CIS664-Knowledge Discovery and Data Mining

Mining Association Rules

Vasileios Megalooikonomou
Dept. of Computer and Information Sciences
Temple University

(based on notes by Jiawei Han and Micheline Kamber)

Agenda

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

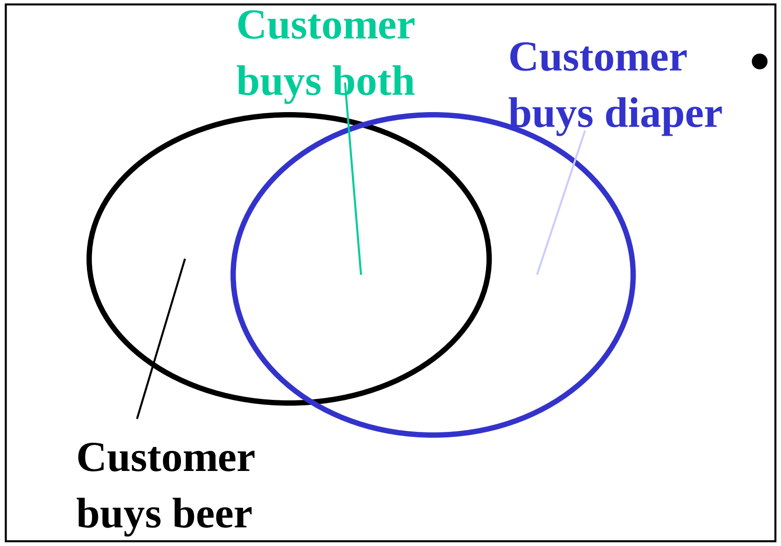
Association Mining?

- Association rule mining:
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
- Applications:
  - Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.
- Examples.
  - Rule form: "Body $\rightarrow$ Head [support, confidence]".
  - buys(x, "diapers") $\rightarrow$ buys(x, "beers") [0.5%, 60%]
  - major(x, "CS") ^ takes(x, "DB") $\rightarrow$ grade(x, "A") [1%, 75%]

Association Rules: Basic Concepts

- Given: (1) database of transactions, (2) each transaction is a list of items (purchased by a customer in a visit)
- Find: all rules that correlate the presence of one set of items with that of another set of items
  - E.g., *98% of people who purchase tires and auto accessories also get automotive services done*
- Applications
  - *$* \Rightarrow$ Maintenance Agreement* (What the store should do to boost Maintenance Agreement sales)
  - *Home Electronics $\Rightarrow *$* (What other products should the store stocks up?)
  - Attached mailing in direct marketing
  - Detecting “ping-pong”ing of patients, faulty “collisions”

Interestingness Measures: Support and Confidence



- Find all the rules $X \,\&\, Y \Rightarrow Z$ with minimum confidence and support
  - support, *s*, probability that a transaction contains {X ⍰ Y ⍰ Z}
  - confidence, *c*, conditional probability that a transaction having {X ⍰ Y} also contains *Z*

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Let minimum support 50%, and minimum confidence 50%, we have

- $A \Rightarrow C$ (50%, 66.6%)
- $C \Rightarrow A$ (50%, 100%)

Association Rule Mining: A Road Map

- Boolean vs. quantitative associations (Based on the types of values handled)
  - buys(x, "SQLServer") ^ buys(x, "DMBook") → buys(x, "DBMiner") [0.2%, 60%]
  - age(x, "30..39") ^ income(x, "42..48K") → buys(x, "PC") [1%, 75%]
- Single dimension vs. multiple dimensional associations (each distinct predicate of a rule is a dimension)
- Single level vs. multiple-level analysis (consider multiple levels of abstraction)
  - What brands of beers are associated with what brands of diapers?
- Extensions
  - Correlation, causality analysis
    - Association does not necessarily imply correlation or causality
  - Maxpatterns (a frequent pattern s.t. any proper subpattern is not frequent) and closed itemsets (if there exist no proper superset c' of c s.t. any transaction containing c also contains c')

Agenda

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Mining Association Rules—An Example

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Min. support 50%
Min. confidence 50%

Frequent Itemset	Support
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

For rule $A \Rightarrow C$:

support = support($\{A \cup C\}$) = 50%

confidence = support($\{A \cup C\}$)/support($\{A\}$) = 66.6%

The Apriori principle:

Any subset of a frequent itemset must be frequent

Mining Frequent Itemsets

- Find the *frequent itemsets*: the sets of items that have minimum support
  - A subset of a frequent itemset must also be a frequent itemset
    - i.e., if {*AB*} is a frequent itemset, both {*A*} and {*B*} should be a frequent itemset
  - Iteratively find frequent itemsets with cardinality from 1 to *k* (*k*-itemset)
- Use the frequent itemsets to generate association rules.

The Apriori Algorithm: Basic idea

- Join Step: C_k is generated by joining L_{k-1} with itself
- Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset
- Pseudo-code:

```
$C_k$: Candidate itemset of size k
$L_k$ : frequent itemset of size k

$L_1$ = {frequent items};
for ($k$ = 1; $L_k$ !=$\emptyset$; $k$++) 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

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

How to Generate Candidates?

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

```
insert into $C_k$
select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$
from $L_{k-1}$ $p$, $L_{k-1}$ $q$
where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
```

- Step 2: pruning

```
forall itemsets $c$ in $C_k$ do
    forall $(k-1)$-subsets $s$ of $c$ do
        if ($s$ is not in $L_{k-1}$) then delete $c$ from $C_k$
```

How to Count Supports of Candidates?

- Why is counting supports of candidates a problem?
  - The total number of candidates can be huge
  - Each transaction may contain many candidates
- Method:
  - Candidate itemsets are stored in a *hash-tree*
  - *Leaf* node of hash-tree contains a list of itemsets and counts
  - *Interior* node contains a hash table
  - *Subset function*: finds all the candidates contained in a transaction

Example of Generating Candidates

- L_3={*abc, abd, acd, ace, bcd*}
- 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*}

Improving Apriori’s Efficiency

- Hash-based itemset counting: A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- Transaction reduction: A transaction that does not contain any frequent k-itemset is useless in subsequent scans
- Partitioning: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
- Sampling: mining on a subset of given data, need a lower support threshold + a method to determine the completeness
- Dynamic itemset counting: add new candidate itemsets immediately (unlike Apriori) when all of their subsets are estimated to be frequent

Is Apriori Fast Enough? — Performance Bottlenecks

- The core of the Apriori algorithm:
  - Use frequent (k – 1)-itemsets to generate candidate frequent k-itemsets
  - Use database scan and pattern matching to collect counts for the candidate itemsets
- The bottleneck of *Apriori*: candidate generation
  - 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.
  - Multiple scans of database:
    - Needs $(n+1)$ scans, n is the length of the longest pattern

Mining Frequent Patterns Without Candidate Generation

- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure
  - highly condensed, but complete for frequent pattern mining
  - avoid costly database scans
- Develop an efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only!

Construct FP-tree from a Transaction DB

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}*	*{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}*

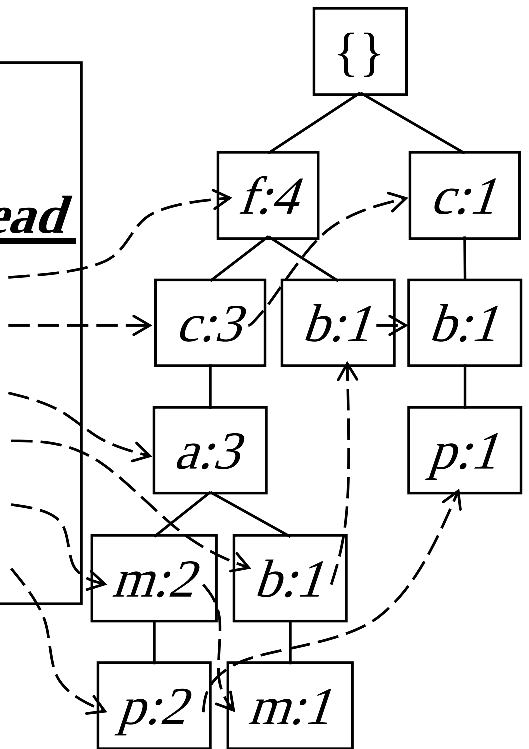
min_support = 0.5

Steps:

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

Header Table

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



Benefits of the FP-tree Structure

- Completeness:
  - never breaks a long pattern of any transaction
  - preserves complete information for frequent pattern mining
- Compactness
  - reduce irrelevant information—infrequent items are gone
  - frequency descending ordering: more frequent items are more likely to be shared
  - never be larger than the original database (if not count node-links and counts)

Mining Frequent Patterns Using FP-tree

- General idea (divide-and-conquer)
  - Recursively grow frequent pattern path using the FP-tree
- Method
  - For each 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)

Major Steps to Mine FP-tree

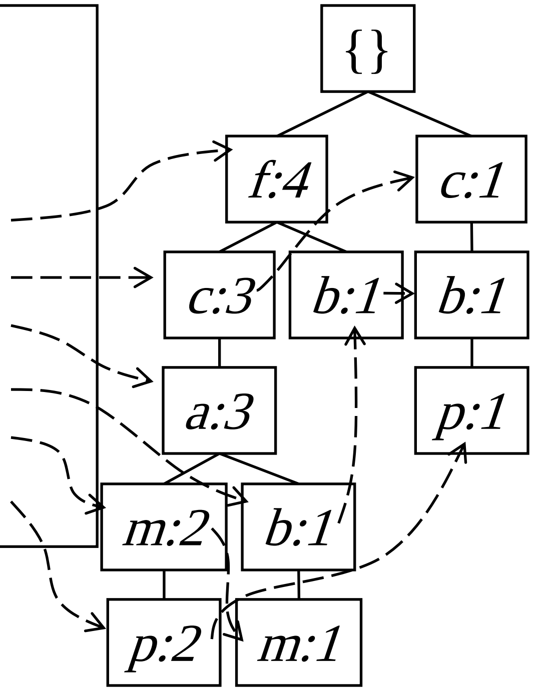
1) Construct conditional pattern base for each node in the FP-tree
2) Construct conditional FP-tree from each conditional pattern-base
3) Recursively mine conditional FP-trees and grow frequent patterns obtained so far
   - If the conditional FP-tree contains a single path, simply enumerate all the patterns

Step 1: From FP-tree to Conditional Pattern Base

- Starting at the frequent header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item
- Accumulate all of transformed prefix paths of that item to form a conditional pattern base

Header Table

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



Conditional **pattern bases**

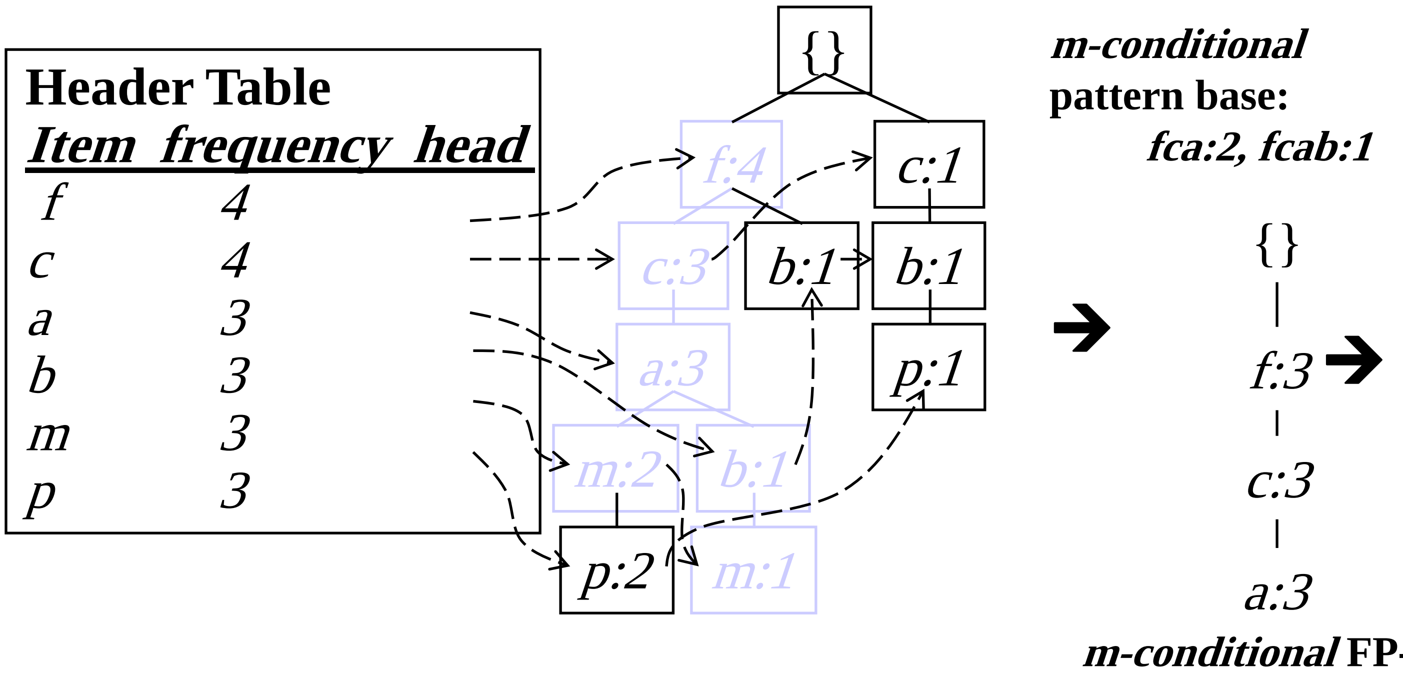
item	***cond. pattern base***
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

Properties of FP-tree for Conditional Pattern Base Construction

- Node-link property
  - For any frequent item a_i, all the possible frequent patterns that contain a_i can be obtained by following a_i's node-links, starting from a_i's head in the FP-tree header
- Prefix path property
  - To calculate the frequent patterns for a node a_i in a path P, only the prefix sub-path of a_i in P need to be accumulated, and its frequency count should carry the same count as node a_i.

Step 2: Construct 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



Header Table

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

m-conditional **pattern base:**

fca:2, fcab:1

m-conditional **FP-tree**

All frequent patterns concerning *m*

m,

fm, cm, am,

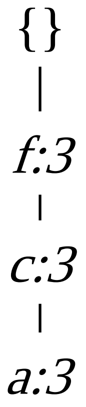
fcm, fam, cam,

fcam

Mining Frequent Patterns by Creating Conditional Pattern-Bases

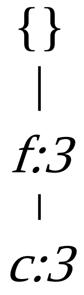
Item	Conditional pattern-base	Conditional FP-tree
p	{(fcam:2), (cb:1)}	{(c:3)}|p
m	{(fca:2), (fcab:1)}	{(f:3, c:3, a:3)}|m
b	{(fca:1), (f:1), (c:1)}	Empty
a	{(fc:3)}	{(f:3, c:3)}|a
c	{(f:3)}	{(f:3)}|c
f	Empty	Empty

Step 3: Recursively mine the conditional FP-tree



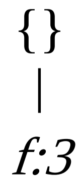
m-conditional **FP-tree**

Cond. pattern base of "am": (fc:3)



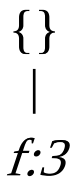
am-conditional **FP-tree**

Cond. pattern base of "cm": (f:3)



cm-conditional **FP-tree**

Cond. pattern base of "cam": (f:3)



cam-conditional **FP-tree**

Single FP-tree Path Generation

- Suppose an FP-tree T has a single path P
- The complete set of frequent pattern of T can be generated by enumeration of all the combinations of the sub-paths of P

$$\begin{tikzcd}
\{\} \arrow[d, dash] \\
f:3 \arrow[d, dash] \\
c:3 \arrow[d, dash] \\
a:3
\end{tikzcd}$$

➔

All frequent patterns concerning *m*

m,

fm, cm, am,

fcm, fam, cam,

fcam

***m-conditional* FP-tree**

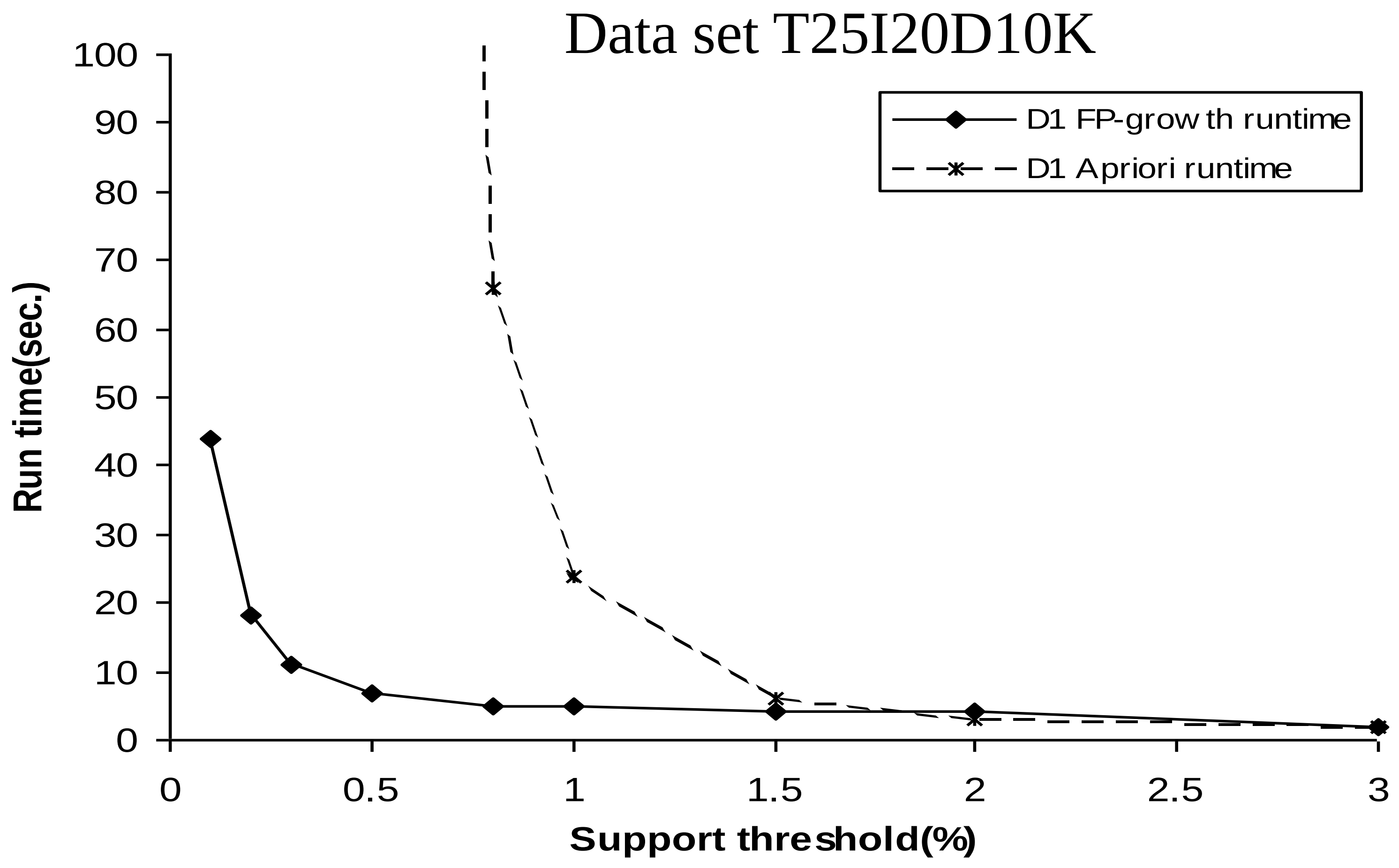
Principles of Frequent Pattern Growth

- Pattern growth property
  - Let α be a frequent itemset in DB, B be α's conditional pattern base, and β be an itemset in B. Then $\alpha \cup \beta$ is a frequent itemset in DB iff β is frequent in B.
- "*abcdef*" is a frequent pattern, if and only if
  - "*abcde*" is a frequent pattern, and
  - "*f*" is frequent in the set of transactions containing "*abcde*"

Why Is Frequent Pattern Growth Fast?

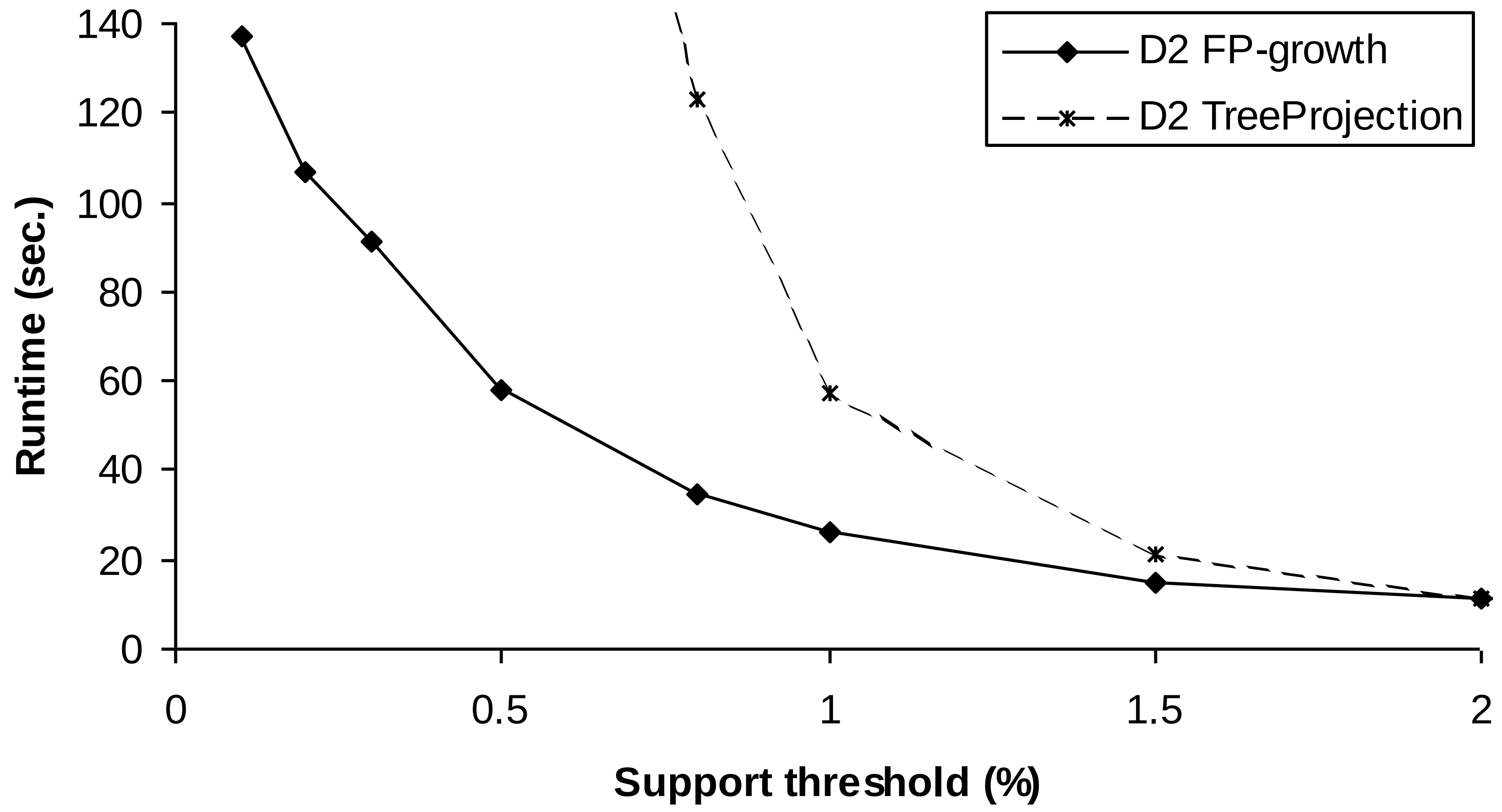
- Our performance study shows
  - FP-growth is an order of magnitude faster than Apriori, and is also faster than tree-projection
- Reasoning
  - No candidate generation, no candidate test
  - Use compact data structure
  - Eliminate repeated database scan
  - Basic operation is counting and FP-tree building

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

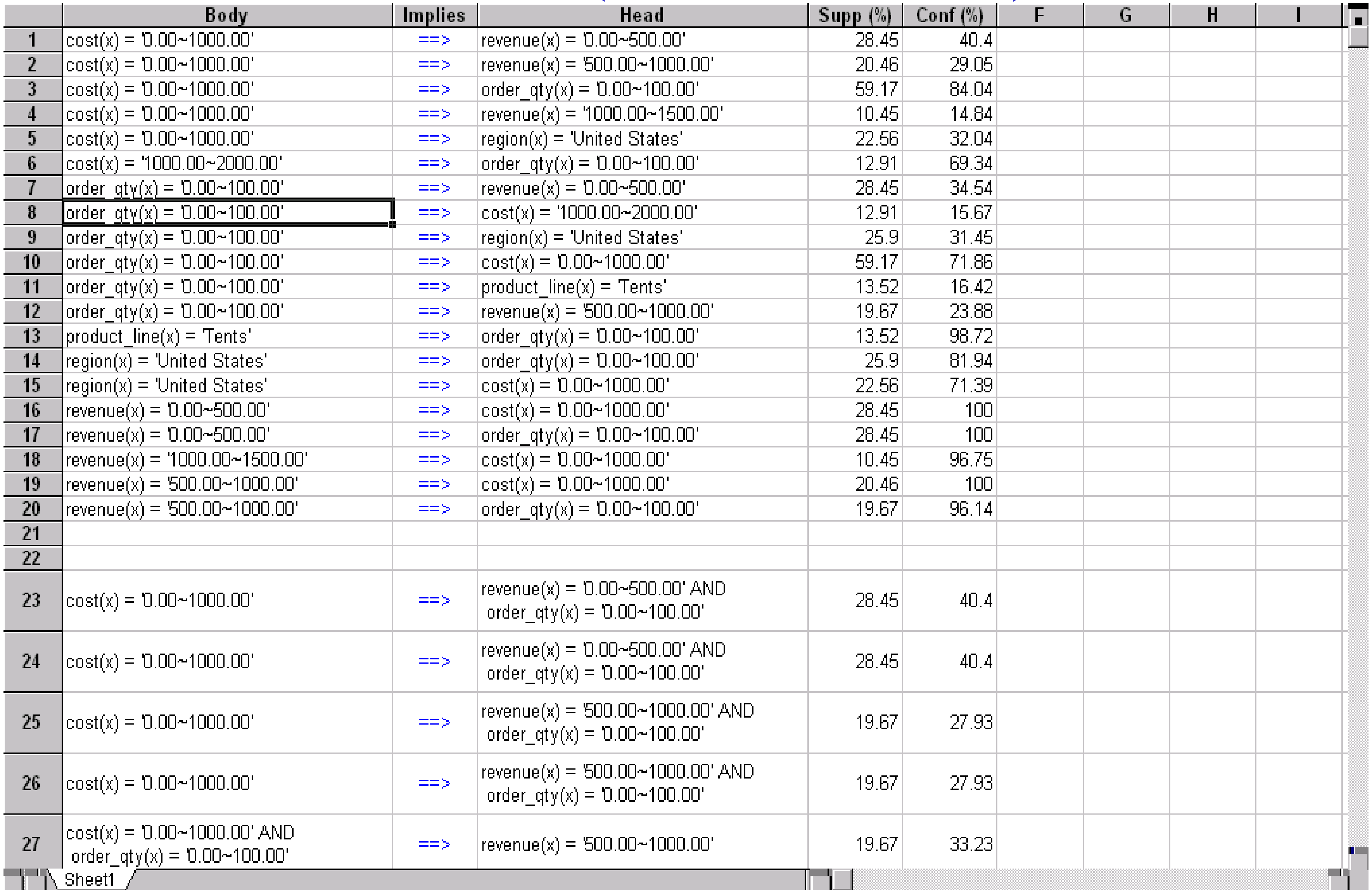


FP-growth vs. Tree-Projection: Scalability with Support Threshold

Data set T25I20D100K



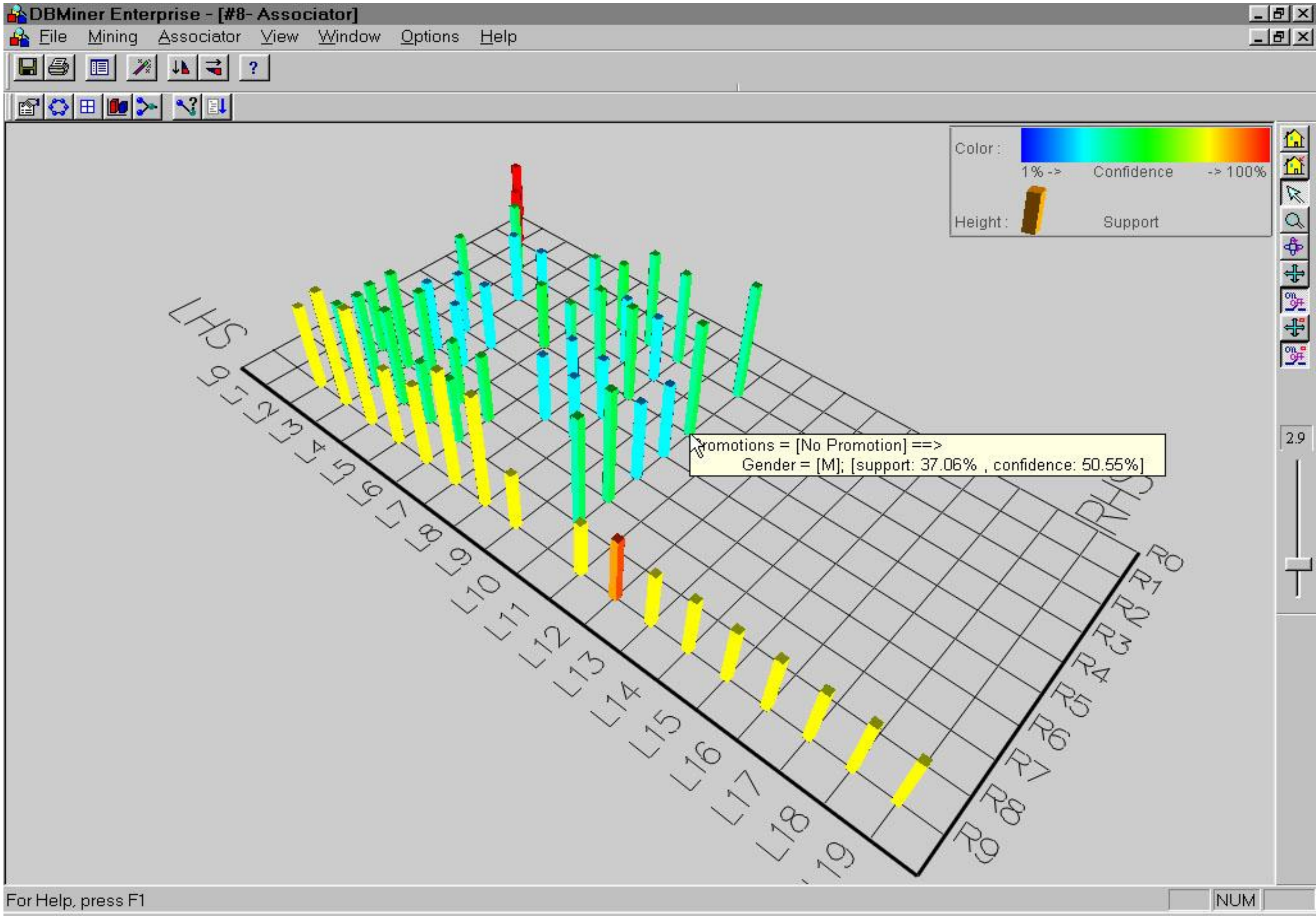
Presentation of Association Rules (Table Form)



	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

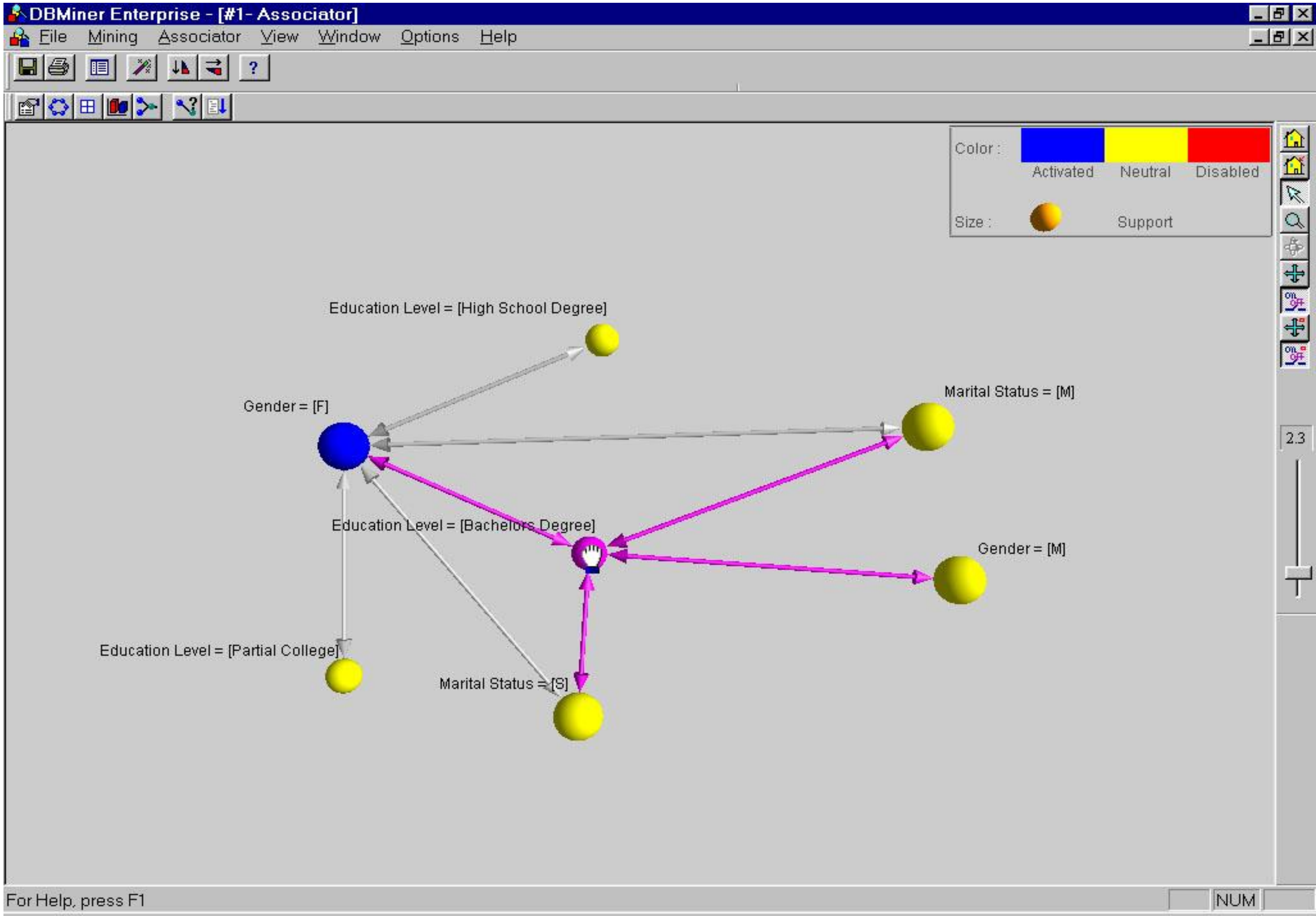
Visualization of Association Rule Using Plane Graph



DBMiner Enterprise - [#8- Associator]
File Mining Associator View Window Options Help
Color : 1% -> Confidence -> 100%
Height : Support
LHS
L0 L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12 L13 L14 L15 L16 L17 L18 L19
RHS
R0 R1 R2 R3 R4 R5 R6 R7 R8 R9
2.9
For Help, press F1
NUM

Promotions = [No Promotion] ==>
Gender = [M]; [support: 37.06% , confidence: 50.55%]

Visualization of Association Rule Using Rule Graph


DBMiner Enterprise - [#1- Associator]
File Mining Associator View Window Options Help
Color : Activated Neutral Disabled
Size : Support
Education Level = [High School Degree]
Gender = [F]
Marital Status = [M]
Education Level = [Bachelors Degree]
Gender = [M]
Education Level = [Partial College]
Marital Status = [S]
2.3
For Help, press F1
NUM

Iceberg Queries

- Icerberg query: Compute aggregates over one or a set of attributes only for those whose aggregate values is above certain threshold
- Example:

```
select P.custID, P.itemID, sum(P.qty)
from purchase P
group by P.custID, P.itemID
having sum(P.qty) >= 10
```

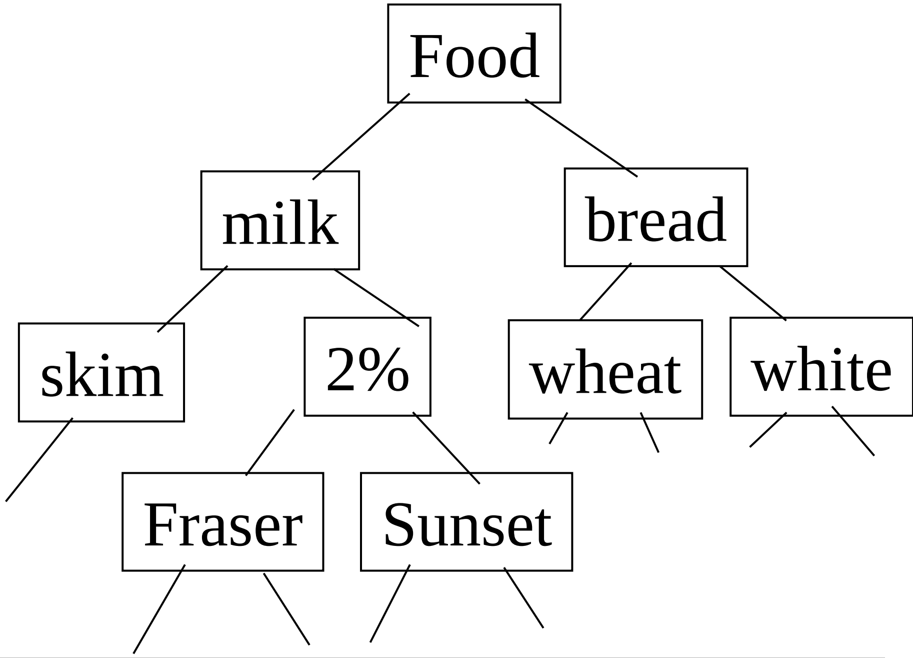
- Compute iceberg queries efficiently by Apriori:
  - First compute lower dimensions
  - Then compute higher dimensions only when all the lower ones are above the threshold

Agenda

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Multiple-Level Association Rules

- Items often form hierarchies.
- Items at the lower level are expected to have lower support.
- Rules regarding itemsets at appropriate levels could be quite useful.
- Transaction database can be encoded based on dimensions and levels
- We can explore shared multi-level mining



TID	Items
T1	{111, 121, 211, 221}
T2	{111, 211, 222, 323}
T3	{112, 122, 221, 411}
T4	{111, 121}
T5	{111, 122, 211, 221, 413}

Mining Multi-Level Associations

- A top_down, progressive deepening approach:
  - First find high-level strong rules:

    milk $\rightarrow$ bread [20%, 60%].
  - Then find their lower-level "weaker" rules:

    2% milk $\rightarrow$ wheat bread [6%, 50%].
- Variations at mining multiple-level association rules.
  - Level-crossed association rules:

    2% *milk* $\rightarrow$ *Wonder wheat bread*
  - Association rules with multiple, alternative hierarchies:

    2% *milk* $\rightarrow$ *Wonder bread*

Multi-level Association: Uniform Support vs. Reduced Support

- Uniform Support: the same minimum support for all levels
  - + One minimum support threshold.   No need to examine itemsets containing any item whose ancestors do not have minimum support.
  - – Lower level items do not occur as frequently. If support threshold
    - too high ⇒ miss low level associations
    - too low ⇒ generate too many high level associations

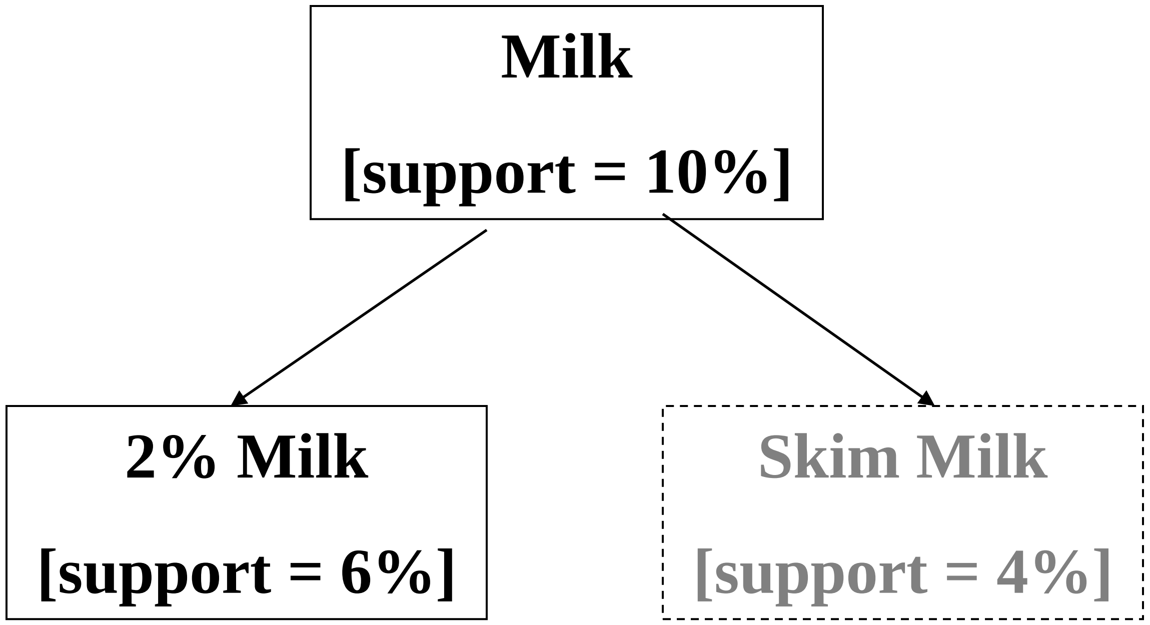
- Reduced Support: reduced minimum support at lower levels
  - There are 4 search strategies:
    - Level-by-level independent
    - Level-cross filtering by k-itemset
    - Level-cross filtering by single item
    - Controlled level-cross filtering by single item (level passage threshold)

Uniform Support

Multi-level mining with uniform support

Level 1
min_sup = 5%

Level 2
min_sup = 5%



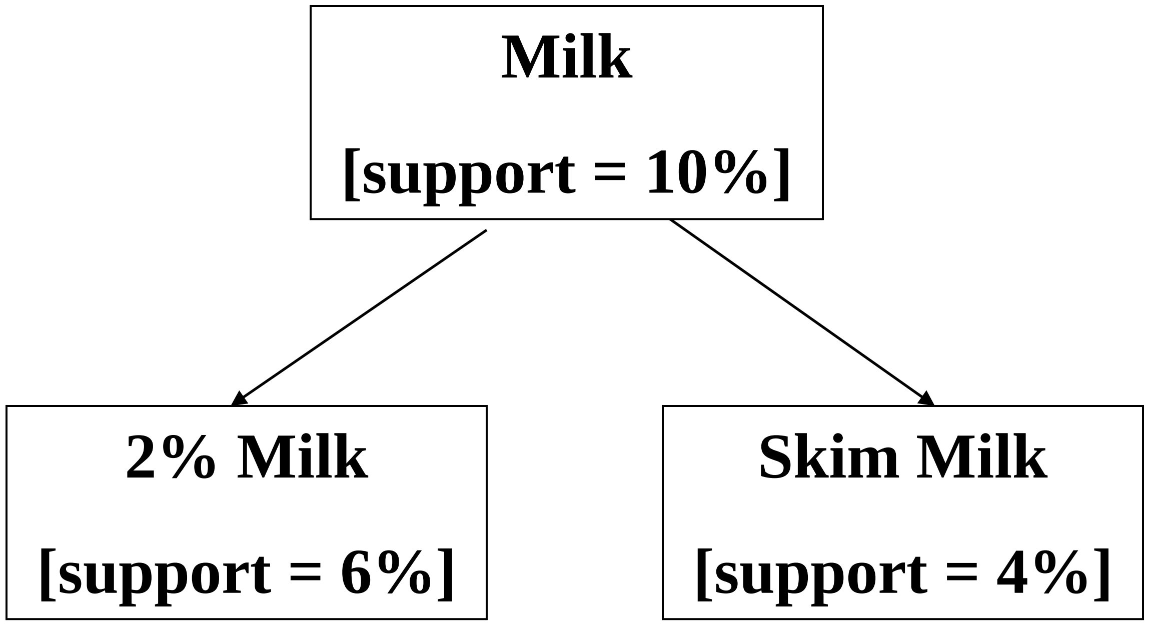
Back

Reduced Support

Multi-level mining with reduced support

Level 1
min_sup = 5%

Level 2
min_sup = 3%



Back

Multi-level Association: Redundancy Filtering

- Some rules may be redundant due to “ancestor” relationships between items.
- Example
  - milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%]
  - 2% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%]
- We say the first rule is an ancestor of the second rule.
- A rule is redundant if its support is close to the “expected” value, based on the rule’s ancestor.

Multi-Level Mining: Progressive Deepening

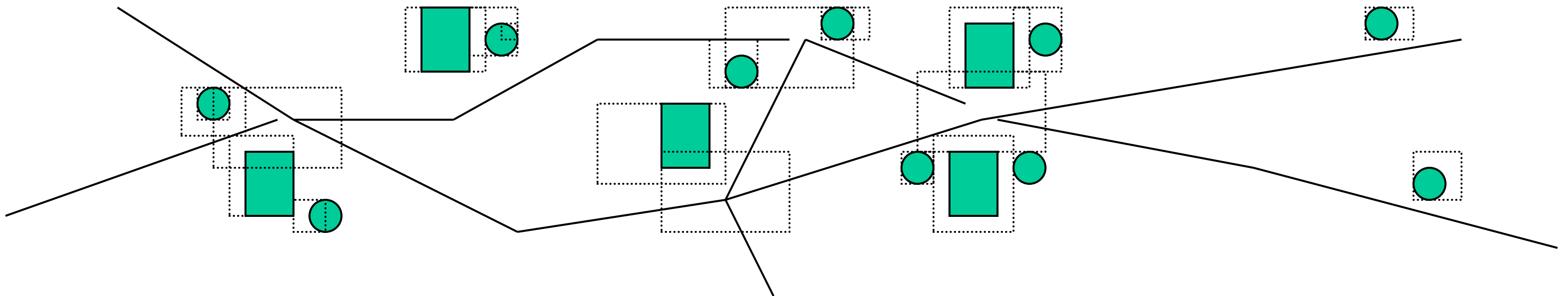
- A top-down, progressive deepening approach:
  - First mine high-level frequent items:
    milk (15%), bread (10%)
  - Then mine their lower-level "weaker" frequent itemsets:
    2% milk (5%), wheat bread (4%)
- Different min_support threshold across multi-levels lead to different algorithms:
  - If adopting the same *min_support* across multi-levels
    then toss *t* if any of *t*'s ancestors is infrequent.
  - If adopting reduced *min_support* at lower levels
    then examine only those descendents whose ancestor's support is frequent/non-negligible.

Progressive Refinement of Data Mining Quality

- Why progressive refinement?
  - Mining operator can be expensive or cheap, fine or rough
  - Trade speed with quality: step-by-step refinement.
- Superset coverage property:
  - Preserve all the positive answers—allow a positive false test but not a false negative test.
- Two- or multi-step mining:
  - First apply rough/cheap operator (superset coverage)
  - Then apply expensive algorithm on a substantially reduced candidate set (Koperski & Han, SSD'95).

Progressive Refinement Mining of Spatial Association Rules

- Hierarchy of spatial relationship:
  - "g_close_to": near_by, touch, intersect, contain, etc.
  - First search for rough relationship and then refine it.
- Two-step mining of spatial association:
  - Step 1: rough spatial computation (as a filter)
    - Using MBR or R-tree for rough estimation.
  - Step2: Detailed spatial algorithm (as refinement)
    - Apply only to those objects which have passed the rough spatial association test (no less than *min_support*)



Agenda

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Multi-Dimensional Association: Concepts

- Single-dimensional rules:

  buys(X, “milk”) $\Rightarrow$ buys(X, “bread”)

- Multi-dimensional rules: ⍰ 2 dimensions or predicates
  - Inter-dimension association rules (*no repeated predicates*)

    age(X,”19-25”) $\wedge$ occupation(X,“student”) $\Rightarrow$ buys(X,“coke”)
  - hybrid-dimension association rules (*repeated predicates*)

    age(X,”19-25”) $\wedge$ buys(X, “popcorn”) $\Rightarrow$ buys(X, “coke”)
- Categorical Attributes
  - finite number of possible values, no ordering among values
- Quantitative Attributes
  - numeric, implicit ordering among values

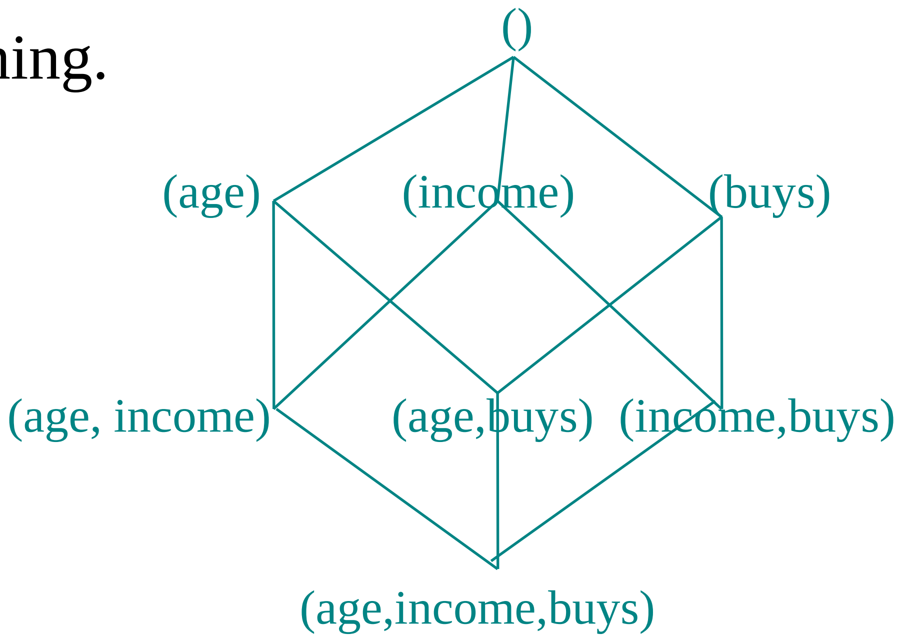
Techniques for Mining MD Associations

- Search for frequent *k*-predicate set:
  - Example: {age, occupation, buys} is a 3-predicate set.
  - Techniques can be categorized by how age is treated:

1. Using static discretization of quantitative attributes
   - Quantitative attributes are statically discretized by using predefined concept hierarchies.
2. Quantitative association rules
   - Quantitative attributes are dynamically discretized into “bins”based on the distribution of the data.
3. Distance-based association rules
   - This is a dynamic discretization process that considers the distance between data points.

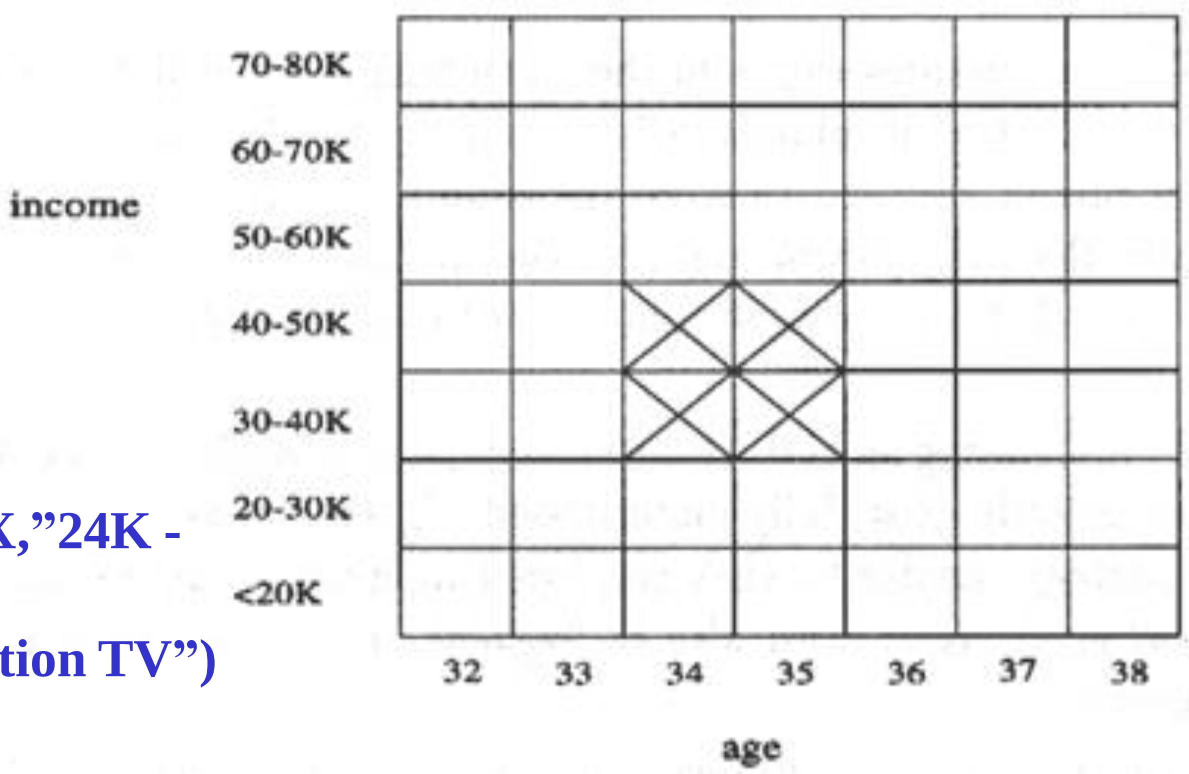
Static Discretization of Quantitative Attributes

- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges.
- In relational database, finding all frequent k-predicate sets will require *k* or *k*+1 table scans.
- Data cube is well suited for mining.
- The cells of an n-dimensional cuboid correspond to the predicate sets.
- Mining from data cubes can be much faster.



Quantitative Association Rules

- Numeric attributes are *dynamically* discretized
  - Such that the confidence or compactness of the rules mined is maximized.
- 2-D quantitative association rules: $A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$
- Cluster "adjacent" association rules to form general rules using a 2-D grid.
- Example:



age(X,"30-34") $\wedge$ income(X,"24K - 48K")
$\Rightarrow$ buys(X,"high resolution TV")

ARCS (Association Rule Clustering System)

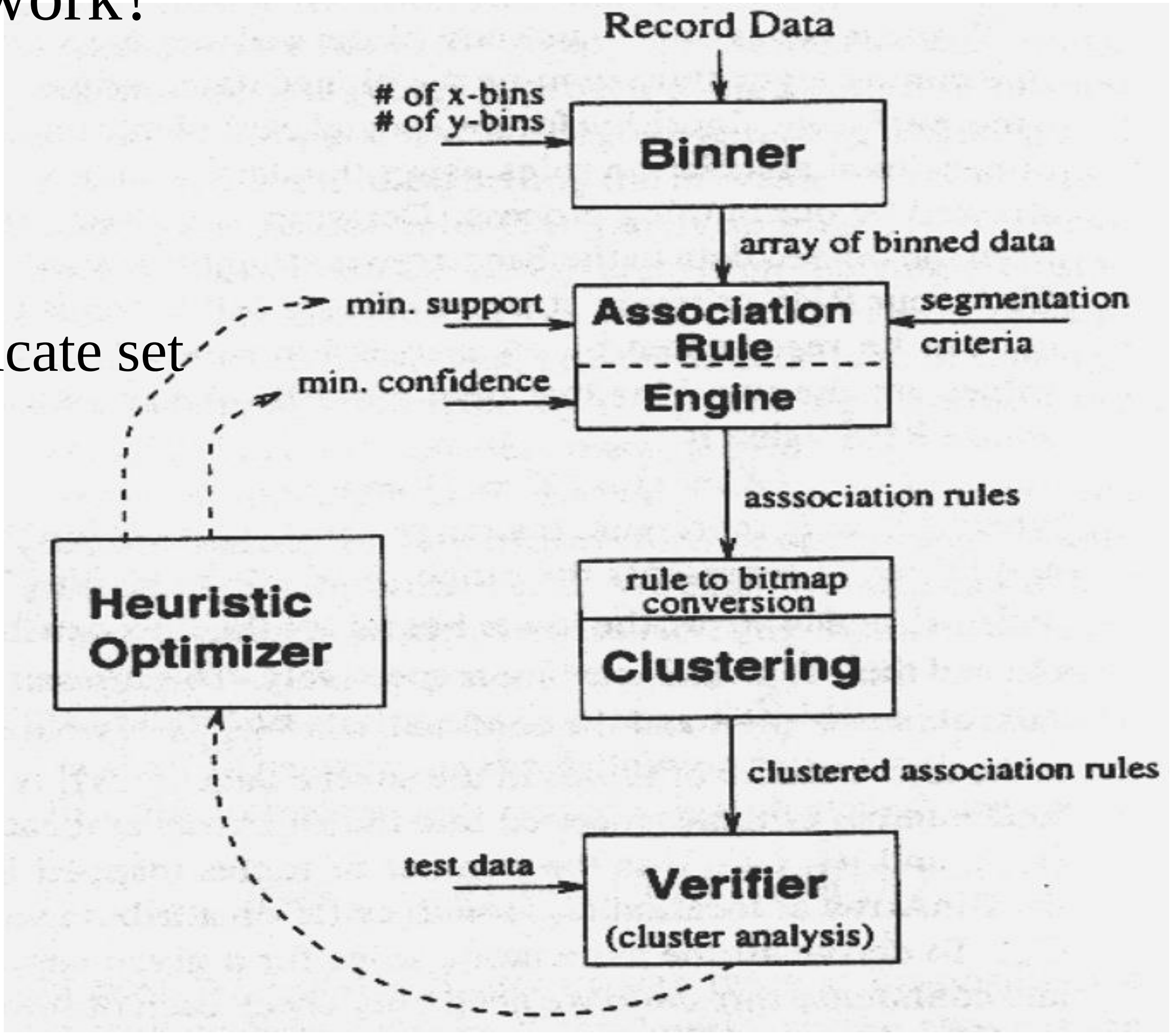
How does ARCS work?

1. Binning

2. Find frequent predicate set

3. Clustering

4. Optimize



Limitations of ARCS

- Only quantitative attributes on LHS of rules.
- Only 2 attributes on LHS. (2D limitation)
- An alternative to ARCS
  - Non-grid-based
  - equi-depth binning
  - clustering based on a measure of *partial completeness (information lost due to partitioning).*
  - "***Mining Quantitative Association Rules in Large Relational Tables***" by R. Srikant and R. Agrawal.

Mining Distance-based Association Rules

- Binning methods do not capture the semantics of interval data

Price($)	Equi-width (width $10)	Equi-depth (depth 2)	Distance-based
7	[0,10]	[7,20]	[7,7]
20	[11,20]	[22,50]	[20,22]
22	[21,30]	[51,53]	[50,53]
50	[31,40]		
51	[41,50]		
53	[51,60]		

- Distance-based partitioning, more meaningful discretization considering:
  - density/number of points in an interval
  - "closeness" of points in an interval

Agenda

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Interestingness Measures

- Objective measures

  Two popular measurements:

  - *support;* and
  - *confidence*

- Subjective measures (Silberschatz & Tuzhilin, KDD95)

  A rule (pattern) is interesting if

  - it is *unexpected* (surprising to the user); and/or
  - *actionable* (the user can do something with it)

- From association to correlation and causal structure analysis.
  - Association does not necessarily imply correlation or causal relationships

Criticism to Support and Confidence

- Example 1: (Aggarwal & Yu, PODS98)
  - Among 5000 students
    - 3000 play basketball
    - 3750 eat cereal
    - 2000 both play basket ball and eat cereal
  - *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%.
  - *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is far more accurate, although with lower support and confidence

	basketball	not basketball	sum(row)
cereal	2000	1750	3750
not cereal	1000	250	1250
sum(col.)	3000	2000	5000

Criticism to Support and Confidence

- Example 2:
  - X and Y: positively correlated,
  - X and Z, negatively related
  - support and confidence of X=>Z dominates
- We need a measure of dependent or correlated events

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

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

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 Interestingness Measures: Interest

- Interest (correlation, lift) $$\frac{P(A \wedge B)}{P(A)P(B)}$$
  - taking both P(A) and P(B) in consideration
  - P(A^B)=P(B)*P(A), if A and B are independent events
  - A and B negatively correlated, if the value is less than 1; otherwise A and B positively correlated

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

Itemset	Support	Interest
X,Y	25%	2
X,Z	37.50%	0.9
Y,Z	12.50%	0.57

Agenda

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Constraint-Based Mining

- Interactive, exploratory mining giga-bytes of data?
  - Could it be real? — Making good use of constraints!
- What kinds of constraints can be used in mining?
  - Knowledge type constraint: classification, association, etc.
  - Data constraint: SQL-like queries
    - Find product pairs sold together in Vancouver in Dec.'98.
  - Dimension/level constraints:
    - in relevance to region, price, brand, customer category.
  - Rule constraints
    - On the form of the rules to be mined (e.g., # of predicates, etc)
    - small sales (price $< \$10$) triggers big sales (sum $> \$200$).
  - Interestingness constraints:
    - Thresholds on measures of interestingness
    - strong rules (min_support $\geq 3\%$, min_confidence $\geq 60\%$).

Rule Constraints in Association Mining

- Two kind of rule constraints:
  - Rule form constraints: meta-rule guided mining.
    - $P(x, y) \wedge Q(x, w) \rightarrow$ takes(x, “database systems”).
  - Rule (content) constraint: constraint-based query optimization (Ng, et al., SIGMOD’98).
    - $sum(LHS) < 100 \wedge min(LHS) > 20 \wedge count(LHS) > 3 \wedge sum(RHS) > 1000$
- 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).
    - $sum(LHS) < min(RHS) \wedge max(RHS) < 5 * sum(LHS)$

Constraint-Based Association Query

- Database: (1) trans (TID, Itemset), (2) itemInfo (Item, Type, Price)
- A constrained assoc. query (CAQ) is in the form of $\{(S_1, S_2)/C\}$,
  - where C is a set of constraints on S_1, S_2 including frequency constraint
- A classification of (single-variable) constraints:
  - Class constraint: $S \subset A$. *e.g.* $S \subset \mathit{Item}$
  - Domain constraint:
    - $S \theta v$, $\theta \in \{ =, \neq, <, \leq, >, \geq \}$. *e.g.* $S.\mathit{Price} < 100$
    - $v \theta S$, θ is $\in$ or $\notin$. *e.g.* $\mathit{snacks} \notin S.\mathit{Type}$
    - $V \theta S$, or $S \theta V$, $\theta \in \{ \subseteq, \subset, \not\subset, =, \neq \}$
      - *e.g.* $\{\mathit{snacks}, \mathit{sodas}\} \subseteq S.\mathit{Type}$
  - Aggregation constraint: $\mathit{agg}(S)\ \theta\ v$, where agg is in $\{\mathit{min}, \mathit{max}, \mathit{sum}, \mathit{count}, \mathit{avg}\}$, and $\theta \in \{ =, \neq, <, \leq, >, \geq \}$.
    - *e.g.* $\mathit{count}(S_1.\mathit{Type}) = 1$, $\mathit{avg}(S_2.\mathit{Price}) < 100$

Constrained Association Query Optimization Problem

- Given a CAQ $= \{ (S_1, S_2) / C \}$, the algorithm should be :
  - sound: It only finds frequent sets that satisfy the given constraints C
  - complete: All frequent sets satisfy the given constraints C are found
- A naïve solution:
  - Apply Apriori for finding all frequent sets, and then to test them for constraint satisfaction one by one.
- Other approach:
  - Comprehensive analysis of the properties of constraints and try to push them as deeply as possible inside the frequent set computation.

Anti-monotone and Monotone Constraints

- A constraint C_a is anti-monotone iff. for any pattern S not satisfying C_a, none of the super-patterns of S can satisfy C_a
- A constraint C_m is monotone iff. for any pattern S satisfying C_m, every super-pattern of S also satisfies it

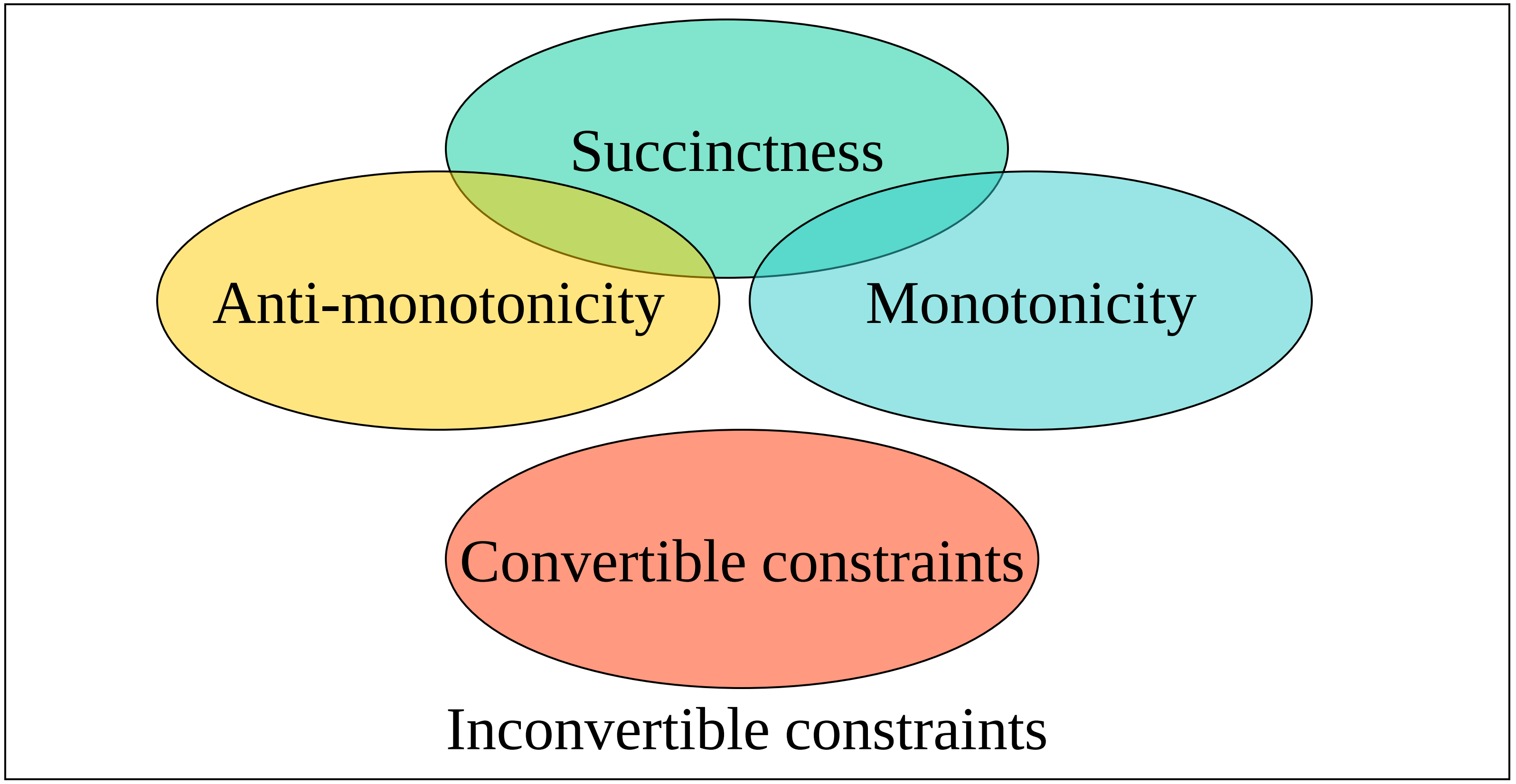
Succinct Constraint

- A subset of item I_s is a succinct set, if it can be expressed as $\sigma_p(I)$ for some selection predicate p, where σ is a selection operator
- $SP \subseteq 2^I$ is a succinct power set, if there is a fixed number of succinct set $I_1, \ldots, I_k \subseteq I$, s.t. SP can be expressed in terms of the strict power sets of $I_1, \ldots, I_k$ using union and minus
- A constraint C_s is succinct provided $SAT_{Cs}(I)$ is a succinct power set

Convertible Constraint

- Suppose all items in patterns are listed in a total order R
- A constraint C is convertible anti-monotone iff a pattern S satisfying the constraint implies that each suffix of S w.r.t. R also satisfies C
- A constraint C is convertible monotone iff a pattern S satisfying the constraint implies that each pattern of which S is a suffix w.r.t. R also satisfies C

Relationships Among Categories of Constraints



Property of Constraints: Anti-Monotone

- Anti-monotonicity: *If a set S violates the constraint, any superset of S violates the constraint.*
- Examples:
  - *sum(S.Price)* $\leq$ *v* is anti-monotone
  - *sum(S.Price)* $\geq$ *v* is not anti-monotone
  - *sum(S.Price)* $=$ *v* is partly anti-monotone
- Application:
  - Push "*sum(S.price)* $\leq$ 1000" deeply into iterative frequent set computation.

Characterization of Anti-Monotonicity Constraints

$S\ \theta\ v,\ \theta \in \{=, \le, \ge\}$	yes
$v \in S$	no
$S \supseteq V$ $S \subseteq V$ $S = V$	no yes partly
$\min(S) \le v$ $\min(S) \ge v$ $\min(S) = v$	no yes partly
$\max(S) \le v$ $\max(S) \ge v$ $\max(S) = v$	yes no partly
$\text{count}(S) \le v$ $\text{count}(S) \ge v$ $\text{count}(S) = v$	yes no partly
$\text{sum}(S) \le v$ $\text{sum}(S) \ge v$ $\text{sum}(S) = v$	yes no partly
$\text{avg}(S)\ \theta\ v,\ \theta \in \{=, \le, \ge\}$	convertible
(frequent constraint)	(yes)

Example of Convertible Constraints: Avg(S) θ V

- Let R be the value descending order over the set of items
  - E.g. I={9, 8, 6, 4, 3, 1}
- Avg(S) $\geq$ v is convertible monotone w.r.t. R
  - If S is a suffix of S_1, avg(S_1) $\geq$ avg(S)
    - {8, 4, 3} is a suffix of {9, 8, 4, 3}
    - avg({9, 8, 4, 3})=6 $\geq$ avg({8, 4, 3})=5
  - If S satisfies avg(S) $\geq$v, so does S_1
    - {8, 4, 3} satisfies constraint avg(S) $\geq$ 4, so does {9, 8, 4, 3}

Property of Constraints: Succinctness

- Succinctness:
  - For any set S_1 and S_2 satisfying C, $S_1 \cup S_2$ satisfies C
  - Given A_1 is the sets of size 1 satisfying C, then any set S satisfying C are based on A_1 , i.e., it contains a subset belongs to A_1 ,
- Example :
  - $sum(S.Price) \geq v$ is not succinct
  - $min(S.Price) \leq v$ is succinct
- Optimization:
  - If C is succinct, then C is pre-counting prunable. The satisfaction of the constraint alone is not affected by the iterative support counting.

Characterization of Constraints by Succinctness

S θ v, θ ∈ { =, ≤, ≥ }	Yes
v ∈ S	yes
S ⊇ V S ⊆ V S = V	yes yes yes
min(S) ≤ v min(S) ≥ v min(S) = v	yes yes yes
max(S) ≤ v max(S) ≥ v max(S) = v	yes yes yes
count(S) ≤ v count(S) ≥ v count(S) = v	weakly weakly weakly
sum(S) ≤ v sum(S) ≥ v sum(S) = v	no no no
avg(S) θ v, θ ∈ { =, ≤, ≥ }	no
(frequent constraint)	(no)

Agenda

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Summary

- Association rule mining
  - probably the most significant contribution from the database community in KDD
  - large number of papers
- Many interesting issues have been explored
- An interesting research direction
  - Association analysis in other types of data: spatial data, multimedia data, time series data, etc.