University of Alberta

Database Laboratory

PAKDD 2004 Tutorial

Advances and Issues in Frequent Pattern Mining

Osmar R. Zaïane
Mohammad El-Hajj
University of Alberta
Canada

What Is Frequent Pattern Mining?

- What is a frequent pattern?
  - Pattern (set of items, sequence, etc.) that occurs together frequently in a database [AIS92]
- Frequent pattern: an important form of regularity
  - What products were often purchased together? — beers and diapers!
  - What are the consequences of a hurricane?
  - What is the next target after buying a PC?

Why Studying Frequent Pattern Mining?

- Frequent pattern mining — Foundation for several essential data mining tasks:
  - association, correlation, causality
  - sequential patterns
  - partial periodicity, cyclic/temporal associations
- Applications:
  - basket data analysis, cross-marketing, catalog design, loss-leader analysis,
  - clustering, classification, Web log sequence, DNA analysis, etc.

General Outline

- Association Rules
- Different Frequent Patterns
- Different Lattice Traversal Approaches
- Different Transactional Layouts
- State-Of-The-Art Algorithms
  - For All Frequent Patterns
  - For Frequent Closed Patterns
  - For Frequent Maximal Patterns
- Adding Constraints
- Parallel and Distributed Mining
- Visualization of Association Rules
- Frequent Sequential Pattern 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%]

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 $P \cap Q = \varnothing$

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

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

Association Rule Mining

Frequent Itemset Mining

FIM 1 abc

Bound by a **support** threshold

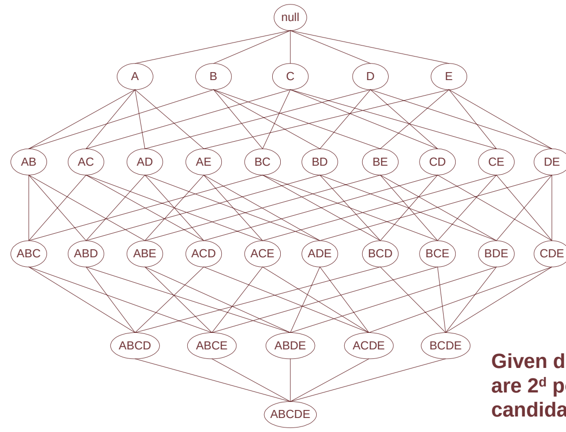
Association Rules Generation

2 ab→c
b→ac

Bound by a **confidence** threshold

- Frequent itemset generation is still computationally expensive

Frequent Itemset Generation



Given d items, there are 2^d possible candidate itemsets

Frequent Itemset Generation

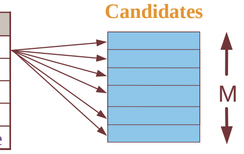
- Brute-force approach (Basic approach):
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database

Transactions

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

N, w

List of Candidates

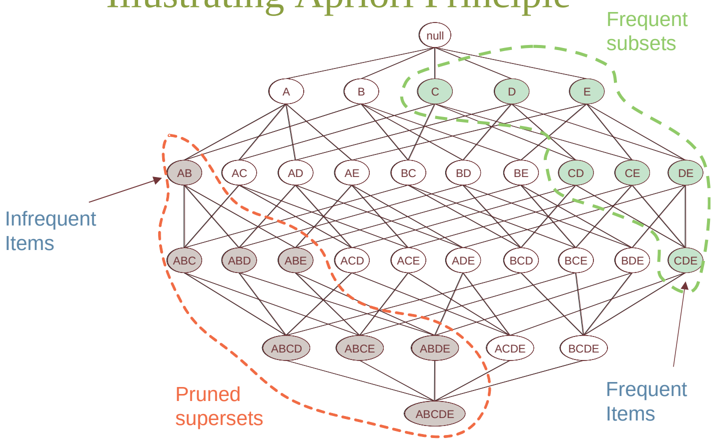


  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => **Expensive since M = 2^d** !!!

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)**

Illustrating Apriori Principle



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

Support>1

Database D

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

Scan D → C_1

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

→ L_1

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

C_2

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

Scan D → C_2

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

→ L_2

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

C_3

itemset
{2 3 5}

Scan D → L_3

itemset	sup
{2 3 5}	2

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($P \rightarrow Q$) = Prob(Q/P) = $\frac{\text{Support}(P \cup Q)}{\text{Support}(P)}$

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

Interestingness Measures

- *play basketball* ⇒ *eat cereal* [40%, 66.7%] is misleading
  - The overall percentage of students eating cereal is 75% which is higher than 66.7%.
- *play basketball* ⇒ *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

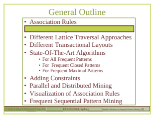
- We need a measure of dependent or correlated events

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

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

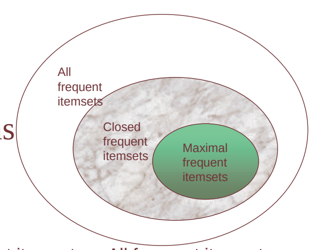
- P(B|A)/P(B) is also called the **lift** of rule A => B

Other Frequent Patterns



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

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



Maximal frequent itemsets ⊆ Closed frequent itemsets ⊆ 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

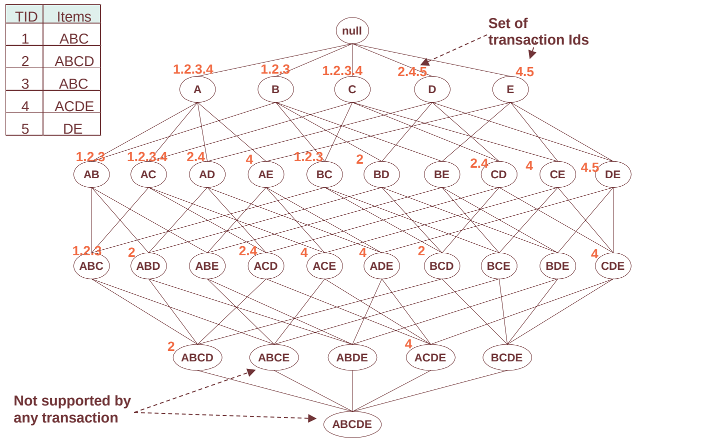
Frequent itemsets

bd	2
abc	2

Frequent Maximal itemsets

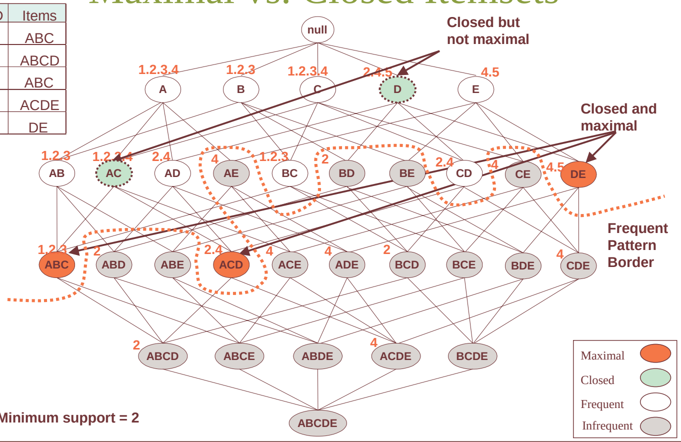
Maximal vs. Closed Itemsets

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

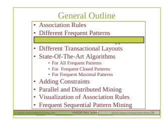


Maximal vs. Closed Itemsets

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

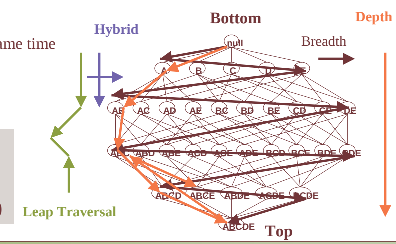


Mining the Pattern Lattice



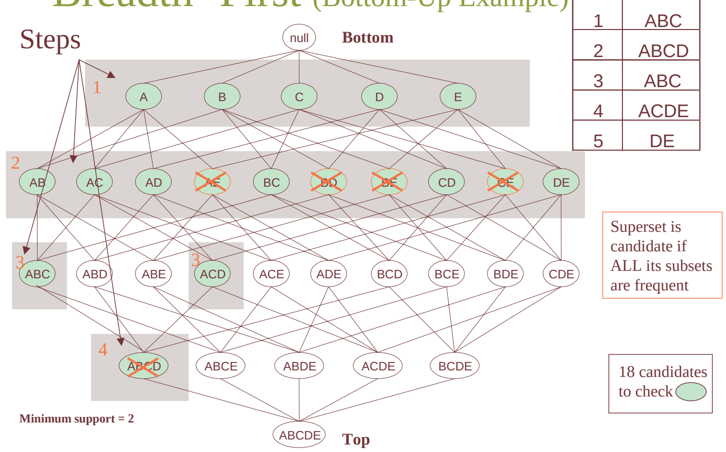
- Breadth-First
  - It uses current items at level k to generate items of level k+1 (many database scans)
- Depth-First
  - It uses a current item at level k to generate all its supersets (favored when mining long frequent patterns)
- Hybrid approach
  - It mines using both direction at the same time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

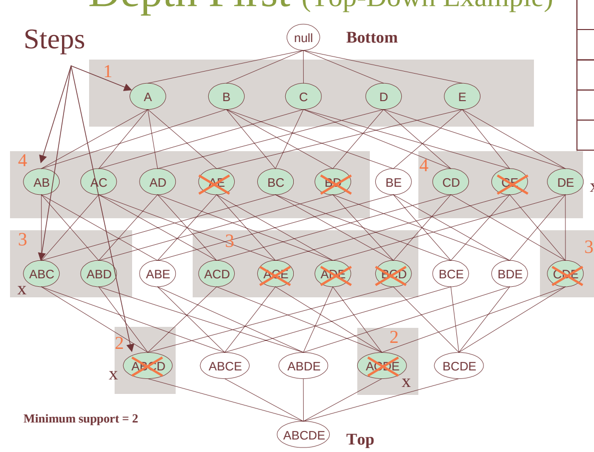


Superset is candidate if ALL its subsets are frequent

18 candidates to check

Depth First (Top-Down Example)

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



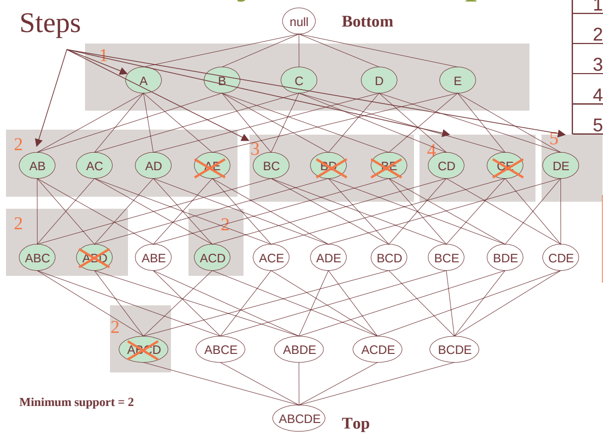
Subset is candidate if it is marked or if one of its supersets is candidate

23 candidates to check

One Hybrid Example

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

Steps



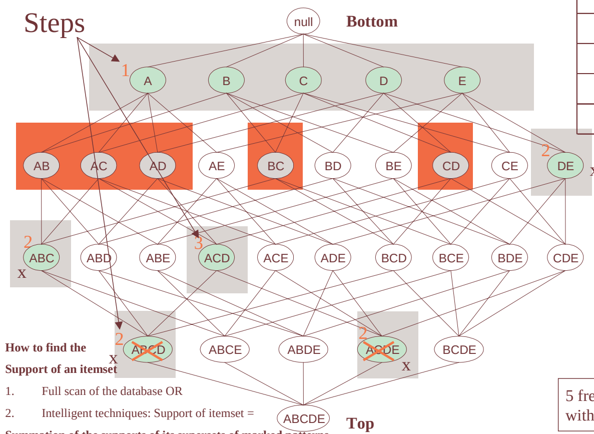
Superset is candidate if ALL its subsets are frequent

19 candidates to check

Leap Traversal Example

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

Steps



Itemset is candidate if it is marked or if it is a subset of more than one <u>infrequent</u> marked superset

10 candidates to check

5 frequent patterns without checking

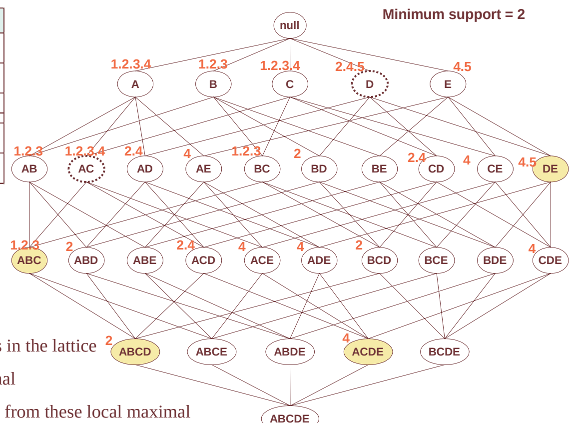
How to find the Support of an itemset

1. Full scan of the database OR
2. Intelligent techniques: Support of itemset = Summation of the supports of its supersets of marked patterns

Finding Maximal using leap traversal approach

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

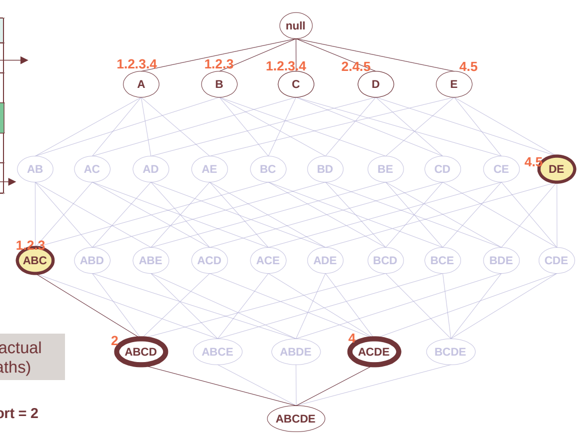
Minimum support = 2



- 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~~

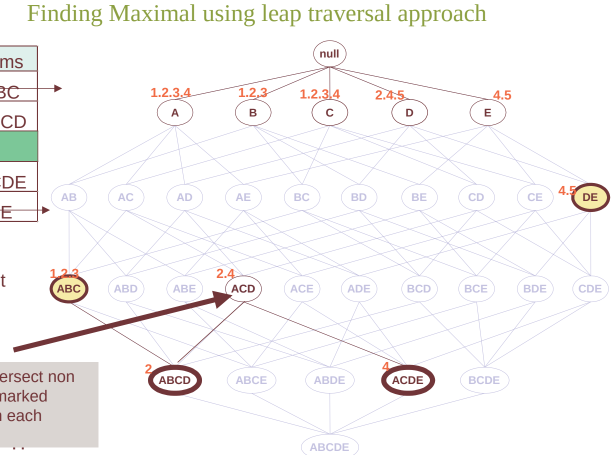


Step1: Define actual paths (Mark paths)

Minimum support = 2

Finding Maximal using leap traversal approach

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

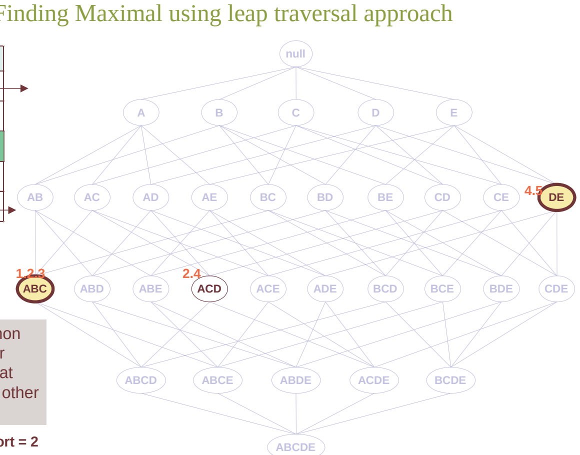


Intersect ABCD with ACDE

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~~

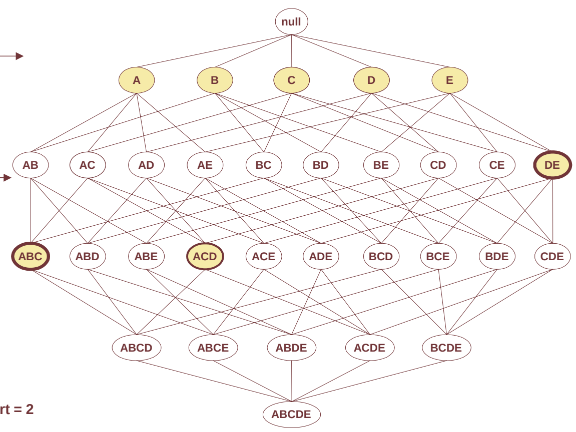


Step3: Remove non frequent paths, or frequent paths that have superset of other frequent paths

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

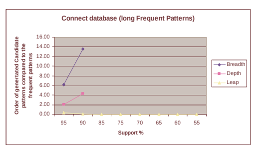
				Total Candidates Created		
Support	Size of Largest	F1-Size	Total Frequent	Breadth	Depth	Leap
95	9	17	2205	15626	6654	3044
90	12	21	27127	394648	144426	29263
80	15	28	119418			120177
75	17	30	176365			177244
70	18	31	627952			628963
65	19	33	1368337			1369585
60	20	36	2908632			2910175
55	21	37	5996892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

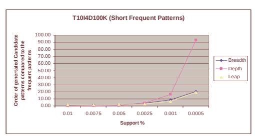
T10I4D100K

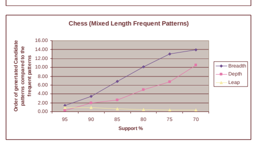
				Total Candidates Created		
Support	Size of Largest	F1-Size	Total Frequent	Breadth	Depth	Leap
1000	3	375	385	20	10	29
750	4	463	561	262	124	291
500	5	569	1073	1492	731	1546
250	8	717	7703	36994	36309	26666
100	10	797	27532	264645	458275	200113
50	10	839	53385	1129779	4923723	1067050

Chess

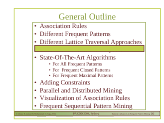
				Total Candidates Created		
Support	Size of Largest	F1-Size	Total Frequent	Breadth	Depth	Leap
95	5	9	78	165	90	136
90	7	13	628	2768	1842	1191
85	8	16	2690	20871	9667	4318
80	10	20	8282	91577	49196	12021
75	11	23	20846	292363	160362	28986
70	13	24	48939	731740	560103	65093







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	0	1	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	1	0	1
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

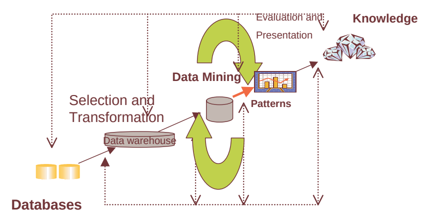
T#	Items				
T1	A	G	D	C	B
T2	B	C	H	E	D
T3	B	D	E	A	M
T4	C	E	F	A	N
T5	A	B	N	O	P
T6	A	C	Q	R	G
T7	A	C	H	I	G
T8	L	E	F	K	B
T9	A	F	M	N	O
T10	C	F	P	J	R
T11	A	D	B	H	I
T12	D	E	B	K	L
T13	M	D	C	G	O
T14	C	F	P	Q	J
T15	B	D	E	F	I
T16	J	E	B	A	D
T17	A	K	E	F	C
T18	C	D	L	B	A

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	(☼,☼)	(☼,☼)	(☼,☼)	(☼,☼)	(☼,☼)	(☼,☼)	(☼,☼)	(☼,☼)	(☼,☼)	(☼,☼)	(☼,☼)

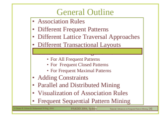
Transactional Layouts

- 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

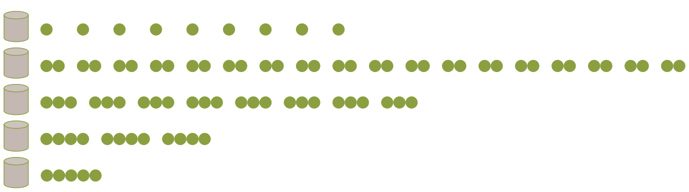
MaxMiner, MAFIA, GENMAX, COFI-MAX

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

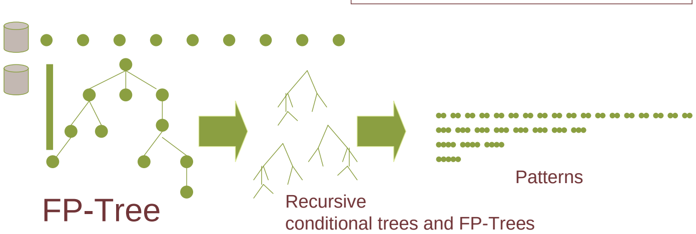
FP-Growth

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

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~~

All-FP-Growth

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

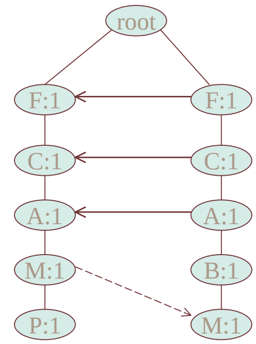
F:5, C:5, A:4, B:4, M:4, P:3

Required Support: **3**

All-FP-Growth

Frequent Pattern Tree

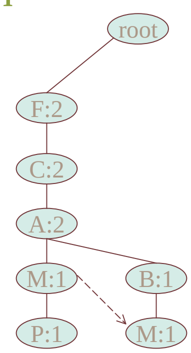
F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B



All-FP-Growth

Frequent Pattern Tree

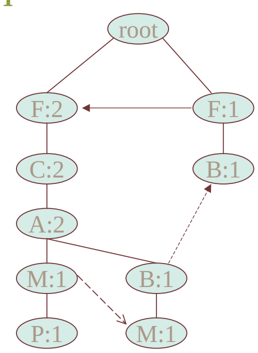
F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B



All-FP-Growth

Frequent Pattern Tree

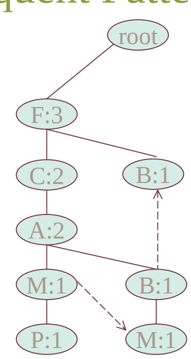
F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B



All-FP-Growth

Frequent Pattern Tree

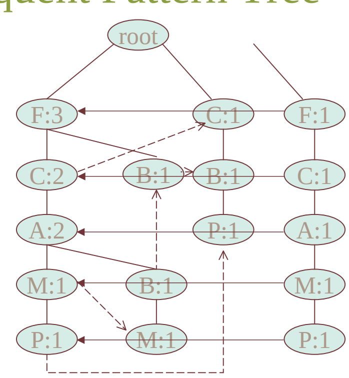
F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B



All-FP-Growth

Frequent Pattern Tree

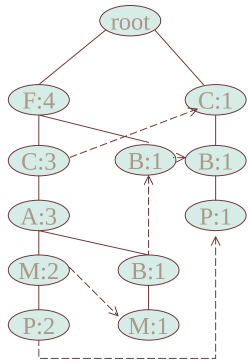
F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B



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

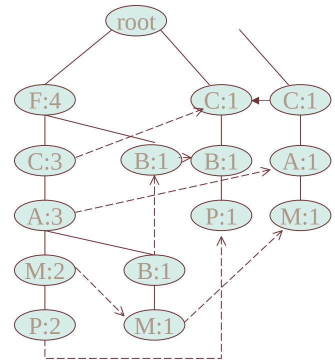
Frequent Pattern Tree



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

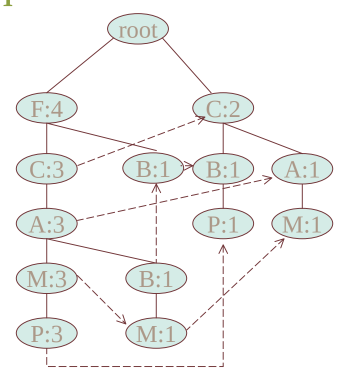
Frequent Pattern Tree



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

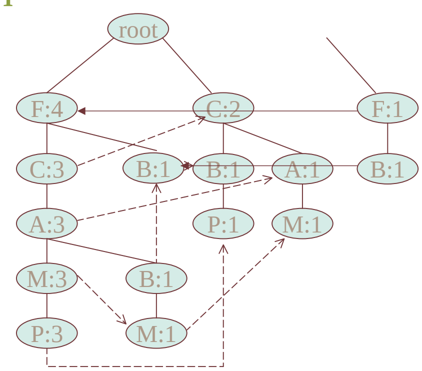
Frequent Pattern Tree



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

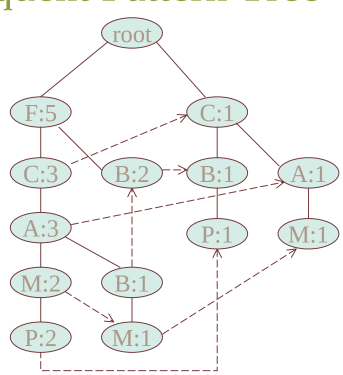
Frequent Pattern Tree



All-FP-Growth

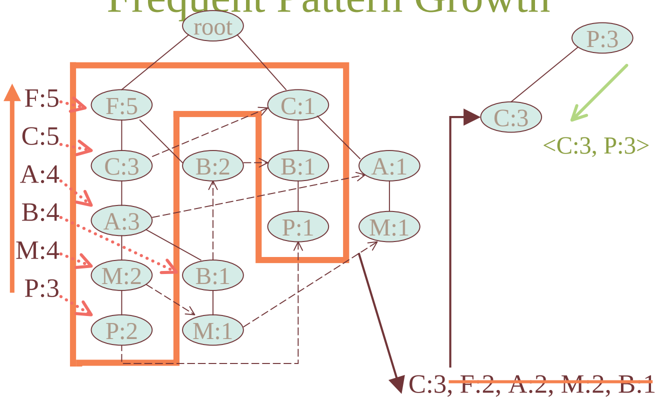
F, C, A, M, P
F, C, A, B, M
F, B
C, B, P
F, C, A, M, P
C, A, M
F, B

Frequent Pattern Tree



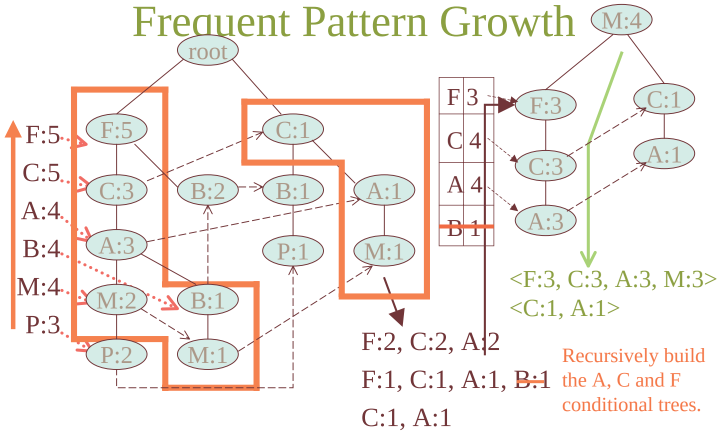
All-FP-Growth

Frequent Pattern Growth



All-FP-Growth

Frequent Pattern Growth

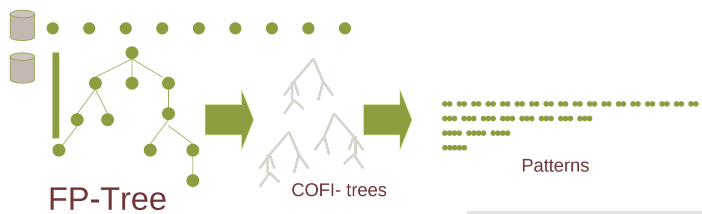


All-COFI

COFI algorithm big picture

COFI

2 I/O scans
reduced candidacy generation
Small memory footprint



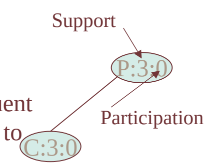
El-Hajj and Zaïane, DaWak'03

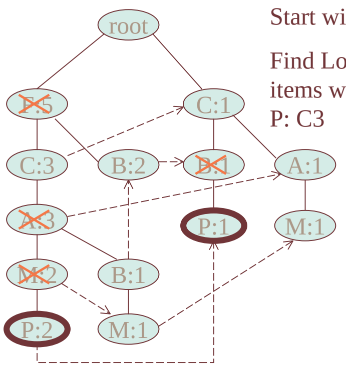
All-COFI

Co-Occurrences Frequent Item tree

Start with item P:

Find Locally frequent items with respect to P: C3



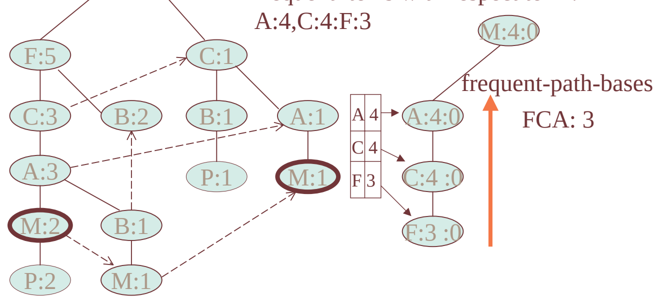


PC:3 frequent-path-base
All subsets of PC:3 are frequent and have the same support

All-COFI

Co-Occurrences Frequent Item tree

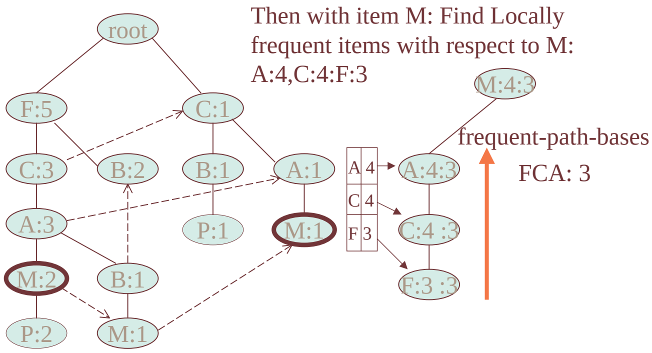
Then with item M: Find Locally frequent items with respect to M: A:4,C:4:F:3



All-COFI

Co-Occurrences Frequent Item tree

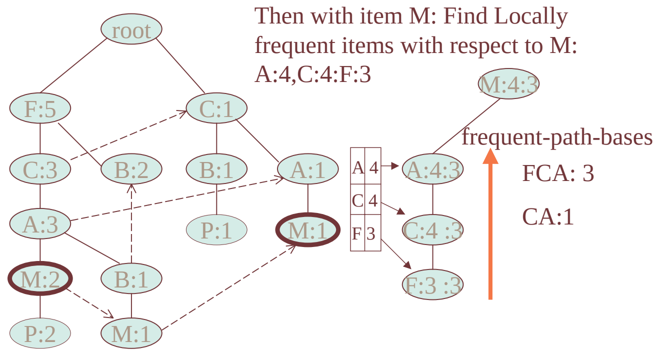
Then with item M: Find Locally frequent items with respect to M: A:4,C:4:F:3



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

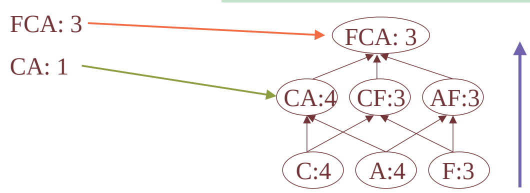
Co-Occurrences Frequent Item tree

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

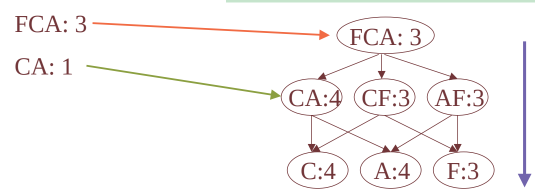
Co-Occurrences Frequent Item tree

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

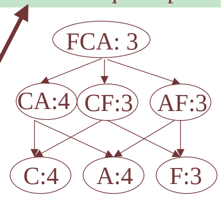
Support of any pattern is the summation of the supports of its supersets of frequent-path-bases

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

$\wedge$

B
1
2
5
7
8
10

$\rightarrow$

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

CLOSED-CHARM

CHARM

Use vertical Data Format

Deliver Closed patterns based on vertical intersections

A		B		AB
1		1		1
4		2		5
5	^	5	→	7
6		7		8
7		8		
8		10		
9				

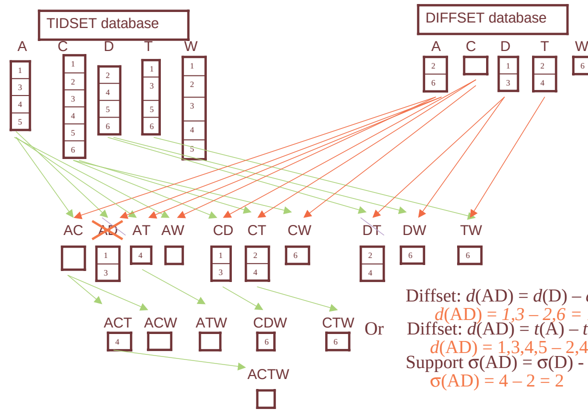
Diffsets: don't store the entire tidsets, only the differences between tidsets

Use Diffset to accelerate mining

M.J.Zaki and C-J. Hsiao. SIAM 02

CLOSED-CHARM

Diffset Intersections (example)



Diffset: $d(AD) = d(D) - d(A)$

$d(AD) = 1,3 - 2,6 = 1,3$

Diffset: $d(AD) = t(A) - t(D)$

$d(AD) = 1,3,4,5 - 2,4,5,6 = 1,3$

Support $\sigma(AD) = \sigma(D) - |d(AD)|$

$\sigma(AD) = 4 - 2 = 2$

CLOSED-CLOSET+

CLOSET+

Use horizontal format of transactions

Based on the FP-Growth Model

Divide the search space

Find closed itemset recursively

2 level hash indexed result tree structure for dense datasets

Pseudo projection based upward-checking for sparse datasets

J. Wang, J. Han, J. Pei. SIGKDD 03

CLOSED-CLOSET+

CLOSET+

- Two-Level hash-indexed results tree
- Compressed result tree structure
- Search space shrinking for subset checking
  - If itemset S_c can be absorbed by another already mined itemset S_a, they have the following relationships:
    1) $sup(S_c)=sup(S_a)$
    2) $length(S_c)<length(S_a)$
    3) $\forall i, j \in S_c \Rightarrow i \in S_a$
  - Measures to enhance the checking
    - *Two-level hash indices – support and itemID*
    - *Record length information in each result tree node*

CLOSED-CLOSET+

CLOSET+

Pseudo-projection based upward checking

- Result-tree may consume much more memory for sparse datasets
- Subset checking without maintenance of history itemsets

CLOSED-CLOSET+

CLOSET+

Pseudo-projection based upward checking

- Result-tree may consume much more memory for sparse datasets
- Subset checking without maintenance of history itemsets
  - For a certain prefix X, as long as we can find any item which
  - (1) appears in each prefix path w.r.t. prefix X, and
  - (2) does not belong to X, any itemset with prefix X will be non-closed, otherwise, if there's no such item, the union of X and the complete set of its locally frequent items with support sup(X) will form a closed itemset.

CLOSED-COFI-CLOSED

COFI-CLOSED

Use Horizontal format

COFI- Mining approach

Leap-traversal approach

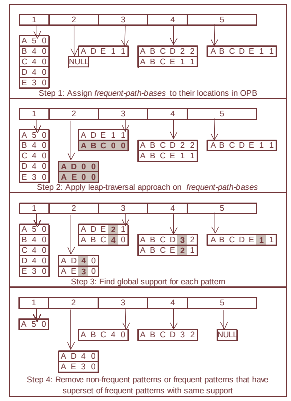
Divide the search space

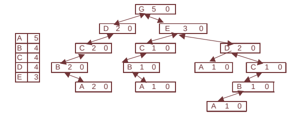
One Global tree to hold the discovered closed patterns

M. El-Hajj and O. Zaïane

CLOSED-COFI-CLOSED

COFI-CLOSED





Example (Finding closed frequent patterns for a COFI-tree)

Frequent Path Bases

ABCD:2, ABCE:1, ADE:1, ABCDE:1

MAXIMAL-MAXMINER

MaxMiner: Mining Max-patterns

- 1st scan: find frequent items
  - A, B, C, D, E
- 2nd scan: find support for
  - AB, AC, AD, AE, **ABCDE**
  - BC, BD, BE, **BCDE**
  - CD, CE, **CDE**

Potential max-patterns

- Since BCDE is a max-pattern, no need to check BCD, BDE, CDE in later scan

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

R. J. Bayardo, ACM SIGMOD'98

MAXIMAL-MAXMINER

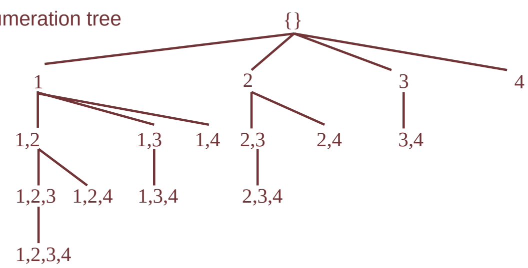
MaxMiner: Mining Max-patterns

–Abandons a bottom-up traversal

–Attempts to "look-ahead"

–Identify a long frequent itemset, prune all its subsets.

•Set-enumeration tree



MAXIMAL-MAFIA

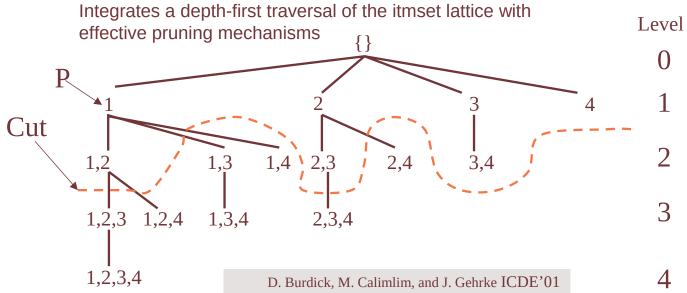
MAFIA

A Maximal Frequent Itemset Algorithm for Transactional Databases

Vertical bitmap presentation

Look Ahead pruning technique

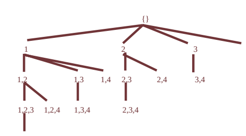
Integrates a depth-first traversal of the itmset lattice with effective pruning mechanisms



D. Burdick, M. Calimlim, and J. Gehrke ICDE'01

MAXIMAL-MAFIA

MAFIA



- PEP (Parent Equivalence Pruning)
  - newNode = C ∪ i (where i is child of C)
  - Check newNode.support == C.support
    - Move i from C.tail to C.head
- FHUT (Frequent Head Union Tail)
  - newNode = C ∪ I (where I is the leftmost child in the tail)
- HUTMFI
  - Check Head Union Tail is in MFI
    - Stop searching and return

MAXIMAL-GENMAX

GenMax:

- Progressive focusing to improve superset checking
- Superset checking techniques
  - Do superset check only for $I_{l+1} \cup P_{l+1}$
  - Using check_status flag
  - Local maximal frequent itemsets
- Reordering the combine set
- Vertical approach.
- Database needs to be loaded to Main Memory

.

K. Jouda and M. Zaki, ICDM'01

MAXIMAL-COFIMAX

COFI-MAX

COFI-Approach

Leap traversal searching

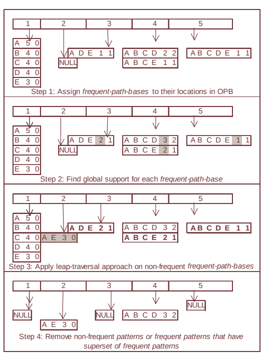
Pruning techniques

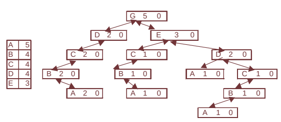
(A) skip building some COFI-trees
(B) Remove all frequent-path-bases that are subset of already found maximal patterns
(C) Count the support of frequent-path-bases early, and remove the frequent ones from the leap-traversal approach

M. El-Hajj and O. Zaïane

MAXIMAL-COFIMAX

COFI-MAX



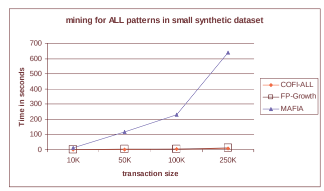


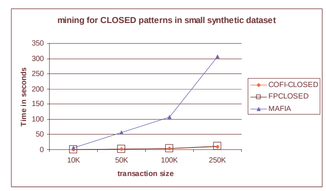
Example (Finding Maximal frequent patterns for a COFI-tree)

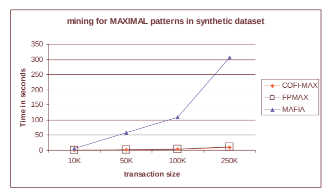
Frequent Path Bases

ABCD:2, ABCE:1, ADE:1, ABCDE:1

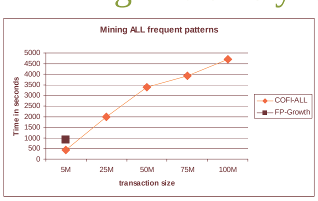
Mining relatively small datasets

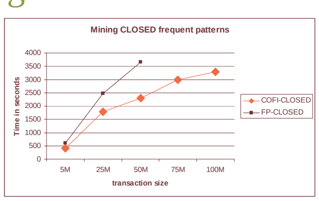


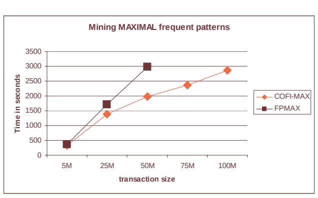




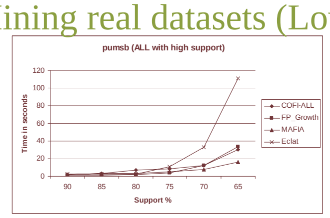
Mining Extremely large datasets

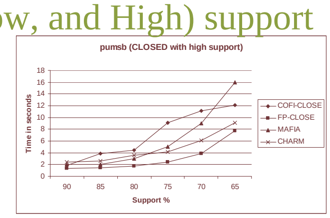


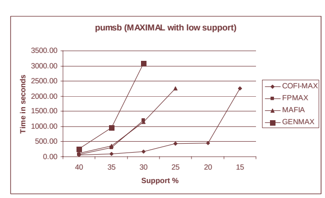




Mining real datasets (Low, and High) support







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

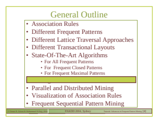
Reparative approaches cannot keep scanning these huge databases many times

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

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 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) ^ Q(x, w) –> 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 ∈ S	yes	no
S ⊇ V	yes	no
S ⊆ V	no	yes
min(S) ≤ v	yes	no
min(S) ≥ v	no	yes
max(S) ≤ v	no	yes
max(S) ≥ v	yes	no
count(S) ≤ v	no	yes
count(S) ≥ v	yes	no
sum(S) ≤ v (a ∈ S, a ≤ 0)	no	yes
sum(S) ≥ v (a ∈ S, a ≤ 0)	yes	no
range(S) ≤ v	no	yes
range(S) ≥ v	yes	no
support(S) ≥ ξ	no	yes
support(S) ≤ ξ	yes	no

State Of The Art

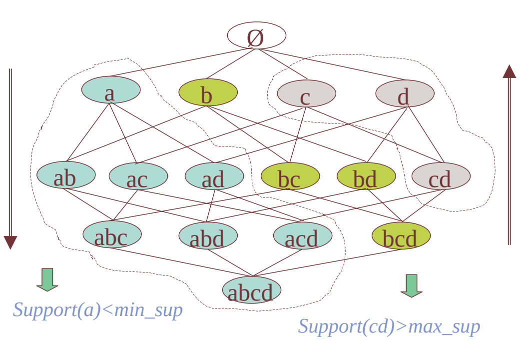
- **Constraint pushing techniques have been proven to be effective in reducing the explored portion of the search space in constrained frequent pattern mining tasks.**
- **Anti-monotone constraints:**
  - **Easy to push …**
  - **Always profitable to do …**
- **Monotone constraints:**
  - **Hard to push …**
  - **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)



All its supersets can be pruned 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

COFI-With Constraints

Anti-Monotone constraint checking

1. Removing individual items that do not satisfy the it anti-monotone constraint. (i.e they do not even participate in the FP-tree Building process)

Reduce FP-tree size

2. For each A-COFI-tree, remove any locally frequent item **B**, where **B** does not satisfy the anti-monotone constraint

Reduce COFI-tree size

3. No need for constraint checking for any A-COFI-tree if the itemset **X** satisfies the anti-monotone constraint, where **X** is the set of all locally frequent items with respect to **A**

Reduce testing

M. El-Hajj and O. Zaïane

COFI-With Constraints

Monotone constraint checking

1. Removing any frequent-path-bases that do not satisfy the monotone constraint.

Reduce lattice traversal

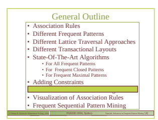
2. Not creating an A-COFI-tree, if all its local items X with the A itemset violate the monotone constraint.

Reduce number of COFI-trees

3. No need for constraint checking for any A-COFI-tree if the item A satisfies the monotone constraint, since any item with A will also satisfy this constraint.

Reduce testing

Parallel KDD systems (Requirements & Design issues)



Algorithm Evaluation

Process Support

Location Transparency

Data Transparency

System Transparency

Security, Quality of Service

Availability, Fault tolerance and Mobility

Challenges and issues

Large size

High dimensionality

Data location

Data skew

Dynamic load balance

Parallel database management system vs. parallel file systems

Parallel I/O minimization

Parallel incremental mining algorithms

Parallel Dividing of the datasets

Parallel merging of discoveries

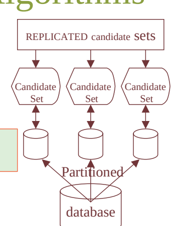
Parallel Association Rule Mining Algorithms

Replication Algorithms

Partitioning Algorithms

Hybrid Algorithms

Replicate Candidate Sets
Partition the databases



- ➢ Count Distribution algorithm (Agrawal R. et al, 1996)
- ➢ Parallel Partition algorithm (Ashok Savasere et al, 1995)
- ➢ Fast Distributed algorithm (David Cheung et al, 1996)
- ➢ Parallel Data Mining algorithm (J.S. Park et al, 1996)

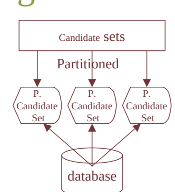
Parallel Association Rule Mining Algorithms

Replication Algorithms

Partitioning Algorithms

Hybrid Algorithms

-Partition the Candidate Set
-Need to Scan the whole data



- ➢ Data Distribution algorithm (Agrawal R. et al, 1996)
- ➢ Intelligent Data Distribution algorithm (E-H Han et all, 1997)
- ➢ Non Partitioned apriori, Simple partitioned, Hash Partitioned apriori and HPA_ELD algorithms (Takahiko et al, 1996)

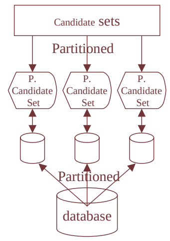
Parallel Association Rule Mining Algorithms

Replication Algorithms

Partitioning Algorithms

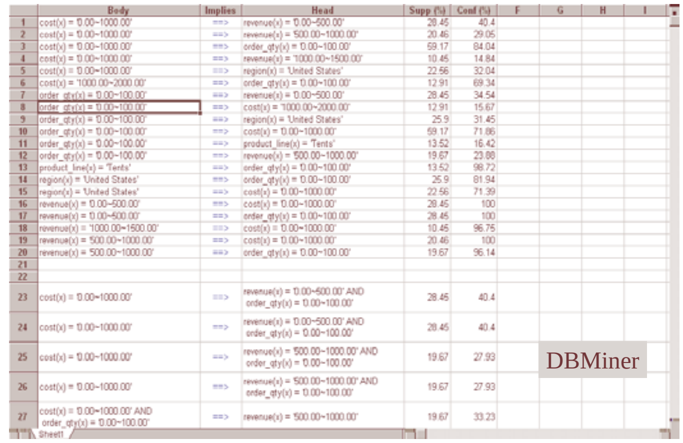
Hybrid Algorithms

-Combine both ideas of replication and partition



- Hybrid Distribution algorithm (E-H Han et all, 1997)
- Multiple Local Frequent Pattern Tree algorithm (Zaïane et al, 2001)

Presentation of Association Rules (Table Form)

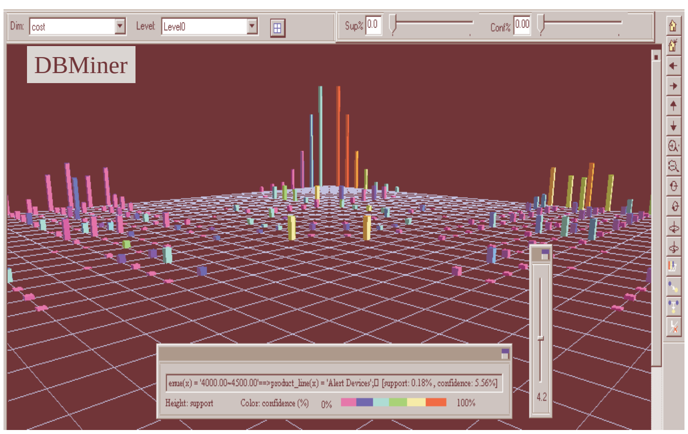


	Body	Implies	Head	Supp (%)	Conf (%)	F	G	H	I
1	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00'	28.45	40.4				
2	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00'	20.46	29.05				
3	cost(x) = '0.00~1000.00'	==>	order_qty(x) = '0.00~100.00'	59.17	84.04				
4	cost(x) = '0.00~1000.00'	==>	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				

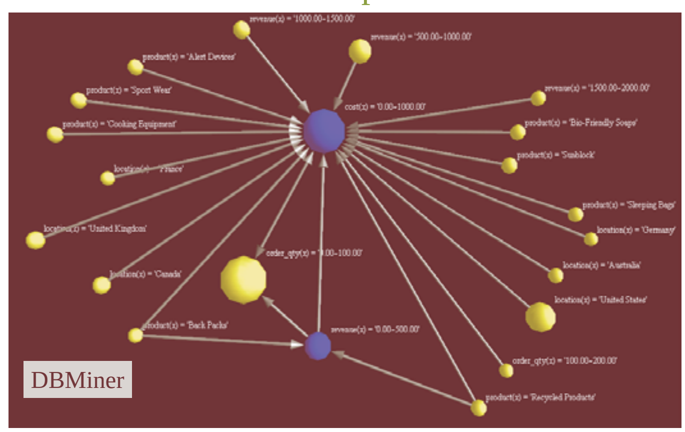
Sheet1

DBMiner

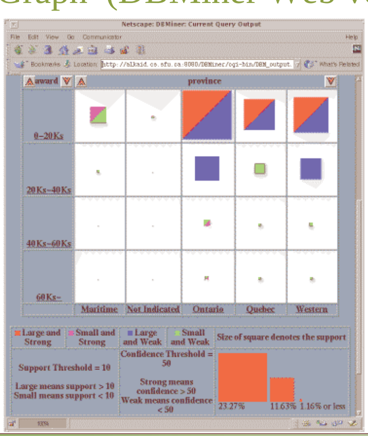
Visualization of Association Rule in Plane Form



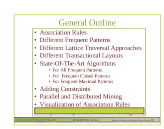
Visualization of Association Rule Using Rule Graph



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



What is Sequence Mining?



- Input
  - A database ***D*** of sequences called *data-sequences*, in which:
    - $I=\{i_1, i_2,\ldots,i_n\}$ is the set of items
    - each sequence is a list of transactions ordered by transaction-time
    - each transaction consists of fields: (sequence-id, transaction-id), transaction-time and a set of items.
- Problem
  - To discover all the sequential patterns with a user-specified minimum support

Input Database: example

Database *D*

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

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

•Subsequence:

A sequence $<a_1, a_2, \ldots, a_n>$ is contained in another sequence $<b_1, b_2, \ldots, b_m>$ if there exists integers $i_1 < i_2 < i_3 < \ldots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, …, $a_n \subseteq b_{in}$

subsequence	sequence	
<(3)(4 5) (8)>	<(7) (3 8) (9)(4 5 6) (8)>	√
<(3) (5)>	< (3 5) >	X
<(3) (5)>	< (3 5) (3 5) >	√

***k*-sequence**: a sequence that consists of *k* items
Examples: both <x,y> and <(x, y)> are 2-sequence

Example

Customer ID	Transaction Time	Items Bought
1	June 1	30
1	June 30	90
2	June 10	10,20
2	June 15	30
2	June 20	40, 60, 70
3	June 25	30, 50, 70
4	June 20	30
4	June 25	40, 70
4	June 30	90
5	June 12	90

Data base Sorted by customer ID and transaction time

Customer ID	Customer Sequence
1	<(30) (90)>
2	<(10 20) (30) (40 60 70)>
3	<(30 50 70)>
4	<(30) (40 70) (90)>
5	<(90)>

Customer sequence version of the database

Sequential Patterns with support = 2 customer, 2 sequences (25% support)
<(30) (90)>
<(30) (40 70)>

Answer Set

Algorithms

- **AprioriAll** (1995)
- **GSP**:Generalized Sequential Patterns (1996)
  - Both proposed by Rakesh Agrawal, Ramakrishnan Srikant
  - Both need multi-scans over the database
  - GSP outperforms AprioriAll and is able to discover generalized sequential patterns
- **SPADE**: (1998)
  - Proposed by Mohammed J. Zaki
  - Sequential Pattern Discovery using Equivalence classes
- **FreeSpan** (2000)
- **PrefixSpan** (2001)
  - Both proposed by Jian Pei et al.
  - Based on FP-Tree and FP-Growth model

References: Frequent-pattern Mining

- B. Lent, A. Swami, and J. Widom. Clustering association rules. ICDE'97, 220-231, Birmingham, England.
- R. Meo, G. Psaila, and S. Ceri. A new SQL-like operator for mining association rules. VLDB'96, 122-133, Bombay, India.
- R.J. Miller and Y. Yang. Association rules over interval data. SIGMOD'97, 452-461, Tucson, Arizona.
- A. Savasere, E. Omiecinski, and S. Navathe. Mining for strong negative associations in a large database of customer transactions. ICDE'98, 494-502, Orlando, FL, Feb. 1998.
- D. Tsur, J. D. Ullman, S. Abitboul, C. Clifton, R. Motwani, and S. Nestorov. Query flocks: A generalization of association-rule mining. SIGMOD'98, 1-12, Seattle, Washington.
- J. Pei, A.K.H. Tung, J. Han. Fault-Tolerant Frequent Pattern Mining: Problems and Challenges. SIGMOD DMKD'01, Santa Barbara, CA.
- Mohammad El-Hajj and Osmar R. Zaïane, Non Recursive Generation of Frequent K-itemsets from Frequent Pattern Tree Representations, in Proc. of 5th International Conference on Data Warehousing and Knowledge Discovery (DaWak'2003), pp 371-380, Prague, Czech Republic, September 3-5, 2003
- Mohammad El-Hajj and Osmar R. Zaïane, Inverted Matrix: Efficient Discovery of Frequent Items in Large Datasets in the Context of Interactive Mining, in Proc. 2003 Int'l Conf. on Data Mining and Knowledge Discovery (ACM SIGKDD), pp 109-118, Washington, DC, USA, August 24-27, 2003

Constraint-base Frequent-pattern Mining

- G. Grahne, L. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. ICDE'00, 512-521, San Diego, CA, Feb. 2000.
- Y. Fu and J. Han. Meta-rule-guided mining of association rules in relational databases. KDOOD'95, 39-46, Singapore, Dec. 1995.
- J. Han, L. V. S. Lakshmanan, and R. T. Ng, "Constraint-Based, Multidimensional Data Mining", COMPUTER (special issues on Data Mining), 32(8): 46-50, 1999.
- L. V. S. Lakshmanan, R. Ng, J. Han and A. Pang, "Optimization of Constrained Frequent Set Queries with 2-Variable Constraints", SIGMOD'99

References: Constraint-base Frequent Pattern Mining

- R. Ng, L.V.S. Lakshmanan, J. Han & A. Pang. "Exploratory mining and pruning optimizations of constrained association rules." SIGMOD'98
- J. Pei, J. Han, and L. V. S. Lakshmanan, "Mining Frequent Itemsets with Convertible Constraints", Proc. 2001 Int. Conf. on Data Engineering (ICDE'01), April 2001.
- J. Pei and J. Han "Can We Push More Constraints into Frequent Pattern Mining?", Proc. 2000 Int. Conf. on Knowledge Discovery and Data Mining (KDD'00), Boston, MA, August 2000.
- R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. KDD'97, 67-73, Newport Beach, California.
- C. Bucila, J. E. Gehrke, D. Kifer, and W. White. Dualminer: A dual-pruning algorithm for itemsets with constraints. In Proc. SIGKDD 2002.
- M. Kamber, J. Han, and J. Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. KDD'97, 207-210, Newport Beach, California.

Mining Maximal and Closed Patterns

- R. J. Bayardo. Efficiently mining long patterns from databases. SIGMOD'98, 85-93, Seattle, Washington.
- J. Pei, J. Han, and R. Mao, "CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets", Proc. 2000 ACM-SIGMOD Int. Workshop on Data Mining and Knowledge Discovery (DMKD'00), Dallas, TX, May 2000.
- N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. ICDT'99, 398-416, Jerusalem, Israel, Jan. 1999.
- M. Zaki. Generating Non-Redundant Association Rules. KDD'00. Boston, MA. Aug. 2000.
- M. Zaki. CHARM: An Efficient Algorithm for Closed Association Rule Mining, TR99-10, Department of Computer Science, Rensselaer Polytechnic Institute.
- M. Zaki, Fast Vertical Mining Using Diffsets, TR01-1, Department of Computer Science, Rensselaer Polytechnic Institute.
- D. Burdick, M. Calimlim, and J. Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In ICDE 2001. IEEE Computer Society, 2001

References: Sequential Pattern Mining

- R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95, 3-14, Taipei, Taiwan.
- R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. EDBT'96.
- J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M.-C. Hsu, "FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining", Proc. 2000 Int. Conf. on Knowledge Discovery and Data Mining (KDD'00), Boston, MA, August 2000.
- H. Mannila, H Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery, 1:259-289, 1997.
- J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth", Proc. 2001 Int. Conf. on Data Engineering (ICDE'01), Heidelberg, Germany, April 2001.
- B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. ICDE'98, 412-421, Orlando, FL.
- S. Ramaswamy, S. Mahajan, and A. Silberschatz. On the discovery of interesting patterns in association rules. VLDB'98, 368-379, New York, NY.
- M.J. Zaki. Efficient enumeration of frequent sequences. CIKM'98. Novermber 1998.
- M.N. Garofalakis, R. Rastogi, K. Shim: SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. VLDB 1999: 223-234, Edinburgh, Scotland.

References: Frequent-pattern Mining in Spatial, Multimedia, Text & Web Databases

- K. Koperski, J. Han, and G. B. Marchisio, "Mining Spatial and Image Data through Progressive Refinement Methods", Revue internationale de gomatique (European Journal of GIS and Spatial Analysis), 9(4):425-440, 1999.
- A. K. H. Tung, H. Lu, J. Han, and L. Feng, "Breaking the Barrier of Transactions: Mining Inter-Transaction Association Rules", Int. Conf. on Knowledge Discovery and Data Mining (KDD'99), San Diego, CA, Aug. 1999, pp. 297-301.
- J. Han, G. Dong and Y. Yin, "Efficient Mining of Partial Periodic Patterns in Time Series Database", Proc. 1999 Int. Conf. on Data Engineering (ICDE'99), Sydney, Australia, March 1999, pp. 106-115.
- H. Lu, L. Feng, and J. Han, "Beyond Intra-Transaction Association Analysis:Mining Multi-Dimensional Inter-Transaction Association Rules", ACM Transactions on Information Systems (TOIS'00), 18(4): 423-454, 2000.
- O. R. Zaiane, M. Xin, J. Han, "Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs," Proc. Advances in Digital Libraries Conf., Santa Barbara, CA, April 1998, pp. 19-29
- O. R. Zaiane, J. Han, and H. Zhu, "Mining Recurrent Items in Multimedia with Progressive Resolution Refinement", Proc. 2000 Int. Conf. on Data Engineering (ICDE'00), San Diego, CA, Feb. 2000, pp. 461-470.

FIM for Classification & Data Cube Computation

- K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. SIGMOD 1999, pp 359-370
- M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman. Computing iceberg queries efficiently. VLDB, 299-310, New York, NY, Aug. 1998.
- J. Han, J. Pei, G. Dong, and K. Wang, Computing Iceberg Data Cubes with Complex Measures, SIGMOD 2001.
- K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. SIGMOD 1999
- B. Liu, W. Hsu and Y. Ma, Integrating classification and association rule mining, SIGKDD 1998, pp 80-86
- M-L. Antonie, O. R. Zaïane, Text Document Categorization by Term Association , in Proc. of the IEEE International Conference on Data Mining (ICDM'2002), pp 19-26, Maebashi City, Japan.

References: Parallel Frequent itemset mining

- Agrawal, R., Shafer, J.: Parallel Mining of Association Rules. IEEE Transactions in Knowledge and Data Eng. 8 (1996) pp. 962-969
- Barry Wilkinson, Michael Allen: Parallel Programming Techniques and applications using networked workstations and parallel computers. Alan Apt, New Jersy, USA. 1999
- D. Cheung, K. Hu, S. Xia: Asynchronous Parallel Algorithm for Mining Association Rules on a Shared-memory Multi-processors. Proc. 10th ACM Symp. Parallel Algorithms and architectures,ACM Press, NY, 1998, pp 279 - 288
- David Wai-Lok Cheung, Jiawei Han, Vincent Ng, Ada Wai-Chee Fu, Yongjian Fu: A Fast Distributed Algorithm for Mining Association Rules. PDIS 1996: pp. 31-42
- David Wai-Lok Cheung, Yongqiao Xiao: Effect of Data Distribution in Parallel Mining of Associations. Data Mining and Knowledge Discovery 3(3): pp.291-314 (1999)
- E-H. Han, G. Karypis, and V. Kumar: Scalable parallel data mining for association rules. In ACM SIGMOD Conf. Management of Data, May 1997
- J. S. Park, M. Chen, and P. S. Yu: Efficient parallel data mining for association rules. In ACM Intl. Conf. Infomration and Knowledge Management, November 1995
- Mohammed J. Zaki and Ching-Tien Ho (editors): Large-scale Parallel Data Mining, Lecture Notes in Artificial Intelligence, State-of-the-Art-Survey, Volume 1759, Springer-Verlag, 2000

References: Parallel Frequent itemset mining

- Mohammed J. Zaki, Parallel and Distributed Association Mining: A Survey, in IEEE Concurrency, special issue on Parallel Mechanisms for Data Mining, Vol. 7, No. 4, pp14-25, December, 1999
- Osmar R. Zaïane, Mohammad El-Hajj, Paul Lu: Fast Parallel Association Rule Mining without Candidacy Generation, in Proc. of the IEEE 2001 International Conference on Data Mining (ICDM'2001), San Jose, CA, USA
- Mohammad El-Hajj and Osmar R. Zaïane, Parallel Association Rule Mining with Minimum Inter-Processor Communication, Fifth International Workshop on Parallel and Distributed Databases (PaDD'2003) in conjunction with the 14th Int' Conf. on Database and Expert Systems Applications DEXA2003, pp 519-523, Prague, Czech Republic, September 1-September 5, 2003
- R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo: Fast discovery of association rules. In Fayyad et al, AAAI Press, Menio Park, Calif pp. 307--328, 1996
- Skillicorn, D: Strategies for Parallel Data Mining. IEEE concurrency 7 (1999) pp. 26-35
- Srinivasan Parthasarathy, Mohammed Zaki, Mitsunori Ogihara, Wei Li: Parallel Data Mining for Association Rules on Shared-memory Systems, in Knowledge and Information Systems, Volume 3, Number 1, pp 1-29, Feb 2001
- Takahiko Shintani,Masaru Kitsuregawa: Hash Based Parallel Algorithms for Mining Association Rules. Proceedings of IEEE Fourth International Conference on Parallel and Distributed Information Systems, pp.19-30, 1996.12
- W. A. Maniatty, Mohammed J. Zaki, A Requirement Analysis for Parallel KDD Systems , 3rd IPDPS Workshop on High Performance Data Mining , Cancun, Mexico, May 2000