



Association Rule Mining II

COMP 790-90 Seminar

BCB 713 Module

Spring 2009

The UNIVERSITY of NORTH CAROLINA at CHAPEL HILL



Bottleneck of Frequent-pattern Mining

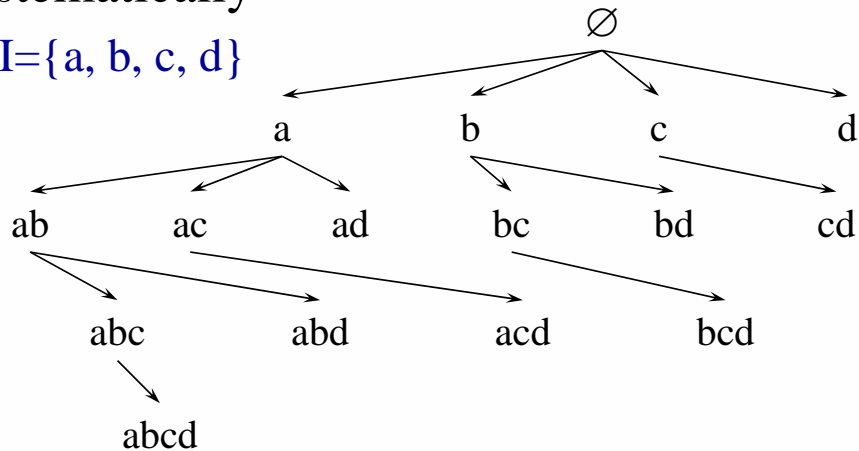
- ▶ Multiple database scans are costly
- ▶ Mining long patterns needs many passes of scanning and generates lots of candidates
 - ▶ To find frequent itemset $i_1 i_2 \dots i_{100}$
 - ▶ # of scans: 100
 - ▶ # of Candidates: $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} = 2^{100} - 1 \approx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



Set Enumeration Tree

- ▶ Subsets of I can be enumerated systematically

- ▶ $I = \{a, b, c, d\}$

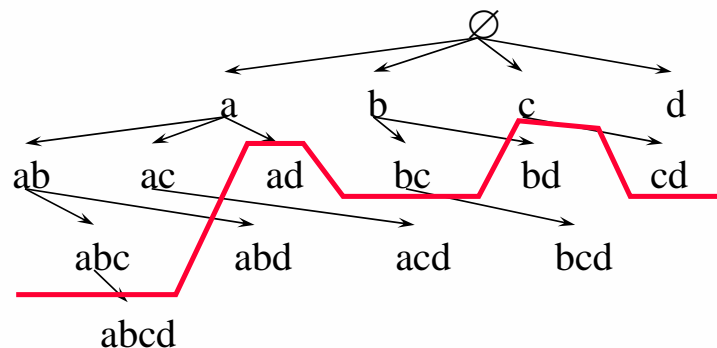


3



Borders of Frequent Itemsets

- ▶ Connected
 - ▶ X and Y are frequent and X is an ancestor of Y
 - all patterns between X and Y are frequent

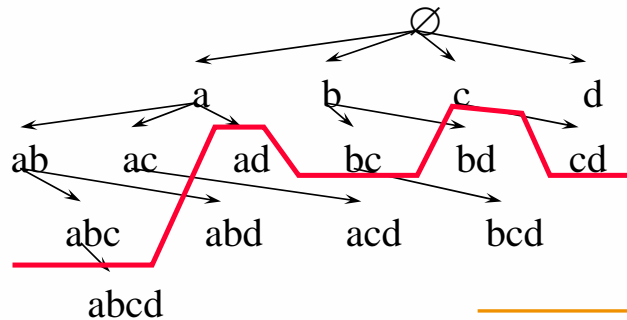


4



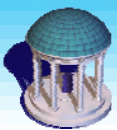
Projected Databases

- ▶ To find a child Xy of X , only X -projected database is needed
 - ▶ The sub-database of transactions containing X
 - ▶ Item y is frequent in X -projected database



5

COMP 790-090 Data Mining: Concepts, Algorithms, and Applications



Tree-Projection Method

- ▶ Find frequent 2-itemsets
- ▶ For each frequent 2-itemset xy , form a projected database
 - ▶ The sub-database containing xy
- ▶ Recursive mining
 - ▶ If $x'y'$ is frequent in xy -proj db, then $xyx'y'$ is a frequent pattern

6

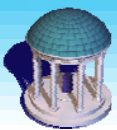
COMP 790-090 Data Mining: Concepts, Algorithms, and Applications



Why Is Tree-projection Fast?

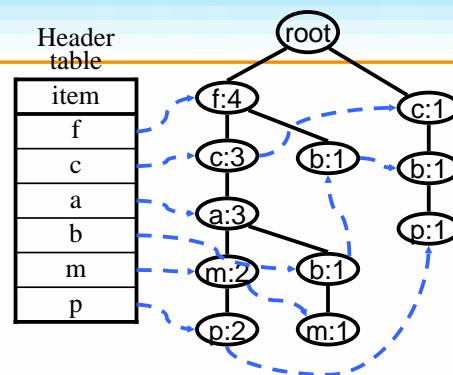
- ▶ A bi-level unfolding of set enumeration tree
- ▶ Major operations
 - ▶ Finding frequent 2-itemsets: faster than matching candidates
 - ▶ Form projected databases
- ▶ AAP'01

7



Compress Database by FP-tree

- ▶ 1st scan: find freq items
 - ▶ Only record freq items in FP-tree
 - ▶ F-list: f-c-a-b-m-p
- ▶ 2nd scan: construct tree
 - ▶ Order freq items in each transaction w.r.t. f-list
 - ▶ Explore sharing among transactions



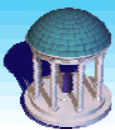
TI D	Items bought	(ordered) freq items
100	f, a, c, d, g, l, m, p	f, c, a, m, p
200	a, b, c, f, l, m, o	f, c, a, b, m
300	b, f, h, j, o	f, b
400	b, c, k, s, p	c, b, p
500	a, f, c, e, l, p, m, n	f, c, a, m, p

8



Benefits of FP-tree

- ▶ Completeness
 - ▶ Never break a long pattern in any transaction
 - ▶ Preserve complete information for freq pattern mining
 - ▶ No need to scan database anymore
- ▶ Compactness
 - ▶ Reduce irrelevant info — infrequent items are gone
 - ▶ Items in frequency descending order (f-list): the more frequently occurring, the more likely to be shared
 - ▶ Never be larger than the original database (not counting node-links and the count fields)



Partition Frequent Patterns

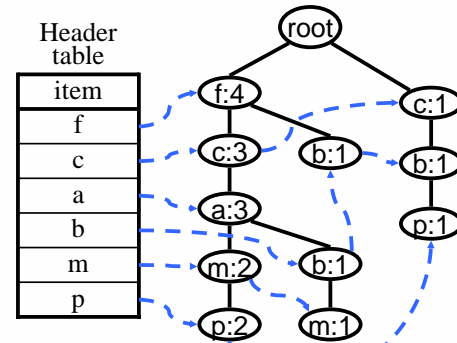
- ▶ Frequent patterns can be partitioned into subsets according to f-list: f-c-a-b-m-p
 - ▶ Patterns containing p
 - ▶ Patterns having m but no p
 - ▶ ...
 - ▶ Patterns having c but no a nor b, m, or p
 - ▶ Pattern f
- ▶ The partitioning is complete and without any overlap



Find Patterns Having Item "p"

- ▶ Only transactions containing p are needed
- ▶ Form p-projected database
 - ▶ Starting at entry p of header table
 - ▶ Follow the side-link of frequent item p
 - ▶ Accumulate all transformed prefix paths of p

p-projected database $TDB|_p$
 fcam: 2
 cb: 1
 Local frequent item: c:3
 Frequent patterns containing p
 p: 3, pc: 3

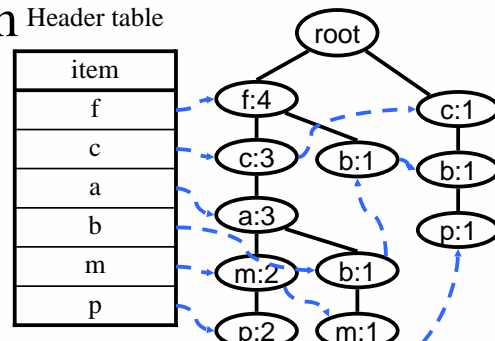
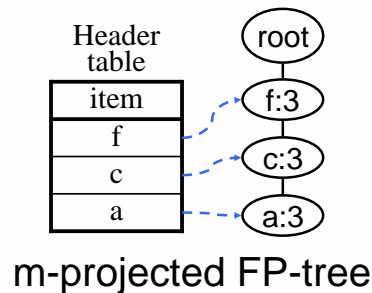


11



Find Patterns Having Item m But No p

- ▶ Form m-projected database $TDB|_m$
 - ▶ Item p is excluded
 - ▶ Contain fca:2, fcab:1
 - ▶ Local frequent items: f, c, a
- ▶ Build FP-tree for $TDB|_m$

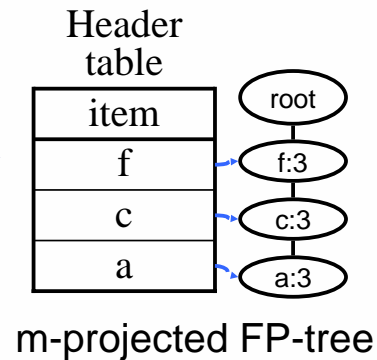


12



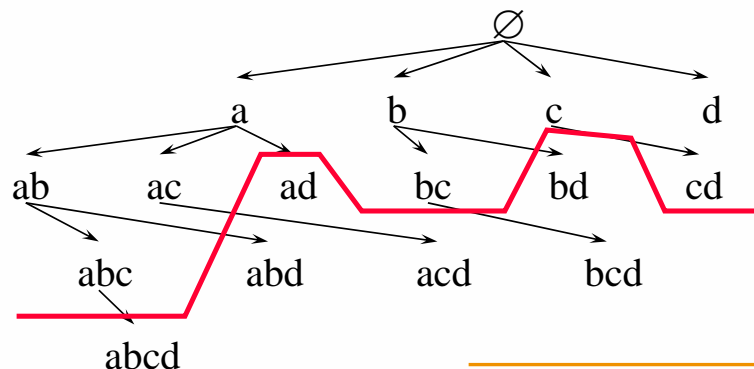
Recursive Mining

- ▶ Patterns having m but no p can be mined recursively
- ▶ Optimization: enumerate patterns from single-branch FP-tree
 - ▶ Enumerate all combination
 - ▶ Support = that of the last item
 - ▶ m, fm, cm, am
 - ▶ fcm, fam, cam
 - ▶ fcam



Borders and Max-patterns

- ▶ Max-patterns: borders of frequent patterns
 - ▶ A subset of max-pattern is frequent
 - ▶ A superset of max-pattern is infrequent





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, DE,

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

- ▶ Baya'98

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

Min_sup=2

Potential max-patterns



Frequent Closed Patterns

- ▶ For frequent itemset X, if there exists no item y s.t. every transaction containing X also contains y, then X is a frequent closed pattern

- ▶ “acdf” is a frequent closed pattern

- ▶ Concise rep. of freq pats

- ▶ Reduce # of patterns and rules

- ▶ N. Pasquier et al. In ICDT'99

Min_sup=2

TID	Items
10	a, c, d, e, f
20	a, b, e
30	c, e, f
40	a, c, d, f
50	c, e, f



CLOSET: Mining Frequent Closed Patterns

- ▶ Flist: list of all freq items in support asc. order
 - ▶ Flist: d-a-f-e-c
- ▶ Divide search space
 - ▶ Patterns having d
 - ▶ Patterns having d but no a, etc.
- ▶ Find frequent closed pattern recursively
 - ▶ Every transaction having d also has cfa → cfad is a frequent closed pattern
- ▶ PHM'00

Min_sup=2

TID	Items
10	a, c, d, e, f
20	a, b, e
30	c, e, f
40	a, c, d, f
50	c, e, f



The CHARM Method

- ▶ Use vertical data format: $t(AB) = \{T1, T12, \dots\}$
- ▶ Derive closed pattern based on vertical intersections
 - ▶ $t(X) = t(Y)$: X and Y always happen together
 - ▶ $t(X) \subset t(Y)$: transaction having X always has Y
- ▶ Use diffset to accelerate mining
 - ▶ Only keep track of difference of tids
 - ▶ $t(X) = \{T1, T2, T3\}$, $t(Xy) = \{T1, T3\}$
 - ▶ $\text{Diffset}(Xy, X) = \{T2\}$



Closed and Max-patterns

- ▶ Closed pattern mining algorithms can be adapted to mine max-patterns
 - ▶ A max-pattern must be closed
- ▶ Depth-first search methods have advantages over breadth-first search ones