Data Mining:
Concepts and Techniques

— Chapter 3 —
inspired from the following textbook

Jiawei Han
Department of Computer Science
University of Illinois at Urbana-Champaign
www.cs.uiuc.edu/~hanj

Chapter 3: Mining Frequent Patterns and Associations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
  - Apriori
  - FP-growth
  - ECLAT
- Mining various kinds of association rules
- Summary

What Is Frequent Pattern Analysis?

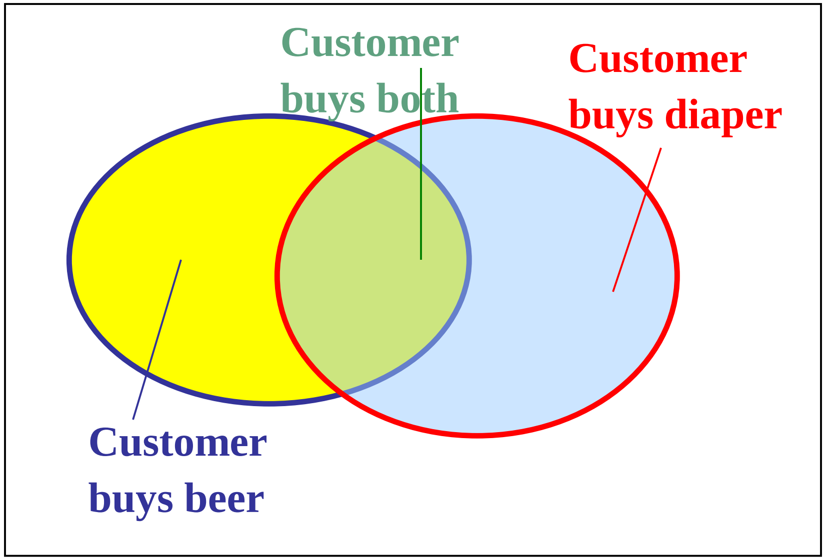
- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Why Is Freq. Pattern Mining Important?

- Discloses an intrinsic and important property of data sets
- Forms the foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: associative classification
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression: fascicles
  - Broad applications

Basic Concepts: Frequent Patterns and Association Rules

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F



- Itemset X = $\{x_1, \ldots, x_k\}$
- Find all the rules $X \rightarrow Y$ with minimum support and confidence
  - support, *s*, probability that a transaction contains X ∪ Y
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

Let $sup_{min} = 50\%$, $conf_{min} = 50\%$

Freq. Pat.: {*A:3, B:3, D:4, E:3, AD:3*}

Association rules:

$A \rightarrow D$ (60%, 100%)

$D \rightarrow A$ (60%, 75%)

Chapter 3: Mining Frequent Patterns and Associations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
  - Apriori
  - FP-growth
  - ECLAT
- Mining various kinds of association rules
- Summary

Scalable Methods for Mining Frequent Patterns

- The downward closure property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If **{b, d, n}** is frequent, so is **{b, d}**
  - i.e., every transaction having {b, d, n} also contains {b, d}
- Scalable mining methods: Three major approaches
  - Apriori (Agrawal & Srikant@VLDB'94)
  - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
  - Vertical data format approach (Charm—Zaki & Hsiao @SDM'02)

Apriori: A Candidate Generation-and-Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1st scan →

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan ←

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3rd scan →

L_3

Itemset	sup
{B, C, E}	2

The Apriori Algorithm

- Pseudo-code:

  C_k: Candidate itemset of size k
  L_k : frequent itemset of size k

  L_1 = {frequent items};
  for (k = 1; L_k !=$\emptyset$; k++) **do begin**
  C_{k+1} = candidates generated from L_k;
  for each transaction t in database do
  increment the count of all candidates in C_{k+1} that are contained in t
  L_{k+1} = candidates in C_{k+1} with min_support
  end
  return $\cup_k L_k$;

Important Details of Apriori

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- How to count supports of candidates?
- Example of Candidate-generation
  - L_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: L_3*L_3
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - C_4={*abcd*}

Challenges of Frequent Pattern Mining

- Challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

Partition: Scan Database Only Twice

- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association in large databases. In *VLDB'95*

Bottleneck of Frequent-pattern Mining

- Multiple database scans are costly
- Mining long patterns needs many passes of scanning and generates lots of candidates
  - To find frequent itemset $i_1 i_2 \ldots i_{100}$
    - # of scans: 100
    - # of Candidates: $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 = 1.27 \times 10^{30}$!
- Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?

Mining Frequent Patterns Without Candidate Generation

- Grow long patterns from short ones using local frequent items
  - "abc" is a frequent pattern
  - Get all transactions having "abc": DB|abc
  - "d" is a local frequent item in DB|abc → abcd is a frequent pattern

Construct FP-tree from a Transaction Database

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

min_support = 3

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

Item	*frequency*
f	*4*
c	*4*
a	*3*
b	*3*
m	*3*
p	*3*

F-list=f-c-a-b-m-p

Construct FP-tree from a Transaction Database

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

min_support = 3

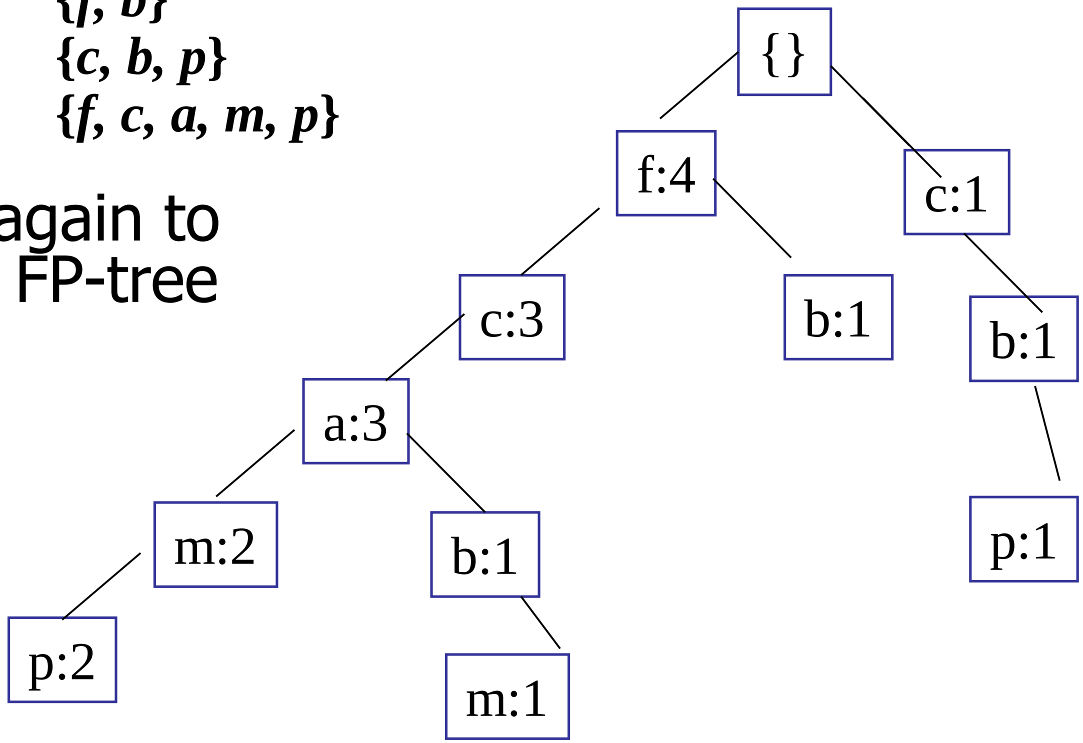
F-list=f-c-a-b-m-p

Construct FP-tree from a Transaction Database

TID	*ordered frequent items*
100	**{*f, c, a, m, p*}**
200	**{*f, c, a, b, m*}**
300	**{*f, b*}**
400	**{*c, b, p*}**
500	**{*f, c, a, m, p*}**

min_support* = *3

Scan DB again to construct FP-tree

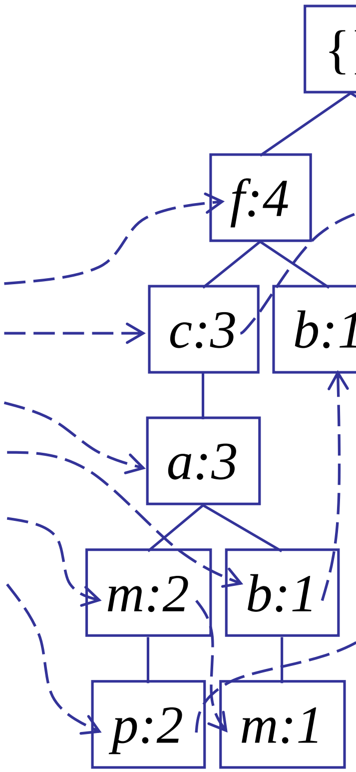


Construct FP-tree from a Transaction Database

TID	***ordered frequent items***
100	**{*f, c, a, m, p*}**
200	**{*f, c, a, b, m*}**
300	**{*f, b*}**
400	**{*c, b, p*}**
500	**{*f, c, a, m, p*}**

min_support* = *3

Scan DB again to construct FP-tree



F-list=f-c-a-b-m-p

Construct FP-tree from a Transaction Database

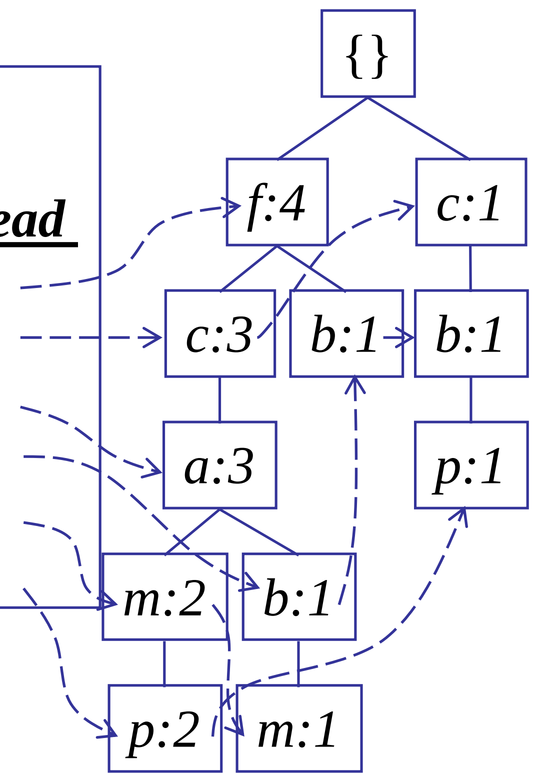
TID	*Items bought*	*(ordered) frequent items*
100	***{f, a, c, d, g, i, m, p}***	***{f, c, a, m, p}***
200	***{a, b, c, f, l, m, o}***	***{f, c, a, b, m}***
300	***{b, f, h, j, o, w}***	***{f, b}***
400	***{b, c, k, s, p}***	***{c, b, p}***
500	***{a, f, c, e, l, p, m, n}***	***{f, c, a, m, p}***

min_support = 3

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

Header Table

Item	*frequency*	*head*
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p	*3*	



F-list=f-c-a-b-m-p

Benefits of the FP-tree Structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)

Partition Patterns and Databases

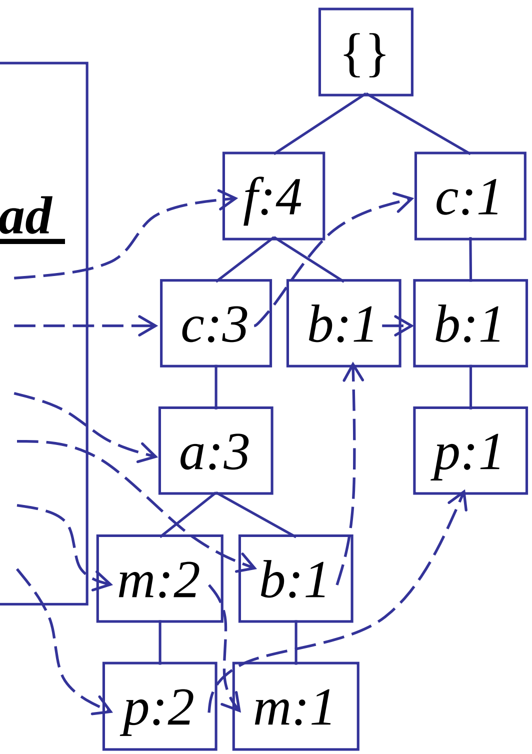
- Frequent patterns can be partitioned into subsets according to f-list
  - F-list=f-c-a-b-m-p
  - Patterns containing p
  - Patterns having m but no p
  - …
  - Patterns having c but no a nor b, m, p
  - Pattern f
- Completeness and non-redundency

Find Patterns Having P From P-conditional Database

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

Header Table

Item	frequency	head
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

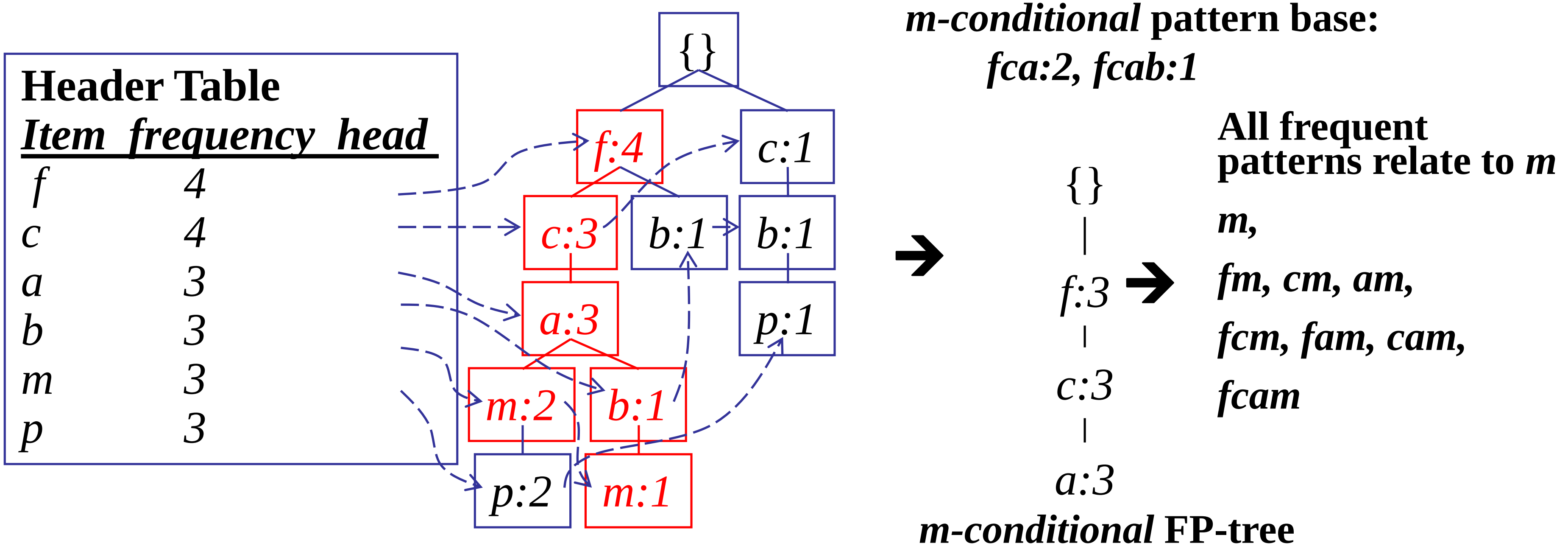


***Conditional* pattern bases**

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

From Conditional Pattern-bases to Conditional FP-trees

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



Recursion: Mining Each Conditional FP-tree

{}
|
f:3
|
c:3
|
a:3

m-conditional **FP-tree**

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

{}
|
f:3
|
c:3

am-conditional **FP-tree**

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

{}
|
f:3

cm-conditional **FP-tree**

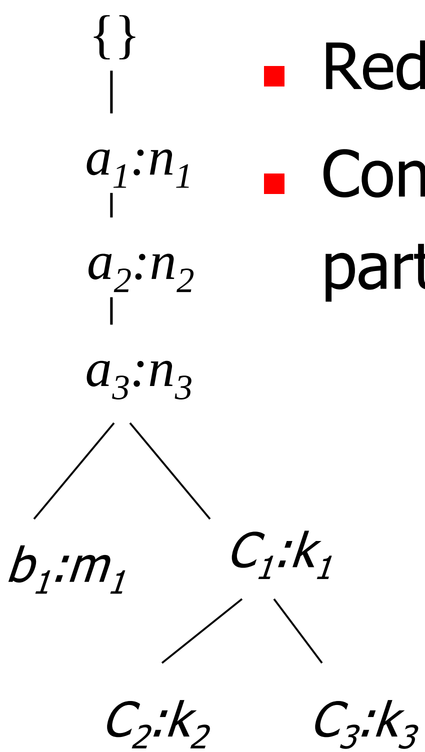
Cond. pattern base of "cam": (f:3)

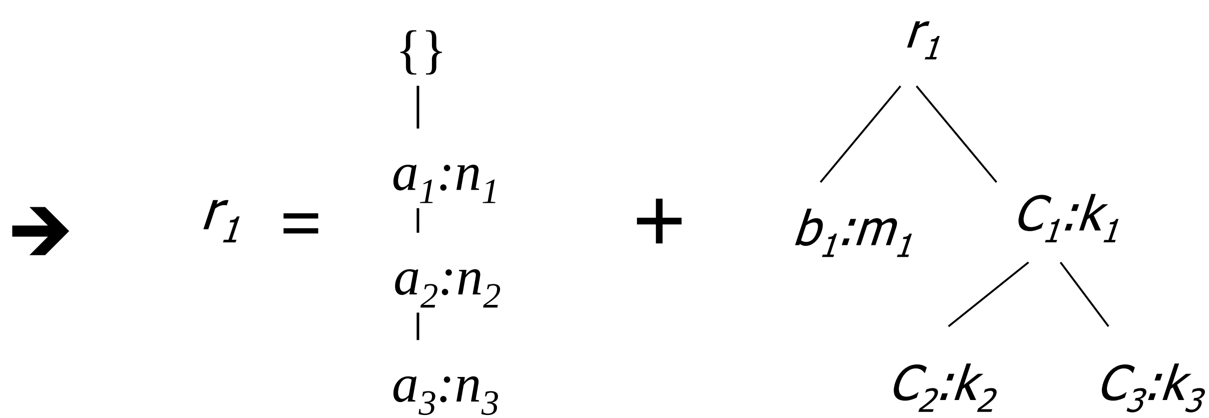
{}
|
f:3

cam-conditional **FP-tree**

A Special Case: Single Prefix Path in FP-tree

- Suppose a (conditional) FP-tree T has a shared single prefix-path P
- Mining can be decomposed into two parts
  - Reduction of the single prefix path into one node
  - Concatenation of the mining results of the two parts

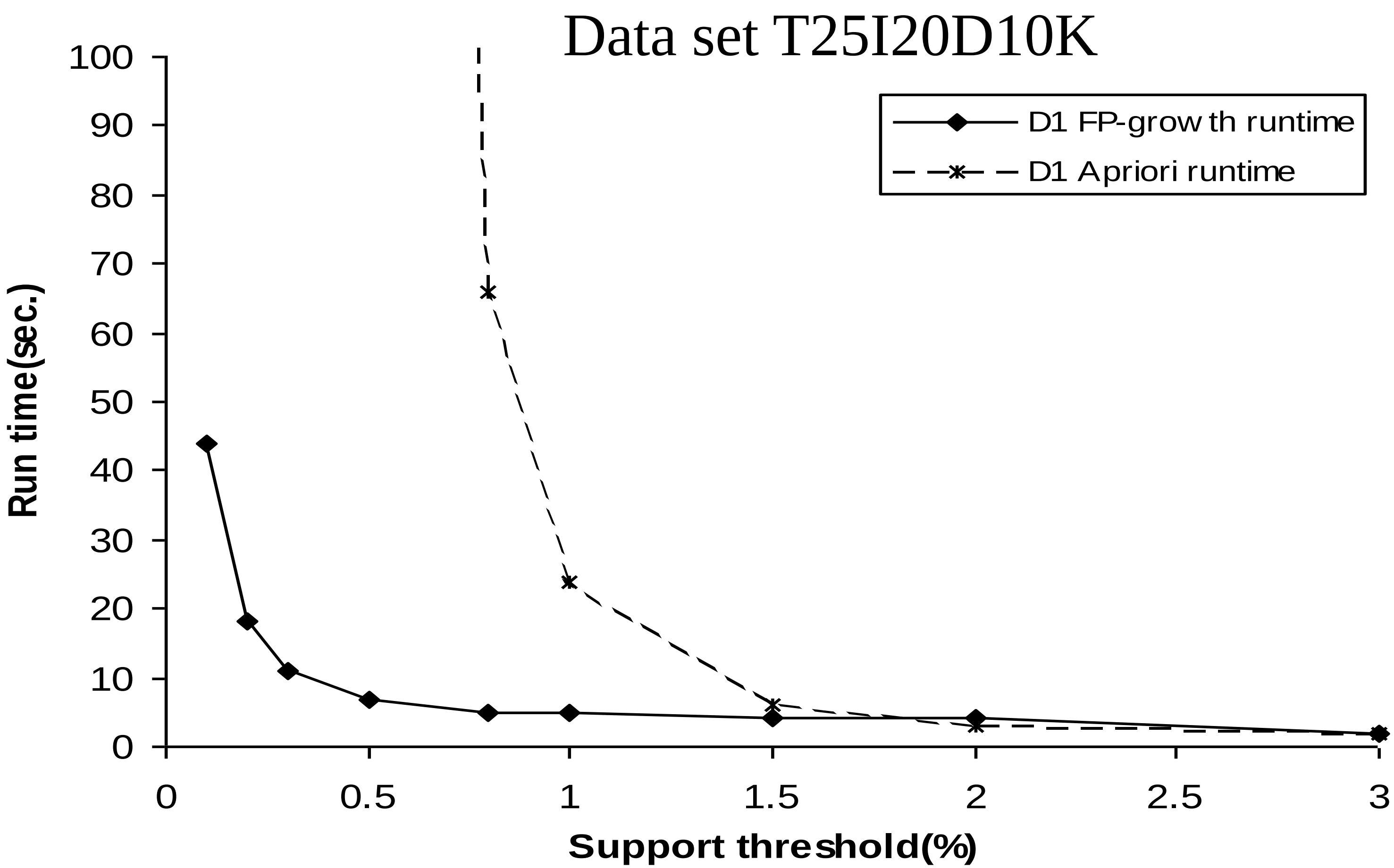




Mining Frequent Patterns With FP-trees

- Idea: Frequent pattern growth
  - Recursively grow frequent patterns by pattern and database partition
- Method
  - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
  - Repeat the process on each newly created conditional FP-tree
  - Until the resulting FP-tree is empty, or it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

FP-Growth vs. Apriori: Scalability With the Support Threshold



Why Is FP-Growth the Winner?

- Divide-and-conquer:
  - decompose both the mining task and DB according to the frequent patterns obtained so far
  - leads to focused search of smaller databases
- Other factors
  - no candidate generation, no candidate test
  - compressed database: FP-tree structure
  - no repeated scan of entire database
  - basic ops—counting local freq items and building sub FP-tree, no pattern search and matching

CHARM: Mining by Exploring Vertical Data Format

- Vertical format: $t(AB) = \{T_{11}, T_{25}, \ldots\}$
  - tid-list: list of trans.-ids containing an itemset
- Deriving closed patterns based on vertical intersections
  - $t(X) = t(Y)$: X and Y always happen together
  - $t(X) \subset t(Y)$: transaction having X always has Y
- Using diffset to accelerate mining
  - Only keep track of differences of tids
  - $t(X) = \{T_1, T_2, T_3\}$, $t(XY) = \{T_1, T_3\}$
  - Diffset $(XY, X) = \{T_2\}$
- Eclat/MaxEclat (Zaki et al. @KDD'97), VIPER(P. Shenoy et al.@SIGMOD'00), CHARM (Zaki & Hsiao@SDM'02)

ECLAT: a simple Algorithm based on the Vertical Approach

ECLAT Algorithm

1. Invert the BD
   a. In the first column place the item
   b. In the second column place the transactions Ids to which the item belongs

2. k = 1 : eliminate all the rows with a number of transactions lower than the minimum support

1. k=k+1: **for** all possible pair of rows **do**
   a. create a new row containing
      i. in the first column the union of the contents of the first columns
      ii. in the second column the **intersection** of contents of the second columns

2. **repeat** steps 2 and 3 **until** no new itemset can be created

ECLAT: Example

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

min_support = 3

itemset	**TIDs**
{a}	100, 200,500
{b}	200,300,400
{c}	100,200,400, 500
{d}	100
{e}	500
{f}	100, 200, 300, 500
{g}	100
{h}	300
{i}	100
{j}	300
{k}	400
{i}	200, 500
{m}	100, 200, 500
{n}	500
{o}	200, 300
{p}	100, 400, 500

itemset	**TIDs**
{a}	100, 200,500
{b}	200,300,400
{c}	100,200,400, 500
{f}	100, 200, 300, 500
{m}	100, 200, 500
{p}	100, 400, 500

L1

ECLAT: Example

itemset	TIDs
{a}	100, 200,500
{b}	200,300,400
{c}	100,200,400, 500
{f}	100, 200, 300, 500
{m}	100, 200, 500
{p}	100, 400, 500

itemset	TIDs
{a,b}	200
{a,c}	100, 200, 500
{a,f}	100,200,500
{a,m}	100, 200, 500
{a,p}	100, 500
{b,c}	200, 400
{b,m}	200, 500
{b,p}	400
{c,f}	100, 200, 500
{c,m}	100, 200, 500
{c,p}	100, 400, 500
{f,m}	100, 200, 500
{f,p}	100, 500
{m,p}	100, 500

itemset	TIDs
{a,c}	100, 200, 500
{a,f}	100,200,500
{a,m}	100, 200, 500
{c,f}	100, 200, 500
{c,m}	100, 200, 500
{c,p}	100, 400, 500
{f,m}	100, 200, 500

L2

$$(n-1)+(n-2)+\ldots+1=\sum_1^{n-1} i = \frac{n(n-1)}{2}=\frac{n^2-n}{2}$$

ECLAT

itemset	TIDs
{a,c}	100, 200, 500
{a,f}	100,200,500
{a,m}	100, 200, 500
{c,f}	100, 200, 500
{c,m}	100, 200, 500
{c,p}	100, 400, 500
{f,m}	100, 200, 500

itemset	TIDs
{a,c,f}	100, 200, 500
{a,c,m}	100,200,500
{a,c,p}	100,500
{a,c,f,m}	100, 200, 500
{a,f,m}	100, 200, 500
{a,f,c,p}	100, 500
{a,m,c,p}	100, 500
{c,f,m}	100, 200, 500
{c,f,p}	100, 500
{c,m,p}	100, 500
{c,p,f,m}	100, 500

itemset	TIDs
{a,c,f}	100, 200, 500
{a,c,m}	100,200,500
{a,c,f,m}	100, 200, 500
{a,f,m}	100, 200, 500
{c,f,m}	100, 200, 500

L3

ECLAT

itemset	TIDs
{a,c,f}	100, 200, 500
{a,c,m}	100,200,500
{a,c,f,m}	100, 200, 500
{a,f,m}	100, 200, 500
{c,f,m}	100, 200, 500

itemset	TIDs
{a,c,f,m}	100, 200, 500

itemset	TIDs
{a,c,f,m}	100, 200, 500

L4

ECLAT :Global Result

itemset	TIDs
{a}	100, 200,500
{b}	200,300,400
{c}	100,200,400, 500
{f}	100, 200, 300, 500
{m}	100, 200, 500
{p}	100, 400, 500

itemset	TIDs
{a,c}	100, 200, 500
{a,f}	100,200,500
{a,m}	100, 200, 500
{c,f}	100, 200, 500
{c,m}	100, 200, 500
{c,p}	100, 400, 500
{f,m}	100, 200, 500

itemset	TIDs
{a,c,f}	100, 200, 500
{a,c,m}	100,200,500
{a,c,f,m}	100, 200, 500
{a,f,m}	100, 200, 500
{c,f,m}	100, 200, 500

L={{a}, {b}, {c}, {f}, {m}, {p}, {a,c}, {a,f}, {a,m}, {c,f}, {c,m}, {c,p}, {f,m}, {a,c,f}, {a,c,m}, {a,c,f,m}, {a,f,m}, {c,f,m}}

Chapter 3: Mining Frequent Patterns and Associations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
  - Apriori
  - FP-growth
  - ECLAT
- Mining various kinds of association rules
- Summary

Mining Association Rules

- A frequent pattern can derive several association rules:
  - an arrow is introduced between the items of the frequent pattern
  - the part appearing before the arrow is called the **antecedent**
  - the part appearing after the arrow is called the **consequent**

- To a rule is associated a support and a confidence
  - the support corresponds to the support of the frequent pattern
  - the confidence is calculated as the probability that the consequent of the rule appears whenever the antecedent appears

Mining Association Rules : Examples

Let consider the frequent patterns calculated by the previous algorithms for the following transactions base:

min_support = 3

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

The set of frequent patterns is:

L={{a}, {b}, {c}, {f}, {m}, {p}, {a,c}, {a,f}, {a,m}, {c,f}, {c,m}, {c,p}, {f,m}, {a,c,f}, {a,c,m}, {a,c,f,m}, {a,f,m}, {c,f,m}}

Mining Association Rules : Examples

If we consider the frequent itemset **{a,c,m}** then the following association rules can be derived:

$\rightarrow$ a,c,m (60%,60%)

a,c $\rightarrow$ m (60%,100%)

a $\rightarrow$ c,m (60%,100%)

If we consider the frequent itemset **{f,c}** then the following association rules can be derived:

c $\rightarrow$ f (60%,75%)

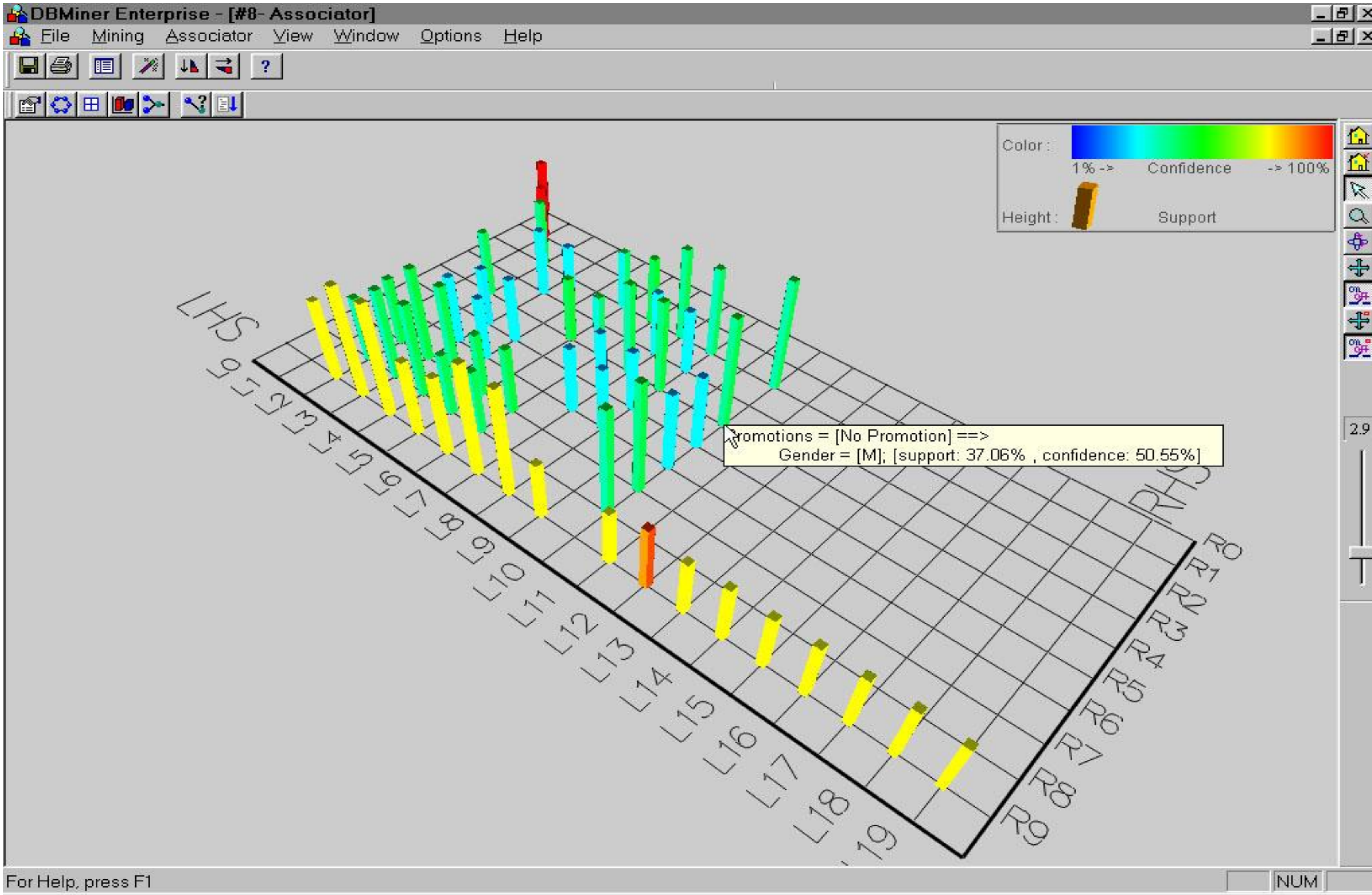
f $\rightarrow$ c (60%,75%)

If we consider the frequent itemset **{f,c,m}** then the following association rules can be derived:

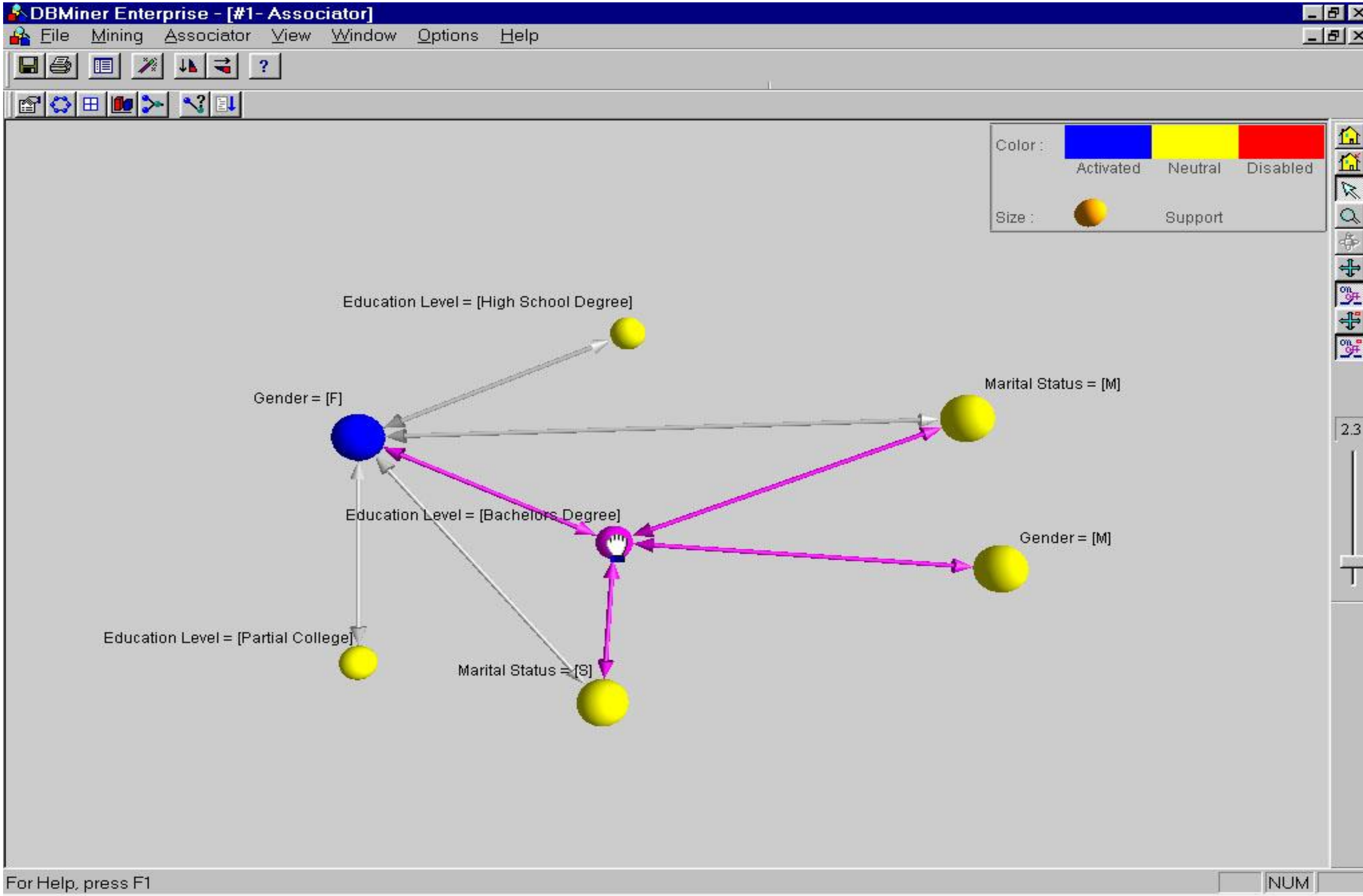
f $\rightarrow$ c,m (60%,75%)

c,m $\rightarrow$ f (60%,100%)

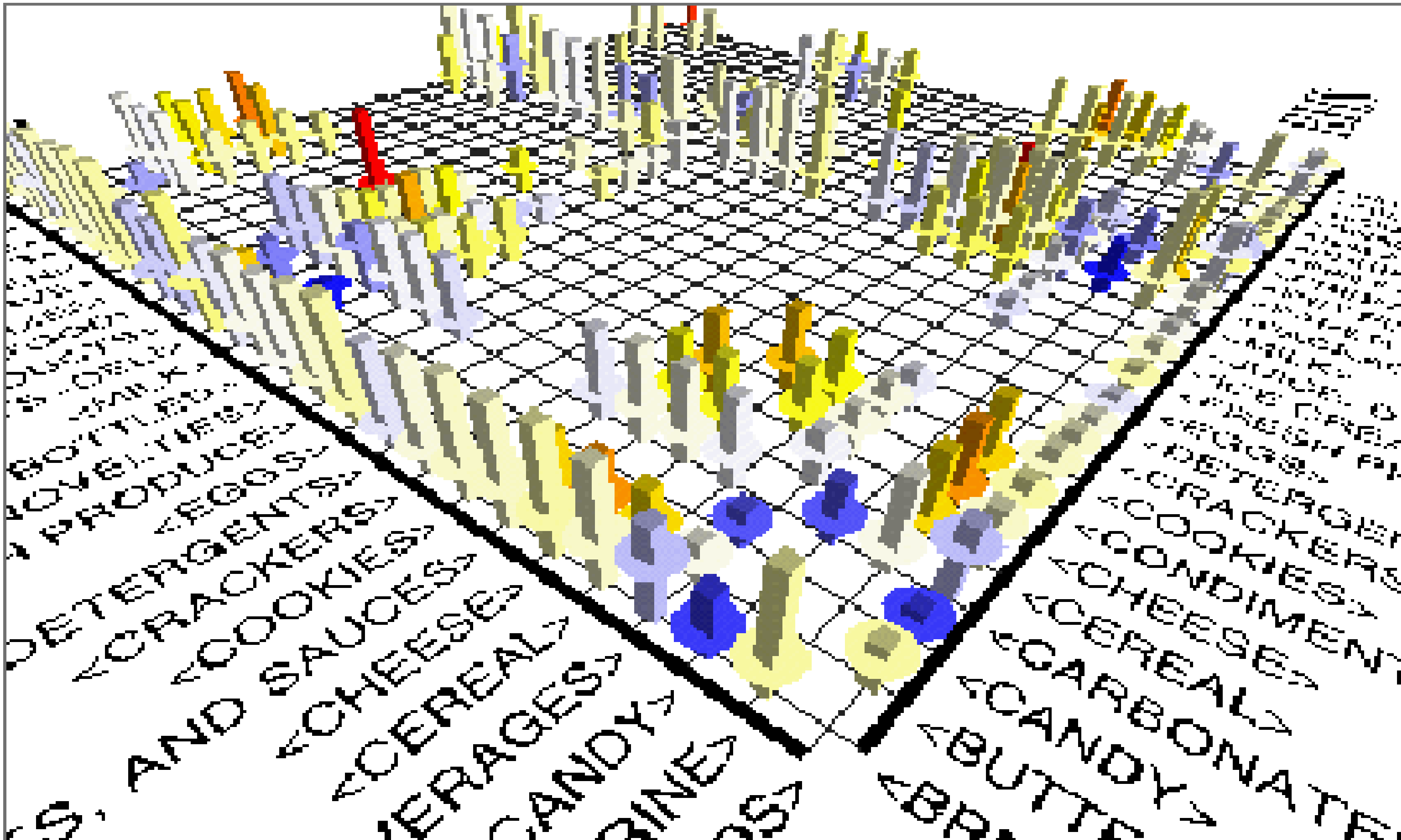
Visualization of Association Rules: Plane Graph



Visualization of Association Rules: Rule Graph



Visualization of Association Rules (SGI/MineSet 3.0)



Frequent-Pattern Mining: Summary

- Frequent pattern mining—an important task in data mining
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Projection-based (FPgrowth)
  - Vertical format approach (ECLAT)
- Mining a variety of rules

Ref: Apriori and Its Improvements

- R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94.
- H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. KDD'94.
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. VLDB'95.
- J. S. Park, M. S. Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95.
- H. Toivonen. Sampling large databases for association rules. VLDB'96.
- S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket analysis. SIGMOD'97.
- S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD'98.

Ref: Depth-First, Projection-Based FP Mining

- R. Agarwal, C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent itemsets. J. Parallel and Distributed Computing:02.
- J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD' 00.
- J. Pei, J. Han, and R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. DMKD'00.
- J. Liu, Y. Pan, K. Wang, and J. Han. Mining Frequent Item Sets by Opportunistic Projection. KDD'02.
- J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining Top-K Frequent Closed Patterns without Minimum Support. ICDM'02.
- J. Wang, J. Han, and J. Pei. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. KDD'03.
- G. Liu, H. Lu, W. Lou, J. X. Yu. On Computing, Storing and Querying Frequent Patterns. KDD'03.

Ref: Vertical Format and Row Enumeration Methods

- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. Parallel algorithm for discovery of association rules. DAMI:97.
- Zaki and Hsiao. CHARM: An Efficient Algorithm for Closed Itemset Mining, SDM'02.
- C. Bucila, J. Gehrke, D. Kifer, and W. White. DualMiner: A Dual-Pruning Algorithm for Itemsets with Constraints. KDD’02.
- F. Pan, G. Cong, A. K. H. Tung, J. Yang, and M. Zaki , CARPENTER: Finding Closed Patterns in Long Biological Datasets. KDD'03.