CSMDM21 - Data Analytics and Mining

Association Rule Mining

Module convenor
Dr. Carmen Lam
carmen.lam@reading.ac.uk
Department of Computer Science

Lecture notes and videos created by
Prof. Giuseppe Di Fatta

Overview

- Association Rule Mining (ARM)
- Apriori algorithm
- Database layout
- Relevant subsets of frequent itemsets
  - Maximal and Closed Frequent Itemsets
- Taxonomy of ARM algorithms
- Extended Association Rules
- Other ARM algorithms
- References

What is Association Rule Mining (ARM)?

- ARM was introduced in 1993:

  R.**A**grawal, T. **I**mielinski, A **S**wami, “Mining Association Rules between Sets of Items in Large Databases”, Proc. of ACM-SIGMOD93 Conference.

- Given a set of attributes (*items*) I, and a set T of objects (*transactions*) containing a subset of the items,

List of Items
I = {Bread, Milk, Diaper, Beer, Eggs, Cok

|I| = 6

|T| = 5

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

- Find rules that express the occurrence of items based on the occurrences of other items in the transactions.

Examples:

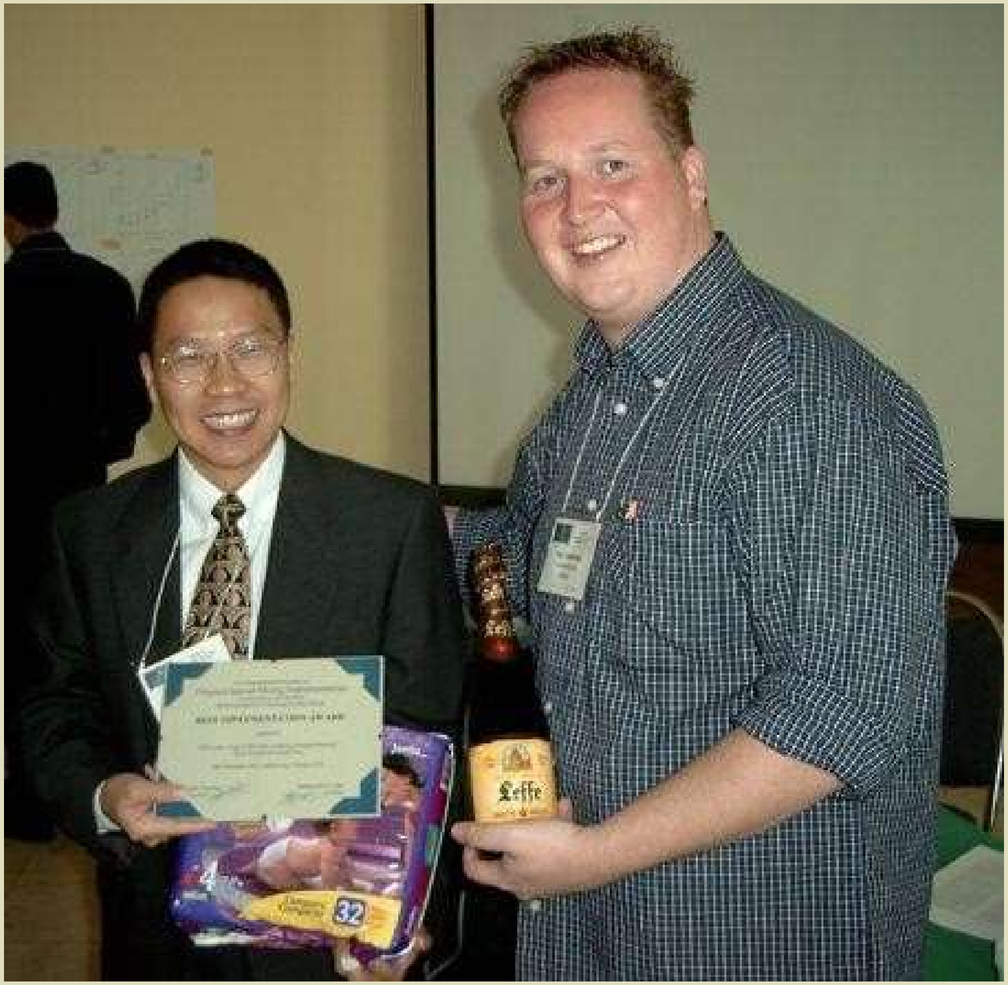
{Diaper} ⟹ {Beer}

{Milk, Bread} ⟹ {Eggs, Coke}

{Beer, Bread} ⟹ {Milk}

NB: Implication “⟹” means co-occurrence, not causality!

Diapers and Beer



Gösta Grahne and Jianfei Zhu’s paper "**Efficiently Using Prefix-trees in Mining Frequent Itemsets**“ received the best implementation award at FIMI03, the Workshop on Frequent Itemset Mining Implementations 2003.

Applications

- ✓ Marketing and Sales Promotion
  - – What other products should the store stocks up?
  - – Shelf management
  - – Customer behavior analysis
- ✓ Prediction of the failure of telecommunication networks
  - – What set of events occur prior to failure of a switching node?
- ✓ Medical applications
  - – In Singapore, about 10 percent of the population is diabetic. In 1992 a regular screening program for the diabetic patients was introduced. Patient information, clinical symptoms, eye-disease diagnosis and treatments, etc., were captured in a database.
  - – As a result of data mining, the doctors gained a much better understanding of how diabetes progresses over time and how different treatments affect its progress.
- ✓ Health Insurance Commission (HIC)
  - – Associations on episode database for pathology services
    - 6.8 million records for 120 attributes (3.5GB)
    - 15 months preprocessing then 2 weeks data mining
  - – Goal: find associations between tests
    - minConf = 50% and minSupp = 1%, 0.5%, 0.25% (1% of 6.8 million = 68,000)
    - Unexpected/unnecessary combination of services
    - Refusing cover would save $550,000 per year
- ✓ Tools for the drug discovery process

Definition of ARM

Association Rule Mining (ARM)

- Let I be a set of items, $I = \{A,B,C,\ldots\}$ with $|I| = d$. A set $X = \{i_1,\ldots,i_k\} \subseteq I$ is called an itemset.
- A transaction over I is a pair t = <tid, S>, where tid is the transaction identifier and S is an itemset. T is the set of transactions ($|T| = n$).
- A transaction t = <tid, S> is said to support an itemset $X \subseteq I$, if $X \subseteq S$.
  - Ex.: the transaction t=<09CF44, {bier, diapers, milk}> does support the itemset X={bier, diapers} and does not support the itemset Y={bier, coke}.
- X and Y are proper sets of items, $X \subset I$ and $Y \subset I$, and $X \cap Y = \varnothing$
- ARM problem: find all frequent and strong rules:

$X \Rightarrow Y$, "if X then Y"

(body $\Rightarrow$ head)
(antecedent $\Rightarrow$ consequent)

with sufficient support and confidence

Support of the rule = $\text{Prob}(X \cup Y)$ ❼ joint probability

Confidence of the rule = $\text{Prob}(Y \mid X)$ ❼ conditional probability

Definition of Frequent Itemset

- Itemset
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items

- Support of an itemset
  - The support of an itemset c, $c \subseteq I$, is the frequency (either absolute or relative value) of its occurrence in the transactions, i.e. the fraction of transactions that contain the itemset.

support count

$$s(c) = Support(c) = \frac{\left|\left\{t \mid t \in T, c \subseteq itemset(t)\right\}\right|}{|T|} = \frac{\sigma(c)}{|T|}$$

- Frequent Itemset (FI)
  - An itemset whose support is greater than or equal to a ***minSup*** threshold.

$$s(c) \geq minSup$$

Interesting Association Rules

- ❑ ARM problem: find all frequent and strong rules:

  $X \Rightarrow Y$, “if X then Y”

  with sufficient support and confidence

- ❑ Given an association rule $X \Rightarrow Y$, where $X, Y \subseteq I$,

  – Def.: **support of a rule** is given by the fraction of the transactions that contain both X and Y.

  $$\text{Support}(X \Rightarrow Y) = \text{Support}(X \cup Y)$$

  – Def.: **confidence of a rule** is a measures how often items in Y appear in transactions that contain X

  $$\text{Confidence}(X \Rightarrow Y) = \text{Support}(X \cup Y) / \text{Support}(X)$$

Given two thresholds minSup and minConf, interesting rules are those such that:

- ➢ $\text{Support}(X \Rightarrow Y) \geq$ ***minSup***
- ➢ $\text{Confidence}(X \Rightarrow Y) \geq$ ***minConf***

Example

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

$$\{\text{Milk}, \text{Diaper}$$

$$s = \frac{o\,(\text{Milk}, \text{Diaper}, \text{Beer})}{|\,\text{T}\,|} = \frac{2}{5} = 0.4$$

$$c = \frac{o\,(\text{Milk}, \text{Diaper}, \text{Beer})}{o\,(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find ALL rules having
  - support ≥ *minSup* threshold
  - confidence ≥ *minConf* threshold

- **Brute-force approach**:
  1. List all possible association rules
  2. Compute the support and confidence for each rule
  3. Prune rules that fail the *minSup* and *minConf* thresholds

⇒ Computationally prohibitive!

Total Number of Rules

- Given a set I with d unique items, |I|=d:
  - the total number of itemsets is the cardinality of the power set, |Power(I)| = 2^d
  - the total number R of possible association rules is:

$$R = \sum_{k=1}^{d-1} \left[\binom{d}{k} \times \sum_{j=1}^{d-k} \binom{d-k}{j} \right]$$

Total Number of Rules

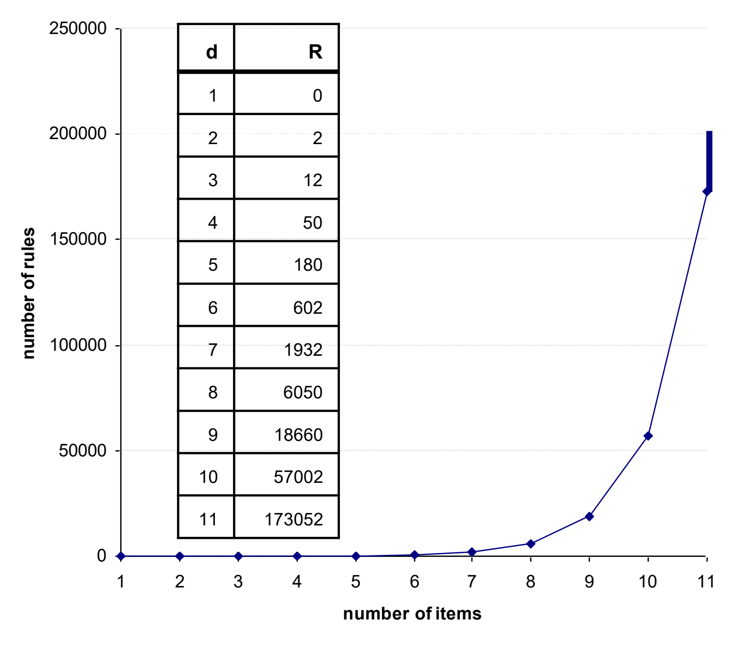
- Given a set I with d unique items, |I|=d:
  - the total number of itemsets is the cardinality of the power set, |Power(I)| = 2^d
  - the total number R of possible association rules is:

Proof derived from the binomial theorem:

$$\sum_{k=0}^{N} \binom{N}{k} \times r^{k} = (1+r)^{N}$$

Total Number of Rules

- Given a set I with d unique items, |I|=d:
  - the total number of itemsets is the cardinality of the power set, |Power(I)| = 2^d
  - the total number R of possible association rules is:



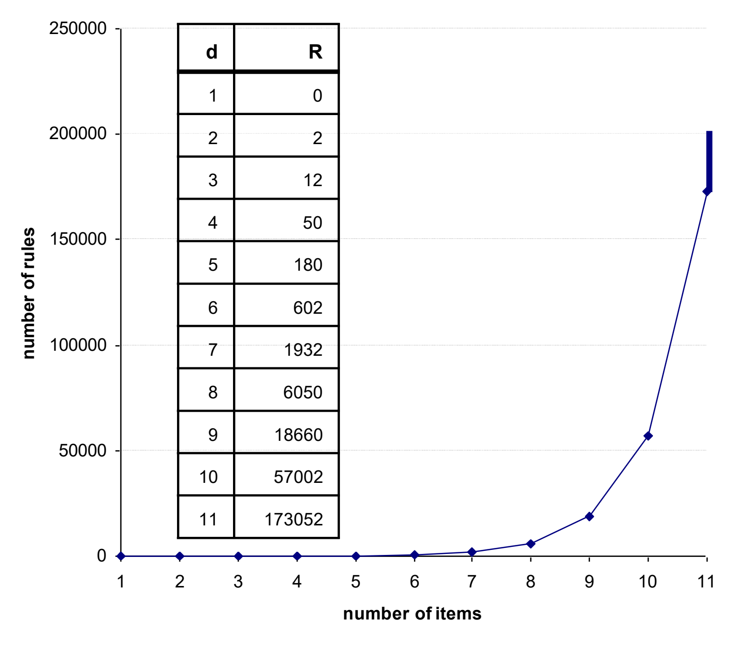
d	R
1	0
2	2
3	12
4	50
5	180
6	602
7	1932
8	6050
9	18660
10	57002
11	173052

Example for I={A,B,C}, d=3:

1. A → B
2. A → C
3. A → BC
4. B → A
5. B → C
6. B → AC
7. C → A
8. C → B
9. C → AB
10. AB → C
11. AC → B
12. BC → A

Total Number of Rules

- Given a set I with d unique items, |I|=d:
  - the total number of itemsets is the cardinality of the power set, |Power(I)| = 2^d
  - the total number R of possible association rules is:



d	R
1	0
2	2
3	12
4	50
5	180
6	602
7	1932
8	6050
9	18660
10	57002
11	173052

Example for I={A,B,C}, d=3:

1.	A❼	B	7.	A❼	BC
2.	A❼	C	8.	B❼	AC
3.	B❼	A	9.	C❼	AB
4.	B❼	C	10.	AB❼	C
5.	C❼	A	11.	AC❼	B
6.	C❼	B	12.	BC❼	A

Support and Confidence

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

Example of Rules:

$\{\text{Milk,Diaper}\} \Rightarrow \{\text{Beer}\}$ (s=0.4, c=0.67)
$\{\text{Milk,Beer}\} \Rightarrow \{\text{Diaper}\}$ (s=0.4, c=1.0)
$\{\text{Diaper,Beer}\} \Rightarrow \{\text{Milk}\}$ (s=0.4, c=0.67)
$\{\text{Beer}\} \Rightarrow \{\text{Milk,Diaper}\}$ (s=0.4, c=0.67)
$\{\text{Diaper}\} \Rightarrow \{\text{Milk,Beer}\}$ (s=0.4, c=0.5)
$\{\text{Milk}\} \Rightarrow \{\text{Diaper,Beer}\}$ (s=0.4, c=0.5)

- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}

- Rules originating from the same itemset have **identical support** but can have **different confidence**.

- Thus, we may **decouple the support and confidence** requirements

Two-Phase Process

Association Rule Mining (ARM)

Discover all association rules <X $\Rightarrow$ Y>

with a minSupp and a minConf

(user parameters)

Two phases:

1. find all Frequent Itemsets (FI)

   ← Most of the complexity

   - all possible itemsets with a minimum support

     $FI = \{c, c \subseteq I | s(c) \geq minSupp\}$

2. build strong association rules
   - Use the frequent itemsets FI to build rules with a minimum confidence:

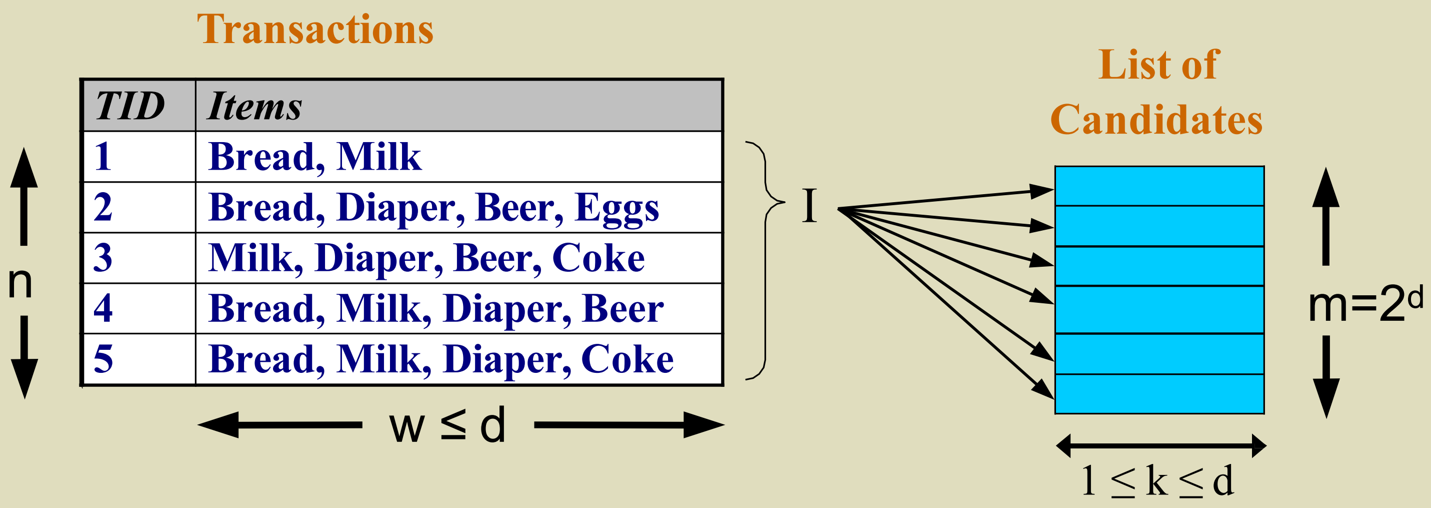
     $Rules = \{X \Rightarrow Y \mid X,Y \subset I, X \cap Y = \emptyset, X \cup Y \in FI, c(X \Rightarrow Y) \geq minConf\}$

Frequent Itemsets Generation

- In order to find all frequent itemsets we have to match each transaction (|T| = n) against every candidate (m=|Power(I)|=2^d, where |I|=d): O(n·2^d).

  Def.: a **candidate itemset** is an itemset that has been generated and is potentially frequent. We still need to count its support.

- Each match requires a subset test, at most (w·k) or max(w,k) comparisons: O(d).



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

Complexity ~ O(n · d · 2^d) => Still exponential!

Frequent Itemsets

How can we efficiently generate FI?

- Reducing the **number of candidates** (m)
  - Complete search: $m=2^d$
  - Use pruning techniques to reduce m
- Reducing the **number of transactions** (n)
  - Reduce size of T as the size of itemsets increases
  - Use a subsample of the transactions
- Reducing the **number of comparisons** ($w \cdot k$)
  - Use efficient data structures to store the candidates or the transactions
  - No need to match every candidate against every transaction

Lattice of Subsets of I={1,2,3,4}

$$\binom{d}{k} = \frac{d!}{k!\cdot(d-k)!}$$

k=0
❼
1

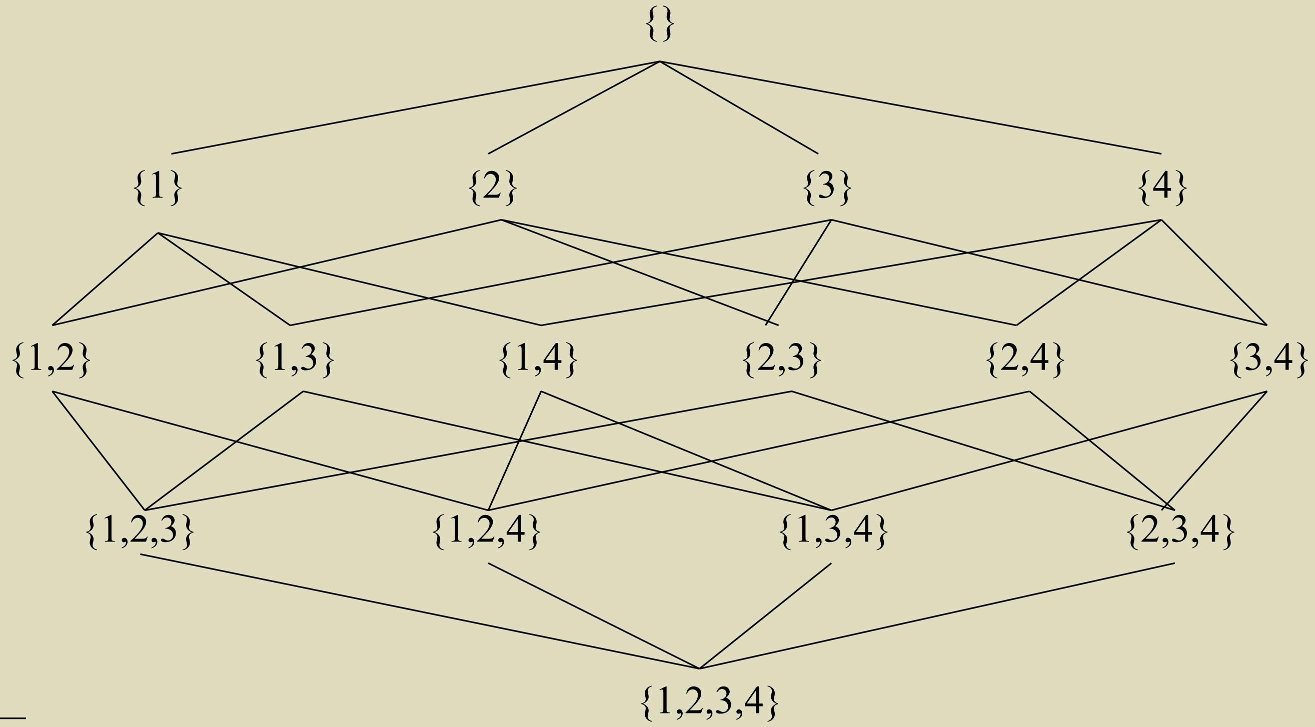
k=1
❼
4

k=2
❼
6

k=3
❼
4

k=4
❼

$$\sum_{k=0}^{d}\binom{d}{k} = 2^{d} = 16$$



{}

{1} {2} {3} {4}

{1,2} {1,3} {1,4} {2,3} {2,4} {3,4}

{1,2,3} {1,2,4} {1,3,4} {2,3,4}

{1,2,3,4}

Each itemset of the lattice is a candidate frequent itemset.

…reducing the number of candidates

Apriori algorithm was introduced in 1994:

"Fast algorithms for mining association rules", R. Agrawal and R. Srikant, in VLDB'94, the Twentieth Very Large Data Base Conference, 1994.

❑ The Apriori principle:

- if an itemset is frequent, then all of its subsets must also be frequent

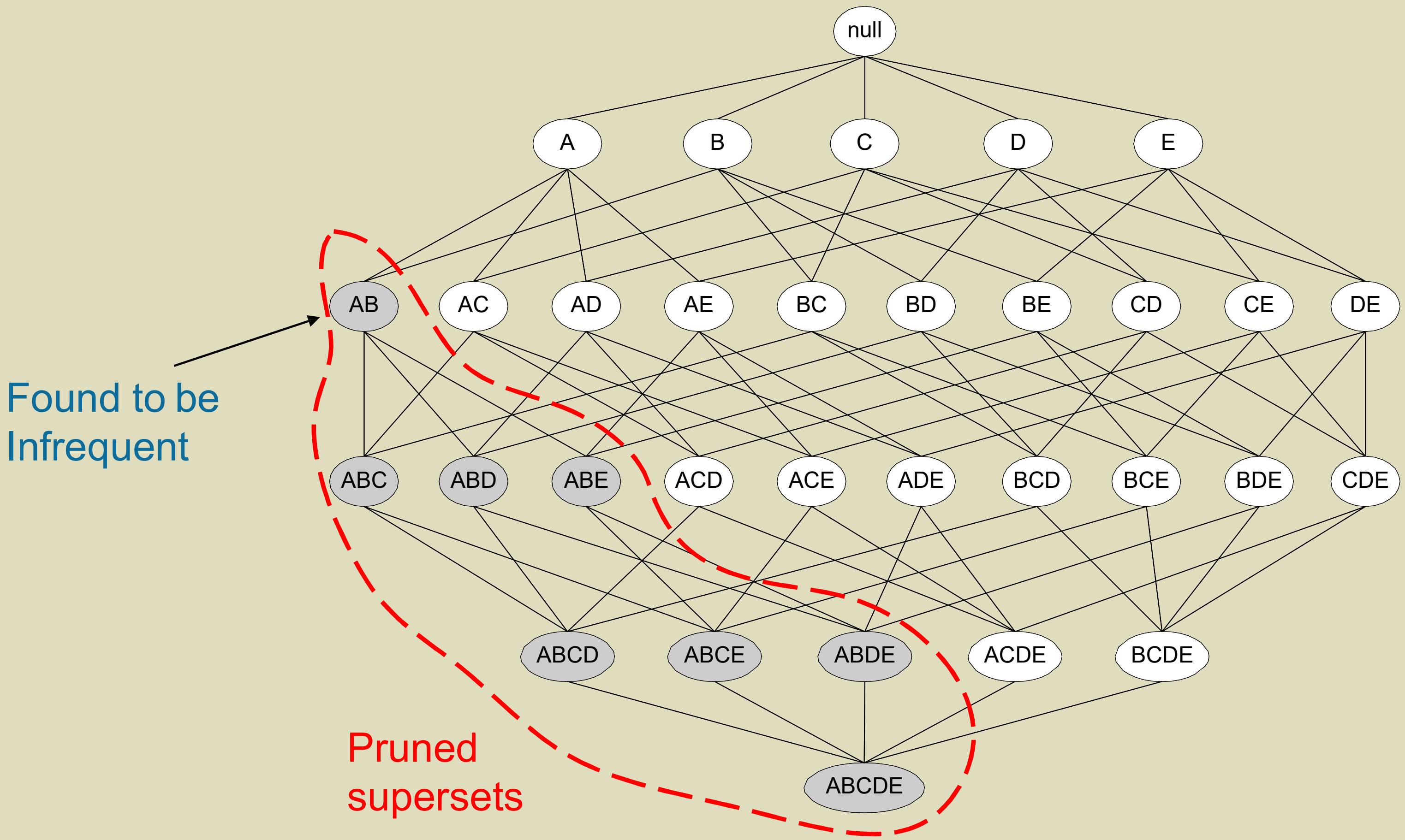
– The Apriori principle holds due to the following property of the support measure:

$$\boxed{\forall X,Y:(X\subseteq Y)\rightarrow \qquad\qquad\qquad\qquad}$$

– The support of an itemset never exceeds the support of any of its subsets

– This is known as the anti-monotone property of the support.

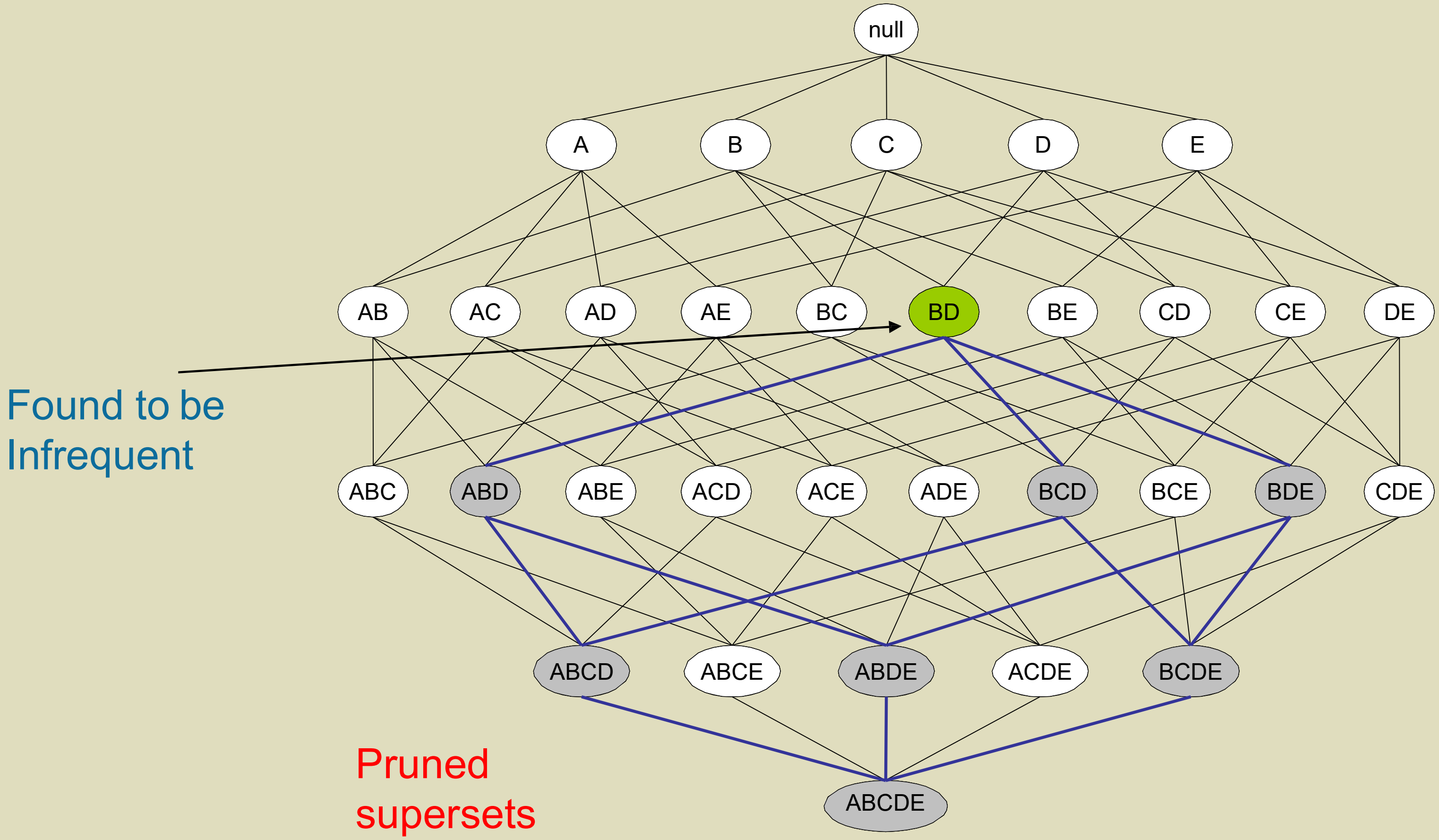
Anti-monotone Property



Found to be Infrequent

Pruned supersets

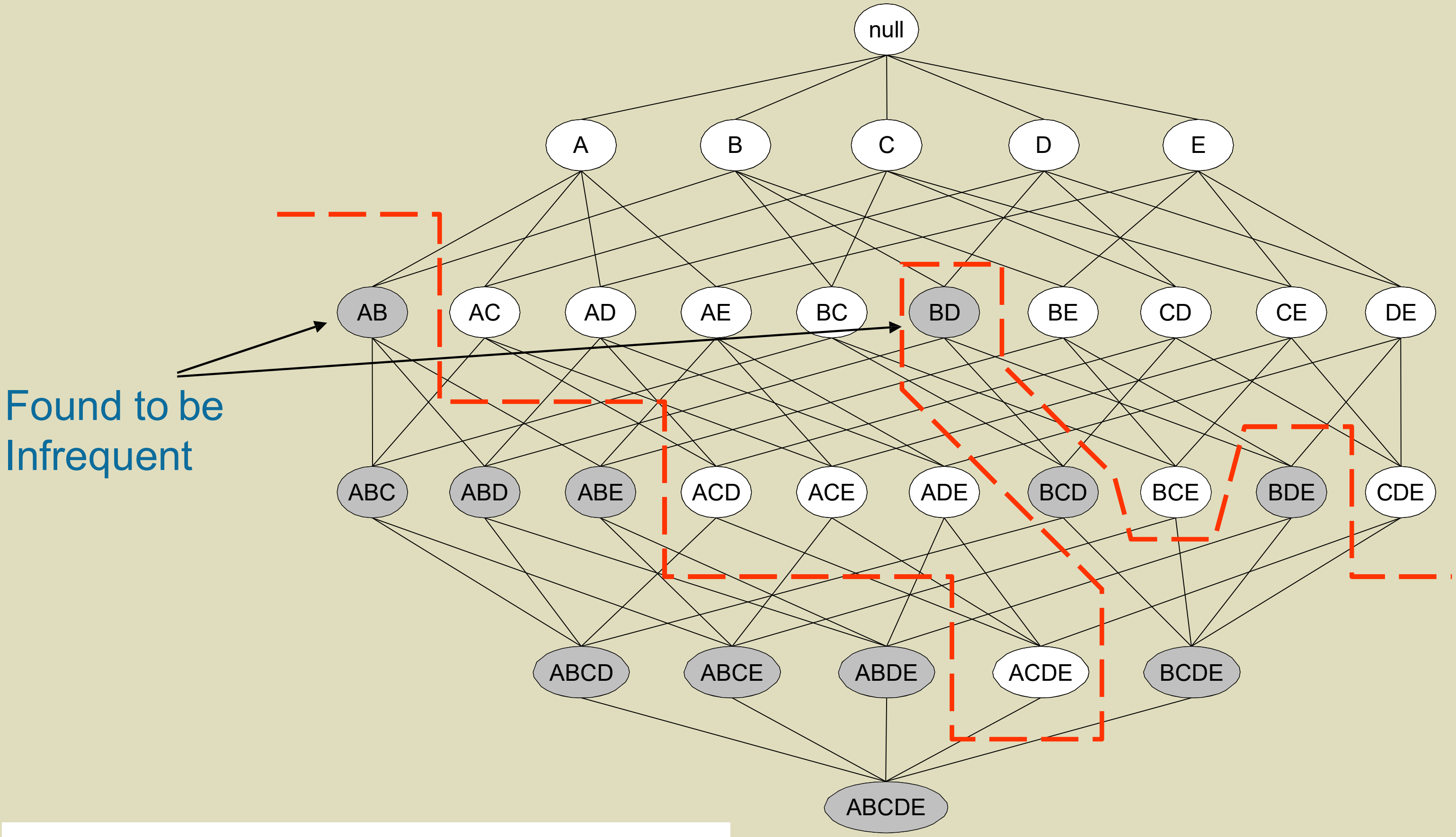
Anti-monotone Property



Found to be Infrequent

Pruned supersets

Border of Frequent Itemsets



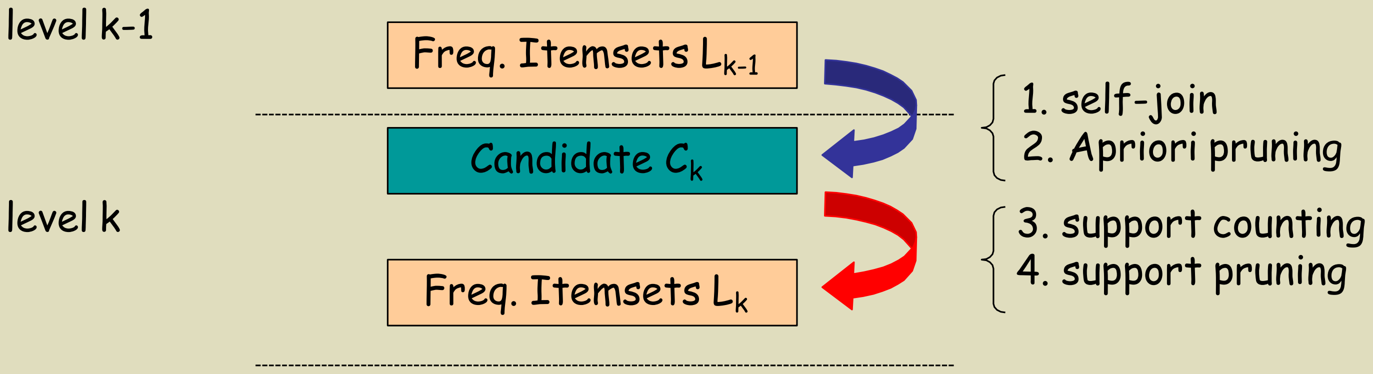
Border of frequent/infrequent itemsets

Apriori Algorithm

- Bread First Search (BFS) of all possible candidates:
  1. At level k, generate all candidate k-itemsets C_k (of length k), using frequent itemsets L_{k-1} generated at the previous level: self-join.
  2. Prune the search tree using the anti-monotone property of the support:

     $x2 \supseteq x1$ ❼ $support(x2) \leq support(x1)$
  3. Count the support of the candidate C_k: one DB scan.
  4. Prune infrequent candidates and generate L_k.

- Efficient computation of the support
  – Candidate itemsets are stored in a hash-tree



level k-1

Freq. Itemsets L_{k-1}

Candidate C_k

level k

Freq. Itemsets L_k

1. self-join
2. Apriori pruning

3. support counting
4. support pruning

Apriori Algorithm

```
$L_1$ = {frequent 1-itemsets};
for ($k$=2; $L_{k-1} \neq \emptyset$; k++) do begin
     $C_k$ = ***apriori-gen***($L_{k-1}$); //New candidates
    for all trans $t \in T$ do begin
        $C_t$ = ***subset***($C_k$, t); //Candidates contained in t
        for all candidates $c \in C_t$ do
            c.count++;
        end
        $L_k = \{c \in C_k \mid c.count \geq minSup\}$
end
return FI = $\cup_k L_k$
```

Apriori-gen Method

Let’s assume that the items in L_{k-1} are listed in an order.

```
$C_k$ = apriori-gen($L_{k-1}$):
    For all itemsets X,Y in $L_{k-1}$ , X[i]=Y[i] for $1 \leq i \leq k-2$, and X[k-1] < Y[k-1]
        I = X U {Y[k-1]}
        for each $J \subset I$, |J|=k-1
          if $J \epsilon L_{k-1}$ then $C_k = C_k \cup I$
```

Candidate Generation: Example

Generation step:

1. {1,2} U {1,3} = {1,2,3}
2. {1,2} U {1,4} = {1,2,4}
3. {1,3} U {1,4} = {1,3,4}

- ✓ {2,3} is not frequent ❼ skipped
- ✓ {2,4} has no frequent partner
- ✓ {3,4} is not frequent ❼ skipped

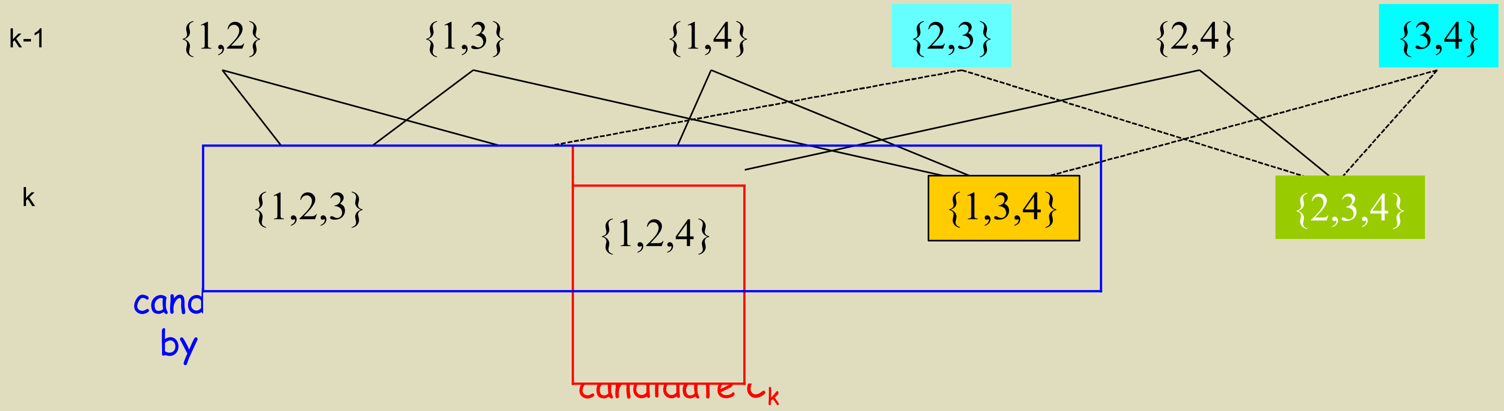
Pruning step:

1. {1,2,3} pruned because its subset {2,3} is infrequent
2. {1,3,4} pruned because its subset {3,4} is infrequent

not frequent

pruned

not generated



…reducing the number of comparisons

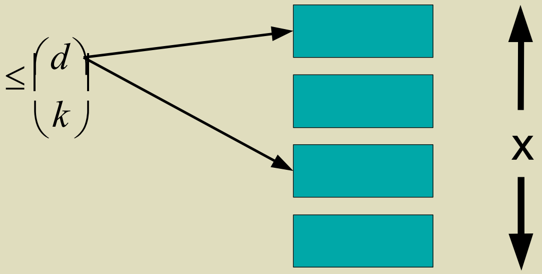
- **Apriori candidate counting**:
  - We need to scan the database of transactions to determine the support of each candidate k-itemset
  - To reduce the number of comparisons, store the candidates in a hash structure (x buckets)
    - Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets

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

Hash Structure

candidate of level k



x buckets to store the candidate of level k

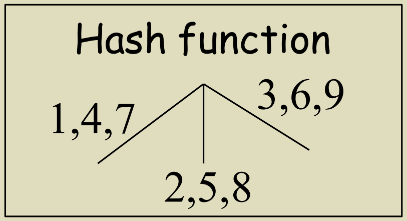
Hash Tree

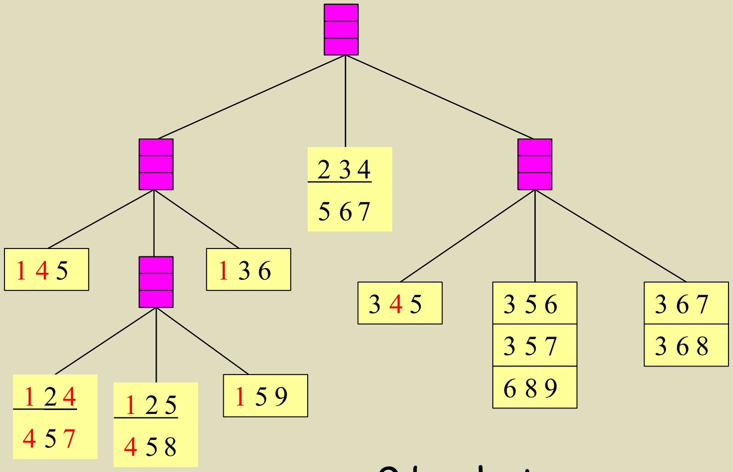
Suppose you have 15 candidate itemsets of length 3:

{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8},
{1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5},
{3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

We need:

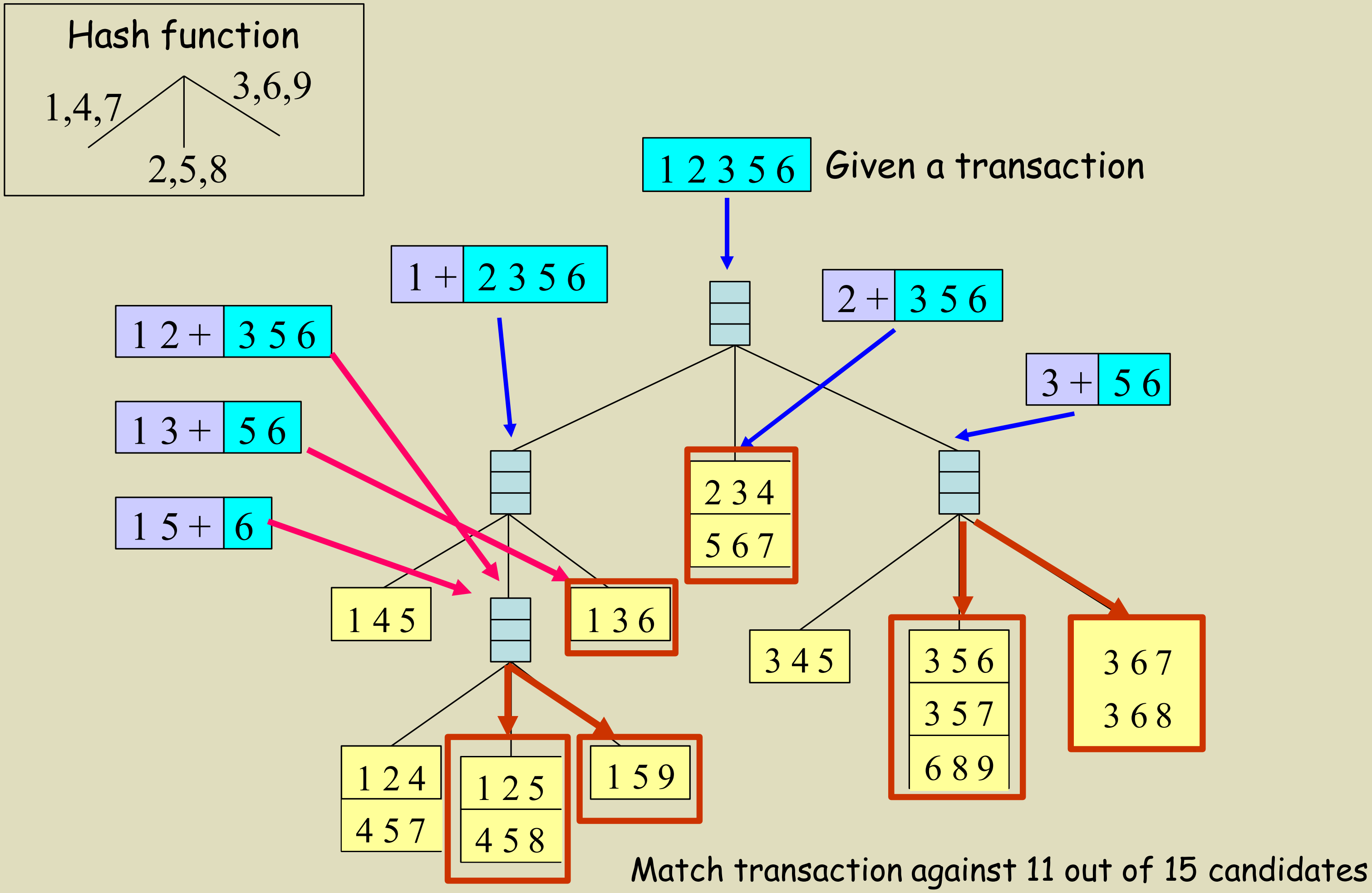
- Hash function
- Max leaf size: max number of itemsets stored in a leaf node
  (if number of candidate itemsets exceeds max leaf size, split the node)





9 buckets

Support Counting



Match transaction against 11 out of 15 candidates

Database Layout

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

- **Horizontal layout**:

TransactionID – list of items (Transactional)

TID	Bread	Beer	Coke	Diapers	Eggs	Milk
1	1	0	0	0	0	1
2	1	1	0	1	1	0
3	0	1	1	1	0	1
4	1	1	0	1	0	1
5	1	0	1	1	0	1

- **Vertical layout**:

Item – List of transactions (TID-list)

TID	Bread	Beer	Coke	Diapers	Eggs	Milk
1	1	0	0	0	0	1
2	1	1	0	1	1	0
3	0	1	1	1	0	1
4	1	1	0	1	0	1
5	1	0	1	1	0	1

TID	1	2	3	4	5
Bread	1	1	0	1	1
Beer	0	1	1	1	0
Coke	0	0	1	0	1
Diapers	0	1	1	1	1
Eggs	0	1	0	0	0
Milk	1	0	1	1	1

- In the **vertical layout** all data for a particular item is available in one record.
  - to count the itemset {beer, diaper} **intersect** TID-list of the item “beer” with TID-list of item “diaper”.

Database Projection

Given a transaction database and an itemset X

- Database ***selection*** operation:
  select only those records that contain X.

- Databases ***projection*** operation:
  delete the items of X from the selected records.

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

{beer}-projection

TID	*Transactions*
2	Bread, Diaper, Eg
3	Milk, Dia
4	

Compact Representations

Some itemsets are **redundant** because they have identical support as their supersets.

A method of reducing the large amount of frequent itemsets is to use so-called **compact or condensed representations**, such as

- **Maximal Frequent Itemset** (MFI) and
- **Closed Frequent Itemset** (CFI).

MFI are frequent itemsets for which none of their supersets is also frequent. Clearly, each frequent itemset is a subset of a maximal frequent itemset.

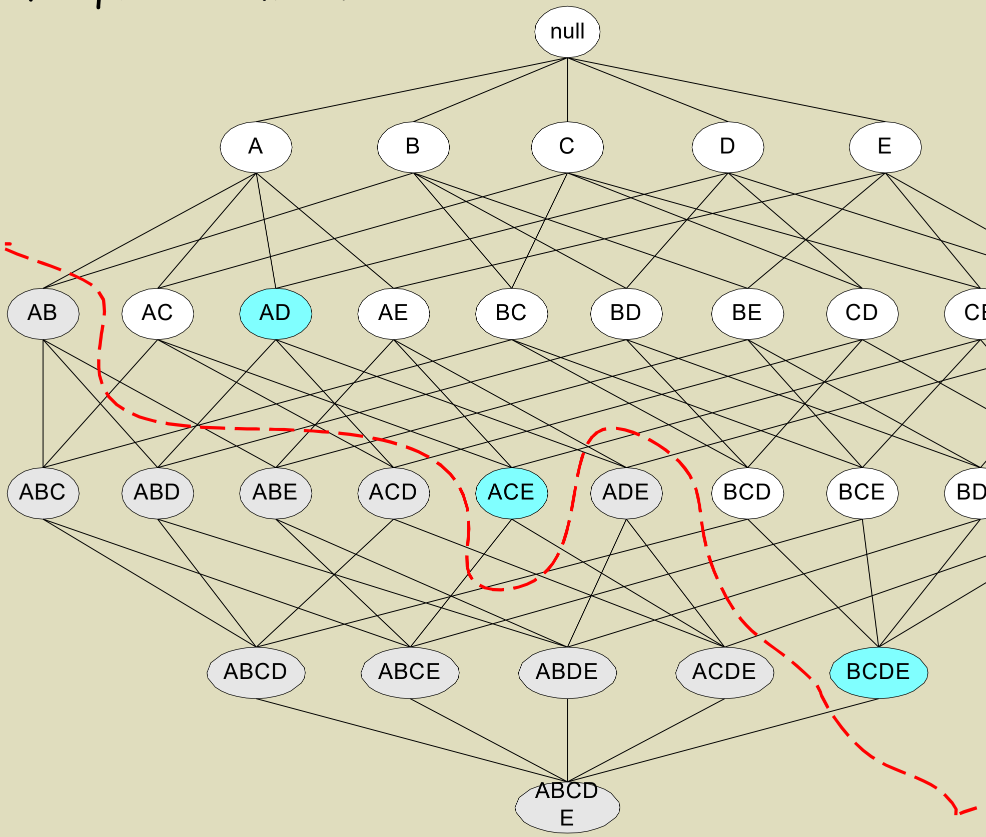
CFI are itemsets that completely characterize their associated set of transactions. That is, a frequent itemset is closed if it contains all items that occur in all transaction in which it is bought, i.e. it is the intersection of its supporting transactions.

The underlying idea of both concepts is that the set of all maximal/closed frequent itemsets represents all frequent itemsets but is far smaller.

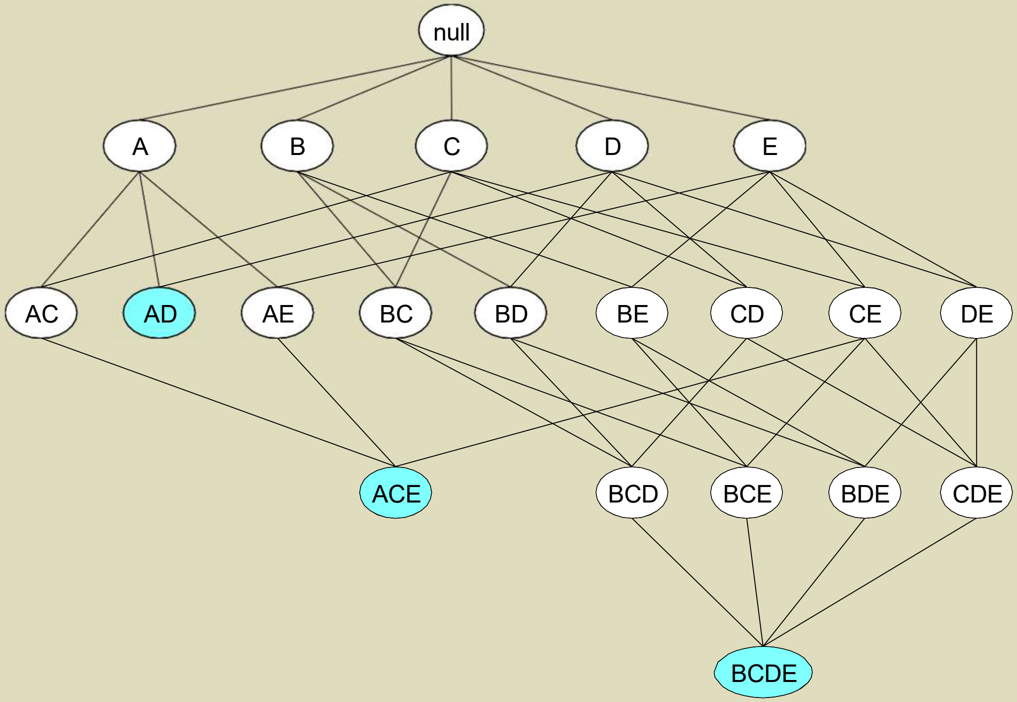
Maximal Frequent Itemsets

***Definition 1**: Maximal Frequent Itemset (MFI)*

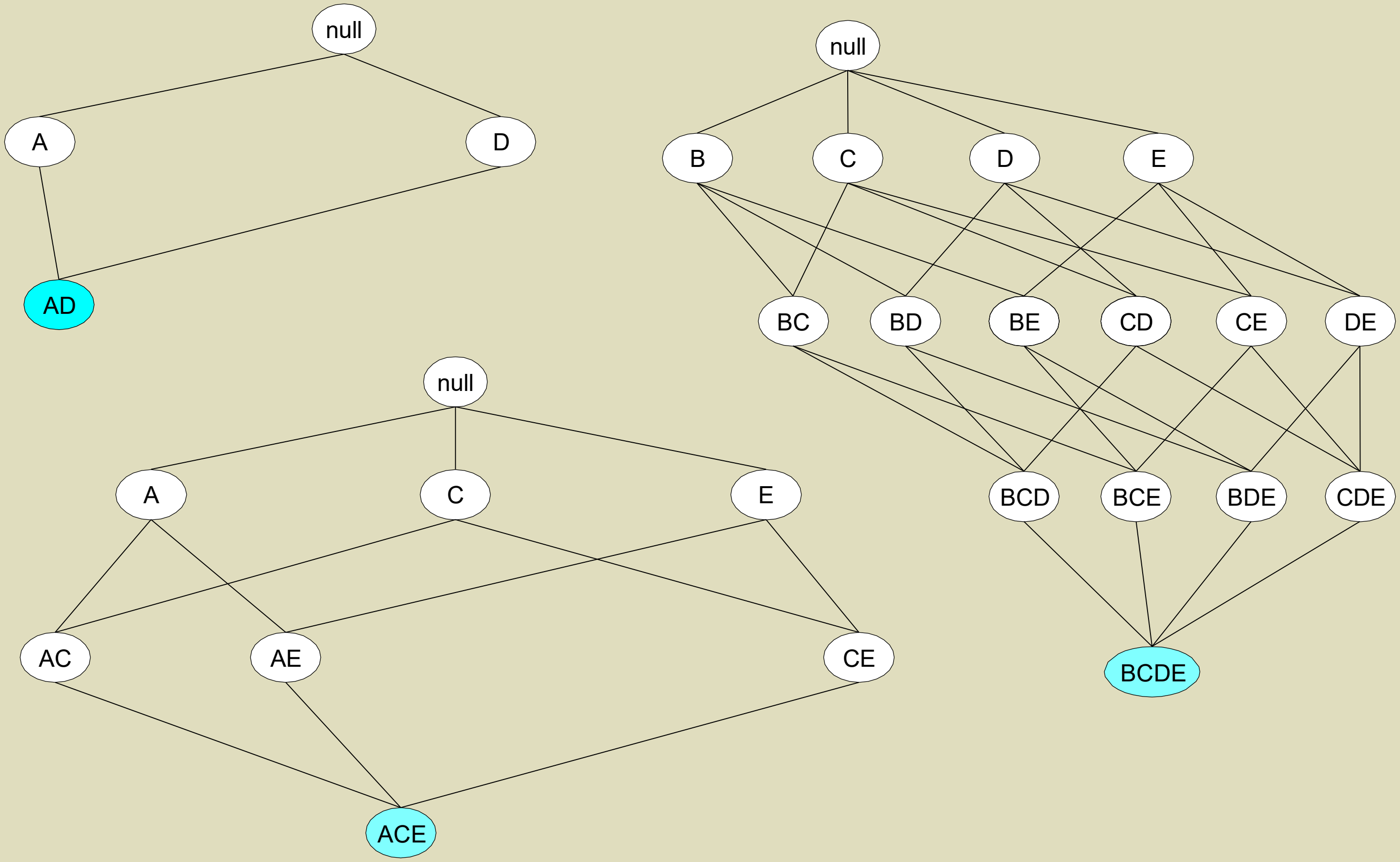
A frequent itemset is maximal if it is not a proper subset of any other frequent itemset.



Maximal Frequent Itemsets



Sublattice induced by MFI



Closed Frequent Itemsets

Problem with maximal frequent itemsets:
support of their subsets is not known and additional DB scans are needed.

***Definition 2**: Closed Frequent Itemset (CFI)*

A closed frequent itemset is a frequent itemset whose support is higher than the supports of all its proper supersets.

(All maximal frequent itemsets are closed.)

Example: minSup=2

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C}

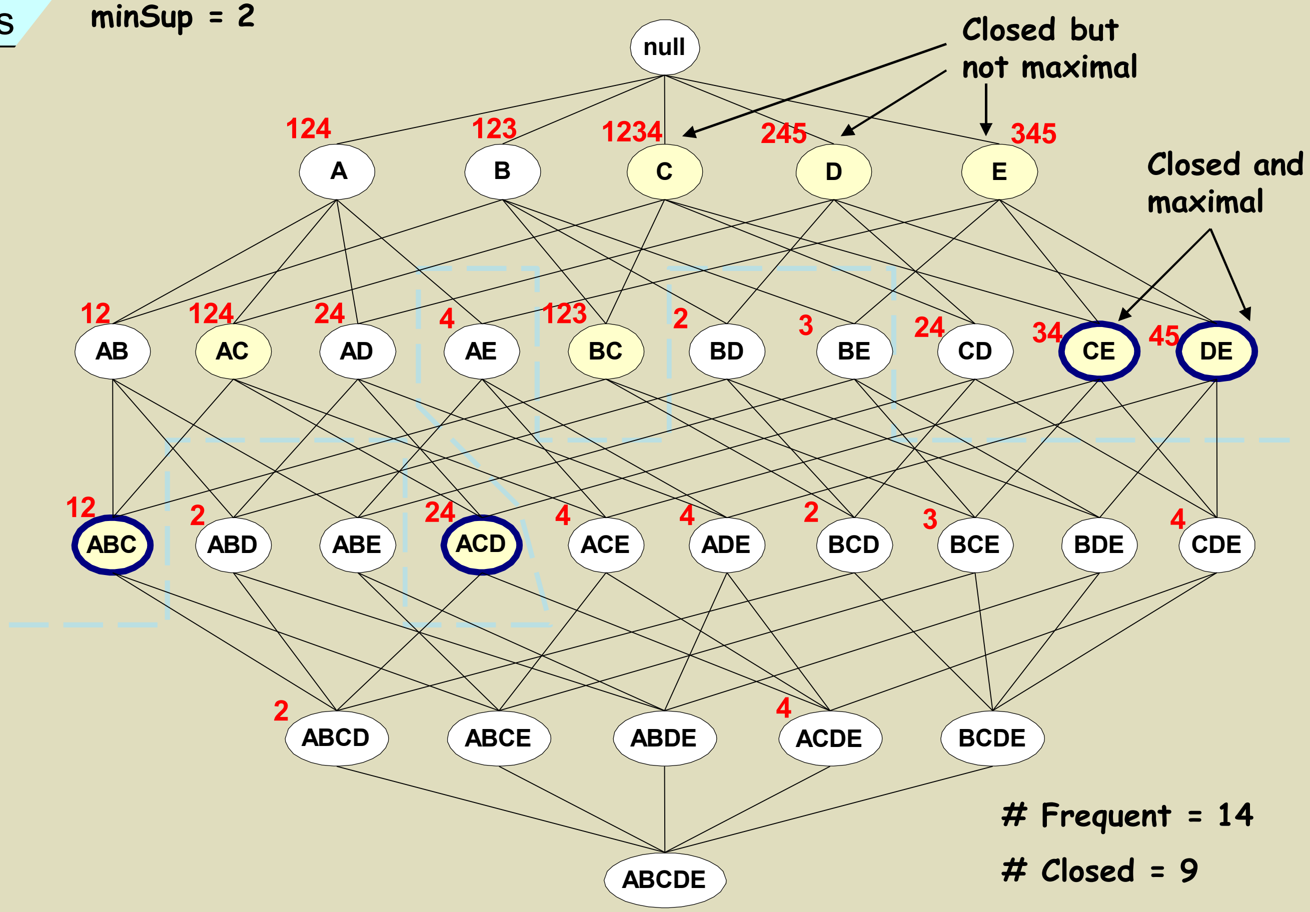
Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	3
{A,B}	4
{A,C}	2
{A,D}	2
{B,C}	3
{B,D}	3
{C,D}	2

Itemset	Suppor
{A,B,C}	
{A,B,D}	
{A,C,D	
{B,	
{	

MFI vs CFI

TID	Items
1	AB
2	A
3	
4	

minSup = 2



Closed but not maximal

Closed and maximal

# Frequent = 14

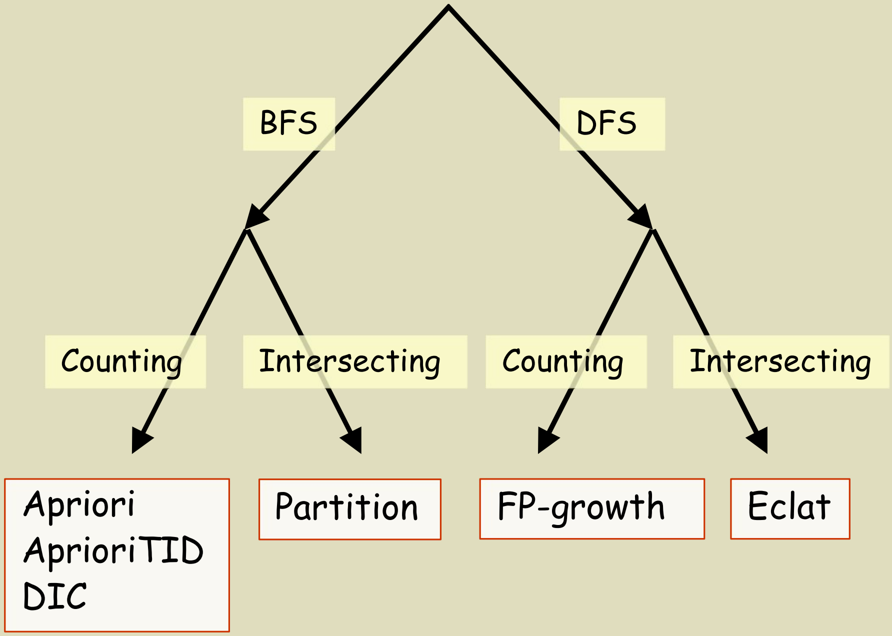
# Closed = 9

# Maximal = 4

Other ARM Algorithms

- AprioriTID and AprioriHybrid
- PARTITION
- DIC
- ECLAT
- Prefix-Tree based
  - FP-Growth
  - FP-Tree

Taxonomy of ARM Approaches

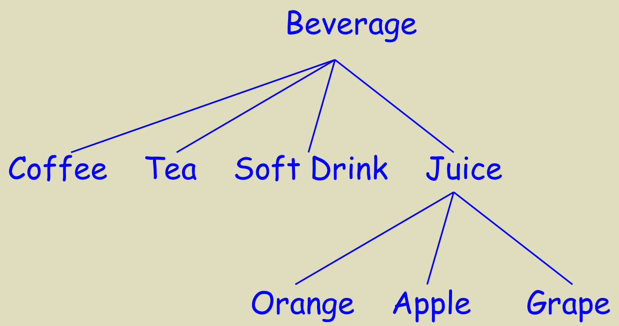

BFS
DFS
Counting
Intersecting
Counting
Intersecting
Apriori
AprioriTID
DIC
Partition
FP-growth
Eclat

Search Space exploration

Support determination

Extended Association Rules

- Generalized Association Rules
  - hierarchical taxonomy (concept hierarchy)



- Quantitative Association Rules
  - categorical and quantitative data
- Interval Data Association Rules
  - E.g., partition the age into 5-year-increment ranges
- Maximal Association Rules
- Sequential Association Rules
  - temporal data
  - E.g., first buy a PC, then CDROMs, and, finally, a digital camera

ARM References

- G. Di Fatta, "Association Rules and Frequent Patterns", Chapter 20333, Encyclopedia of Bioinformatics and Computational Biology, Elsevier, Vol.1, 2019, pp. 367-373.
- R. Agrawal, T. Imielinski, and A. Swami, Mining Association Rules Between Sets of Items in Large Database. Proc. ACM-SIGMOD International Conference, May 1993.
- R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules. Proc International Conference on Very Large Databases, Santiago, Chile, September 1994.
- R. Srikant and R. Agrawal, Mining Quantitative Association Rules in Large Relational Tables, SIGMOD, June 1996.
- Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In Proceedings of ACM SIGMOD'00,pages 1–12, May 2000.
- M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In D. Heckerman, H. Mannila, and D. Pregibon, editors, Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, pages 283–286. AAAI Press, 1997.
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In Dayal et al. [17], pages 432–444.
- H. Toivonen. Sampling large databases for association rules. In T.M. Vijayaraman, A.P. Buchmann, C. Mohan, and N.L. Sarda, editors, Proceedings 22nd International Conference on Very Large Data Bases, pages 134–145. Morgan Kaufmann, 1996.
- S. Brin, R. Motwani, J.D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, volume 26(2) of SIGMOD Record, pages 255–264. ACM Press, 1997.
- Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh, "Algorithms for Association Rule Mining - A General Survey and Comparison", ACM SIGKDD, July 2000.