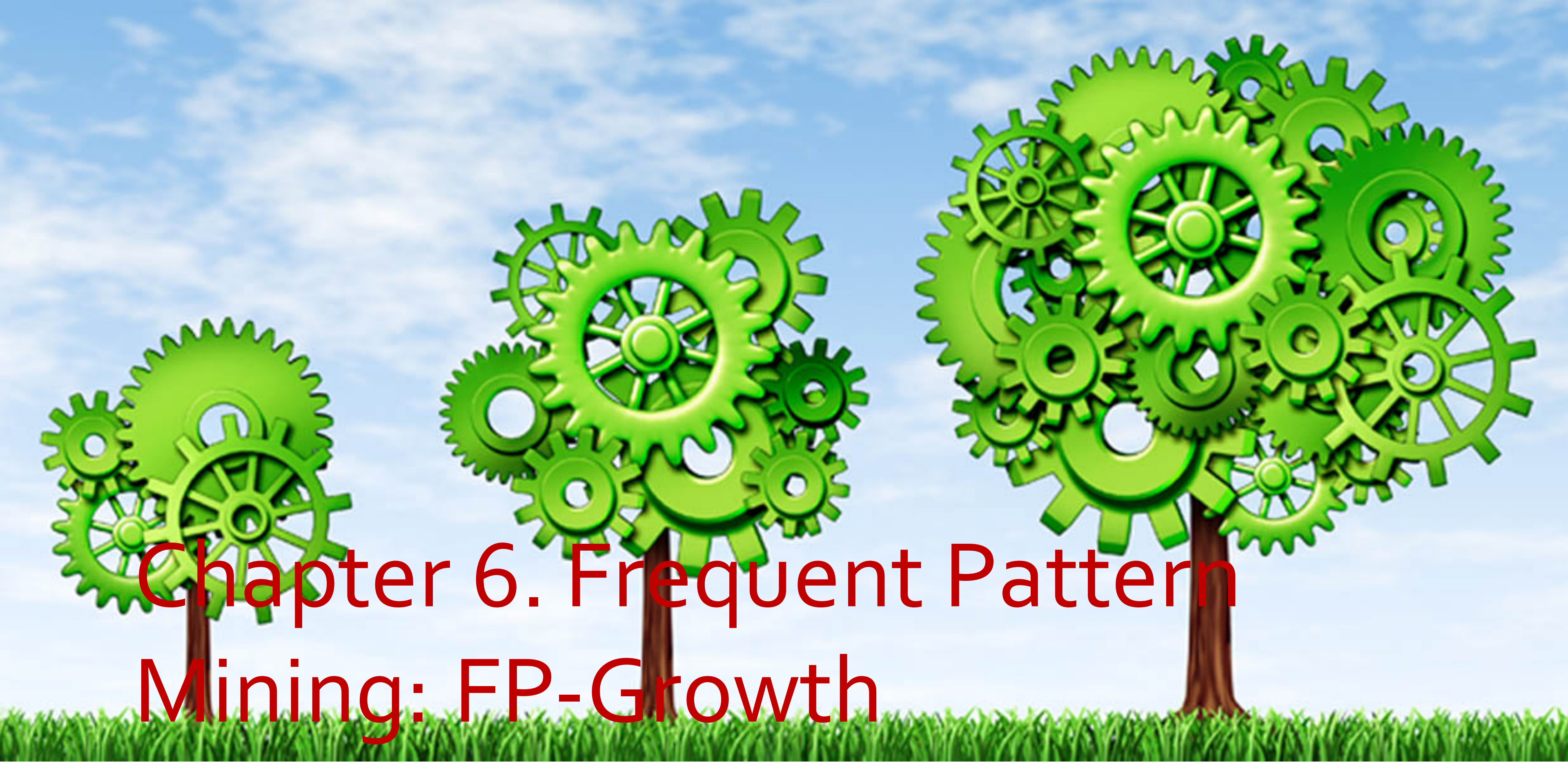


Chapter 6. Frequent Pattern Mining: FP-Growth

Meng Jiang

CSE 40647/60647 Data Science Fall 2017

Introduction to Data Mining

Frequent Pattern Mining Methods

- Apriori
- **ECLAT**
- **FP-Growth**

Exploring Vertical Data Format: ECLAT

- ECLAT (Equivalence Class Transformation): A depth-first search algorithm using set intersection [Zaki et al. @KDD'97]
- Tid-List: List of transaction-ids containing an itemset
- Vertical format: t(e) = $\{T_{10}, T_{20}, T_{30}\}$; t(a) = $\{T_{10}, T_{20}\}$; t(ae) = $\{T_{10}, T_{20}\}$
- **Deriving frequent patterns based on vertical intersections**

A transaction DB in Horizontal Data Format

Tid	Itemset
10	a, c, d, e
20	a, b, e
30	b, c, e

The transaction DB in Vertical Data Format

Item	TidList
a	10, 20
b	20, 30
c	10, 30
d	10
e	10, 20, 30

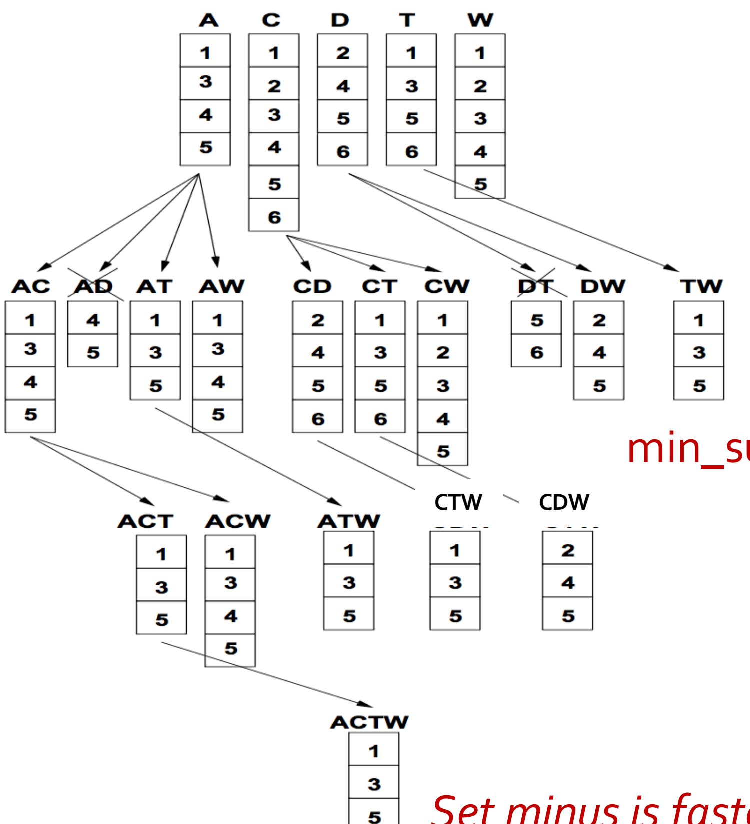
http://www.cs.rpi.edu/~zaki/PaperDir/SIGKDD03-diffsets.pdf

ECLAT: Diffset Based Mining

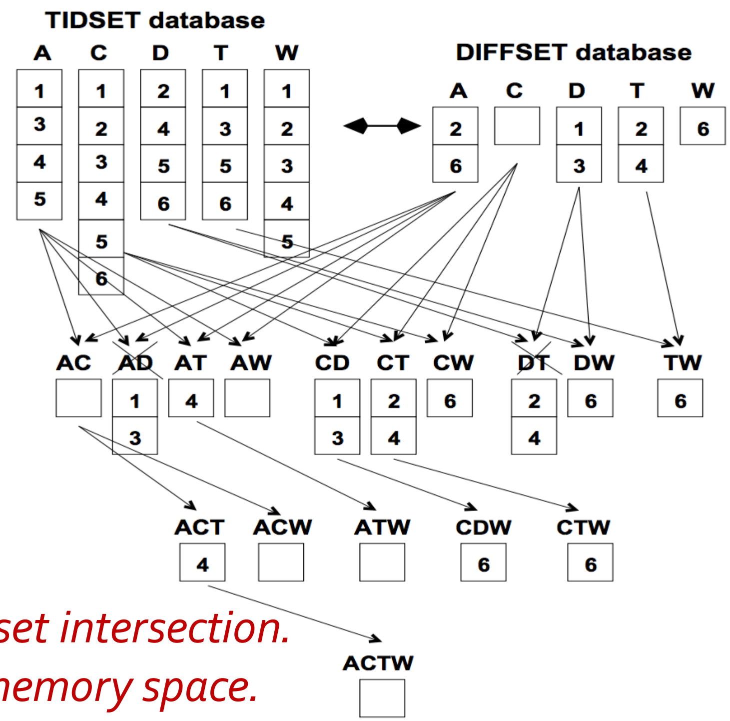
- Using diffset to accelerate mining
  - Only keep track of differences of tids

P: itemset; X, Y: items; d: diffset;
t: transaction set/list; σ: support

$$d(XY) = t(X) - t(Y)$$
$$= d(Y) - d(X)$$
$$d(PXY) = t(PX) - t(PY)$$
$$= d(PY) - d(PX)$$
$$\sigma(PX) = \sigma(P) - |d(PX)|$$



min_sup = 3



Set minus is faster than set intersection.
Set minus requires less memory space.

FPGrowth: Mining Frequent Patterns by Pattern Growth

- Idea: Frequent pattern growth (FPGrowth)
  - Find frequent single items and partition the database based on each such item
  - Recursively grow frequent patterns by doing the above for each partitioned database (also called *conditional database*)
  - To facilitate efficient processing, an efficient data structure, FP-tree, can be constructed
- Mining becomes
  - Recursively construct and mine (conditional) FP-trees
  - Until the resulting FP-tree is empty, or until it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

YouTube: https://www.youtube.com/watch?v=LXx1xKF9oDg

Example: Construct FP-tree from a Transactional DB

TID	Items in the Transaction	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*}

Answer:
f:4, a:3, c:4, b:3, m:3, p:3;
fm: 3, cm: 3, am: 3, cp:3;
fcm: 3, fam:3, cam: 3;
fcam: 3.

1. Scan DB once, find single item frequent pattern:

Let min_support = 3

f:4, a:3, c:4, b:3, m:3, p:3

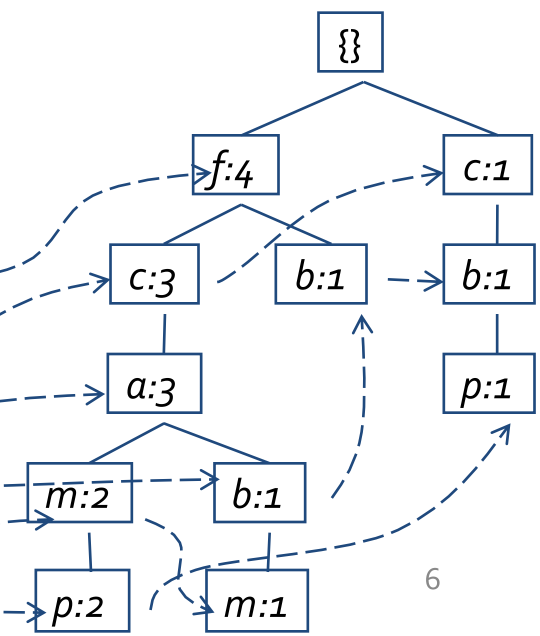
2. Sort frequent items in frequency descending order, f-list

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

3. Scan DB again, construct FP-tree

Header Table

Item	Frequency	Header
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

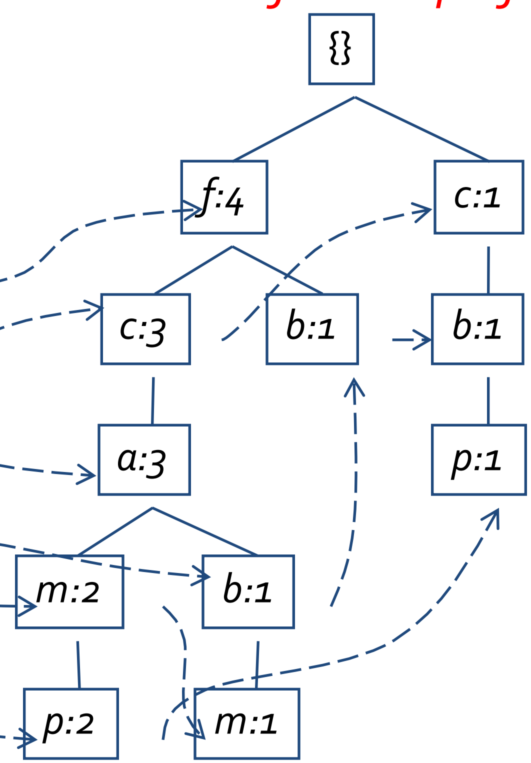


Divide and Conquer
Based on Patterns and Data

- Pattern mining can be partitioned according to current patterns
  - Patterns containing p: p's conditional database: *fcam:2*, *cb:1*
  - Patterns having m but no p: m's conditional database: *fca:2*, *fcab:1*
  - ....
- *p*'s conditional pattern base: *transformed prefix paths* of item *p*

min_support = 3

Item	Frequency	Header
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



***Conditional* pattern bases**

Item	*Conditional 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*

Mine Each Conditional Pattern-Base Recursively

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*

min_support = 3

For each conditional pattern-base

- Mine single-item patterns
- Construct its **cond. FP-tree** & mine it

p-conditional PB: ***fcam:2, cb:1 → c: 3***

m-conditional PB: ***fca:2, fcab:1 → fca: 3***

b-conditional PB: ***fca:1, f:1, c:1 → ϕ***

a-conditional PB: ***fc:3 → fc:3***

c-conditional PB: ***f:3 → f:3***

Mine Each Conditional Pattern-Base Recursively

***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***

min_support = 3

For each conditional pattern-base

– Mine single-item patterns
– Construct its **cond. FP-tree** & **mine** it

p-conditional PB: ***fcam:2, cb:1*** → ***c: 3***

m-conditional PB: ***fca:2, fcab:1*** → ***fca: 3***

b-conditional PB: ***fca:1, f:1, c:1*** → ***ϕ***

a-conditional PB: ***fc:3*** → ***fc:3***

c-conditional PB: ***f:3*** → ***f:3***

{}
|
f:3
|
c:3
|
a:3
m-cond. FP-tree

{}
|
f:3
|
c:3
am-cond. FP-tree

{}
|
f:3
cm-cond. FP-tree

{}
|
f:3
cam-cond. FP-tree

mine(<f:3, c:3, a:3>|m)

→ (am:3) + mine(<f:3, c:3>|am)

→ (cam:3) + (fam:3) + mine (<f:3>|cam)

→ (fcam:3)

→ (cm:3) + mine(<f:3>|cm)

→ (fcm:3)

→ (fm:3)

Mine Each Conditional Pattern-Base Recursively

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***

min_support = 3

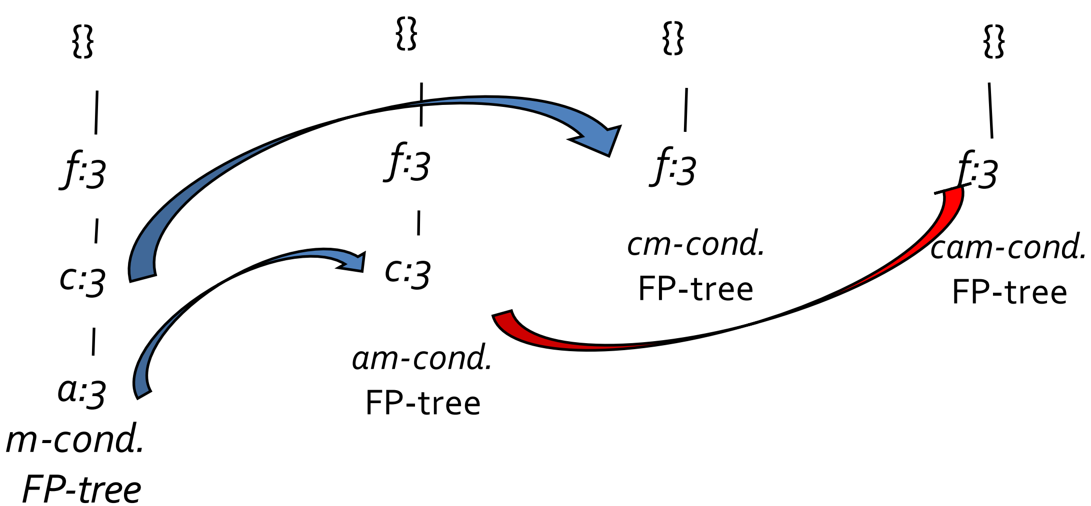
For each conditional pattern-base

- Mine single-item patterns
- Construct its cond. FP-tree & mine it

p-conditional PB: ***fcam:2, cb:1 → c: 3***

m-conditional PB: ***fca:2, fcab:1 → fca: 3***

b-conditional PB: ***fca:1, f:1, c:1 → ϕ***



Actually, for single branch FP-tree, all frequent patterns can be generated in one shot

m: 3
fm: 3, cm: 3, am: 3
fcm: 3, fam:3, cam: 3
fcam: 3

Mine Each Conditional Pattern-Base Recursively

***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*

min_support = 3

For each conditional pattern-base

- Mine single-item patterns
- Construct its **cond. FP-tree** & **mine** it

p-conditional PB: ***fcam:2, cb:1 → c: 3***

m-conditional PB: ***fca:2, fcab:1 → fca: 3***

b-conditional PB: ***fca:1, f:1, c:1 → ϕ***

a-conditional PB: ***fc:3 → fc:3***

c-conditional PB: ***f:3 → f:3***

Mine Each Conditional Pattern-Base Recursively

***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*

min_support = 3

For each conditional pattern-base

- Mine single-item patterns
- Construct its **cond. FP-tree** & **mine** it

p-conditional PB: ***fcam:2, cb:1*** → ***c: 3***

m-conditional PB: ***fca:2, fcab:1*** → ***fca: 3***

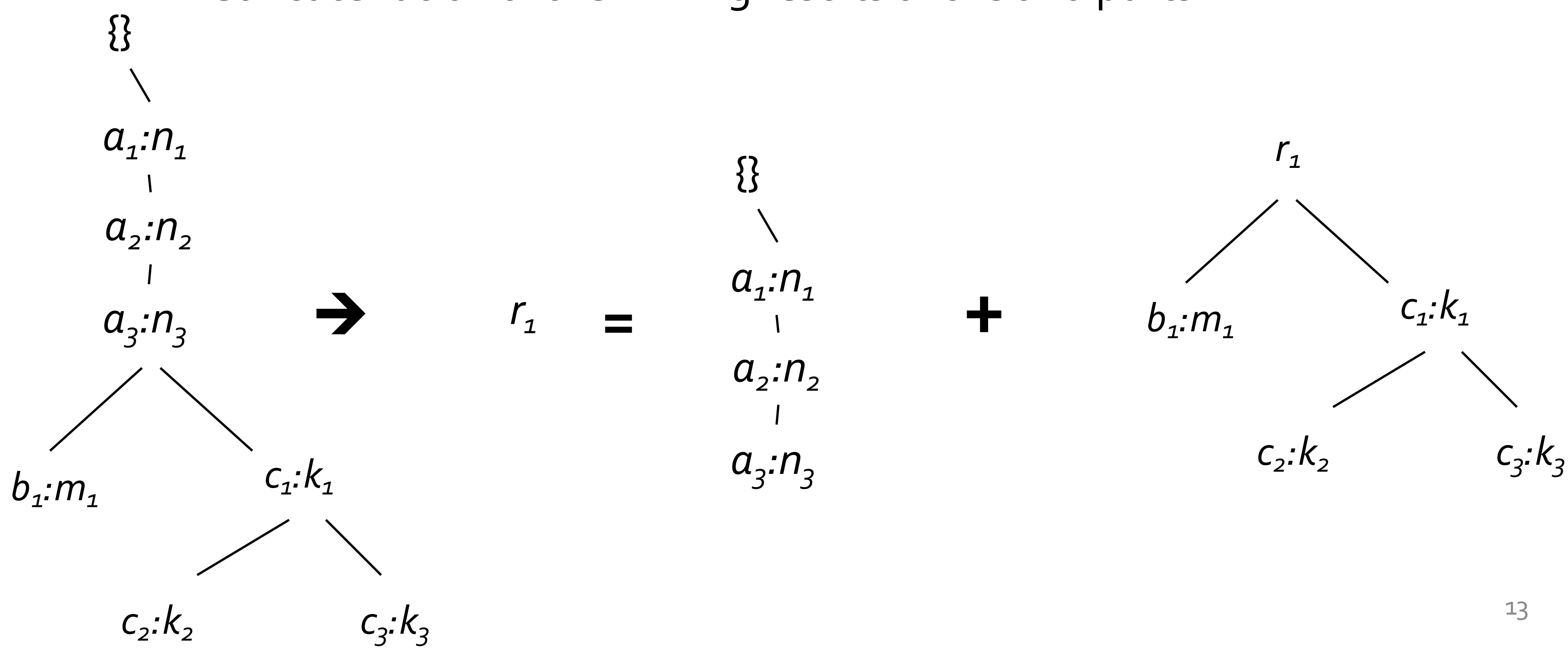
b-conditional PB: ***fca:1, f:1, c:1*** → ***ϕ***

a-conditional PB: ***fc:3*** → ***fc:3***

c-conditional PB: ***f:3*** → ***f:3***

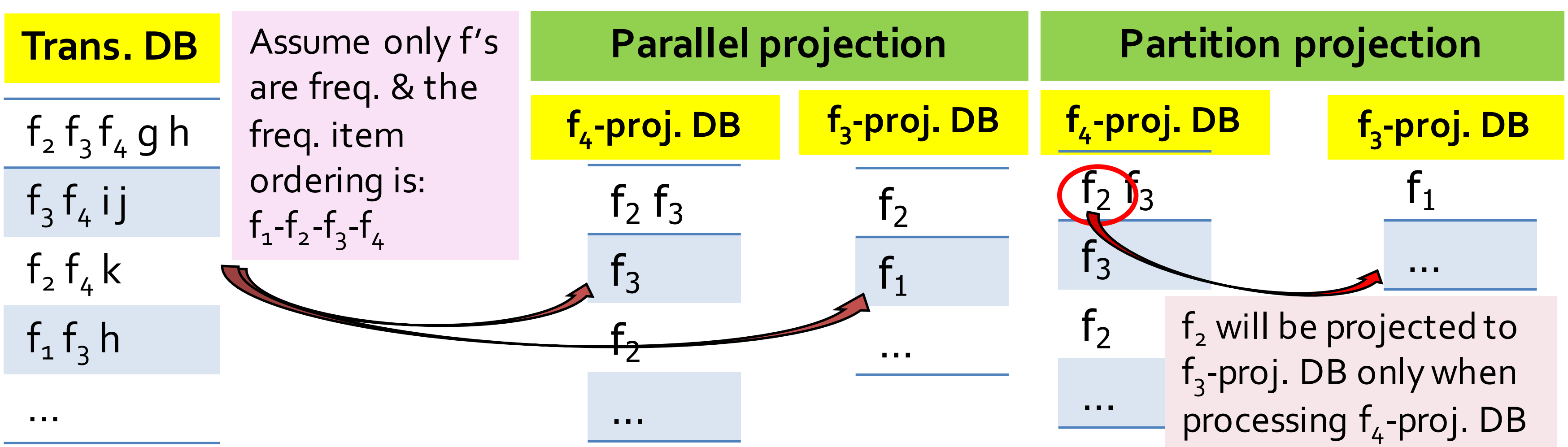
A Special Case: Single Prefix Path in FP-tree

- 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



Scaling FP-growth by Database Projection

- What if FP-tree cannot fit in memory? — DB projection
  - Project the DB based on patterns
  - Construct & mine FP-tree for each projected DB
- Parallel projection vs. partition projection
  - Parallel projection: Project the DB on each frequent item
    - Space costly, all partitions can be processed in parallel
  - Partition projection: Partition the DB in order
    - Passing the unprocessed parts to subsequent partitions



Discussion

- Compare Apriori, ECLAT, and FP-Growth.
  - Strong points of each
  - Weak points of each

References

- R. Agrawal, T. Imielinski, and A. Swami, “Mining association rules between sets of items in large databases”, in Proc. of SIGMOD'93
- R. J. Bayardo, “Efficiently mining long patterns from databases”, in Proc. of SIGMOD'98
- N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, “Discovering frequent closed itemsets for association rules”, in Proc. of ICDT'99
- J. Han, H. Cheng, D. Xin, and X. Yan, “Frequent Pattern Mining: Current Status and Future Directions”, Data Mining and Knowledge Discovery, 15(1): 55-86, 2007
- R. Agrawal and R. Srikant, “Fast algorithms for mining association rules”, VLDB'94
- A. Savasere, E. Omiecinski, and S. Navathe, “An efficient algorithm for mining association rules in large databases”, VLDB'95
- J. S. Park, M. S. Chen, and P. S. Yu, “An effective hash-based algorithm for mining association rules”, SIGMOD'95
- S. Sarawagi, S. Thomas, and R. Agrawal, “Integrating association rule mining with relational database systems: Alternatives and implications”, SIGMOD'98
- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, “Parallel algorithm for discovery of association rules”, Data Mining and Knowledge Discovery, 1997
- J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation”, SIGMOD’00

References (cont.)

- M. J. Zaki and Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining", SDM'02
- J. Wang, J. Han, and J. Pei, "CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets", KDD'03
- C. C. Aggarwal, M.A., Bhuiyan, M. A. Hasan, "Frequent Pattern Mining Algorithms: A Survey", in Aggarwal and Han (eds.): Frequent Pattern Mining, Springer, 2014
- C. C. Aggarwal and P. S. Yu. A New Framework for Itemset Generation. PODS'98
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97
- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94
- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE'03
- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02
- T. Wu, Y. Chen and J. Han, Re-Examination of Interestingness Measures in Pattern Mining: A Unified Framework, Data Mining and Knowledge Discovery, 21(3):371-397, 2010