Principles of Knowledge Discovery in Data

Fall 2004

Chapter 6: Mining Association Rules

Dr. Osmar R. Zaïane

University of Alberta

Course Content

- Introduction to Data Mining
- Data warehousing and OLAP
- Data cleaning
- Data mining operations
- Data summarization
- **Association analysis**
- Classification and prediction
- Clustering
- Web Mining
- Spatial and Multimedia Data Mining
- *Other topics if time permits*

Chapter 6 Objectives

Understand association analysis in large datasets and get a brief introduction to the different types of association rule mining

Association Rules Outline

- What is association rule mining?
- How do we mine single-dimensional boolean associations?
- How do we mine multilevel associations?
- How do we mine multidimensional associations?
- Can we constrain the association mining?

What Is Association Mining?

- **Association rule mining searches for relationships between items in a dataset**:
  - Finding association, correlation, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
  - Rule form: "**Body → Head [support, confidence]**".
- **Examples:**
  - buys(x, "bread") → buys(x, "milk") [0.6%, 65%]
  - major(x, "CS") ^ takes(x, "DB") → grade(x, "A") [1%, 75%]

Association Rule Mining

mining association rules (Agrawal et. al SIGMOD93)
Fast algorithm (Agrawal et. al VLDB94)
Partitioning (Navathe et. al VLDB95)
Hash-based (Park et. al SIGMOD95)
Multilevel A.R. (Han et. al. VLDB95)
Generalized A.R. (Srikant et. Al. VLDB95)
Quantitative A.R. (Srikant et. al SIGMOD96)
Incremental mining (Cheung et. al ICDE96)
Parallel mining (Agrawal et. al TKDE96)
Distributed mining (Cheung et. al PDIS96)
Meta-ruleguided mining (Kamber et al. KDD97)
Direct Itemset Counting (Brin et. al SIGMOD97)
N-dimensional A.R. (Lu et. al DMKD'98)
Constraint A.R. (Ng et. al SIGMOD'98)
A.R. with recurrent items (Zaïane et. al ICDE'00)
FP without Candidate gen. (Han et. al SIGMOD'00)

And many many others:
Spatial AR; Sequence Associations;AR for multimedia; AR in time series;AR with progressive refinement; etc.

Basic Concepts

A transaction is a set of items: $T=\{i_a, i_b,\ldots i_t\}$

$T \subset I$, where I is the set of all possible items $\{i_1, i_2,\ldots i_n\}$

D, the task relevant data, is a set of transactions.

An association rule is of the form:
$P \rightarrow Q$, where $P \subset I$, $Q \subset I$, and $P \cap Q = \emptyset$

Basic Concepts (con't)

$P \rightarrow Q$ holds in D with support s
and
$P \rightarrow Q$ has a confidence c in the transaction set D.

Support($P \rightarrow Q$) = Probability($P \cup Q$)
Confidence($P \rightarrow Q$)=Probability(Q/P)

Itemsets

A set of items is referred to as itemset.

An itemset containing k items is called **k-itemset**.

An items set can also be seen as a conjunction of items (or a predicate)

Support and Confidence

- **Support** of $P = P_1 \wedge P_2 \wedge ... \wedge P_n$ in ***D***
  - $\sigma(P/D)$ is the percentage of transactions T in *D* satisfying P. (number of T by cardinality of *D*).
- **Confidence** of a rule $P \rightarrow Q$
  - $\varphi(P \rightarrow Q/ D)$ ratio $\sigma((P \wedge Q)/ D)$ by $\sigma(P/ D)$
- Thresholds:
  - *minimum support* σ'
  - *minimum confidence* φ'

Strong Rules

- **Frequent (or large) predicate** P in set D
  - support of P larger than minimum support,
- Rule $P \rightarrow Q$ (c%) is **strong**
  - predicate $(P \wedge Q)$ is frequent (or large),
  - c is larger than minimum confidence.

Different Kinds of Association Rules

- **Boolean vs. Quantitative Associations**
  - Association on discrete vs. continuous data
  - Ex. Age(X,30-45) $\wedge$ Income(X, 50K-75K) ➔ Buys(X, SUVcar)

  - **Boolean Association Rules**
  - **Quantitative Association Rules**

Different Kinds of Association Rules

- **Single dimension vs. multiple dimensional associations**
  - Based on the dimensions in data involved.
  - One predicate then single dimension. More predicates then multi-dimensions.
  - Ex. Buys(X, bread) ➔ Buys(X, milk)

    Age(X,30-45) ∧ Income(X, 50K-75K) ➔ Buys(X, SUVcar)

  - ➢ **Single-dimensional Association Rules**
  - ➢ **Multi-dimensional Association Rules**

Different Kinds of Association Rules

- **Single level vs. multiple-level analysis**
  - Based on the level of abstractions involved.
  - Find association rules at different levels of abstraction.
  - Ex. Buys(X, bread) ➔ Buys(X, milk)

    Buys(X, Wheat Bread) ➔ Buys(X, Formost 2% milk)


  - ➢ **Single-level Association Rules**
  - ➢ **Multi-level Association Rules**

Different Kinds of Association Rules

- **Single occurrence vs. multiple occurrences**
  - One item may occur more than once in the transaction.
  - Not the presence of the item is important but its frequency.
  - Ex. Buys(X, bread, 2) ➔ Buys(X, milk, 1)


  - ➢ **Single-occurrence-items Association Rules**
  - ➢ **Recurrent-items Association Rules**

Different Kinds of Association Rules

- **Simple vs. constraint-based**
  - Constraints can be added on the rules to be discovered

- **Association vs. correlation analysis**
  - Association does not necessarily imply correlation.

$$\frac{P(A \wedge B)}{P(A)P(B)} \quad =1? \quad >1? \quad <1?$$

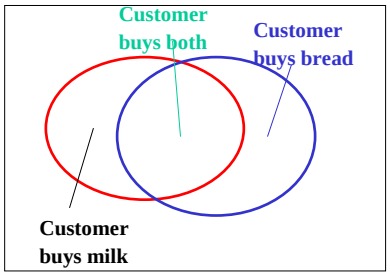
Association Rules Outline

- What is association rule mining?
- How do we mine single-dimensional boolean associations?
- How do we mine multilevel associations?
- How do we mine multidimensional associations?
- Can we constrain the association mining?

How do we Mine Association Rules?

- **Input**
  - A database of transactions
  - Each transaction is a list of items (Ex. purchased by a customer in a visit)
- Find all rules that associate the presence of one set of items with that of another set of items.
  - Example: *98% of people who purchase tires and auto accessories also get automotive services done*
  - There are no restrictions on the number of items in the head or body of the rule.

Rule Measures: Support and Confidence



Find all the rules X & Y ➔ Z with minimum confidence and support
- support, ***s***, probability that a transaction contains {X, Y, Z}
- confidence, ***c***, conditional probability that a transaction having {X, Y} also contains Z.

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Let minimum support 50%, and minimum confidence 50%, we have
- A ➔ C (50%, 66.6%)
- C ➔ A (50%, 100%)

Mining Association Rules

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Min. support 50%
Min. confidence 50%

Frequent Itemset	Support
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

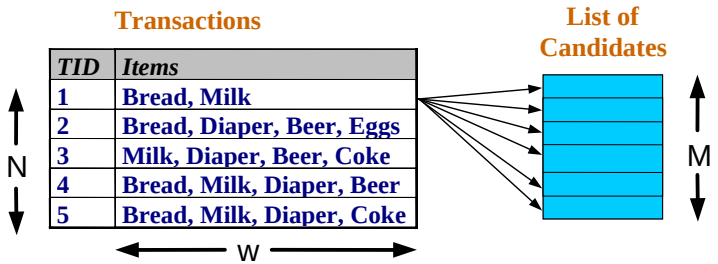
For rule A ➔ C:

support = support({A, C}) = 50%

confidence = support({A, C})/support({A}) = 66.6%

Frequent Itemset Generation

- Brute-force approach (Basic approach):
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database



  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => **Expensive since M = 2^d !!!**

An Influential Mining Methodology — The *Apriori* Algorithm

- The *Apriori* method:
  - Proposed by Agrawal & Srikant 1994
  - A similar level-wise algorithm by Mannila et al. 1994
- Major idea:
  - A subset of a frequent itemset must be frequent
    - E.g., if **{beer, diaper, nuts}** is frequent, **{beer, diaper}** must be.
      Any itemset that is infrequent, its superset cannot be frequent!
  - **A powerful, scalable candidate set pruning technique:**
    - **It reduces candidate k-itemsets dramatically (for k > 2)**

Mining Frequent Itemsets: the Key Step

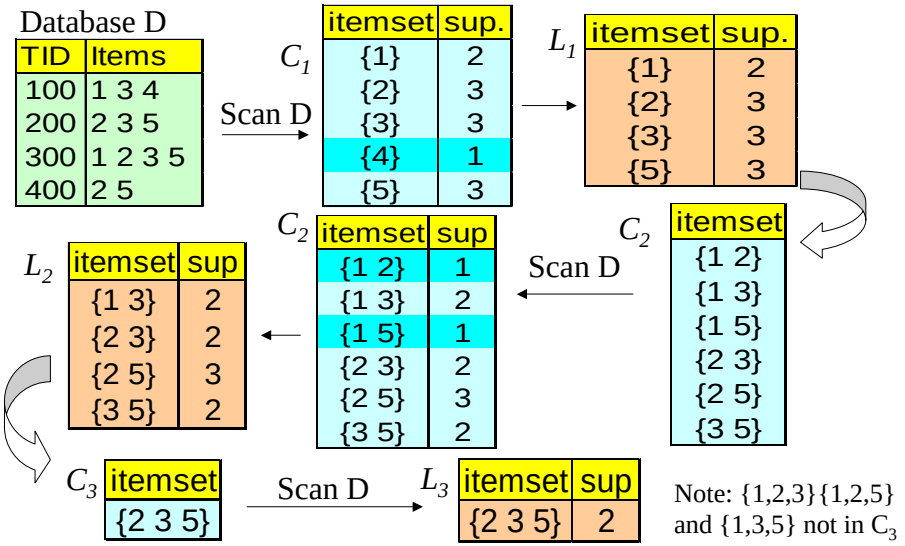
- Find the *frequent itemsets*: the sets of items that have minimum support
  - A subset of a frequent itemset must also be a frequent itemset, i.e., if {*AB*} is a frequent itemset, both {*A*} and {*B*} should be frequent itemsets
  - Iteratively find frequent itemsets with cardinality from 1 to *k* (*k*-itemsets)
- Use the frequent itemsets to generate association rules.

The Apriori Algorithm

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

L_1 = {frequent items};
for (k = 1; L_k !=∅; 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 $\cup_k L_k$;

The Apriori Algorithm -- Example



Note: {1,2,3}{1,2,5} and {1,3,5} not in C_3

Generating Association Rules from Frequent Itemsets

- Only strong association rules are generated.
- Frequent itemsets satisfy minimum support threshold.
- Strong AR satisfy minimum confidence threshold.

- Confidence(A➔B) = Prob(B/A) = $\frac{\text{Support}(A \cup B)}{\text{Support}(A)}$

For each frequent itemset, **f**, generate all non-empty subsets of **f**.
For every non-empty subset **s** of **f** **do**
output rule **s➔(f-s)** if support(**f**)/support(**s**) ≥ min_confidence
end

Improving efficiency of Apriori

- **Reducing the number of scans**
  (there are k DB scans for k-itemsets)
- **Eliminating scans by indexing** (Hashing)
- **Reducing sizes and number of transactions**
  (no need for non frequent items)
- **Partitioning**

Optimization: Direct Hash and Pruning

- DHP: Direct Hash and Pruning (Park, Chen and Yu, SIGMOD'95).
  - Reduce the size of candidate sets to minimize the cost
  - Reduce the size of the transaction database as well
- Using a hash table to keep track the counts of 2-itemset. Using the counts to prune C_2 (C_2 is usually the largest)
- An item in transaction *t* can be trimmed if it does not appear in at least *k* of the candidate *k*-itemsets in *t*.

Optimization: The Partitioning Algorithm

- Partition (Savasere, Omiecinski, & Navathe, VLDB'95).
  - Divide database into *n* partitions.
  - A frequent item must be frequent in at least one partition.
  - Process one partition in main memory at a time:
    - For each partition, generate frequent itemsets using the Apriori algorithm
    - also form *tidlist* for all itemsets to facilitate counting in the merge phase
  - After all partitions are processed, the local frequent itemsets are merged into global frequent sets; support can be computed from the *tidlists*.

Optimization: Sampling and Itemset Counting

- Sampling (Toivonen. VLDB'96).
  - A probabilistic approach finds association rules in about one pass.
- Dynamic Itemset Counting (Brin et. al. SIGMOD'97)
  - Reducing the number of scans over the transactions by starting to count itemsets dynamically during scans
  - Using data structure to keep track of counters and reordering items to reduce increment costs

Incremental Update of Discovered Rules

- Partitioned derivation and incremental updating.
- A fast updating algorithm, FUP (Cheung et al.'96)
  - View a database: original $DB \cup$ incremental *db*.
  - A *k*-itemset (for any *k*),
    - * **frequent** in $DB \cup db$ if frequent in both *DB* and *db*.
    - * **non frequent** in $DB \cup db$ if also in both *DB* and *db*.
  - For those only frequent in *DB*, merge corresponding counts in *db*.
  - For those only frequent in *db*, search *DB* to update their itemset counts.
- Similar methods can be adopted for data removal and update.
- Principles applicable to distributed/parallel mining.

Parallel and Distributed Mining

- **PDM** (Park et al.'95):
  - Use a hashing technique (DHP-like) to identify candidate *k*-itemsets from the local databases.
- **Count Distribution** (Agrawal & Shafer'96):
  - An extension of the Apriori algorithm.
  - May require a lot of messages in count exchange.
- **FDM** (Cheung et al.'96).
  - Observation: If an itemset *X* is globally large, there exists partition *Di* such that *X* and all its subsets are locally large at *Di*.
  - Candidate sets are those which are also local candidates in some component database, plus some message passing optimizations.

Interestingness Measures

- *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading
  - The overall percentage of students eating cereal is 75% which is higher than 66.7%.
- *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is less deceptive, although with lower support and confidence
- Measure of dependent/correlated events: lift

$$corr_{A,B} = \frac{P(A \cup B)}{P(A)P(B)}$$

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

Adding Correlations or Lifts to Support and Confidence

- Example
  - X and Y: positively correlated,
  - X and Z, negatively related
  - support and confidence of X=>Z dominates

X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

- We need a measure of dependent or correlated events

$$corr_{A,B} = \frac{P(A \cup B)}{P(A)P(B)}$$

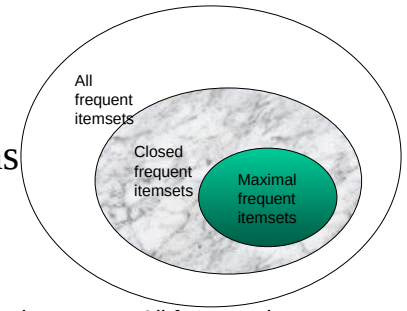
Rule	Support	Confidence
X=>Y	25%	50%
X=>Z	37.50%	75%

- P(B|A)/P(B) is also called the **lift** of rule A => B

Other Frequent Patterns

- Frequent pattern $\{a_1, \ldots, a_{100}\} \rightarrow \binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100}-1 = 1.27*10^{30}$ frequent sub-patterns!

- Frequent Closed Patterns
- Frequent Maximal Patterns
- All Frequent Patterns



Maximal frequent itemsets $\subseteq$ Closed frequent itemsets $\subseteq$ All frequent itemset

Frequent Closed Patterns

- For frequent itemset X, if there exists no item y such that every transaction containing X also contains y, then X is a frequent closed pattern
- In other words, frequent itemset X is closed if there is no item y, not already in X, that always accompanies X in all transactions where X occurs.
- Concise representation of frequent patterns. Can generate all frequent patterns with their support from frequent closed ones.
- Reduce number of patterns and rules
- N. Pasquier et al. In ICDT'99

{abcd}
{abc}
{bd}
Transactions
Support = 2

a	2
b	3
c	2
d	2
ab	2
ac	2
bc	2
bd	2
abc	2

Frequent itemsets

b	3
bd	2
abc	2

Frequent Closed itemsets

Frequent Maximal Patterns

- Frequent itemset X is maximal if there is no other frequent itemset Y that is superset of X.
- In other words, there is no other frequent pattern that would include a maximal pattern.
- More concise representation of frequent patterns but the information about supports is lost.
- Can generate all frequent patterns from frequent maximal ones but without their respective support.
- R. Bayardo. In SIGMOD'98

{abcd}
{abc}
{bd}
Transactions
Support = 2

a	2
b	3
c	2
d	2
ab	2
ac	2
bc	2
bd	2
abc	2

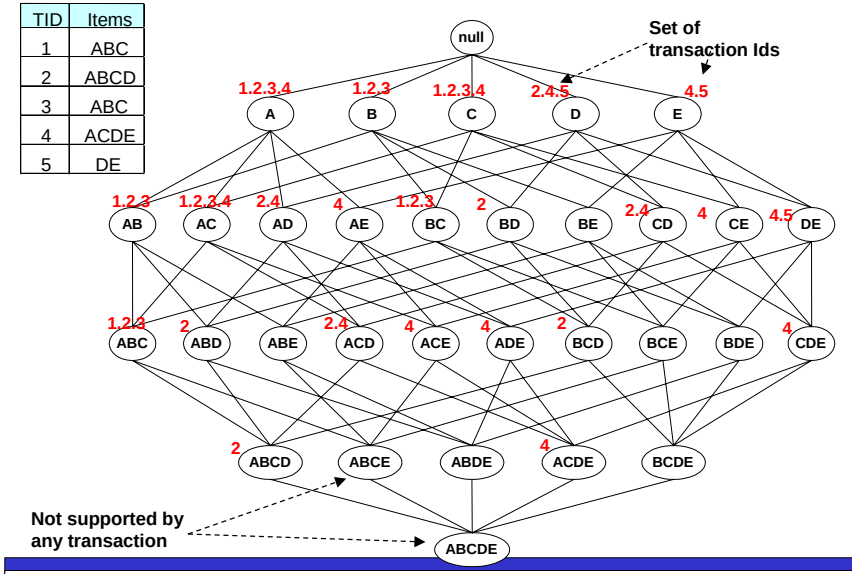
Frequent itemsets

bd	2
abc	2

Frequent Maximal itemsets

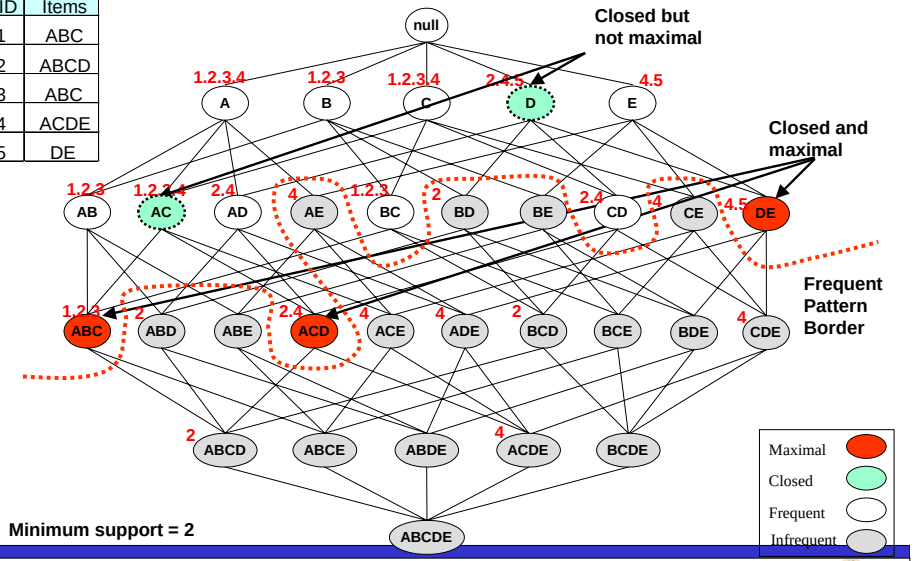
Maximal vs. Closed Itemsets

TID	Items
1	ABC
2	ABCD
3	ABC
4	ACDE
5	DE

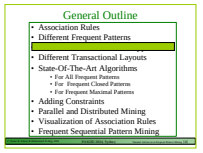


Maximal vs. Closed Itemsets

TID	Items
1	ABC
2	ABCD
3	ABC
4	ACDE
5	DE

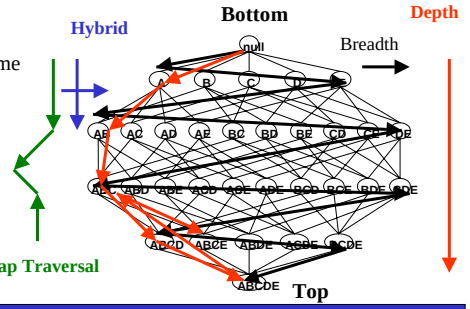


Mining the Pattern Lattice

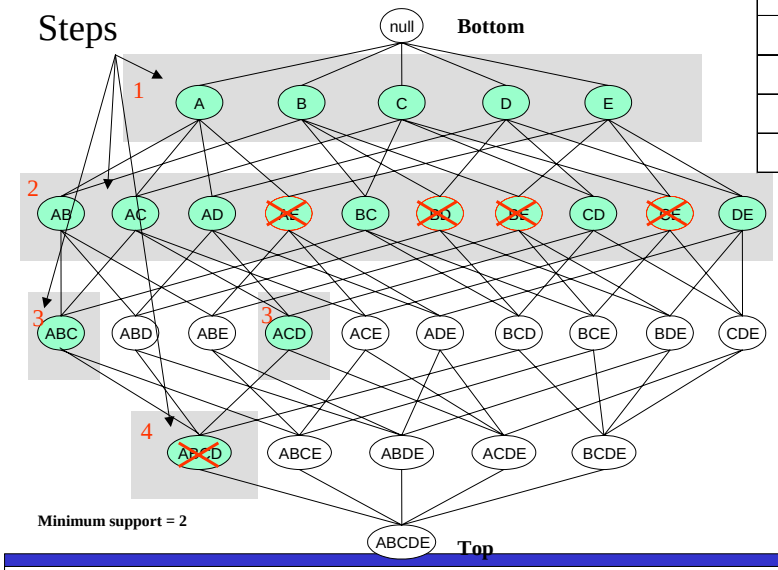


- Breadth-First
  - It uses current items at level k to generate items of level k+1 (many database scans)
- Depth-First
  - It uses a current item at level k to generate all its supersets (favored when mining long frequent patterns)
- Hybrid approach
  - It mines using both direction at the same time
- Leap traversal approach
  - Jumps to selected nodes

There is also the notion of :
Top-down (level k then level k+1)
Bottom-up (level k+1 then level k)



Breadth- First (Bottom-Up Example)

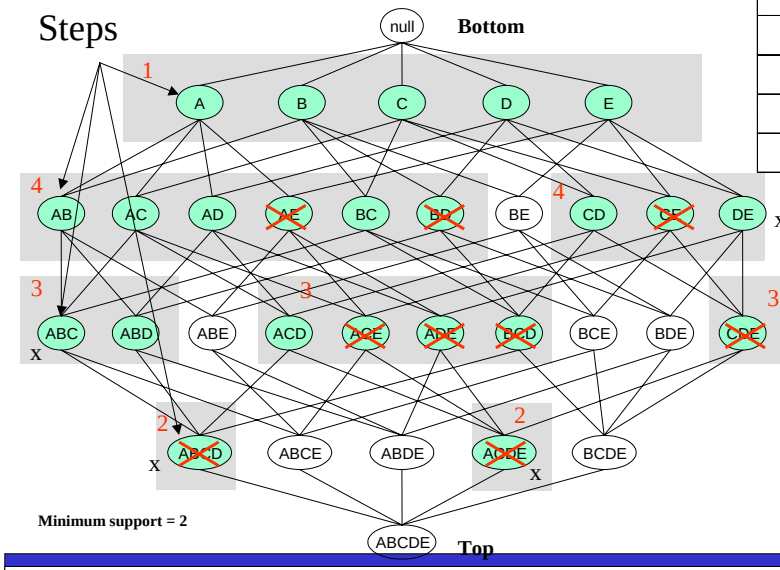


TID	Items
1	ABC
2	ABCD
3	ABC
4	ACDE
5	DE

Superset is candidate if ALL its subsets are frequent

18 candidates to check

Depth First (Top-Down Example)

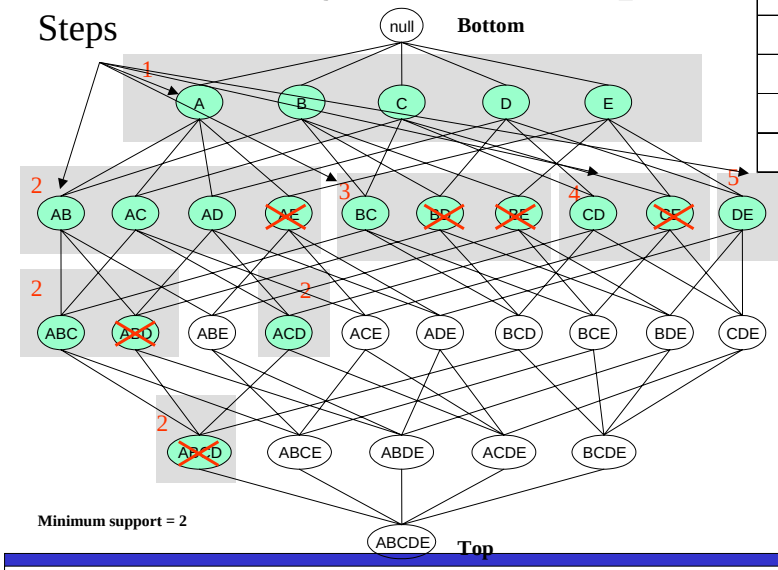


TID	Items
1	ABC
2	ABCD
3	ABC
4	ACDE
5	DE

Subset is candidate if it is marked or if one of its supersets is candidate

23 candidates to check

One Hybrid Example

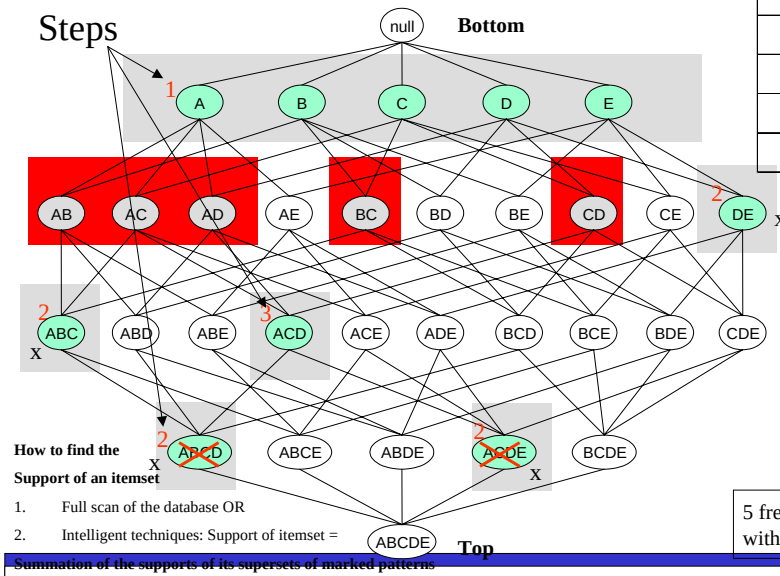


TID	Items
1	ABC
2	ABCD
3	ABC
4	ACDE
5	DE

Superset is candidate if ALL its subsets are frequent

19 candidates to check

Leap Traversal Example



TID	Items
1	ABC
2	ABCD
3	ABC
4	ACDE
5	DE

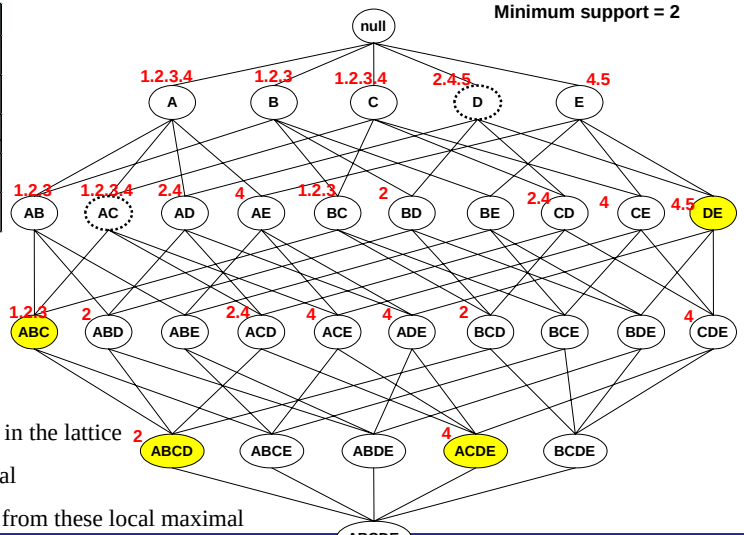
Itemset is candidate if it is marked or if it is a subset of more than one infrequent marked superset

10 candidates to check

5 frequent patterns without checking

Finding Maximal using leap traversal approach

TID	Items
1,3	ABC
2	ABCD
3	ABC
4	ACDE
5	DE



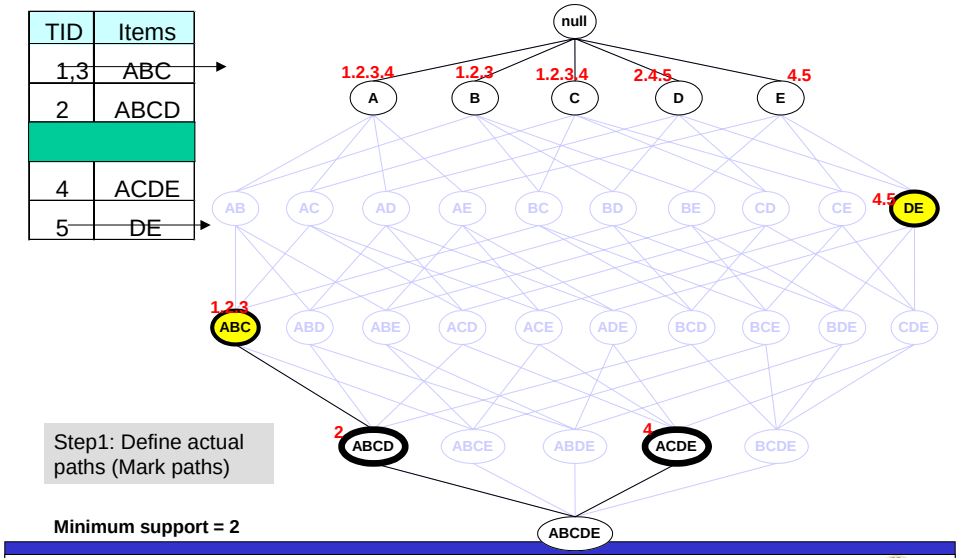
-Selectively jumps in the lattice

-Find local Maximal

-Generate patterns from these local maximal

Finding Maximal using leap traversal approach

TID	Items
1,3	ABC
2	ABCD
4	ACDE
5	DE

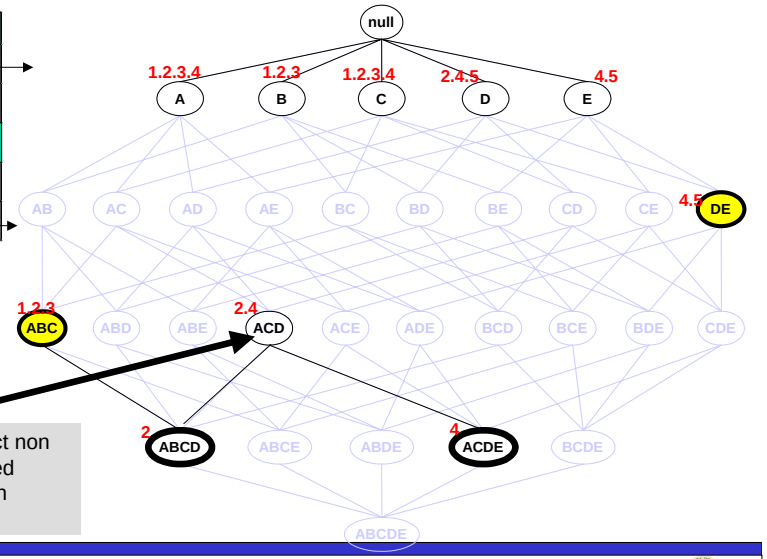


Step1: Define actual paths (Mark paths)

Minimum support = 2

Finding Maximal using leap traversal approach

TID	Items
1,3	ABC
2	ABCD
4	ACDE
5	DE

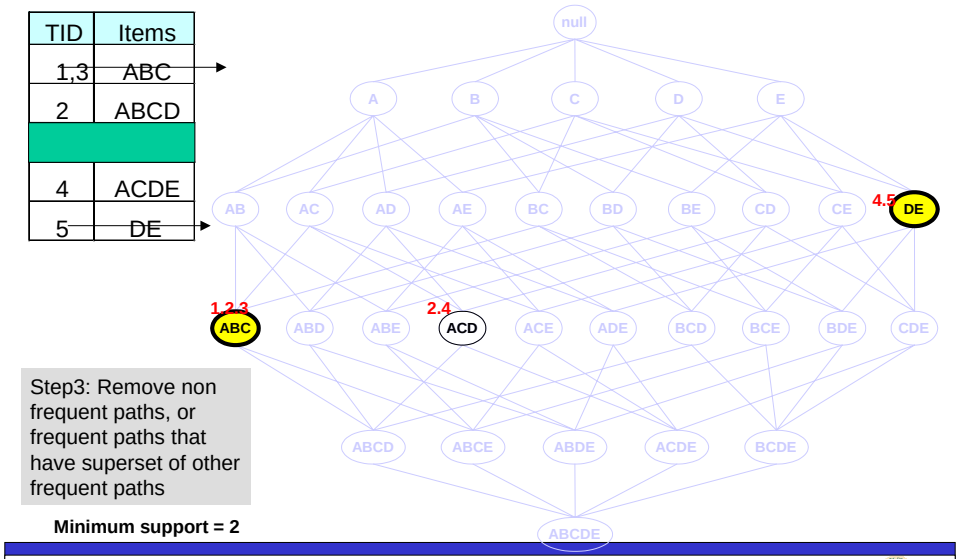


Intersect ABCD with ACDE

Step2: Intersect non frequent marked paths with each others

Finding Maximal using leap traversal approach

TID	Items
1,3	ABC
2	ABCD
4	ACDE
5	DE

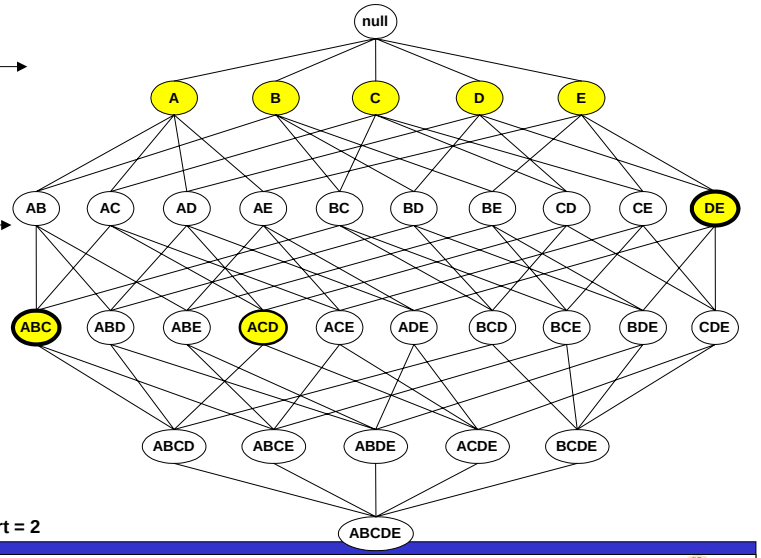


Step3: Remove non frequent paths, or frequent paths that have superset of other frequent paths

Minimum support = 2

Finding Maximal using leap traversal approach

TID	Items
1,3	ABC
2	ABCD
4	ACDE
5	DE



Minimum support = 2

When to use a Given Strategy

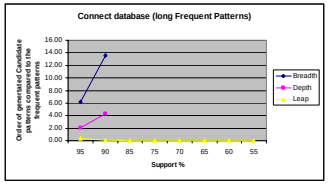
- Breadth First
  - Suitable for short frequent patterns
  - Unsuitable for long frequent patterns
- Depth First
  - Suitable for long frequent patterns
  - In general not scalable when long candidate patterns are not frequent
- Leap Traversal
  - Suitable for cases having short and long frequent patterns simultaneously

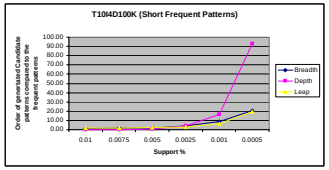
Empirical Tests

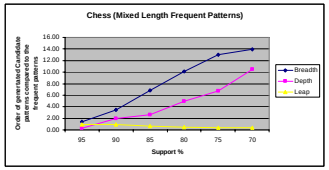
Connect						
				Total Candidates Created		
Support	Size of Largest	F1-Size	Total Frequent	Breadth	Depth	Leap
95	9	17	2205	15626	6654	3044
90	12	21	27127	394648	144426	29263
80	15	28	119418			120177
75	17	30	176365			177244
70	18	31	627952			628963
65	19	33	1368337			1369585
60	20	36	2908632			2910175
55	21	37	5996892			5998715

T10I4D100K						
				Total Candidates Created		
Support	Size of Largest	F1-Size	Total Frequent	Breadth	Depth	Leap
1000	3	375	385	20	10	29
750	4	463	561	262	124	291
500	5	569	1073	1492	731	1546
250	8	717	7703	36994	36309	26666
100	10	797	27532	264645	458275	200113
50	10	839	53385	1129779	4923723	1067050

Chess						
				Total Candidates Created		
Support	Size of Largest	F1-Size	Total Frequent	Breadth	Depth	Leap
95	5	9	78	165	90	136
90	7	13	628	2768	1842	1191
85	8	16	2690	20871	9667	4318
80	10	20	8282	91577	49196	12021
75	11	23	20846	292363	160362	28986
70	13	24	48939	731740	560103	65093







Transactional Layouts

- Horizontal Layout

Each transaction is recorded as a list of items

Transaction ID	Items				
1	A	G	D	C	B
2	B	C	H	E	D
3	B	D	E	A	M
4	C	E	F	A	N
5	A	B	N	O	P
6	A	C	Q	R	G
7	A	C	H	I	G
8	L	E	F	K	B
9	A	F	M	N	O
10	C	F	P	J	R
11	A	D	B	H	I
12	D	E	B	K	L
13	M	D	C	G	O
14	C	F	P	Q	J
15	B	D	E	F	I
16	J	E	B	A	D
17	A	K	E	F	C
18	C	D	L	B	A

Candidacy generation can be removed (FP-Growth)

Superfluous Processing

Transactional Layouts

- Vertical Layout

Tid-list is kept for each item

Transaction ID	Items
1	A G D C B
2	B C H E D
3	B D E A M
4	C E F A N
5	A B N O P
6	A C Q R G
7	A C H I G
8	L E F K B
9	A F M N O
10	C F P J R
11	A D B H I
12	D E B K L
13	M D C G O
14	C F P Q J
15	B D E F I
16	J E B A D
17	A K E F C
18	C D L B A

Items	Transaction ID
A	1 3 4 5 6 7 9 11 16 17 18
B	1 2 3 5 8 11 12 15 16 18
C	1 2 4 6 7 10 13 14 17 18
D	1 2 3 11 12 13 15 16 18
E	2 3 4 8 12 15 16 17
F	4 8 9 10 14 15 17
G	1 6 7 13
H	2 7 11
I	7 11 15
J	10 14 16
K	8 12 17
L	8 12 18
M	3 9 13
N	4 5 9
O	5 9 13
P	5 9 13
Q	6 14
R	6 10

Minimize Superfluous Processing

Candidacy generation is required

Transactional Layouts

- Bitmap Layout

Matrix : Rows represent transactions
Columns represent item
If item exists in transaction
then cell value = 1 else cell value = 0

T#	Items
T1	A G D C B
T2	B C H E D
T3	B D E A M
T4	C E F A N
T5	A B N O P
T6	A C Q R G
T7	A C H I G
T8	L E F K B
T9	A F M N O
T10	C F P J R
T11	A D B H I
T12	D E B K L
T13	M D C G O
T14	C F P Q J
T15	B D E F I
T16	J E B A D
T17	A K E F C
T18	C D L B A

Transaction ID	items																	
T#	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
T2	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
T3	1	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0
T4	1	0	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0
T5	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0
T6	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1
T7	1	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
T8	0	1	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0
T9	1	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0
T10	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0
T11	1	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0
T12	0	1	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0
T13	0	0	1	1	0	0	1	0	0	0	0	0	1	0	1	0	0	0
T14	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0
T15	0	1	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0
T16	1	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0
T17	1	0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0
T18	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Similar to horizontal layout.
Suitable for datasets with small dimensionality

Transactional Layouts

- Inverted Matrix Layout

El-Hajj and Zaiane, ACM SIGKDD'03

Minimize Superfluous Processing

Candidacy generation can be reduced

Appropriate for Interactive Mining

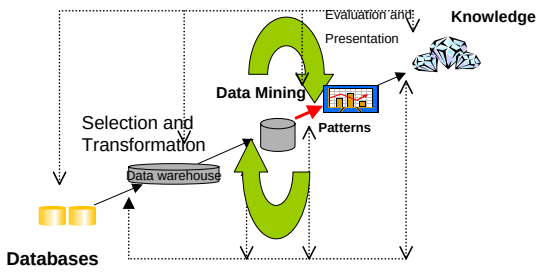
T#	Items
T1	A G D C B
T2	B C H E D
T3	B D E A M
T4	C E F A N
T5	A B N O P
T6	A C Q R G
T7	A C H I G
T8	L E F K B
T9	A F M N O
T10	C F P J R
T11	A D B H I
T12	D E B K L
T13	M D C G O
T14	C F P Q J
T15	B D E F I
T16	J E B A D
T17	A K E F C
T18	C D L B A

Loc	Index		Transactional Array										
			1	2	3	4	5	6	7	8	9	10	11
1	R	2	(2,1)	(3,2)									
2	Q	2	(12,2)	(3,3)									
3	P	3	(4,1)	(9,1)	(9,2)								
4	O	3	(5,2)	(5,3)	(6,3)								
5	N	3	(13,1)	(17,4)	(6,2)								
6	M	3	(14,2)	(13,3)	(12,4)								
7	L	3	(8,1)	(8,2)	(15,9)								
8	K	3	(13,2)	(14,5)	(13,7)								
9	J	3	(13,4)	(13,5)	(14,7)								
10	I	3	(11,2)	(11,3)	(13,6)								
11	H	3	(14,1)	(12,3)	15,4)								
12	G	4	**(15,1)**	(16,4)	(16,5)	(15,6)							
13	F	7	(14,3)	(14,4)	(18,7)	(16,6)	(16,8)	(14,6)	(14,8)				
14	E	8	(15,2)	(15,3)	(16,3)	(17,5)	(15,5)	(15,7)	(15,8)	(16,9)			
15	D	9	**(16,1)**	(16,2)	(17,2)	(17,6)	(17,7)	(16,7)	(17,8)	(17,9)	(16,10)		
16	C	10	**(17,1)**	(17,2)	(18,3)	(18,5)	(18,6)	(¤, ¤)	(¤, ¤)	(¤, ¤)	(18,10)	(17,10)	
17	B	10	**(18,1)**	(¤, ¤)	(18,2)	(18,4)	(¤, ¤)	(18,8)	(¤, ¤)	(¤, ¤)	(18,9)	(18,11)	
18	A	11	(¤, ¤)	(¤, ¤)	(¤, ¤)	(¤, ¤)	(¤, ¤)	(¤, ¤)	(¤, ¤)	(¤, ¤)	(¤, ¤)	(¤, ¤)	(¤, ¤)

Why The Matrix Layout?

Interactive mining

Changing the support level means expensive steps (whole process is redone)



Why The Matrix Layout?

Repetitive tasks, (I/O) readings (Superfluous Processing)

T#	Items				
T1	A	G	D	C	B
T2	B	C	H	E	D
T3	B	D	E	A	M
T4	C	E	F	A	N
T5	A	B	N	O	P
T6	A	C	Q	R	G
T7	A	C	H	I	G
T8	L	E	F	K	B
T9	A	F	M	N	O
T10	C	F	P	J	R
T11	A	D	B	H	I
T12	D	E	B	K	L
T13	M	D	C	G	O
T14	C	F	P	Q	J
T15	B	D	E	F	I
T16	J	E	B	A	D
T17	A	K	E	F	C
T18	C	D	L	B	A

Support > 4

Frequent 1-itemsets {A, B, C, D, E, F}
Non frequent items {G, H, I, J, K, L, M, N, O, P, Q, R}

T#	Items				
T1	A		D	C	B
T2	B	C		E	D
T3	B	D	E	A	
T4	C	E	F	A	
T5	A	B			
T6	A	C			
T7	A	C			
T8		E	F		B
T9	A	F			
T10	C	F			
T11	A	D	B		
T12	D	E	B		
T13		D	C		
T14	C	F			
T15	B	D	E	F	
T16		E	B	A	D
T17	A		E	F	C
T18	C	D		B	A

Why The Matrix Layout?

Repetitive tasks, (I/O) readings (Superfluous Processing)

T#	Items				
T1	A	G	D	C	B
T2	B	C	H	E	D
T3	B	D	E	A	M
T4	C	E	F	A	N
T5	A	B	N	O	P
T6	A	C	Q	R	G
T7	A	C	H	I	G
T8	L	E	F	K	B
T9	A	F	M	N	O
T10	C	F	P	J	R
T11	A	D	B	H	I
T12	D	E	B	K	L
T13	M	D	C	G	O
T14	C	F	P	Q	J
T15	B	D	E	F	I
T16	J	E	B	A	D
T17	A	K	E	F	C
T18	C	D	L	B	A

Support > 9

Frequent 1-itemsets {A, B, C}
Non frequent items
{D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R}

T#	Items			
T1	A		C	B
T2	B	C		
T3	B		A	
T4	C		A	
T5	A	B		
T6	A	C		
T7	A	C		
T8				B
T9	A			
T10	C			
T11	A		B	
T12			B	
T13			C	
T14	C			
T15	B			
T16		B	A	
T17	A			C
T18	C		B	A

Transactional Layouts

- Inverted Matrix Layout

Support > 4

Loc	Index	Transactional Array										
		1	2	3	4	5	6	7	8	9	10	11
1	R 2	(2,1)	(3,2)									
2	Q 2	(12,2)	(3,3)									
3	P 3	(4,1)	(9,1)	(9,2)								
4	O 3	(5,2)	(5,3)	(6,3)								
5	N 3	(13,1)	(17,4)	(6,2)								
6	M 3	(14,2)	(13,3)	(12,4)								
7	L 3	(8,1)	(8,2)	(15,9)								
8	K 3	(13,2)	(14,5)	(13,7)								
9	J 3	(13,4)	(13,5)	(14,7)								
10	I 3	(11,2)	(11,3)	(13,6)								
11	H 3	(14,1)	(12,3)	15,4)								
12	G 4	(15,1)	(16,4)	(16,5)	(15,6)							
13	F 7	(14,3)	(14,4)	(18,7)	(16,6)	(16,8)	(14,6)	(14,8)				
14	E 8	(15,2)	(15,3)	(16,3)	(17,5)	(15,5)	(15,7)	(15,8)	(16,9)			
15	D 9	(16,1)	(16,2)	(17,2)	(17,6)	(17,7)	(16,7)	(17,8)	(17,9)	(16,10)		
16	C 10	(17,1)	(17,2)	(18,3)	(18,5)	(18,6)	(☺,☺)	(☺,☺)	(☺,☺)	(18,10)	(17,10)	
17	B 10	(18,1)	(☺,☺)	(18,2)	(18,4)	(☺,☺)	(18,8)	(☺,☺)	(☺,☺)	(18,9)	(18,11)	
18	A 11	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)

Border
Support

Transactional Layouts

- Inverted Matrix Layout

Loc	Index	Transactional Array										
		1	2	3	4	5	6	7	8	9	10	11
13	F 7	(14,3)	(14,4)	(18,7)	(16,6)	(16,8)	(14,6)	(14,8)				
14	E 8	(15,2)	(15,3)	(16,3)	(17,5)	(15,5)	(15,7)	(15,8)	(16,9)			
15	D 9	(16,1)	(16,2)	(17,2)	(17,6)	(17,7)	(16,7)	(17,8)	(17,9)	(16,10)		
16	C 10	(17,1)	(17,2)	(18,3)	(18,5)	(18,6)	(☺,☺)	(☺,☺)	(☺,☺)	(18,10)	(17,10)	
17	B 10	(18,1)	(☺,☺)	(18,2)	(18,4)	(☺,☺)	(18,8)	(☺,☺)	(☺,☺)	(18,9)	(18,11)	
18	A 11	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)	(☺,☺)

Transactional Layouts

- Inverted Matrix Layout

Loc	Index		Transactional Array										
			1	2	3	4	5	6	7	8	9	10	11
13	F	7	(14,3)	(14,4)	(18,7)	(16,6)	(16,8)	(14,6)	(14,8)				
14	E	8	(15,2)	(15,3)	(16,3)	(17,5)	(15,5)	(15,7)	(15,8)	(16,9)			
15	D	9	(16,1)	(16,2)	(17,2)	(17,6)	(17,7)	(16,7)	(17,8)	(17,9)	(16,10)		
16	C	10	(17,1)	(17,2)	(18,3)	(18,5)	(18,6)	(☹,☹)	(☹,☹)	(☹,☹)	(18,10)	(17,10)	
17	B	10	(18,1)	(☹,☹)	(18,2)	(18,4)	(☹,☹)	(18,8)	(☹,☹)	(☹,☹)	(18,9)	(18,11)	
18	A	11	(☹,☹)	(☹,☹)	(☹,☹)	(☹,☹)	(☹,☹)	(☹,☹)	(☹,☹)	(☹,☹)	(☹,☹)	(☹,☹)	(☹,☹)

T#	Items			
T1	A	D	C	B
T2	B	C	E	D
T3	B	D	E	A
T4	C	E	F	A
T5	A	B		
T6	A	C		
T7	A	C		
T8	E	F	B	
T9	A	F		
T10	C	F		
T11	A	D	B	
T12	D	E	B	
T13	D	C		
T14	C	F		
T15	B	D	E	F
T16	E	B	A	D
T17	A	E	F	C
T18	C	D	B	A

The Algorithms
(State of the Art)

All

Apriori, FP-Growth, COFI*, ECLAT

Closed

CHARM, CLOSET+,COFI-CLOSED

Maximal

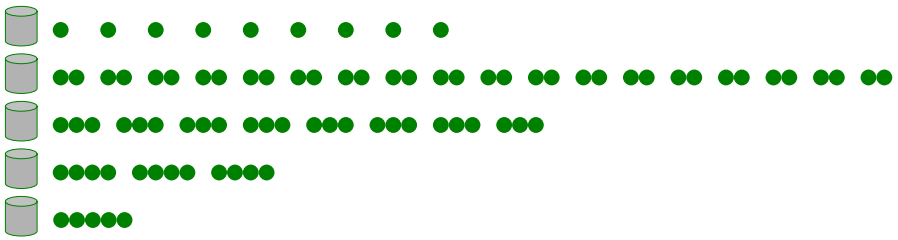
MaxMiner, MAFIA, GENMAX, COFI-MAX

All-Apriori

Apriori

Repetitive I/O scans

Huge Computation to generate candidate items



R. Agrawal, R. Srikant, VLDB'94

All-Apriori

Problems with Apriori

- Generation of candidate itemsets are expensive (Huge candidate sets)
  - 10^4 frequent 1-itemset will generate 10^7 candidate 2-itemsets
  - To discover a frequent pattern of size 100, e.g., $\{a_1, a_2, \ldots, a_{100}\}$, one needs to generate $2^{100} \approx 10^{30}$ candidates.
- High number of data scans

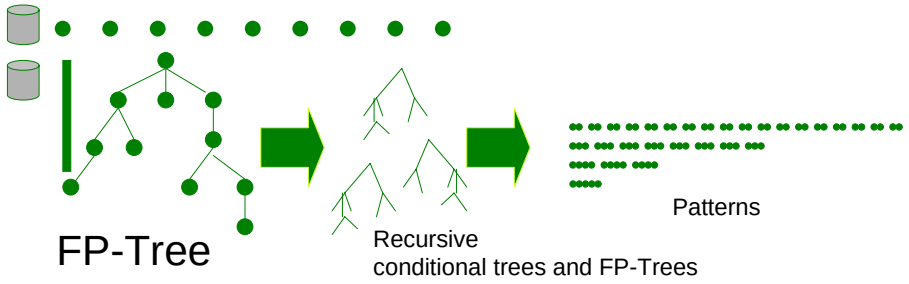
Frequent Pattern Growth

- First algorithm that allows frequent pattern mining without generating candidate sets
- Requires Frequent Pattern Tree

All-FP-Growth

FP-Growth

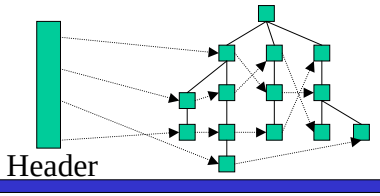
- 2 I/O scans
- Reduced candidacy generation
- High memory requirements
- Claims to be 1 order of magnitude faster than Apriori



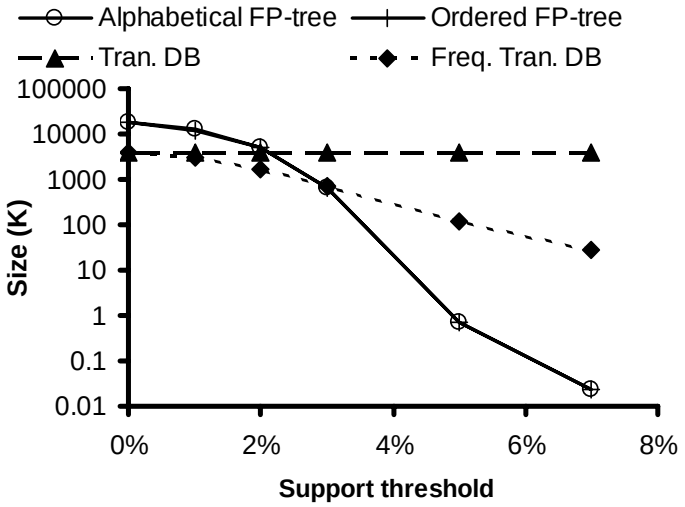
J. Han, J. Pei, Y. Yin, SIGMOD'00

Frequent Pattern Tree

- Prefix tree
- Each node contains the item name, frequency and pointer to another node of the same kind
- Frequent item header that contains item names and pointer to the first node in FP tree



Database Compression Using FP-tree (T10I4D100k)



All-FP-Growth

Frequent Pattern Tree

F, A, C, D, G, I, M, P
A, B, C, F, L, M, O
B, F, H, J, O
A, F, C, E, L, P, M, N
B, C, K, S, P
F, M, C, B, A

Required Support: **3**

F:5, C:5, A:4, B:4, M:4, P:3 D:1 E:1 G:1 H:1 I:1 J:1 K:1 L:1 O:1

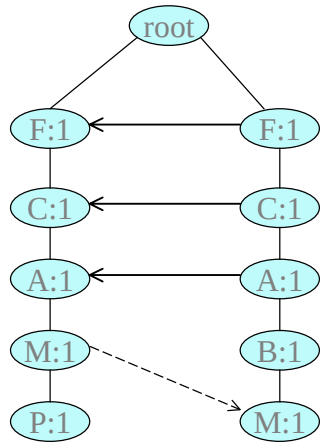
Frequent Pattern Tree

Original Transaction	Ordered frequent items
F, A, C, D, G, I, M, P	F, C, A, M, P
A, B, C, F, L, M, O	F, C, A, B, M
B, F, H, J, O	F, B
A, F, C, E, L, P, M, N	C, B, P
B, C, K, S, P	F, C, A, M, P
F, M, C, B, A	F, C, A, M
F, B, D	F, B

F:5, C:5, A:4, B:4, M:4, P:3 Required Support: **3**

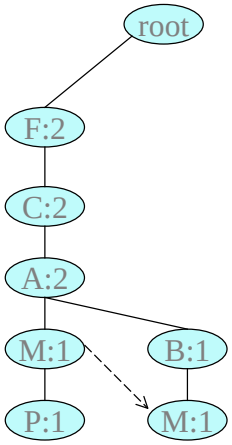
F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B

Frequent Pattern Tree



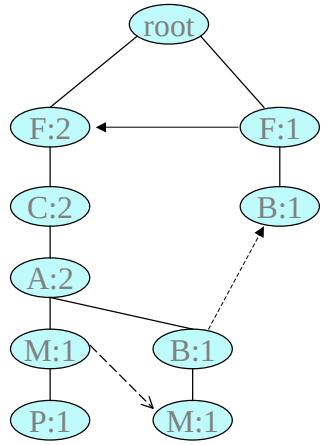
F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B

Frequent Pattern Tree



F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B

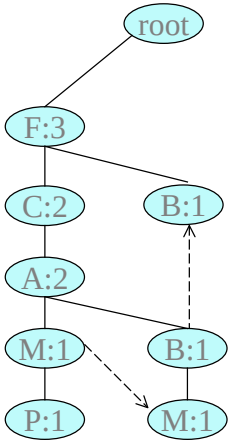
Frequent Pattern Tree



All-FP-Growth

F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B

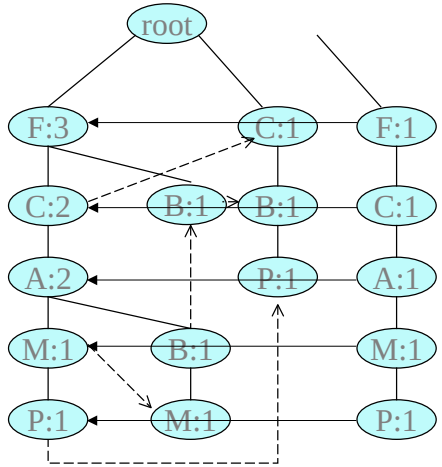
Frequent Pattern Tree



All-FP-Growth

F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B

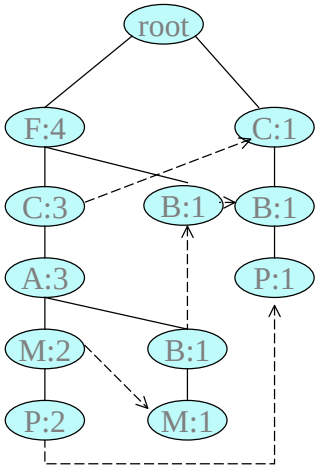
Frequent Pattern Tree



All-FP-Growth

F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B

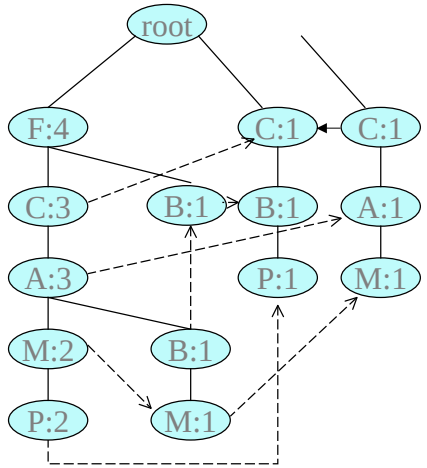
Frequent Pattern Tree



All-FP-Growth

F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B

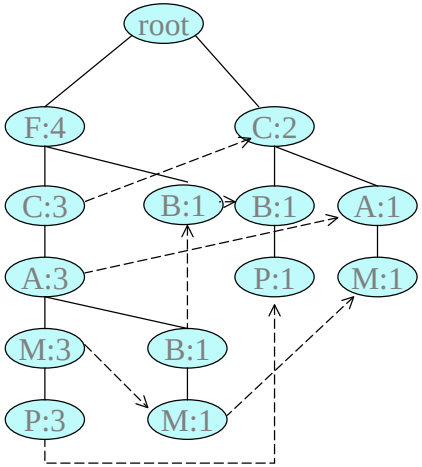
Frequent Pattern Tree



All-FP-Growth

F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B

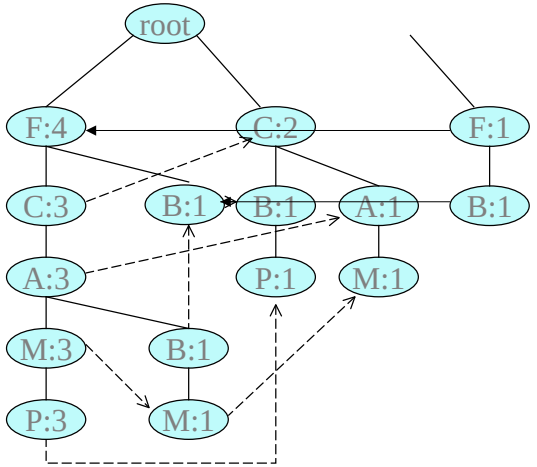
Frequent Pattern Tree



All-FP-Growth

F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B

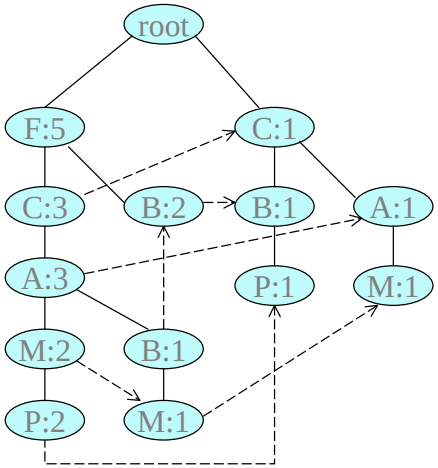
Frequent Pattern Tree



All-FP-Growth

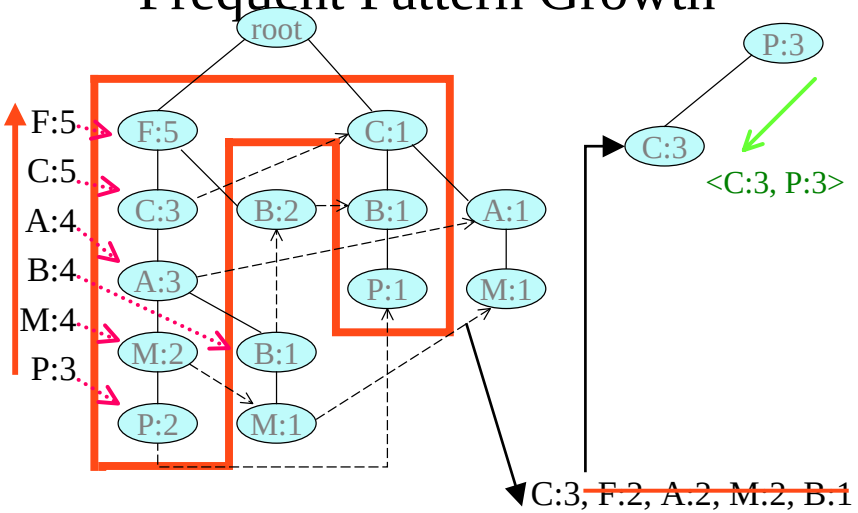
F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B

Frequent Pattern Tree

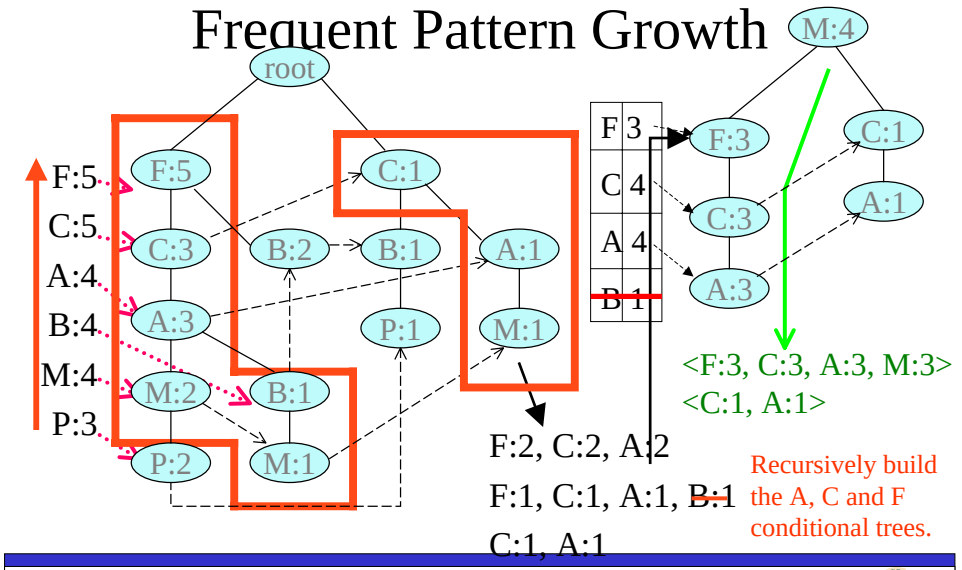


All-FP-Growth

Frequent Pattern Growth



All-FP-Growth

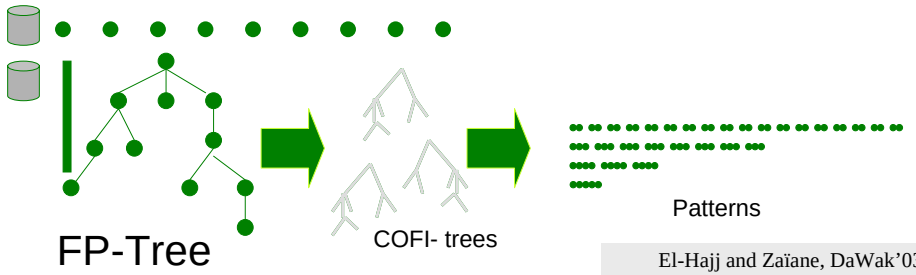


All-COFI

COFI algorithm big picture

COFI

- 2 I/O scans
- reduced candidacy generation
- Small memory footprint



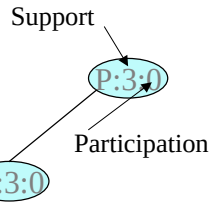
El-Hajj and Zaïane, DaWak'03

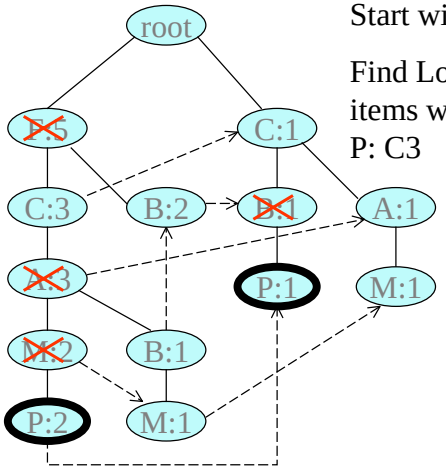
All-COFI

Co-Occurrences Frequent Item tree

Start with item P:

Find Locally frequent items with respect to P: C3



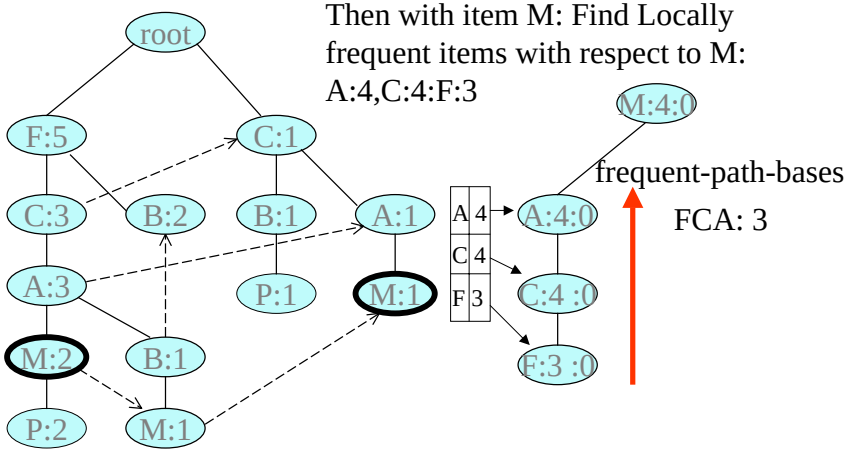


PC:3 frequent-path-base
All subsets of PC:3 are frequent and have the same support

All-COFI

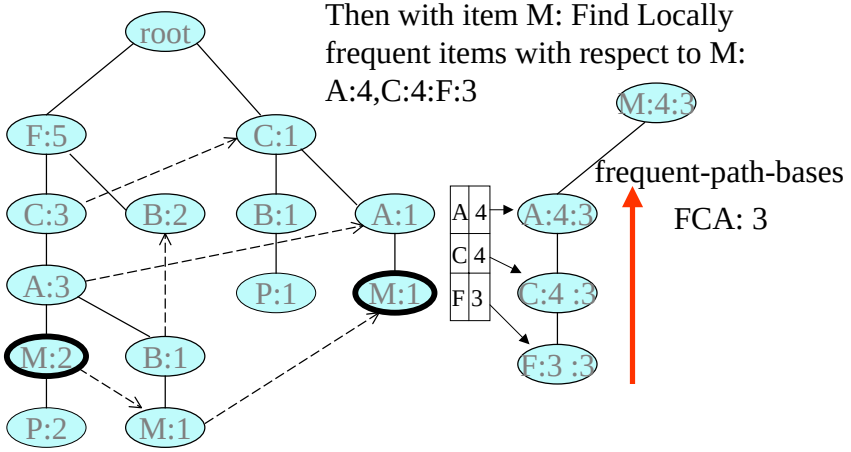
Co-Occurrences Frequent Item tree

Then with item M: Find Locally frequent items with respect to M: A:4,C:4:F:3



All-COFI

Co-Occurrences Frequent Item tree



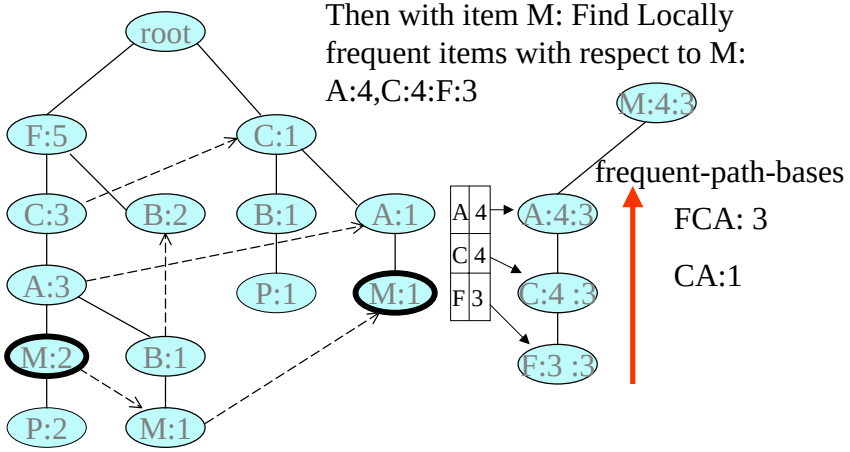
Then with item M: Find Locally frequent items with respect to M: A:4,C:4:F:3

frequent-path-bases

FCA: 3

All-COFI

Co-Occurrences Frequent Item tree



Then with item M: Find Locally frequent items with respect to M: A:4,C:4:F:3

frequent-path-bases

FCA: 3

CA:1

All-COFI

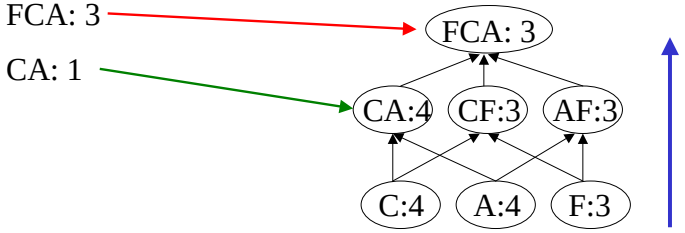
Co-Occurrences Frequent Item tree

How to mine frequent-path-bases

Three approaches:

1: Bottom-Up

Support of any pattern is the summation of the supports of its supersets of frequent-path-bases



All-COFI

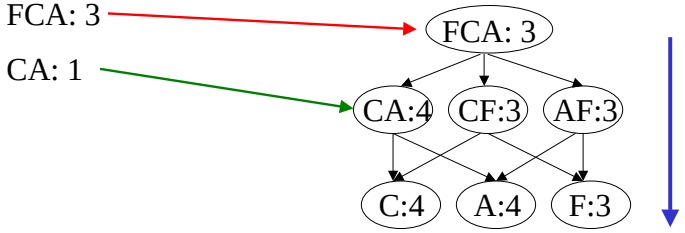
Co-Occurrences Frequent Item tree

How to mine frequent-path-bases

Three approaches:

2: Top-down

Support of any pattern is the summation of the supports of its supersets of frequent-path-bases



All-COFI

Co-Occurrences Frequent Item tree

How to mine frequent-path-bases

Three approaches:

3: Leap-Traversal

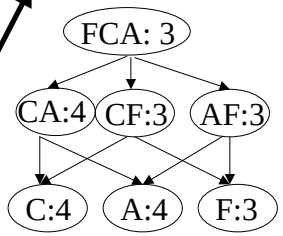
Support of any pattern is the summation of the supports of its supersets of frequent-path-bases

1) Intersect non frequent path bases

FCA: 3 ∩ CA:1 = CA

2) Find subsets of the only frequent paths (sure to be frequent)

3) Find the support of each pattern



All-ECLAT

ECLAT

- For each item, store a list of transaction ids (tids)

Horizontal Data Layout

TID	Items
1	A,B,E
2	B,C,D
3	C,E
4	A,C,D
5	A,B,C,D
6	A,E
7	A,B
8	A,B,C
9	A,C,D
10	B

Vertical Data Layout

A	B	C	D	E
1	1	2	2	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

TID-list

M.J.Zaki IEEE transactions on Knowledge and data Engineering 00

All-ECLAT

ECLAT

- Determine support of any k-itemset by intersecting tid-lists of two of its (k-1) subsets.

A
1
4
5
6
7
8
9

∧

B
1
2
5
7
8
10

→

AB
1
5
7
8

All-ECLAT

ECLAT

Find all frequent patters with respect to item A

AB, AC, …. ABC, ABD, ACD, ABCD ……..

Then it finds all frequent patters with respect to item B

BC, BD, …. BCD, BDE, BCDE ……..

- 3 traversal approaches:
  - top-down, bottom-up and hybrid
- Advantage: very fast support counting, Few scans of database (best case 2)
- Disadvantage: intermediate tid-lists may become too large for memory

Other Algorithms for Other Patterns

Algorithms for Closed Patterns and Maximal Patterns will be discussed in class with paper presentations.

Which algorithm is the winner?

Not clear yet

With relatively small datasets we can find different winners

1. By using different datasets
2. By changing the support level
3. By changing the implementations

Which algorithm is the winner?

What about Extremely large datasets (hundreds of millions of transactions)?

Most of the existing algorithms do not run on such sizes

Vertical approaches and Bitmaps approaches cannot load the transactions in Main Memory

Reparative approaches cannot keep scanning these huge databases many times

Requirements: We need algorithms that

1) do not require multiple scans of the database

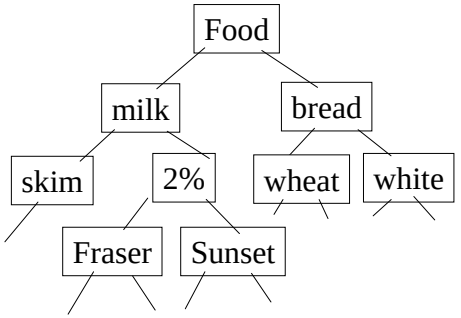
2) Leave small foot print in Main Memory at any given time

Association Rules Outline

- What is association rule mining?
- How do we mine single-dimensional boolean associations?
- How do we mine multilevel associations?
- How do we mine multidimensional associations?
- Can we constrain the association mining?

Multiple-Level Association Rules

- Items often form hierarchy.
- Items at the lower level are expected to have lower support.
- Rules regarding itemsets at appropriate levels could be quite useful.
- Transaction database can be encoded based on dimensions and levels
- It is smart to explore shared multi-level mining (Han & Fu,VLDB'95).



TID	Items
T1	{111, 121, 211, 221}
T2	{111, 211, 222, 323}
T3	{112, 122, 221, 411}
T4	{111, 121}
T5	{111, 122, 211, 221, 413}

Mining Multi-Level Associations

- **A top_down, progressive deepening approach.**
  - First find high-level strong rules:
    milk → bread [20%, 60%].
  - Then find their lower-level "weaker" rules:
    2% milk → wheat bread [6%, 50%].
- **Variations at mining multiple-level association rules.**
  - Level-crossed association rules:
    2% *milk* → ***Wonder wheat*** *bread*
  - Association rules with multiple, alternative hierarchies:
    2% *milk* → *Wonder bread*

Multi-Level Mining: Progressive Deepening

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

  When one threshold is set for all levels; if support too high, it is possible to miss meaningful associations at low level; if support too low, it is possible to generate uninteresting rules
- Different minimum support threshold across multi-levels lead to different algorithms.

Approaches to Mining Multi-level Association Rules

- Uniform minimum support for all levels
  - Same support σ for all levels
  - Avoid examining itemsets containing items whose ancestor is not frequent.
  - Simpler, but it is unlikely that lower level items are as frequent as higher level items.

Approaches to Mining Multi-level Association Rules

- Reduced minimum support at lower levels

Examine only those descendents whose ancestor's support is frequent or non-negligible (controlled).

– **Level-by-level independent**

Full depth search

– **Level-cross filtering by single item**

A specific association is examined from a more general one => items are examined only if parents are frequent.

– **Level-cross filtering by k-itemsets**

Frequency of ancestry examined for k-itemsets and not just items

Association Rules Outline

- What is association rule mining?
- How do we mine single-dimensional boolean associations?
- How do we mine multilevel associations?
- How do we mine multidimensional associations?
- Can we constrain the association mining?

Mining Multi-Dimensional Associations

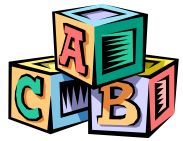
- **Multi-dimensional vs. transaction-based associations**
  - Multi-dimensional (linking different attributes)
    - major(x, "cs") ^ region(x, "oxford") → gpa(x,"good").
  - Transaction-based (linking the same kind of attributes)
    - takes(x, "chemistry") ^ takes (x, "biology") → takes(x, "bio-chemistry").
- **Multi-level association (drilling on any dimension)**
  - Lower levels often adopt lower *min_support* thresholds.
- **Method**:
  - Construct data cube (with count/frequency aggregated)
  - Perform level-wise/dimension-wise search in the data cube (Kamber et al., KDD'97).

Categorical and Quantitative

In a multidimensional context there are:

•Categorical dimensions (attributes)

•Ex. Occupation, Location, etc.



•Quantitative dimensions (attributes)

•Ex. Price, Age, etc.

Apriori, as it is, does not handle quantitative data.

Quantitative Association Rules

RecordID	Age	Married	NumCars
100	23	No	1
200	25	Yes	1
300	29	No	0
400	34	Yes	2
500	38	Yes	2

Sample Rules	Support	Confidence
<age:30..39> and <married: yes> ==> <numCars:2>	40%	100%
<NumCars: 0..1> ==> <Married: No>	40%	66.70%

Mapping Quantitative to Boolean

- One possible solution is to map the problem to the Boolean association rules:
  - discretizing a non-categorical attribute to intervals
    - Age [20,29], [30,39],...
  - forming Boolean records
    - categorical attributes: each value becomes one item
    - non-categorical attributes: each interval becomes one item

RecordID	Age	Married	NoCars
100	23	No	1
500	38	Yes	2

RecID	Age: 20..29	Age: 30..39	Married: Yes	Married: No	Cars: 0	Cars: 1	Cars: 2
100	1	0	0	1	0	1	0
500	0	1	1	0	0	0	1

Mining Quantitative Association Rules

- Problems with the mapping
  - too few intervals: lost information
  - too low support: too many rules
- Solutions
  - using the supports of an itemset and its generalizations to determine the intervals
  - Binning (equi-width,equi-dept,distance based)
  - using interest measure to control the number of association rules

Association Rules Outline

- What is association rule mining?
- How do we mine single-dimensional boolean associations?
- How do we mine multilevel associations?
- How do we mine multidimensional associations?
- Can we constrain the association mining?

Constraint-based Data Mining

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

Restricting Association Rules

•Useful for interactive and ad-hoc mining
•Reduces the the set of association rules discovered and confines them to more relevant rules.
• **Before mining**
✓ Knowledge type constraints: classification, etc.
✓ Data constraints: SQL-like queries (DMQL)
✓ Dimension/level constraints: relevance to some dimensions and some concept levels.
• **While mining**
✓ Rule constraints: form, size, and content.
✓ Interestingness constraints: support, confidence, correlation.
• **After mining**
✓ Querying association rules

Constrained Frequent Pattern Mining: A Mining Query Optimization Problem

- Given a frequent pattern mining query with a set of constraints C, the algorithm should be
  - sound: it only finds frequent sets that satisfy the given constraints *C*
  - complete: all frequent sets satisfying the given constraints *C* are found
- A naïve solution
  - First find all frequent sets, and then test them for constraint satisfaction
- More efficient approaches:
  - Analyze the properties of constraints comprehensively
  - Push them as deeply as possible inside the frequent pattern computation.

Rule Constraints in Association Mining

- Two kind of rule constraints:
  - Rule form constraints: meta-rule guided mining.
    - $P(x, y) \wedge Q(x, w) \rightarrow$ takes(x, "database systems").
  - Rule content constraint: constraint-based query optimization (where and having clauses)(Ng, et al., SIGMOD'98).
    - sum(LHS) < 100 ^ min(LHS) > 20 ^ count(LHS) > 3 ^ sum(RHS) > 1000
- **1-variable vs. 2-variable constraints** (Lakshmanan, et al. SIGMOD'99):
  - 1-var: A constraint confining only one side (L/R) of the rule, e.g., as shown above.
  - 2-var: A constraint confining both sides (L and R).
    - sum(LHS) < min(RHS) ^ max(RHS) < 5* sum(LHS)

Anti-Monotonicity in Constraint-Based Mining

- Anti-monotonicity
  - *When an intemset S **violates** the constraint, so does any of its superset*
  - *sum(S.Price)* ≤ *v* is anti-monotone
  - *sum(S.Price)* ≥ *v* is not anti-monotone
- Example. C: range(S.profit) ≤ 15 is anti-monotone
  - Itemset *ab* violates C
  - So does every superset of *ab*

TDB (min_sup=2)

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

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

Monotonicity in Constraint-Based Mining

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

TDB (min_sup=2)

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

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

Which Constraints Are Monotone or Anti-Monotone?

SQL-based Constraints

Constraint	Monotone	Anti-Monotone
$v \in S$	yes	no
$S \supseteq V$	yes	no
$S \subseteq V$	no	yes
$min(S) \leq v$	yes	no
$min(S) \geq v$	no	yes
$max(S) \leq v$	no	yes
$max(S) \geq v$	yes	no
$count(S) \leq v$	no	yes
$count(S) \geq v$	yes	no
$sum(S) \leq v$ ($a \in S, a \leq 0$)	no	yes
$sum(S) \geq v$ ($a \in S, a \leq 0$)	yes	no
$range(S) \leq v$	no	yes
$range(S) \geq v$	yes	no
$support(S) \geq \xi$	no	yes
$support(S) \leq \xi$	yes	no

State Of The Art

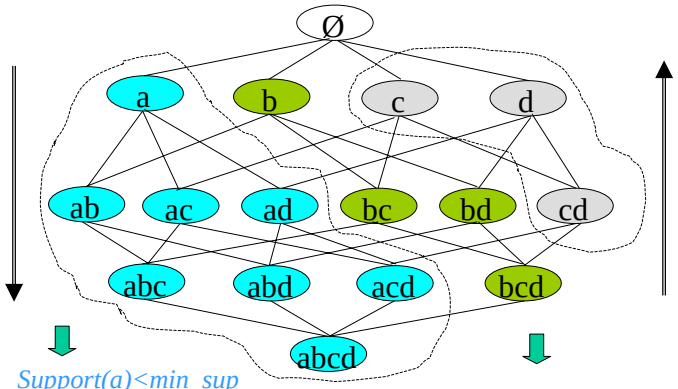
- **Constraint pushing techniques have been proven to be effective in reducing the explored portion of the search space in constrained frequent pattern mining tasks.**
- **Anti-monotone constraints:**
  - **Easy to push …**
  - **Always profitable to do …**
- **Monotone constraints:**
  - **Hard to push …**
  - **Should we push them, or not?**

Dual Miner

C. Bucil, J. Gherke, D. Kiefer and W. White, SIGKDD'02

- A dual pruning algorithm for itemsets with constraints
- Based on MAFIA

1. BITMAP approach
2. Does not compute frequent items with their support
3. Performance issues (Many tests are required)



Support(a)<min_sup

All its supersets can be pruned

Support(cd)>max_sup

All its subsets can be pruned

FP-Growth with constraints

Checks for only monotone constraints (plus the support, which is an anti-monotone constraint).

Once a frequent itemset satisfies the monotone constraint, all frequent itemsets having it as a prefix also are guaranteed to satisfy the constraint

J. Pei, J. Han, L. Lakshmanan, ICDE'01

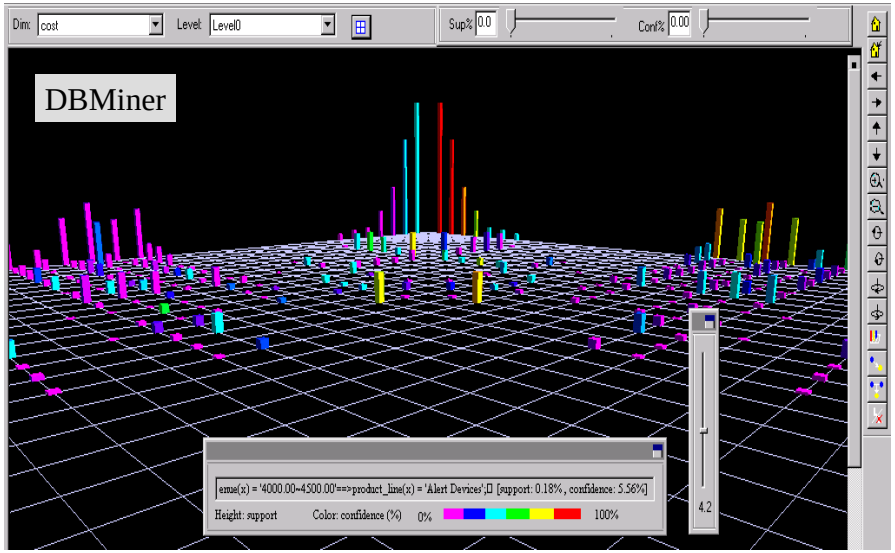
Presentation of Association Rules (Table Form)

	Body	Implies	Head	Supp (%)	Conf (%)
1	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00'	28.45	40.4
2	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00'	20.46	29.05
3	cost(x) = '0.00~1000.00'	==>	order_qty(x) = '0.00~100.00'	59.17	84.04
4	cost(x) = '0.00~1000.00'	==>	revenue(x) = '1000.00~1500.00'	10.45	14.84
5	cost(x) = '0.00~1000.00'	==>	region(x) = 'United States'	22.56	32.04
6	cost(x) = '1000.00~2000.00'	==>	order_qty(x) = '0.00~100.00'	12.91	69.34
7	order_qty(x) = '0.00~100.00'	==>	revenue(x) = '0.00~500.00'	28.45	34.54
8	order_qty(x) = '0.00~100.00'	==>	cost(x) = '1000.00~2000.00'	12.91	15.67
9	order_qty(x) = '0.00~100.00'	==>	region(x) = 'United States'	25.9	31.45
10	order_qty(x) = '0.00~100.00'	==>	cost(x) = '0.00~1000.00'	59.17	71.86
11	order_qty(x) = '0.00~100.00'	==>	product_line(x) = 'Tents'	13.52	16.42
12	order_qty(x) = '0.00~100.00'	==>	revenue(x) = '500.00~1000.00'	19.67	23.88
13	product_line(x) = 'Tents'	==>	order_qty(x) = '0.00~100.00'	13.52	98.72
14	region(x) = 'United States'	==>	order_qty(x) = '0.00~100.00'	25.9	81.94
15	region(x) = 'United States'	==>	cost(x) = '0.00~1000.00'	22.56	71.39
16	revenue(x) = '0.00~500.00'	==>	cost(x) = '0.00~1000.00'	28.45	100
17	revenue(x) = '0.00~500.00'	==>	order_qty(x) = '0.00~100.00'	28.45	100
18	revenue(x) = '1000.00~1500.00'	==>	cost(x) = '0.00~1000.00'	10.45	96.75
19	revenue(x) = '500.00~1000.00'	==>	cost(x) = '0.00~1000.00'	20.46	100
20	revenue(x) = '500.00~1000.00'	==>	order_qty(x) = '0.00~100.00'	19.67	96.14
21					
22					
23	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00' AND order_qty(x) = '0.00~100.00'	28.45	40.4
24	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00' AND order_qty(x) = '0.00~100.00'	28.45	40.4
25	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00' AND order_qty(x) = '0.00~100.00'	19.67	27.93
26	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00' AND order_qty(x) = '0.00~100.00'	19.67	27.93
27	cost(x) = '0.00~1000.00' AND order_qty(x) = '0.00~100.00'	==>	revenue(x) = '500.00~1000.00'	19.67	33.23

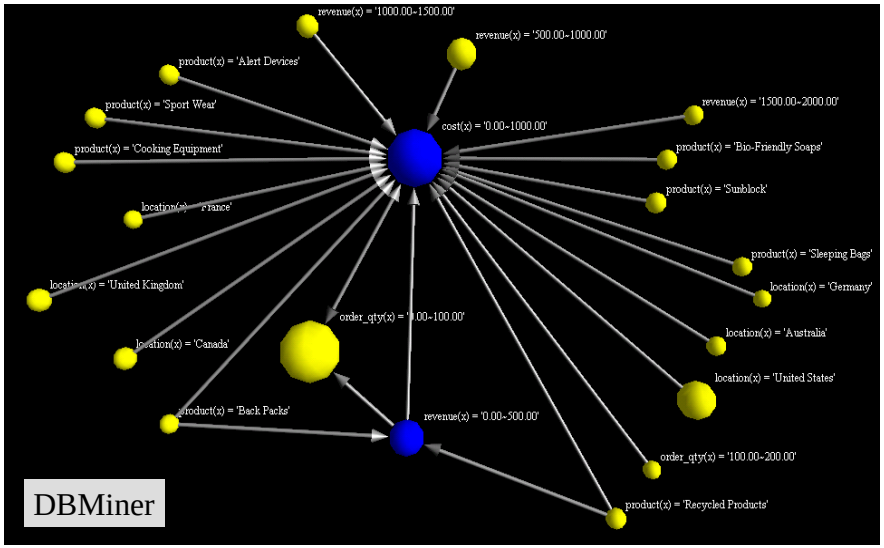
Sheet1

DBMiner

Visualization of Association Rule in Plane Form



Visualization of Association Rule Using Rule Graph



Visualization of Association Rule Using Table Graph (DBMiner Web version)