Data Mining:

Concepts and Techniques

Pattern Discovery

Classification

Clustering

(3rd ed.)

— Chapter 6 —

Slides Courtesy of Textbook

Chapter 6: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Frequent Itemset Mining Methods
- Which Patterns Are Interesting?—Pattern Evaluation Methods
- Summary

Motivation

Pattern: (Regularity)

Entity: a set of attributes

- Useful information
  - Association
  - The more items and interaction, the more information
- Finding inherent regularities in data
  - What products were often purchased together? Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to a new drug?
  - Can we automatically classify web documents by their regular patterns?

Method: Frequent Pattern Analysis

Assiciation

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- Problem definition - beginning of data mining
  - First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- An intrinsic and important property of datasets
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Classification: discriminative, frequent pattern analysis
  - Cluster analysis: frequent pattern-based clustering

Market-Basket data

Basic Concepts: Frequent Patterns

Input: data = {Transactions}

No Brand, price and quantity

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

Let minsup = 50%

Freq. Pat.:
{Beer}:3, {Nuts}:3, {Diaper}:4,
{Eggs}:3, {Beer, Diaper}:3

Find all itemsets X with min support

- Transactions: each is an itemset.
- itemset: A set of one or more items
- k-itemset $X = \{x_1, \ldots, x_k\}$ {Beer} {Beer,Nuts}
- *(absolute) support*, or, *support count* of X: Frequency or occurrence of an itemset X S({Beer}) = 3
- *(relative) support* of X: Fraction of transactions that contains X (i.e., the probability that a transaction contains X)
- An itemset X is *frequent* if X's support is no less than a *minsup* threshold (count or fraction)

Basic Concepts: Association Rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

Let minsup = 50%, minconf = 50%

Association rules: (many more!)

- *Beer* → *Diaper* (60%, 100%)
- *Diaper* → *Beer* (60%, 75%)

P(X,Y)

Find all rules X → Y with minimum support and confidence S({B,D})>=3

- support, *s*, probability that a transaction contains X ∪ Y
  - s=#T(X∪Y)/#T(Total)
- confidence, *c,* conditional probability that a transaction having X also contains Y
  - c=#T(X∪Y)/#T(X) S(B and D) / S(B)
- A rule X→Y is an association rule if s(X→Y) is no less than minsup and c(X→Y) is no less than minconf

P(Y|X)

Closed Patterns and Max Patterns

- Too many patterns, information can be redundant
  - A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$ contains $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 = 1.27 * 10^{30}$ sub-patterns!

any subset of any size

减的1是0 itemset

- Consider longer patterns
  - More items included, more information (few patterns)
  - Solution: *Mine closed patterns and max patterns*

Closed Patterns and Max Patterns

{B}: 3 Not closed {B,D}: 3   Closed, so this pattern is interesting   {B,D,C}:1

- An itemset X is closed if X is *frequent* and there exists *no super-pattern* X' ɔ X that *has the same support as* X.

- An itemset X is a max pattern if X is frequent and there exists no super-pattern X' ɔ X that *is also frequent* (i.e., >= minsup).

Max implies closed, but not versa vice

Closed Patterns and Max Patterns

- Closed patterns: support(super(X)) < support(X)
- Max patterns: support(super(X)) < minsup

- Example: Suppose a DB contains only two transactions
  - $<a_1, \ldots, a_{100}>, <a_1, \ldots, a_{50}>$ (Let minsup = 1)
- What is the set of closed itemset?
  - $\{a_1, \ldots, a_{100}\}$: 1, $\{a_1, \ldots, a_{50}\}$: 2
- What is the set of max pattern?
  - $\{a_1, \ldots, a_{100}\}$: 1
- What is the set of all patterns?
  - $\{a_1\}$: 2, …, $\{a_1, a_2\}$: 2, …, $\{a_1, a_{51}\}$: 1, …, $\{a_1, a_2, \ldots, a_{100}\}$: 1
  - A big number: $2^{100} - 1$? Why?

Chapter 5: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Frequent Itemset Mining Methods
- Which Patterns Are Interesting?—Pattern Evaluation Methods
- Summary

Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach

- FPGrowth: A Frequent Pattern-Growth Approach

- Mining Closed patterns and Max patterns

Frequent Pattern Mining Motivations

- Nature of frequent pattern mining
  - Count the number of occurrences (frequency)
  - Compare to a minimum support threshold
- The downward closure property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}

Apriori: A Candidate Generation & Test Approach

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

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

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

1st scan

C_1

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

L_1

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

C_2

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

2nd scan

C_2

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

L_2

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

C_3

Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

Finally, Generating AR From Frequent

S{B,C,E}>=min-sup

- For each frequent itemset L, generate all X, Y such that XUY = L (and X and Y non-overlappin).

- Check confidence C(X → Y) = Support(XUY = L)/Support(X) >= minsup

- If so, output X → Y.

B -> CE
C -> BE
E -> BC

BC -> E
BE -> C
CE ->B

The Apriori Algorithm (Pseudo-Code)

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

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

Implementation of Apriori

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

- Agree on k-1 items
- Only join with alphabetically later itemsets

Further Improvement of Apriori

- Major computational challenges
  - Multiple scans of transaction database
    the same
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach
- FPGrowth: A Frequent Pattern-Growth Approach
- Mining Closed patterns and Max patterns

Pattern-Growth Approach: Mining Frequent Patterns Without Candidate Generation

- Bottlenecks of the Apriori approach
  - Breadth-first (i.e., level-wise) search
  - Expensive candidate generation and test
    - Often generates a huge number of candidates
- The FPGrowth Approach (J. Han, J. Pei, and Y. Yin, SIGMOD' 00)
  - Depth-first search
  - Avoid explicit candidate generation
- Major philosophy: Grow long patterns from short ones using local frequent items only
  - Build a tree data structure to represent the transaction database
  - Do counting on the tree representation

Construct FP-tree from a Transaction Database

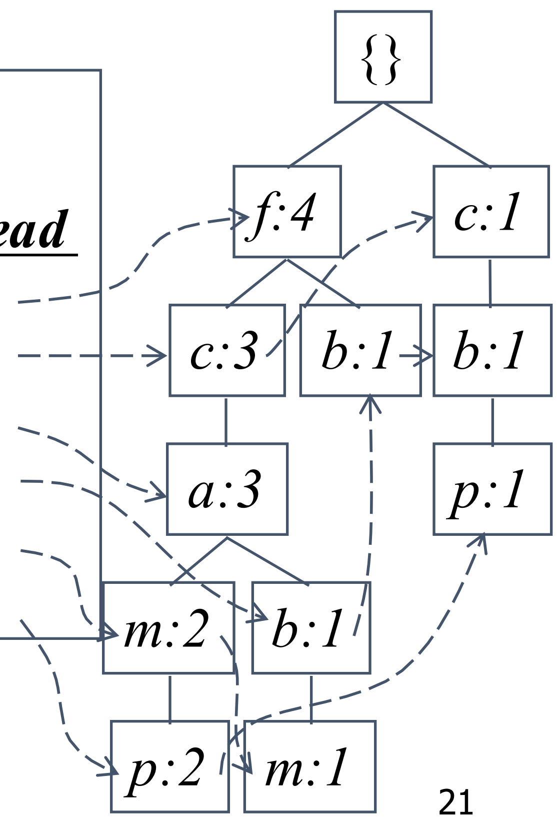
TID	***Items bought***	***(ordered) frequent items***
100	***{f, a, c, d, g, i, m, p}***	***{f, c, a, m, p}***
200	***{a, b, c, f, l, m, o}***	***{f, c, a, b, m}***
300	***{b, f, h, j, o, w}***	***{f, b}***
400	***{b, c, k, s, p}***	***{c, b, p}***
500	***{a, f, c, e, l, p, m, n}***	***{f, c, a, m, p}***

min_support = 3

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

Header Table

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



FP-tree Construction Example

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

- FP-tree is a compression of the original database
  - Omit lease frequent items
  - Share prefixes
- Why order by frequency?
  - More likely to share nodes (more compression)

Counting Frequent Patterns on FP-tree

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

Header Table

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

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

Divide and Conquer

- Partition frequent patterns into subsets according to F-list
- Patterns containing p
- Patterns having m but not p
- …
- Patterns having c but not a,b,m,p
- Pattern f

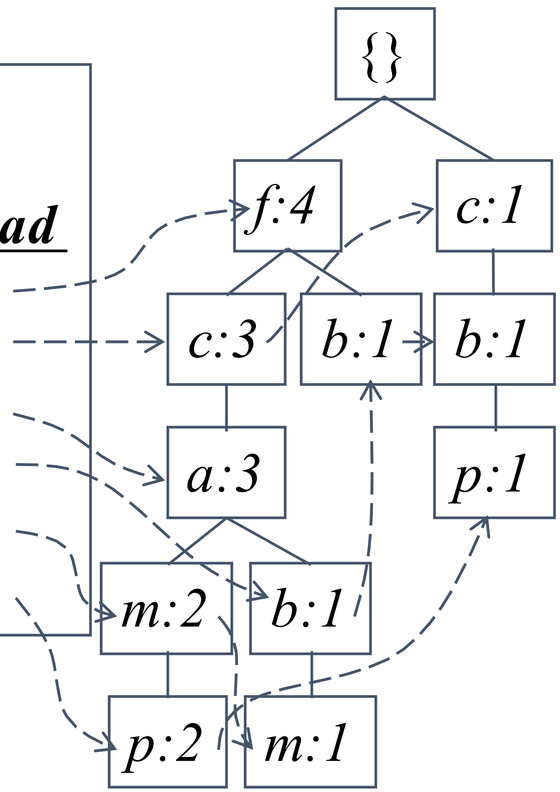
Completeness and no redundancy

Find Conditional Pattern-base

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

Header Table

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

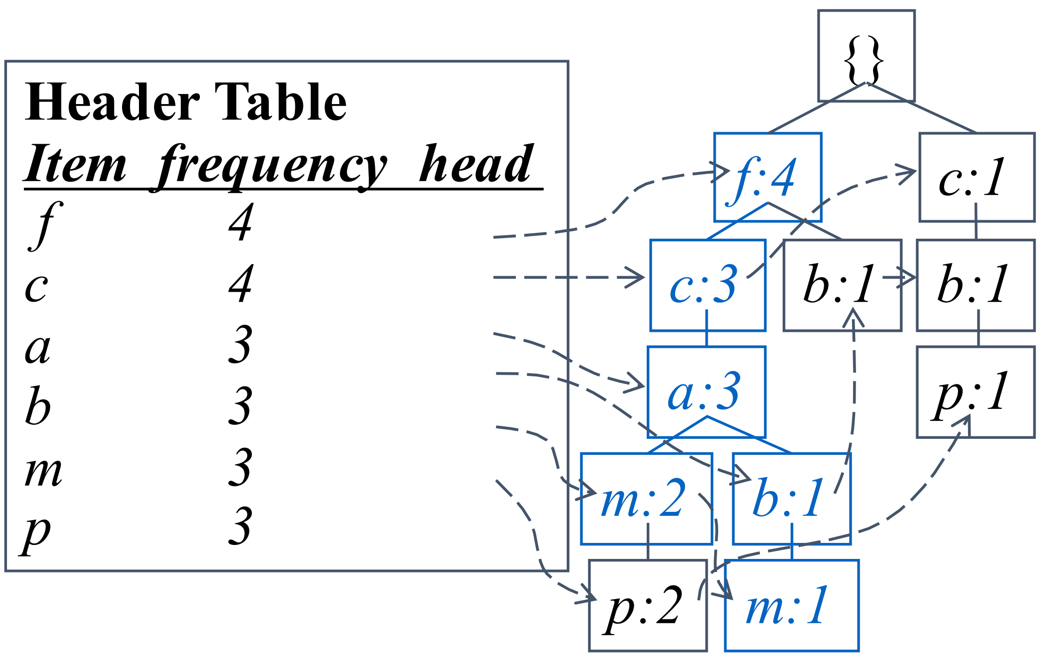


***Conditional* pattern bases**

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

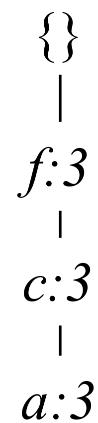
From Conditional Pattern-base to Conditional FP-tree

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



m-conditional **pattern base:**

fca:2, fcab:1



m-conditional **FP-tree**

All frequent patterns relate to *m*

m,

fm, cm, am,

fcm, fam, cam,

fcam

Recursion: Mining Each Conditional FP-tree

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

m-conditional **FP-tree**

Cond. pattern base of “am”: (fc:3)

{}
|
f:3
|
c:3

am-conditional **FP-tree**

Cond. pattern base of “cm”: (f:3)

{}
|
f:3

cm-conditional **FP-tree**

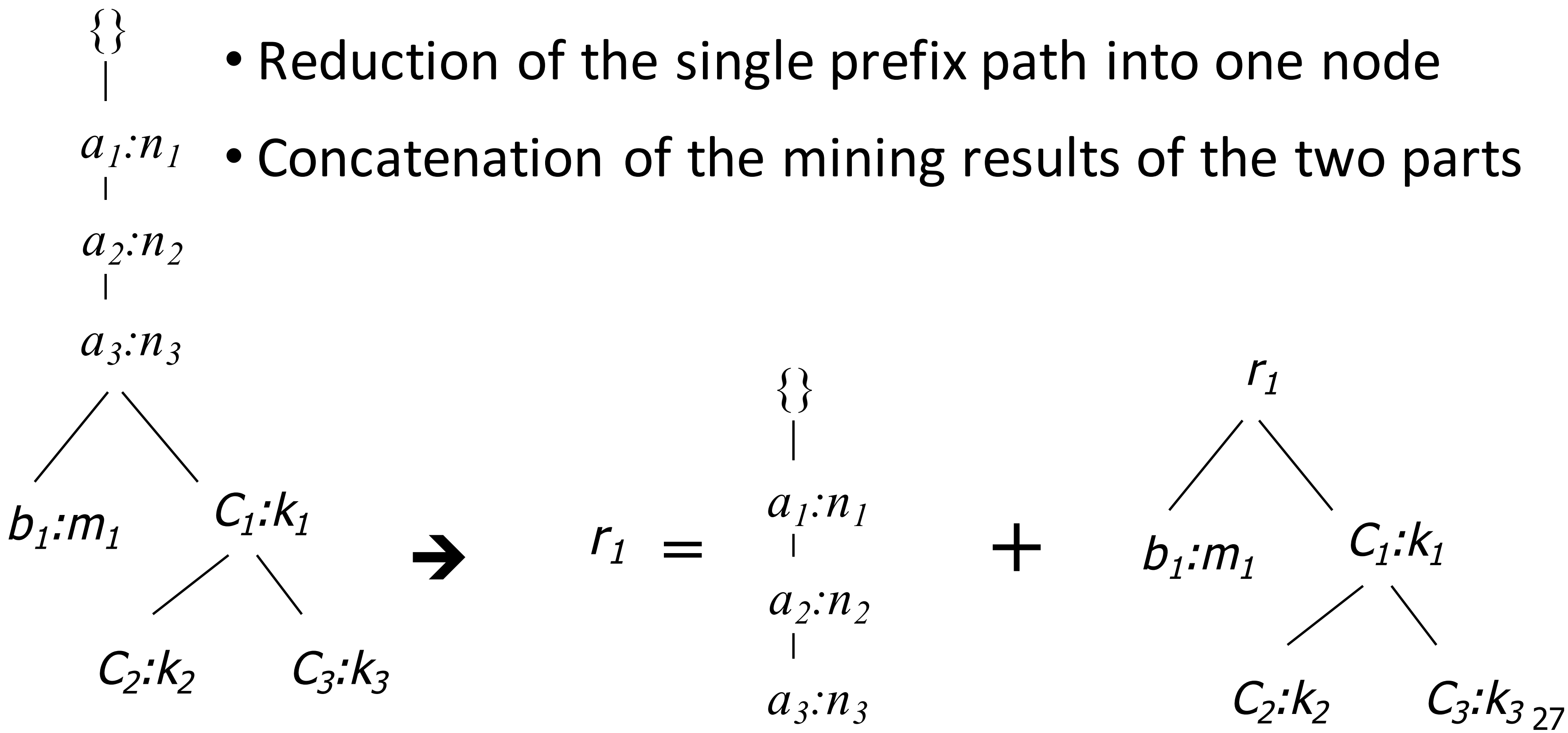
Cond. pattern base of “cam”: (f:3)

{}
|
f:3

cam-conditional **FP-tree**

A Special Case: Single Prefix Path in FP-tree

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



Implementation of FP-Growth

- Step 1: FP-tree construction
  - Scan the database *D* once. Collect the set of frequent items, and sort it in support count descending order as *L*. Create the root of *FP_tree* and label its item-name as "*null*".
  - For each transaction T_i in *D*, select and sort T_i to the order of *L*. Let the sorted frequent item list in Trans be *[p|P]*, where *p* is the first element and *P* is the remaining list. Call *insert_tree([p|P], FP_tree)*

Procedure *insert_tree([p|P], T)*

```
if T has a child N such that N.item-name = p.item-name then
    N.count++;
else
    create a new node N;
    N.count=1; T.child->N; p.node_link->N;
if P is nonempty then
    call insert_tree(P, N);
```

See textbook 6.2.4 Figure 6.9

Implementation of FP-Growth

- Step 2: Mining FP-tree. Call *FP_growth(FP_tree, null)*

procedure *FP_growth(Tree, α)*

if *Tree* contains a single path *P* then

for each combination (denoted as *β*) of the nodes in the path P

generate pattern $\beta \cup \alpha$ with *support_count = minimum support_count of nodes in β;*

else for each a_i in the header of *Tree* {

generate pattern $\beta = a_i \cup \alpha$ with *support_count =* a_i*.support_count*;
construct *β*'s conditional pattern base

construct *β*'s conditional FP tree $Tree_\beta$;

if $Tree_\beta$ is nonempty **then**

call *FP_growth*($Tree_\beta$, β); }

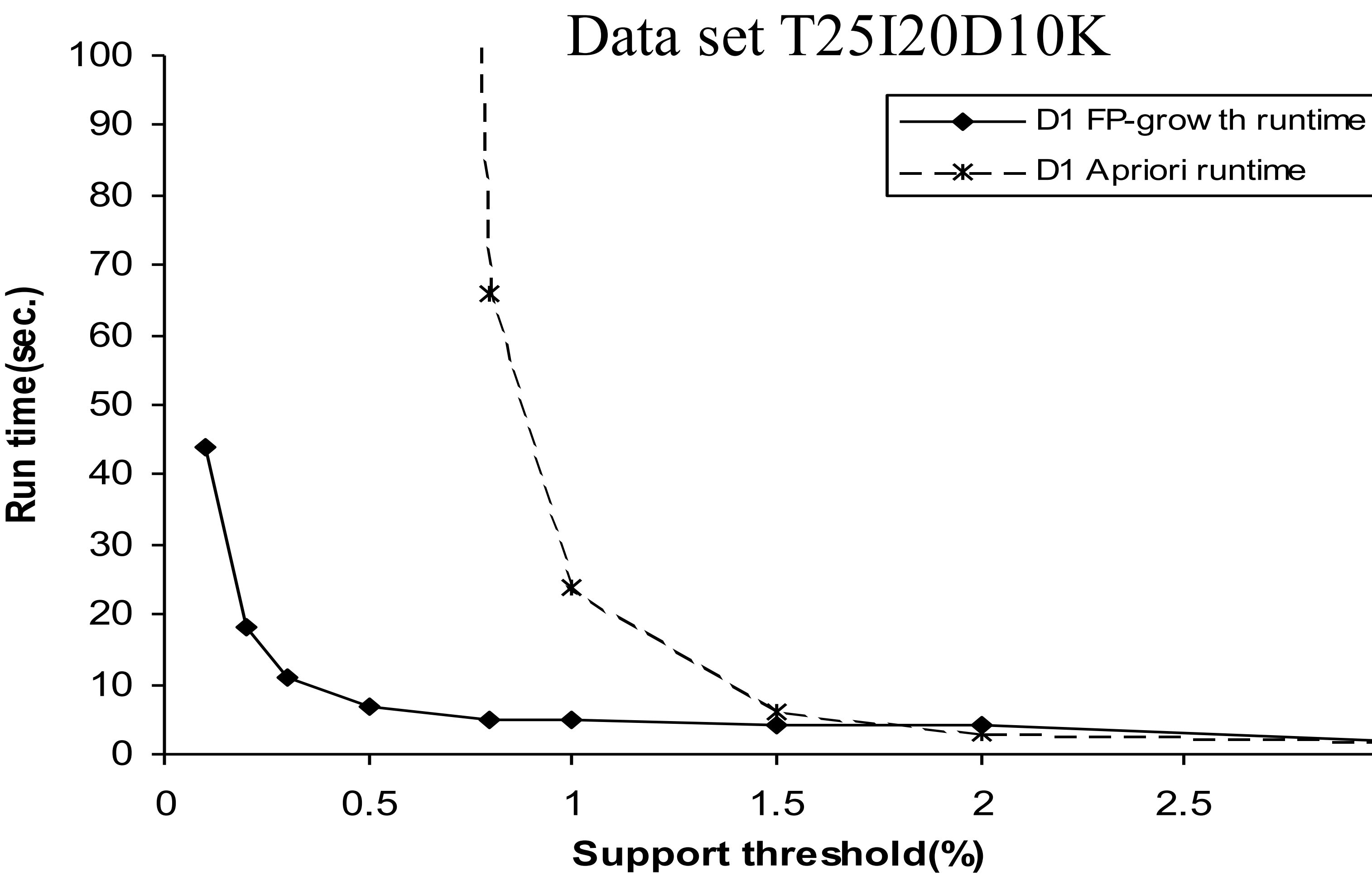
FP-Growth Summery

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

Benefits of the FP-tree Structure

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

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



Further Improvement of FP-Growth

- What about if FP-tree cannot fit in memory?
  - DB projection
- First partition a database into a set of projected DBs
- Then construct and mine FP-tree for each projected DB
- Parallel projection vs. partition projection techniques

Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach
- FPGrowth: A Frequent Pattern-Growth Approach
- Mining Closed patterns and Max patterns

Why Mining Closed and Max Patterns?

- Mining frequent itemsets often generates a large number of frequent itemsets
- Mining frequent closed and max itemsets has the same power as mining the complete set of frequent itemsets, but it substantially reduces redundant rules to be generated
  - Increase both efficiency and effectiveness

TDB
$(a_1 a_2 \ldots a_{100})$
$(a_1 a_2 \ldots a_{50})$

min_sup=1
min_conf=50%

2^{100}-1 frequent itemsets
$a_1, \ldots, a_{100}, a_1 a_2, \ldots, a_{99} a_{100}, \ldots, a_1 a_2 \ldots a_{100}$

2 frequent closed itemsets
$a_1 a_2 \ldots a_{100}, a_1 a_2 \ldots a_{50}$

1 frequent max itemsets
$a_1 a_2 \ldots a_{100}$

CLOSET: Mining Frequent Closed Patterns

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

min_sup =2

Step 1. Generate frequent item list(f_list)

List of frequent items in support descending order

f_list=<c:4, e:4, f:4, a:3, d:2>

CLOSET: Mining Frequent Closed Patterns

- All possible frequent closed patterns can be divided into 5 non-overlapping search spaces based on f_list, i.e. x-conditional database($TDB|_x$)
  - $TDB|_d$: The ones containing d
  - $TDB|_a$: The ones containing a but no d
  - $TDB|_f$: The ones containing f but no a nor d
  - $TDB|_e$: The ones containing e but no f, a nor d
  - $TDB|_c$: The ones containing only c

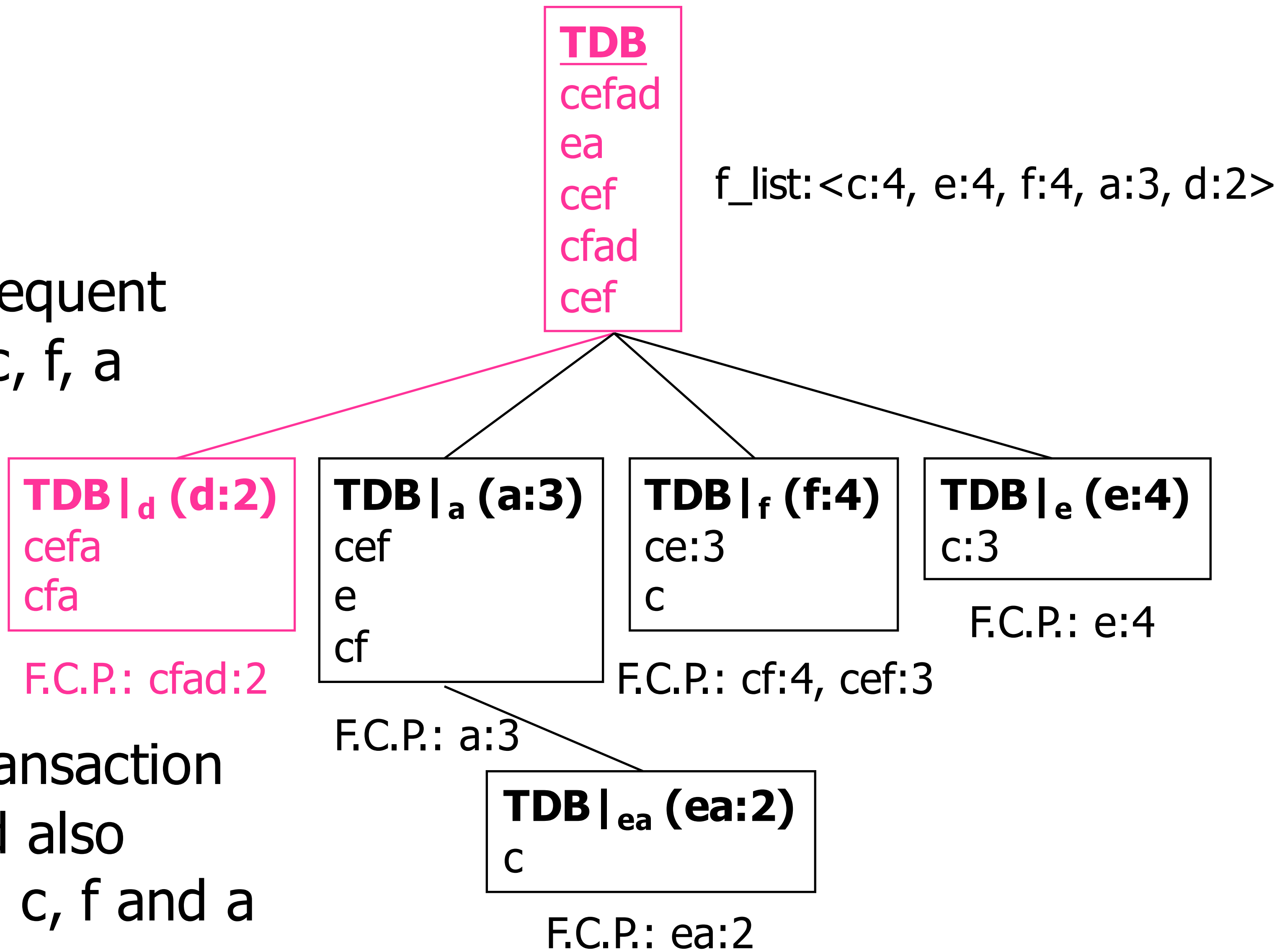
Conditional DB	Itemsets
$TDB\|_d$	cefa, cfa
$TDB\|_a$	cef, e, cf
$TDB\|_f$	ce:3, c
$TDB\|_e$	c:3
$TDB\|_c$	{}

Step 2. Divide Search Space

f_list=<c:4, e:4, f:4, a:3, d:2>

Find Frequent Closed Patterns Containing d

Local frequent items: c, f, a

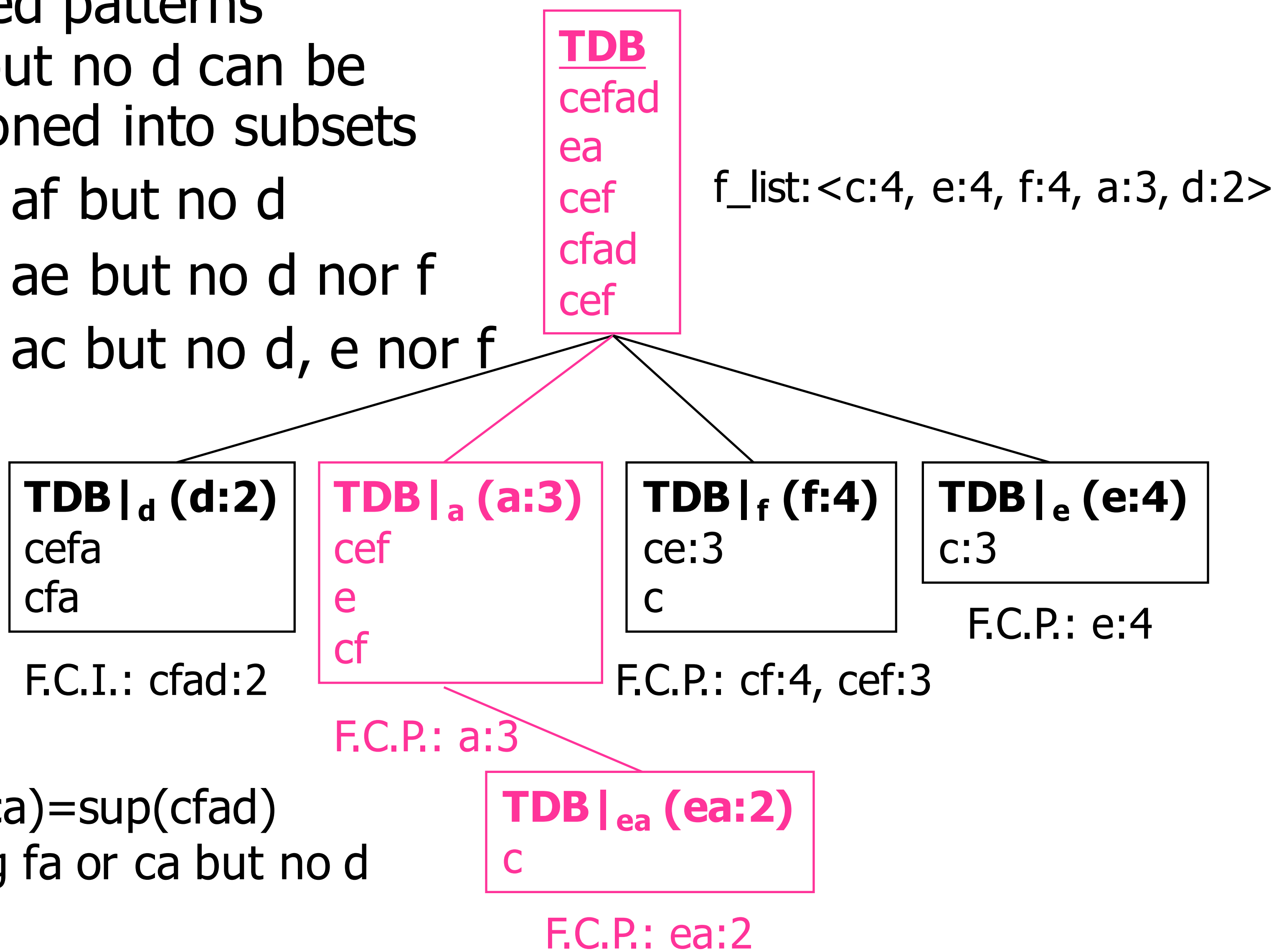


Every transaction having d also contains c, f and a

Find Frequent Closed Patterns Containing a but No d

Frequent closed patterns containing a but no d can be further partitioned into subsets

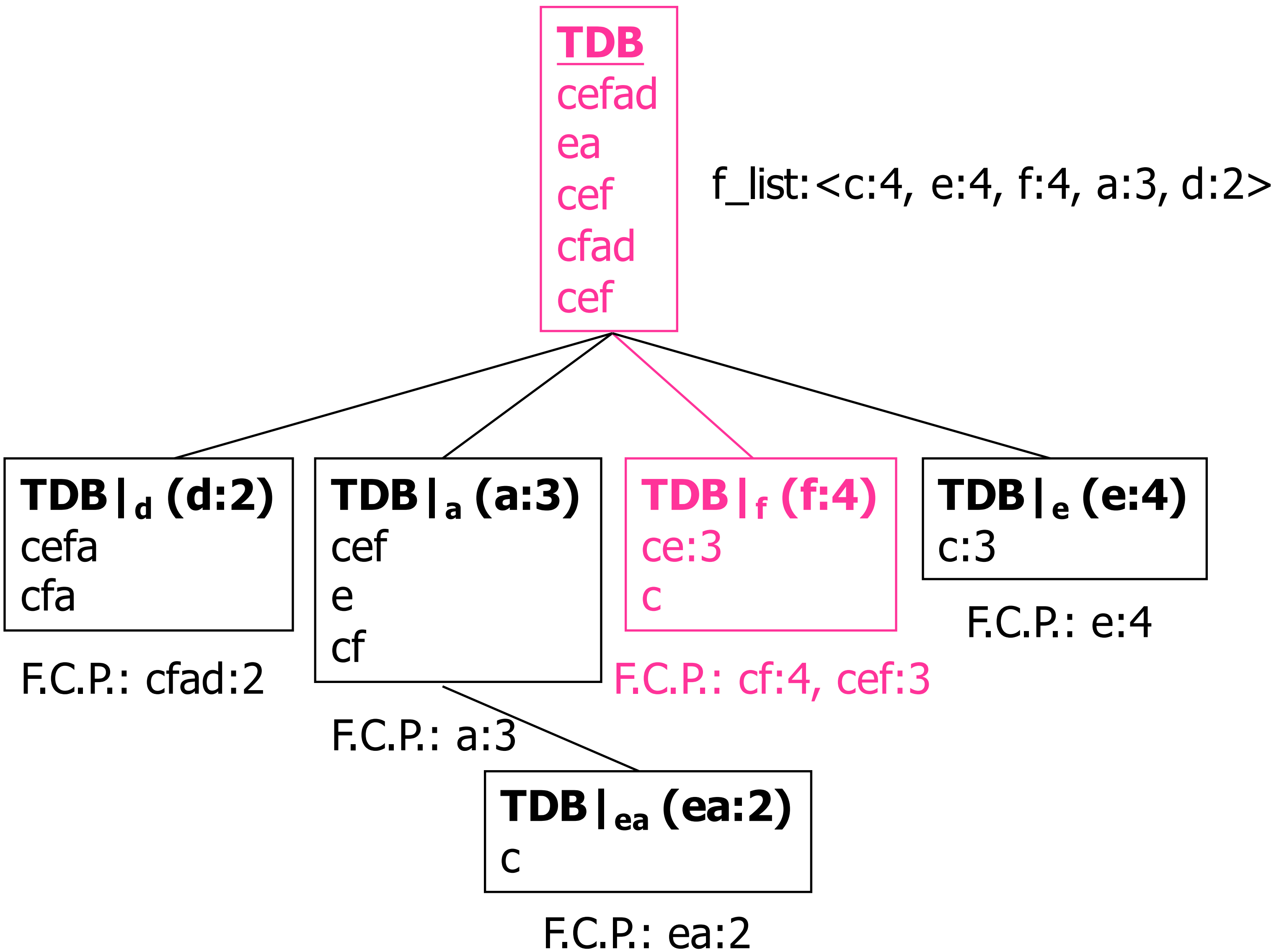
- Ones having af but no d
- Ones having ae but no d nor f
- Ones having ac but no d, e nor f



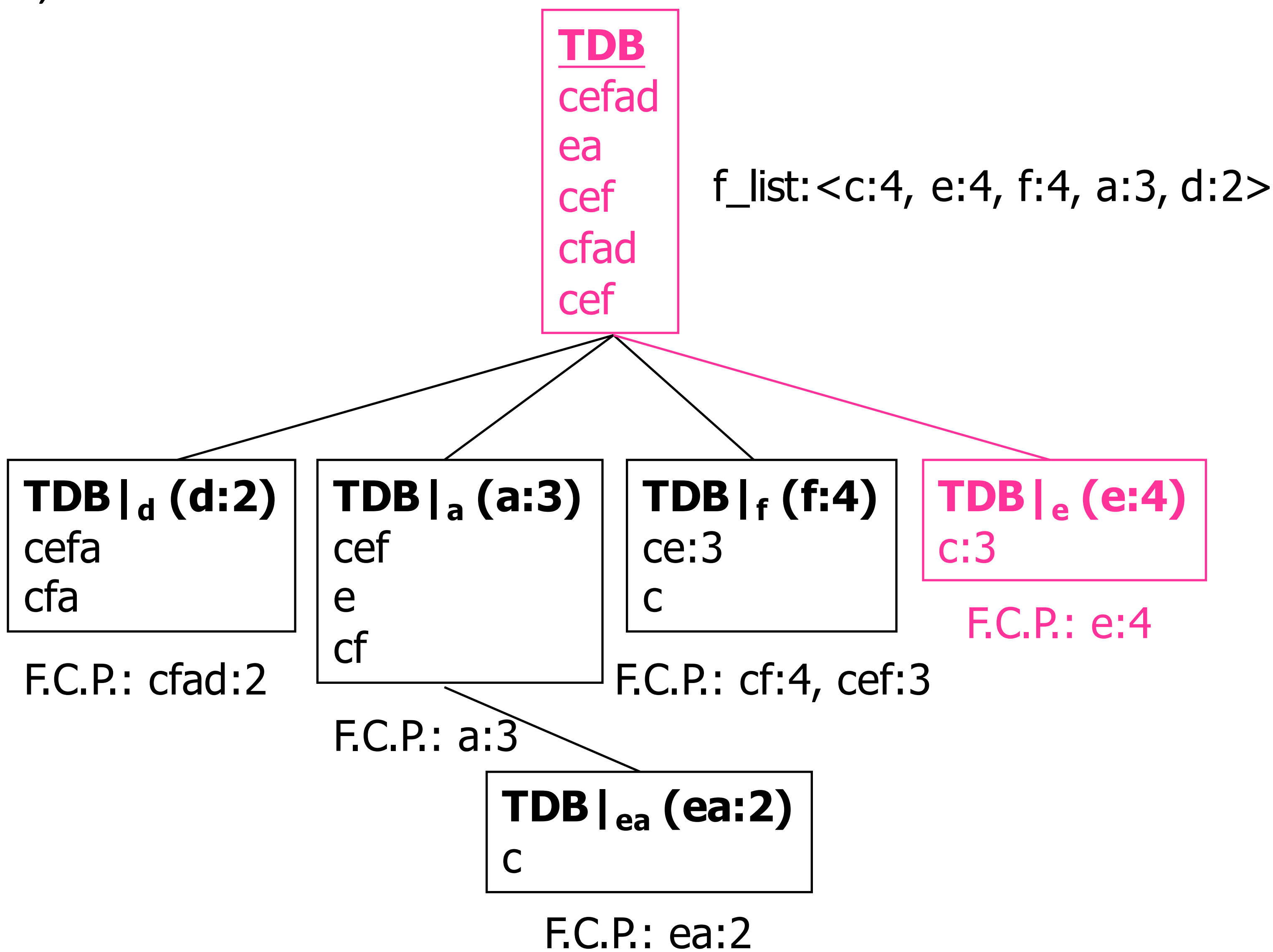
sup(fa)=sup(ca)=sup(cfad)

No FCP having fa or ca but no d

Find Frequent Closed Patterns Containing f but No a Nor d



Find Frequent Closed Patterns Containing e but No f, a Nor d

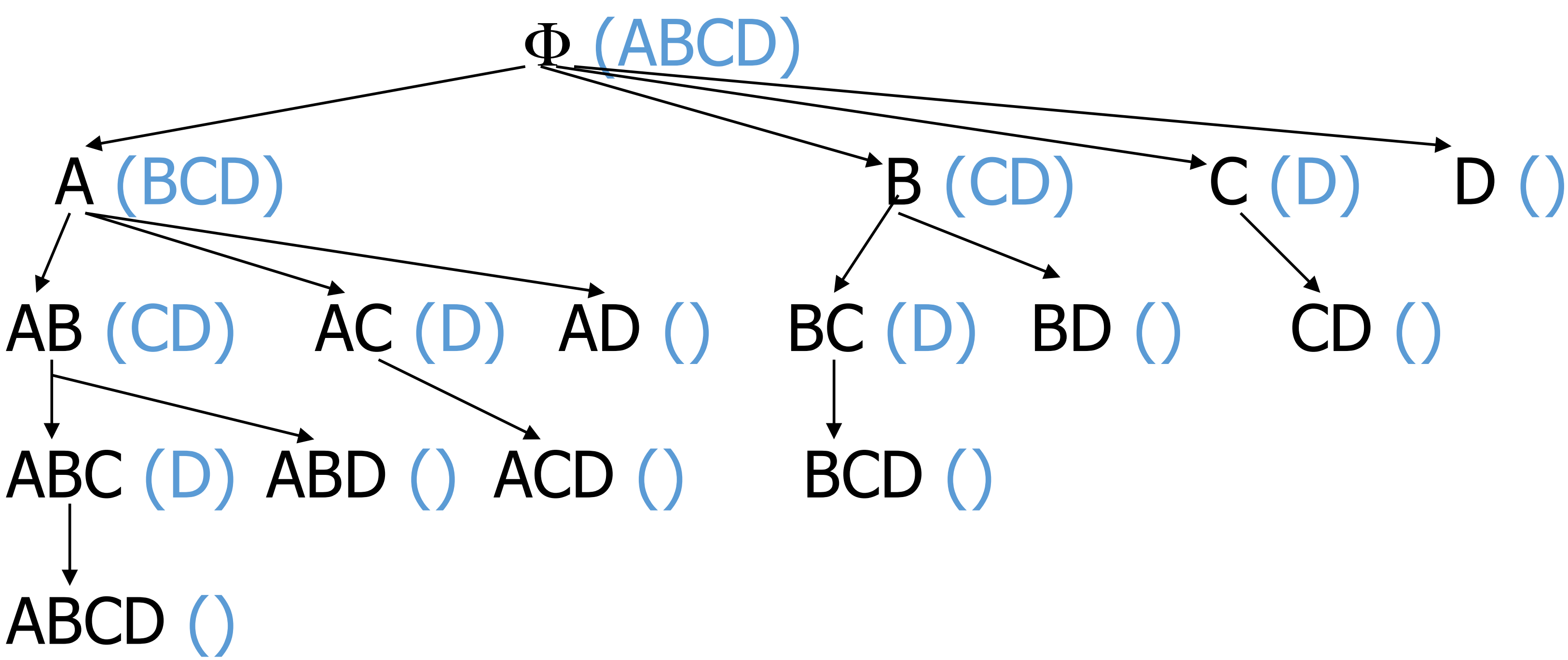


Find Frequent Closed Patterns Containing Only c

- sup(c)=sup(cf), c is not a closed itemset
- In summary, the set of frequent closed itemsets is {acdf:2, a:3, ae:2, cf:4, cef:3, e:4}

MaxMiner: Mining Max patterns

- Intuition: generate the complete set-enumeration tree one level at a time, while prune if applicable.

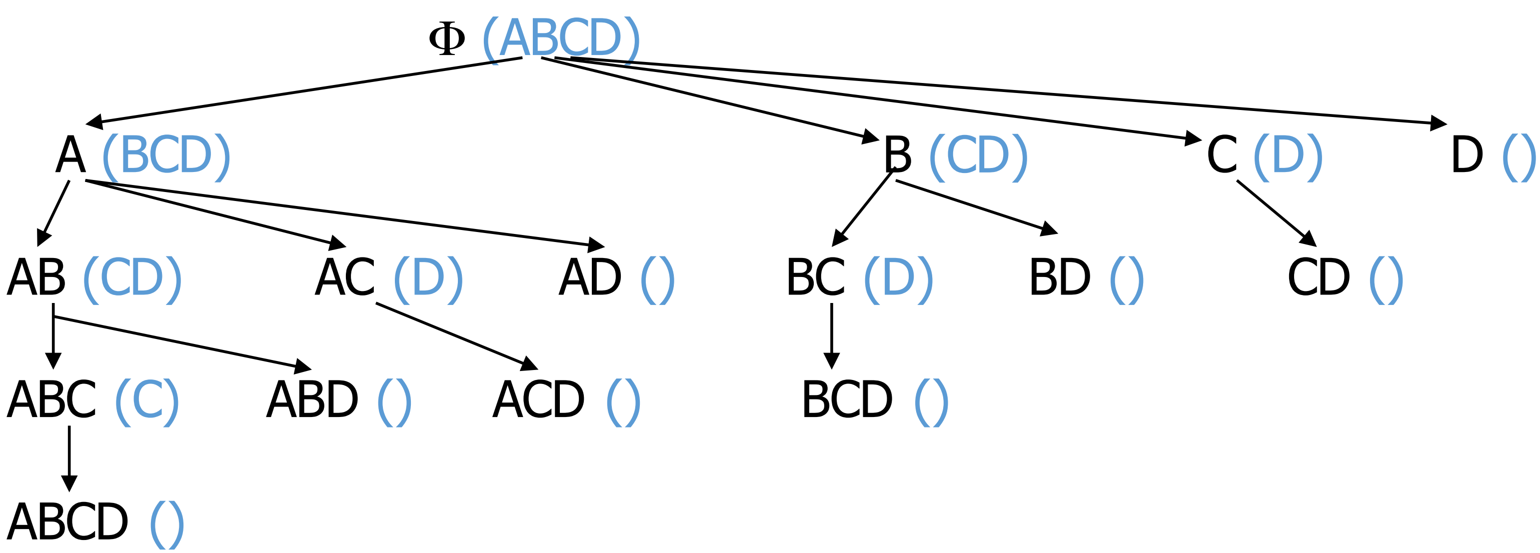


MaxMiner: Mining Max patterns

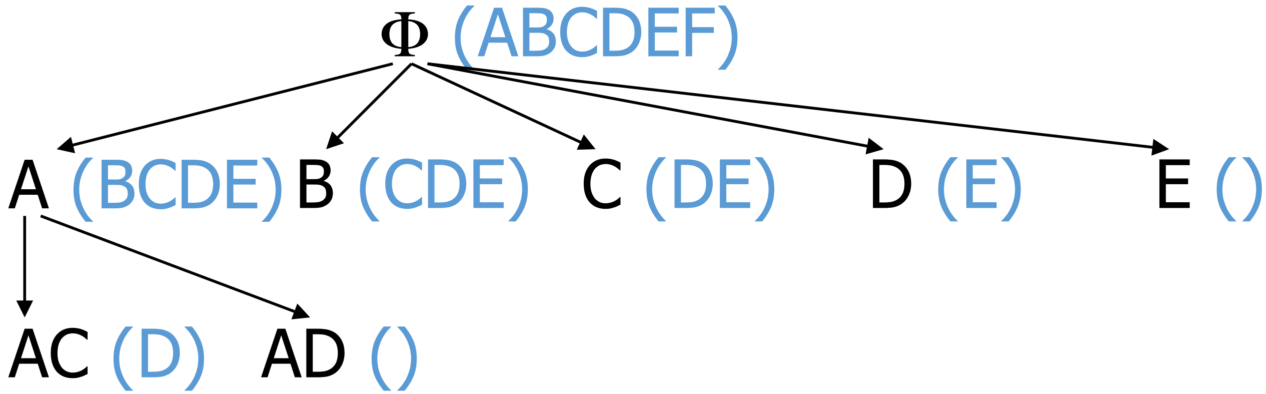
- Initially, generate one node $N=\Phi(ABCD)$, where head of N is $h(N)=\Phi$ and tail of N is $t(N)=\{A,B,C,D\}$.
- Consider expanding N,
  - If $h(N) \cup t(N)$ is frequent, do not expand N and prune the whole sub-tree.
  - Else: do local pruning. If for some $i \in t(N)$, $h(N) \cup \{i\}$ is NOT frequent, remove i from t(N) before expanding N:
- Apply global pruning: When a max pattern is identified, prune all nodes across sub-tree where $h(N) \cup t(N)$ is a sub-set of it

MaxMiner: Mining Max patterns

- Initially, generate one node N=Φ(ABCD), where head of N is h(N)=Φ and tail of N is t(N)={A,B,C,D}.
- Check the frequency of ABCD and AB, AC, AD.
  - If ABCD is frequent, prune the whole sub-tree.
  - If AC is NOT frequent, remove C from the parenthesis before expanding.
- When a max pattern is identified (e.g. ABCD), prune all nodes (e.g. A, B, C and D) where h(N)∪t(N) is a sub-set of it.



Example



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

Min_sup=2

Max patterns:

Computational Complexity of Frequent Itemset Mining

- How many itemsets are potentially to be generated in the worst case?
  - The number of frequent itemsets to be generated is senstive to the minsup threshold
  - When minsup is low, there exist potentially an exponential number of frequent itemsets
  - The worst case: M^N where M: # distinct items, and N: max length of transactions
- The worst case complexty vs. the expected probability
  - Ex. Suppose Walmart has 10^4 kinds of products
    - The chance to pick up one product 10^{-4}
    - The chance to pick up a particular set of 10 products: $\sim 10^{-40}$
    - What is the chance this particular set of 10 products to be frequent 10^3 times in 10^9 transactions?

Chapter 5: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Frequent Itemset Mining Methods
- Which Patterns Are Interesting?—Pattern Evaluation Methods
- Summary

Interestingness Measure: Correlations (Lift)

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

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

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

$$lift(B,C) = \frac{2000/5000}{3000/5000 * 3750/5000} = 0.89$$

$$lift(B,\neg C) = \frac{1000/5000}{3000/5000 * 1250/5000} = 1.33$$

Are *lift* and χ^2 Good Measures of Correlation?

- *"Buy walnuts ⇒ buy milk* [1%, 80%]" is misleading if 85% of customers buy milk
- Support and confidence are not good to indicate correlations
- Over 20 interestingness measures have been proposed (see Tan, Kumar, Sritastava @KDD'02)

symbol	measure	range	formula
ϕ	ϕ-coefficient	-1...1	$\frac{P(A,B)-P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
Q	Yule's Q	-1 ... 1	$\frac{P(A,B)P(\overline{A},\overline{B})-P(A,\overline{B})P(\overline{A},B)}{P(A,B)P(\overline{A},\overline{B})+P(A,\overline{B})P(\overline{A},B)}$
Y	Yule's Y	-1 ... 1	$\frac{\sqrt{P(A,B)P(\overline{A},\overline{B})}-\sqrt{P(A,\overline{B})P(\overline{A},B)}}{\sqrt{P(A,B)P(\overline{A},\overline{B})}+\sqrt{P(A,\overline{B})P(\overline{A},B)}}$
k	Cohen's	-1 ... 1	$\frac{P(A,B)+P(\overline{A},\overline{B})-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A)P(B)-P(\overline{A})P(\overline{B})}$
PS	Piatetsky-Shapiro's	-0.25 ... 0.25	$P(A,B)-P(A)P(B)$
F	Certainty factor	-1 ... 1	$\max(\frac{P(B\mid A)-P(B)}{1-P(B)}, \frac{P(A\mid B)-P(A)}{1-P(A)})$
AV	added value	-0.5 ... 1	$\max(P(B\mid A)-P(B), P(A\mid B)-P(A))$
K	Klosgen's Q	-0.33 ... 0.38	$\sqrt{P(A,B)}\max(P(B\mid A)-P(B), P(A\mid B)-P(A))$
g	Goodman-kruskal's	0 ... 1	$\frac{\Sigma_j \max_k P(A_j,B_k)+\Sigma_k \max_j P(A_j,B_k)-\max_j P(A_j)-\max_k P(B_k)}{2-\max_j P(A_j)-max_k P(B_k)}$
M	Mutual Information	0 ... 1	$\frac{\Sigma_i \Sigma_j P(A_i,B_j) \log \frac{P(A_i,B_j)}{P(A_i)P(B_J)}}{\min(-\Sigma_i P(A_i) \log P(A_i) \log P(A_i), -\Sigma_i P(B_i) \log P(B_i) \log P(B_i))}$
J	J-Measure	0 ... 1	$\max(P(A,B)\log(\frac{P(B\mid A)}{P(B)}) + P(A\overline{B})\log(\frac{P(\overline{B}\mid A)}{P(\overline{B})}))$
			$P(A,B)\log(\frac{P(A\mid B)}{P(A)}) + P(\overline{A}B)\log(\frac{P(\overline{A}\mid B)}{P(\overline{A})})$
G	Gini index	0 ... 1	$\max(P(A)[P(B\mid A)^2 + P(\overline{B}\mid A)^2] + P(\overline{A}[P(B\mid\overline{A})^2 + P(\overline{B}\mid\overline{A})^2] - P(B)^2 - P(\overline{B})^2,$
			$P(B)[P(A\mid B)^2 + P(\overline{A}\mid B)^2] + P(\overline{B}[P(A\mid\overline{B})^2 + P(\overline{A}\mid\overline{B})^2] - P(A)^2 - P(\overline{A})^2)$
s	support	0 ... 1	$P(A,B)$
c	confidence	0 ... 1	$max(P(B\mid A), P(A\mid B))$
L	Laplace	0 ... 1	$\max(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2})$
IS	Cosine	0 ... 1	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
γ	coherence(Jaccard)	0 ... 1	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
α	all_confidence	0 ... 1	$\frac{P(A,B)}{\max(P(A),P(B))}$
o	odds ratio	0 ... ∞	$\frac{P(A,B)P(\overline{A},\overline{B})}{P(\overline{A},B)P(A,\overline{B})}$
V	Conviction	0.5 ... ∞	$\max(\frac{P(A)P(\overline{B})}{P(A\overline{B})}, \frac{P(B)P(\overline{A})}{P(B\overline{A})})$
λ	lift	0 ... ∞	$\frac{P(A,B)}{P(A)P(B)}$
S	Collective strength	0 ... ∞	$\frac{P(A,B)+P(\overline{AB})}{P(A)P(B)+P(\overline{A})P(\overline{B})} \times \frac{1-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A,B)-P(\overline{AB})}$
χ^2	χ^2	0 ... ∞	$\Sigma_i \frac{(P(A_i)-E_i)^2}{E_i}$

Chapter 5: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Frequent Itemset Mining Methods
- Which Patterns Are Interesting?—Pattern Evaluation Methods
- Summary

Summary

- Basic concepts: association rules, support-confident framework, closed and max patterns
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Projection-based (FPgrowth, CLOSET+, ...)
- Which patterns are interesting?
  - Pattern evaluation methods