining Compressed or Approximate Patterns
- Sequential Pattern Mining
- Graph Pattern Mining
- Summary
Sequence Databases & Sequential Patterns

- Transaction databases, time-series databases vs. sequence databases
- Frequent patterns vs. (frequent) sequential patterns
- Applications of sequential pattern mining
  - Customer shopping sequences:
    - First buy computer, then CD-ROM, and then digital camera, within 3 months.
  - Medical treatments, natural disasters (e.g., earthquakes), science & eng. processes, stocks and markets, etc.
  - Telephone calling patterns, Weblog click streams
  - Program execution sequence data sets
  - DNA sequences and gene structures
What Is Sequential Pattern Mining?

- Given a set of sequences, find the complete set of frequent subsequences

A sequence database:

<table>
<thead>
<tr>
<th>SID</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&lt;a(abc)(ac)d(cf)&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;(ad)c(bc)(ae)&gt;</td>
</tr>
<tr>
<td>30</td>
<td>&lt;(ef)(ab)(df)cb&gt;</td>
</tr>
<tr>
<td>40</td>
<td>&lt;eg(af)cbc&gt;</td>
</tr>
</tbody>
</table>

- An element may contain a set of items
- 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 \( \text{min\_sup} = 2 \), <(ab)c> is a sequential pattern

Sequential pattern mining: find the complete set of patterns, satisfying the minimum support (frequency) threshold
Sequential Pattern Mining Algorithms

- Concept introduction and an initial Apriori-like algorithm
  - Agrawal & Srikant: Mining sequential patterns, ICDE’95
- Requirement: efficient, scalable, complete, minimal database scans, and be able to incorporate various kinds of user-specific constraints
- Representative algorithms
  - **GSP** (Generalized Sequential Patterns): Srikant & Agrawal @ EDBT’96
  - Vertical format-based mining: **SPADE** (Zaki@Machine Learning’00)
  - Pattern-growth methods: **PrefixSpan** (Pei, Han et al. @ICDE’01)
  - Constraint-based sequential pattern mining (SPIRIT: Garofalakis, Rastogi, Shim@VLDB’99; Pei, Han, Wang @ CIKM’02)
  - Mining closed sequential patterns: **CloSpan** (Yan, Han et al. @SDM’03)
The Apriori Property of Sequential Patterns

- 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>

<table>
<thead>
<tr>
<th>Seq. ID</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&lt;(bd)cb(ac)&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;(bf)(ce)b(fg)&gt;</td>
</tr>
<tr>
<td>30</td>
<td>&lt;(ah)(bf)abf&gt;</td>
</tr>
<tr>
<td>40</td>
<td>&lt;(be)(ce)d&gt;</td>
</tr>
<tr>
<td>50</td>
<td>&lt;a(bd)bcb(ade)&gt;</td>
</tr>
</tbody>
</table>

Given **support threshold**

*min_sup = 2*
GSP—Generalized Sequential Pattern Mining

- GSP (Generalized Sequential Pattern) mining algorithm
  - proposed by Agrawal and Srikant, EDBT’96
- Outline of the method
  - Initially, every item in DB is a candidate of length-1
  - for each level (i.e., sequences of length-k) do
    - scan database to collect support count for each candidate sequence
    - generate candidate length-(k+1) sequences from length-k frequent sequences using Apriori
  - repeat until no frequent sequence or no candidate can be found
- Major strength: Candidate pruning by Apriori
Finding Length-1 Sequential Patterns

- Examine GSP using an example
- Initial candidates: all singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Scan database once, count support for candidates

\[\text{min\_sup} = 2\]

<table>
<thead>
<tr>
<th>Seq. ID</th>
<th>Sequence</th>
<th>Cand</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&lt;(bd)cb(ac)&gt;</td>
<td>&lt;a&gt;</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>&lt;(bf)(ce)b(fg)&gt;</td>
<td>&lt;b&gt;</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>&lt;(ah)(bf)abf&gt;</td>
<td>&lt;c&gt;</td>
<td>4</td>
</tr>
<tr>
<td>40</td>
<td>&lt;(be)(ce)d&gt;</td>
<td>&lt;d&gt;</td>
<td>3</td>
</tr>
<tr>
<td>50</td>
<td>&lt;a(bd)bcb(ade)&gt;</td>
<td>&lt;e&gt;</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;f&gt;</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;g&gt;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;h&gt;</td>
<td>1</td>
</tr>
</tbody>
</table>
# GSP: Generating Length-2 Candidates

51 length-2 Candidates

<table>
<thead>
<tr>
<th></th>
<th>&lt;a&gt;</th>
<th>&lt;b&gt;</th>
<th>&lt;c&gt;</th>
<th>&lt;d&gt;</th>
<th>&lt;e&gt;</th>
<th>&lt;f&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;a&gt;</td>
<td>&lt;a&gt;</td>
<td>&lt;aa&gt;</td>
<td>&lt;ab&gt;</td>
<td>&lt;ac&gt;</td>
<td>&lt;ad&gt;</td>
<td>&lt;ae&gt;</td>
</tr>
<tr>
<td>&lt;b&gt;</td>
<td>&lt;ba&gt;</td>
<td>&lt;bb&gt;</td>
<td>&lt;bc&gt;</td>
<td>&lt;bd&gt;</td>
<td>&lt;be&gt;</td>
<td>&lt;bf&gt;</td>
</tr>
<tr>
<td>&lt;c&gt;</td>
<td>&lt;ca&gt;</td>
<td>&lt;cb&gt;</td>
<td>&lt;cc&gt;</td>
<td>&lt;cd&gt;</td>
<td>&lt;ce&gt;</td>
<td>&lt;cf&gt;</td>
</tr>
<tr>
<td>&lt;d&gt;</td>
<td>&lt;da&gt;</td>
<td>&lt;db&gt;</td>
<td>&lt;dc&gt;</td>
<td>&lt;dd&gt;</td>
<td>&lt;de&gt;</td>
<td>&lt;df&gt;</td>
</tr>
<tr>
<td>&lt;e&gt;</td>
<td>&lt;ea&gt;</td>
<td>&lt;eb&gt;</td>
<td>&lt;ec&gt;</td>
<td>&lt;ed&gt;</td>
<td>&lt;ee&gt;</td>
<td>&lt;ef&gt;</td>
</tr>
<tr>
<td>&lt;f&gt;</td>
<td>&lt;fa&gt;</td>
<td>&lt;fb&gt;</td>
<td>&lt;fc&gt;</td>
<td>&lt;fd&gt;</td>
<td>&lt;fe&gt;</td>
<td>&lt;ff&gt;</td>
</tr>
</tbody>
</table>

Without Apriori property,
8*8+8*7/2=92 candidates

Apriori prunes 44.57% candidates
The GSP Mining Process

5\textsuperscript{th} scan: 1 cand. 1 length-5 seq. pat. 

4\textsuperscript{th} scan: 8 cand. 6 length-4 seq. pat.

3\textsuperscript{rd} scan: 46 cand. 19 length-3 seq. pat. 20 cand. not in DB at all

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

1\textsuperscript{st} scan: 8 cand. 6 length-1 seq. pat.

\( min\_sup = 2 \)

<table>
<thead>
<tr>
<th>Seq. ID</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>(&lt;(bd)cb(ac)&gt;)</td>
</tr>
<tr>
<td>20</td>
<td>(&lt;(bf)(ce)b(fg)&gt;)</td>
</tr>
<tr>
<td>30</td>
<td>(&lt;(ah)(bf)abf&gt;)</td>
</tr>
<tr>
<td>40</td>
<td>(&lt;(be)(ce)d&gt;)</td>
</tr>
<tr>
<td>50</td>
<td>(&lt;a(bd)bcb(ade)&gt;)</td>
</tr>
</tbody>
</table>
The SPADE Algorithm

- SPADE (Sequential PAtrern Discovery using Equivalent Class) developed by Zaki 2001
- A vertical format sequential pattern mining method
- A sequence database is mapped to a large set of
  - Item: <SID, EID>
- Sequential pattern mining is performed by
  - growing the subsequences (patterns) one item at a time by Apriori candidate generation
### The SPADE Algorithm

<table>
<thead>
<tr>
<th>SID</th>
<th>EID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>abc</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>ac</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>cf</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>ad</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>c</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>bc</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>ae</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>ef</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>ab</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>df</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>b</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>e</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>g</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>af</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>b</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>c</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SID</th>
<th>EID</th>
<th>SID</th>
<th>EID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SID</th>
<th>EID (a)</th>
<th>EID (b)</th>
<th>EID (a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SID</th>
<th>EID (a)</th>
<th>EID (b)</th>
<th>EID (a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SID</th>
<th>EID (a)</th>
<th>EID (b)</th>
<th>EID (a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
Bottlenecks of GSP and SPADE

- A huge set of candidates could be generated
  - 1,000 frequent length-1 sequences generate a huge number of length-2 candidates!
    \[
    1000 \times 1000 + \frac{1000 \times 999}{2} = 1,499,500
    \]
- Multiple scans of database in mining
- Breadth-first search
- Mining long sequential patterns by growing from shorter patterns
  - Needs an exponential number of short candidates
  - A length-100 sequential pattern needs \(10^{30}\) candidate sequences!
    \[
    \sum_{i=1}^{100} \binom{100}{i} = 2^{100} - 1 \approx 10^{30}
    \]
PrefixSpan: Mining Sequential Patterns by Prefix Projections

- Prefix and suffix
  - Given sequence <a(abc)(ac)d(cf)>
  - Prefixes: <a>, <aa>, <a(ab)> and <a(abc)>

- PrefixSpan Mining framework
  - Step 1: find length-1 sequential patterns
    - <a>, <b>, <c>, <d>, <e>, <f>
  - Step 2: divide search space. The complete set of seq. pat. can be partitioned into 6 subsets:
    - The ones having prefix <a>;
    - The ones having prefix <b>;
    - ...;
    - The ones having prefix <f>
Finding Seq. Patterns with Prefix <a>

- Only need to consider projections w.r.t. <a>
  - <a>-projected database:
    - <(abc)(ac)d(cf)>
    - <(_d)c(bc)(ae)>  
    - <(_b)(df)cb>
    - <(_f)cbc>

- Find all the length-2 seq. pat. Having prefix <a>: <aa>, <ab>, <(ab)>, <ac>, <ad>, <af>
  - Further partition into 6 subsets
    - Having prefix <aa>:
    - ...
    - Having prefix <af>

<table>
<thead>
<tr>
<th>SID</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&lt;a(abc)(ac)d(cf)&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;(ad)c(bc)(ae)&gt;</td>
</tr>
<tr>
<td>30</td>
<td>&lt;(ef)(ab)(df)cb&gt;</td>
</tr>
<tr>
<td>40</td>
<td>&lt;eg(af)cbc&gt;</td>
</tr>
</tbody>
</table>
Completeness of PrefixSpan

**SDB**

<table>
<thead>
<tr>
<th>SID</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&lt;a(abc)(ac)d(cf)&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;ad)c(bc)(ae)&gt;</td>
</tr>
<tr>
<td>30</td>
<td>&lt;(ef)(ab)(df)cb&gt;</td>
</tr>
<tr>
<td>40</td>
<td>&lt;eg(af)cbc&gt;</td>
</tr>
</tbody>
</table>

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

Having prefix <a>

**<a>-projected database**

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

Having prefix <aa>

**<aa>-proj. db**

Having prefix <af>

**<af>-proj. db**

Having prefix <b>

**<b>-projected database**

Length-2 sequential patterns:

- <aa>, <ab>, <(ab)>, <ac>, <ad>, <af>

Having prefix <c>, ..., <f>

... ...

Major strength of PrefixSpan:

- No candidate sequence needs to be generated
- Projected databases keep shrinking
Speed-up by Pseudo-Projection

- Major cost of PrefixSpan: Constructing projected databases
  - Postfixes of sequences often appear repeatedly in recursive projected databases
- When (projected) database can be held in main memory, use pointers to form pseudo-projections
  - Pointer to the sequence
  - Offset of the postfix

s = <a(abc)(ac)d(cf)>

\[ s|<a>: (\_\_\_\_, 2) \quad <(abc)(ac)d(cf)> \]
\[ s|<ab>: (\_\_\_\_, 4) \quad <(_c)(ac)d(cf)> \]
Pseudo-Projection vs. Physical Projection

- Pseudo-projection avoids physically copying postfixes
  - Efficient in running time and space when database can be held in main memory
- However, it is not efficient when database cannot fit in main memory
  - Disk-based random accessing is very costly
- Suggested Approach:
  - Integration of physical and pseudo-projection
  - Swapping to pseudo-projection when the data set fits in memory
Performance of Sequential Pattern Mining Algorithms

Performance comparison:
with pseudo-projection vs. without pseudo-projection

Performance comparison on data set C10T8S8I8

Performance comparison on Gazelle data set

Performance comparison: with pseudo-projection vs. without pseudo-projection
A closed sequential pattern $s$: there exists no superpattern $s'$ such that $s' \supset s$, and $s'$ and $s$ have the same support.

Which one is closed? $<\text{abc}> : 20$, $<\text{abcd}> : 20$, $<\text{abcde}> : 15$

Why mine close seq. patterns?
- Reduces the number of (redundant) patterns but attains the same expressive power.
- Property: If $s' \supset s$, closed iff two project DBs have the same size.
- Using Backward Subpattern and Backward Superpattern pruning to prune redundant search space.
Performance Comparison: CloSpan vs. PrefixSpan
Constraint-Based Seq.-Pattern Mining

- Constraint-based sequential pattern mining
  - Constraints: User-specified, for focused mining of desired patterns
  - How to explore efficient mining with constraints? — Optimization

- Classification of constraints
  - Anti-monotone: E.g., \( \text{value}_\text{sum}(S) < 150, \text{min}(S) > 10 \)
  - Monotone: E.g., \( \text{count}(S) > 5, S \supseteq \{\text{PC, digital_camera}\} \)
  - Succinct: E.g., \( \text{length}(S) \geq 10, S \in \{\text{Pentium, MS/Office, MS/Money}\} \)
  -Convertible: E.g., \( \text{value}_{\text{avg}}(S) < 25, \text{profit}_{\text{sum}}(S) > 160, \text{max}(S)/\text{avg}(S) < 2, \text{median}(S) - \text{min}(S) > 5 \)
  - Inconvertible: E.g., \( \text{avg}(S) - \text{median}(S) = 0 \)
From Sequential Patterns to Structured Patterns

- Sets, sequences, trees, graphs, and other structures
  - Transaction DB: Sets of items
    - \{\{i_1, i_2, ..., i_m\}, ...\}
  - Seq. DB: Sequences of sets:
    - \{<\{i_1, i_2\}, ..., \{i_m, i_n, i_k\}>, ...\}
  - Sets of Sequences:
    - \{\{<i_1, i_2>, ..., <i_m, i_n, i_k>\}, ...\}
  - Sets of trees: \{t_1, t_2, ..., t_n\}
  - Sets of graphs (mining for frequent subgraphs):
    - \{g_1, g_2, ..., g_n\}
- Mining structured patterns in XML documents, biochemical structures, etc.
Alternative to Sequential Patterns: Episodes and Episode Pattern Mining

- Alternative patterns: Episodes and regular expressions
  - Serial episodes: A $\rightarrow$ B
  - Parallel episodes: A & B
  - Regular expressions: $(A|B)C^*(D \rightarrow E)$

- Methods for episode pattern mining
  - Method 1: Variations of Apriori/GSP-like algorithms
  - Method 2: Projection-based pattern growth
    - Can you work out the details?
  - Question: What is the difference between mining episodes and constraint-based pattern mining?
Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Pattern Mining in Multi-Level, Multi-Dimensional Space
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Mining Compressed or Approximate Patterns
- Sequential Pattern Mining
- Graph Pattern Mining
- Summary
Graph Pattern Mining

- **Frequent subgraphs**
  - A (sub)graph is **frequent** if its *support* (occurrence frequency) in a given dataset is no less than a *minimum support* threshold

- Applications of graph pattern mining
  - Mining biochemical structures
  - Program control flow analysis
  - Mining XML structures or Web communities
  - Building blocks for graph classification, clustering, compression, comparison, and correlation analysis
Example: Frequent Subgraphs

GRAPH DATASET

(A) [Chemical Structure]

(B) [Chemical Structure]

(C) [Chemical Structure]

FREQUENT PATTERNS
(MIN SUPPORT IS 2)

(1) [Chemical Structure]

(2) [Chemical Structure]
Properties of Graph Mining Algorithms

- Search order
  - breadth vs. depth
- Generation of candidate subgraphs
  - apriori vs. pattern growth
- Elimination of duplicate subgraphs
  - passive vs. active
- Support calculation
  - embedding store or not
- Discover order of patterns
  - path $\rightarrow$ tree $\rightarrow$ graph
Apriori-Based Approach

\[ G \rightarrow G_1 \]
\[ G' \rightarrow G_2 \]
\[ G'' \rightarrow G_n \]

JOIN

\( k \)-edge

\( (k+1) \)-edge
Apriori-Based, Breadth-First Search

- Methodology: breadth-search, joining two graphs

- AGM (Inokuchi, et al. PKDD’00)
  - generates new graphs with one more node

- FSG (Kuramochi and Karypis ICDM’01)
  - generates new graphs with one more edge
Pattern Growth Method

$G_1 \rightarrow (k+1)$-edge

$G_2 \rightarrow (k+2)$-edge

duplicate graph

$G_n \rightarrow \ldots$

$k$-edge

$G \rightarrow \ldots$
GSPAN (Yan and Han ICDM’02)

Right-Most Extension

Theorem: Completeness

The Enumeration of Graphs using Right-most Extension is COMPLETE
- Flatten a graph into a sequence using depth first search

DFS Code

- e0: (0,1)
- e1: (1,2)
- e2: (2,0)
- e3: (2,3)
- e4: (3,1)
- e5: (2,4)
Let $Z$ be the set of DFS codes of all graphs. Two DFS codes $a$ and $b$ have the relation $a \leq b$ (DFS Lexicographic Order in $Z$) if and only if one of the following conditions is true. Let

\begin{align*}
a &= (x_0, x_1, \ldots, x_n) \quad \text{and} \\
b &= (y_0, y_1, \ldots, y_n),
\end{align*}

(i) if there exists $t$, $0 \leq t \leq \min(m,n)$, $x_k = y_k$ for all $k$, s.t. $k < t$, and $x_t < y_t$

(ii) $x_k = y_k$ for all $k$, s.t. $0 \leq k \leq m$ and $m \leq n$. 
DFS Code Extension

Let \( a \) be the minimum DFS code of a graph \( G \) and \( b \) be a non-minimum DFS code of \( G \). For any DFS code \( d \) generated from \( b \) by one right-most extension,

(i) \( d \) is not a minimum DFS code,
(ii) \( \text{min}_\text{dfs}(d) \) cannot be extended from \( b \), and
(iii) \( \text{min}_\text{dfs}(d) \) is either less than \( a \) or can be extended from \( a \).

**THEOREM [ RIGHT-EXTENSION ]**
The DFS code of a graph extended from a Non-minimum DFS code is NOT MINIMUM.
GASTON (Nijssen and Kok KDD’04)

- Extend graphs directly
- Store embeddings
- Separate the discovery of different types of graphs
  - path $\rightarrow$ tree $\rightarrow$ graph
  - Simple structures are easier to mine and duplication detection is much simpler
Graph Pattern Explosion Problem

- If a graph is frequent, all of its subgraphs are frequent — the Apriori property
- An $n$-edge frequent graph may have $2^n$ subgraphs
- Among 422 chemical compounds which are confirmed to be active in an AIDS antiviral screen dataset, there are 1,000,000 frequent graph patterns if the minimum support is 5%
Closed Frequent Graphs

- Motivation: Handling graph pattern explosion problem
- Closed frequent graph
  - A frequent graph $G$ is *closed* if there exists no supergraph of $G$ that carries the same support as $G$
- If some of $G$'s subgraphs have the same support, it is unnecessary to output these subgraphs (nonclosed graphs)
- *Lossless compression*: still ensures that the mining result is complete
At what condition, can we stop searching their children i.e., early termination?

If $G$ and $G'$ are frequent, $G$ is a subgraph of $G'$. If in any part of the graph in the dataset where $G$ occurs, $G'$ also occurs, then we need not grow $G$, since none of $G$’s children will be closed except those of $G'$. 
Handling Tricky Exception Cases

(graph 1)

(graph 2)

(pattern 1)

(pattern 2)
The AIDS antiviral screen compound dataset from NCI/NIH

The dataset contains 43,905 chemical compounds

Among these 43,905 compounds, 423 of them belong to CA, 1081 are of CM, and the remaining are in class CI
Discovered Patterns

20%  

10%  

5%
Performance (1): Run Time

![Graph showing run time per pattern (msec) vs. minimum support (in %) for MoFa, gSpan, FFSM, and Gaston]
Performance (2): Memory Usage

Memory usage (GB) vs. Minimum support (in %)
Performance Comparison: Frequent vs. Closed

# of Patterns: Frequent vs. Closed

Runtime: Frequent vs. Closed

CA

Number of patterns

Minimum support

Run time (sec)

Minimum support
Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Pattern Mining in Multi-Level, Multi-Dimensional Space
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Mining Compressed or Approximate Patterns
- Pattern Exploration and Application
- Summary
Summary

- Roadmap: Many aspects & extensions on pattern mining
- Mining patterns in multi-level, multi-dimensional space
- Mining rare and negative patterns
- Constraint-based pattern mining
- Specialized methods for mining high-dimensional data and colossal patterns
- Mining compressed or approximate patterns
- Pattern exploration and understanding: Semantic annotation of frequent patterns
Ref: Mining Multi-Level and Quantitative Rules

- J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. VLDB'95.
Ref: Mining Other Kinds of Rules

- F. Zhu, X. Yan, J. Han, P. S. Yu, and H. Cheng, “Mining Colossal Frequent Patterns by Core Pattern Fusion”, ICDE'07.
Ref: Constraint-Based Pattern Mining

- R. Ng, L.V.S. Lakshmanan, J. Han & A. Pang. Exploratory mining and pruning optimizations of constrained association rules. SIGMOD'98
- G. Grahne, L. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. ICDE'00
- J. Pei, J. Han, and L. V. S. Lakshmanan. Mining Frequent Itemsets with Convertible Constraints. ICDE'01
- J. Pei, J. Han, and W. Wang, Mining Sequential Patterns with Constraints in Large Databases, CIKM'02
- F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. ExAnte: Anticipated Data Reduction in Constrained Pattern Mining, PKDD'03
- F. Zhu, X. Yan, J. Han, and P. S. Yu, “gPrune: A Constraint Pushing Framework for Graph Pattern Mining”, PAKDD'07
Ref: Mining Sequential and Structured Patterns

- J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. ICDE'01.
- M. Kuramochi and G. Karypis. Frequent Subgraph Discovery. ICDM'01.
- X. Yan, J. Han, and R. Afshar. CloSpan: Mining Closed Sequential Patterns in Large Datasets. SDM'03.
- X. Yan and J. Han. CloseGraph: Mining Closed Frequent Graph Patterns. KDD'03.
K. Koperski and J. Han, Discovery of Spatial Association Rules in Geographic Information Databases, SSD’95.

O. R. Zaiane, M. Xin, J. Han, Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. ADL'98.

O. R. Zaiane, J. Han, and H. Zhu, Mining Recurrent Items in Multimedia with Progressive Resolution Refinement. ICDE'00.

D. Gunopulos and I. Tsoukatos. Efficient Mining of Spatiotemporal Patterns. SSTD'01.
Ref: Mining Frequent Patterns in Time-Series Data

- B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. ICDE'98.
- J. Han, G. Dong and Y. Yin, Efficient Mining of Partial Periodic Patterns in Time Series Database, ICDE'99.
- H. Lu, L. Feng, and J. Han. Beyond Intra-Transaction Association Analysis: Mining Multi-Dimensional Inter-Transaction Association Rules. TOIS:00.
- B.-K. Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris. Online Data Mining for Co-Evolving Time Sequences. ICDE'00.
Ref: FP for Classification and Clustering

- B. Liu, W. Hsu, Y. Ma. Integrating Classification and Association Rule Mining. KDD’98.
- W. Li, J. Han, and J. Pei. CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. ICDM'01.
- J. Yang and W. Wang. CLUSEQ: efficient and effective sequence clustering. ICDE’03.
- X. Yin and J. Han. CPAR: Classification based on Predictive Association Rules. SDM'03.
- H. Cheng, X. Yan, J. Han, and C.-W. Hsu, Discriminative Frequent Pattern Analysis for Effective Classification”, ICDE'07.
Ref: Stream and Privacy-Preserving FP Mining

- G. Manku and R. Motwani. Approximate Frequency Counts over Data Streams. VLDB’02.
- Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. Multi-Dimensional Regression Analysis of Time-Series Data Streams. VLDB'02.
- C. Giannella, J. Han, J. Pei, X. Yan and P. S. Yu. Mining Frequent Patterns in Data Streams at Multiple Time Granularities, Next Generation Data Mining:03.
Ref: Other Freq. Pattern Mining Applications


- T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapenyuk. Mining Database Structure; or How to Build a Data Quality Browser. SIGMOD'02.

- K. Wang, S. Zhou, J. Han. Profit Mining: From Patterns to Actions. EDBT’02.
Ref: Mining Sequential Patterns

- J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. ICDE'01 (TKDE’04).
- J. Pei, J. Han and W. Wang, Constraint-Based Sequential Pattern Mining in Large Databases, CIKM'02.
- X. Yan, J. Han, and R. Afshar. CloSpan: Mining Closed Sequential Patterns in Large Datasets. SDM'03.
- J. Wang and J. Han, BIDE: Efficient Mining of Frequent Closed Sequences, ICDE'04.
- H. Cheng, X. Yan, and J. Han, IncSpan: Incremental Mining of Sequential Patterns in Large Database, KDD'04.
- J. Han, G. Dong and Y. Yin, Efficient Mining of Partial Periodic Patterns in Time Series Database, ICDE'99.
- J. Yang, W. Wang, and P. S. Yu, Mining asynchronous periodic patterns in time series data, KDD'00.
Chapter 7: Advanced Frequent Pattern Mining

- Frequent Pattern and Association Mining: A Road Map
- Pattern Mining in Multi-Level, Multi-Dimensional Space
  - Mining Multilevel Association
  - Mining Multi-Dimensional Association
  - Mining Quantitative Association Rules
- Exploring Alternative Approaches to Improve Efficiency and Scalability
  - Mining Closed and Max Patterns
  - Scalable Pattern Mining in High-Dimensional Data
  - Mining Colossal Patterns
- Mining Beyond Typical Frequent Patterns
  - Mining Infrequent and Negative Patterns
  - Mining Compressed and Approximate Patterns
- Constraint-Based Frequent Pattern Mining
  - Metarule-Guided Mining of Association Rules
  - Constraint-Based Pattern Generation: Monotonicity, Anti-monotonicity, Succinctness, and Data Anti-monotonicity
  - Convertible Constraints: Ordering Data in Transactions
- Advanced Applications of Frequent Patterns
  - Towards pattern-based classification and cluster analysis
  - Context Analysis: Generating Semantic Annotations for Frequent Patterns
- Summary
Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Pattern Mining in Multi-Level, Multi-Dimensional Space
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Mining Compressed or Approximate Patterns
- Pattern Exploration and Application
- Summary
How to Understand and Interpret Patterns?

- Do they all make sense?
- What do they mean?
- How are they useful?

**Semantic Information**

Not all frequent patterns are useful, only meaningful ones ...

**Annotate patterns with semantic information**
A Dictionary Analogy

Word: “pattern” – from Merriam-Webster

Non-semantic info.

Definitions indicating semantics

Synonyms

Related Words

MODEL 2, archetype, beau ideal, ensample, example, exemplar, ideal, figure, paradigm, standard

component
2

synonym

Related Word

original

Order 8, method, orderliness, plan, system

arrangement, constellation
Semantic Analysis with Context Models

- Task 1: Model the context of a frequent pattern
  
  Based on the Context Model ...

- Task 2: Extract strongest context indicators

- Task 3: Extract representative transactions

- Task 4: Extract semantically similar patterns
**Annotating DBLP Co-authorship & Title Pattern**

**Database:**

<table>
<thead>
<tr>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>X.Yan, P. Yu, J. Han</td>
<td>Substructure Similarity Search in Graph Databases</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Frequent Patterns**

- \( P_1: \{x_{yan}, j_{han}\} \)
- \( P_2: \text{“substructure search”} \)

**Frequent Itemset**

- \( P_2: \text{“substructure search”} \)

**Semantic Annotations**

- **Pattern** = \{xifeng_yan, jiawei_han\}

<table>
<thead>
<tr>
<th>Pattern</th>
<th>{x_{yan}, j_{han}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non</td>
<td>Sup = …</td>
</tr>
<tr>
<td>CI</td>
<td>{p_{yu}}, graph pattern, …</td>
</tr>
<tr>
<td>Trans.</td>
<td>gSpan: graph-base……</td>
</tr>
<tr>
<td>SSPs</td>
<td>{j_{wang}}, {j_{han}, p_{yu}}, …</td>
</tr>
</tbody>
</table>

**Context Units**

- \(<\{p_{yu}, j_{han}\}, \{d_{xin}\}, …, \text{“graph pattern”}, …, \text{“substructure similarity”}, …>\)

**Annotation Results:**

- **Context Indicator (CI)**
  - graph; \{philip_yu\}; mine close; graph pattern; sequential pattern; …

- **Representative Transactions (Trans)**
  - > gSpan: graph-base substructure pattern mining;
  - > mining close relational graph connect constraint; …

- **Semantically Similar Patterns (SSP)**
  - \{jiawei_han, philip_yu\}; \{jian_pei, jiawei_han\}; \{jiong_yang, philip_yu, wei_wang\}; …
Frequent Subgraph Mining Approaches

- Apriori-based approach
  - AGM/AcGM: Inokuchi, et al. (PKDD’00)
  - FSG: Kuramochi and Karypis (ICDM’01)
  - PATH#: Vanetik and Gudes (ICDM’02, ICDM’04)
  - FFSM: Huan, et al. (ICDM’03)

- Pattern growth approach
  - MoFa, Borgelt and Berthold (ICDM’02)
  - gSpan: Yan and Han (ICDM’02)
  - Gaston: Nijssen and Kok (KDD’04)
PATH (Vanetik and Gudes ICDM’02, ’04)

- Apriori-based approach
- Building blocks: edge-disjoint path

A graph with 3 edge-disjoint paths

- construct frequent paths
- construct frequent graphs with 2 edge-disjoint paths
- construct graphs with $k+1$ edge-disjoint paths from graphs with $k$ edge-disjoint paths
- repeat
FFSM (Huan, et al. ICDM’03)

- Represent graphs using canonical adjacency matrix (CAM)
- Join two CAMs or extend a CAM to generate a new graph
- Store the embeddings of CAMs
  - All of the embeddings of a pattern in the database
  - Can derive the embeddings of newly generated CAMs
MoFa (Borgelt and Berthold ICDM’02)

- Extend graphs by adding a new edge
- Store embeddings of discovered frequent graphs
  - Fast support calculation
  - Also used in other later developed algorithms such as FFSM and GASTON
- Expensive Memory usage
- Local structural pruning