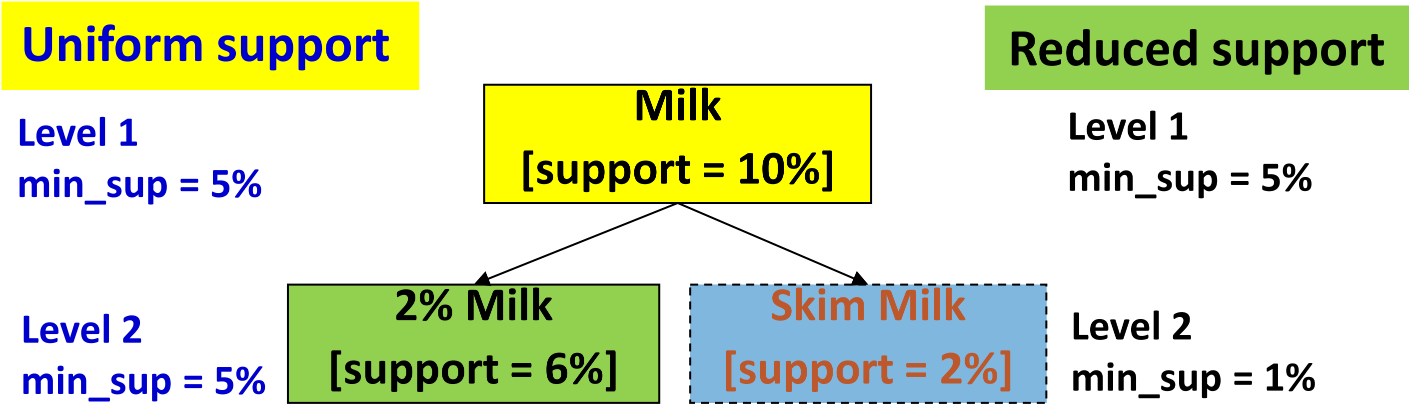
Mining Multiple-Level Associations

Mining Multiple-Level Frequent Patterns

- Items often form hierarchies
  - Ex.: Dairyland 2% milk; Wonder wheat bread
- How to set min-support thresholds?
  - Uniform min-support across multiple levels (reasonable?)
  - Level-reduced min-support: Items at the lower level are expected to have lower support
- Efficient mining: *Shared* multi-level mining
  - Use the lowest min-support to pass down the set of candidates



Redundancy Filtering at Mining Multi-Level Associations

- Multi-level association mining may generate many redundant rules
- Redundancy filtering: Some rules may be redundant due to "ancestor" relationships between items

  (Suppose the 2% milk sold is about ¼ of milk sold in gallons)

  - milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%] (1)
  - 2% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%] (2)
- A rule is *redundant* if its support is close to the "expected" value, according to its "ancestor" rule, and it has a similar confidence as its "ancestor"
  - Rule (1) is an ancestor of rule (2), which one to prune?

Customized Min-Supports for Different Kinds of Items

- We have used the same min-support threshold for all the items or item sets to be mined in each association mining
- In reality, some items (e.g., diamond, watch, …) are valuable but less frequent
- It is necessary to have customized min-support settings for different kinds of items
- One Method: Use group-based "individualized" min-support
  - E.g., {diamond, watch}: 0.05%; {bread, milk}: 5%; …
  - How to mine such rules efficiently?
    - Existing scalable mining algorithms can be easily extended to cover such cases

Mining Multi-Dimensional Associations

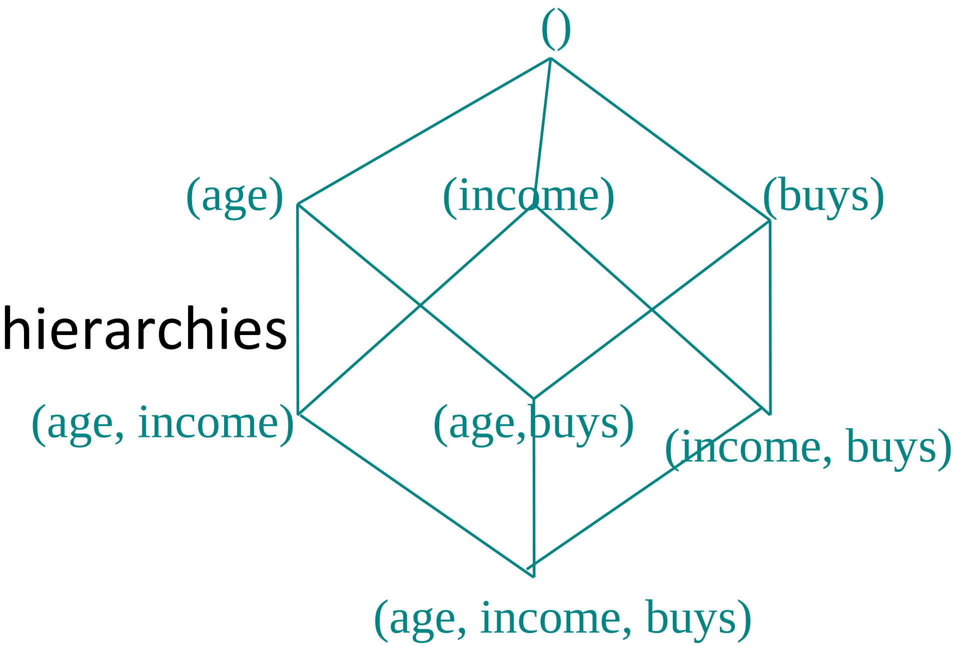
Mining Multi-Dimensional Associations

- Single-dimensional rules (e.g., items are all in "product" dimension)
  - buys(X, "milk") $\Rightarrow$ buys(X, "bread")
- Multi-dimensional rules (i.e., items in $\geq$ 2 dimensions or predicates)
  - Inter-dimension association rules (*no repeated predicates*)
    - age(X, "18-25") $\wedge$ occupation(X, "student") $\Rightarrow$ buys(X, "coke")
  - Hybrid-dimension association rules (*repeated predicates*)
    - age(X, "18-25") $\wedge$ buys(X, "popcorn") $\Rightarrow$ buys(X, "coke")
- Attributes can be categorical or numerical
  - Categorical Attributes (e.g., *profession, product*: no ordering among values): Data cube for inter-dimension association
  - Quantitative Attributes: Numeric, implicit ordering among values—discretization, clustering, and gradient approaches

Mining Quantitative Associations

Mining Quantitative Associations

- Mining associations with numerical attributes
  - Ex.: Numerical attributes: age and salary
- Methods
  - Static discretization based on predefined concept hierarchies
    - Data cube-based aggregation
  - Dynamic discretization based on data distribution
  - Clustering: Distance-based association
    - First one-dimensional clustering, then association
  - Deviation analysis:
    - Gender = female $\Rightarrow$ Wage: mean=\$7/hr (overall mean = \$9)



Mining Extraordinary Phenomena in Quantitative Association Mining

- Mining extraordinary (i.e., interesting) phenomena
  - Ex.: Gender = female $\Rightarrow$ Wage: mean=$7/hr (overall mean = $9)
  - LHS: a subset of the population
  - RHS: an extraordinary behavior of this subset
- The rule is accepted only if a statistical test (e.g., Z-test) confirms the inference with high confidence
- Subrule: Highlights the extraordinary behavior of a subset of the population of the super rule
  - Ex.: (Gender = female) ^ (South = yes) $\Rightarrow$ mean wage = $6.3/hr
- Rule condition can be categorical or numerical (quantitative rules)
  - Ex.: Education in [14-18] (yrs) $\Rightarrow$ mean wage = $11.64/hr
- Efficient methods have been developed for mining such extraordinary rules (e.g., Aumann and Lindell@KDD'99)

Mining Negative Correlations

Rare Patterns vs. Negative Patterns

- Rare patterns
  - Very low support but interesting (e.g., buying Rolex watches)
  - How to mine them? Setting individualized, group-based min-support thresholds for different groups of items
- Negative patterns
  - Negatively correlated: Unlikely to happen together
  - Ex.: Since it is unlikely that the same customer buys both a Ford Expedition (an SUV car) and a Ford Fusion (a hybrid car), buying a Ford Expedition and buying a Ford Fusion are likely negatively correlated patterns
  - How to define negative patterns?

Defining Negative Correlated Patterns

- A support-based definition
  - If itemsets A and B are both frequent but rarely occur together, i.e., sup(A U B) << sup (A) × sup(B)
  - Then A and B are negatively correlated

Does this remind you the definition of *lift?*

- Is this a good definition for large transaction datasets?
- Ex.: Suppose a store sold two needle packages A and B 100 times each, but only one transaction contained both A and B
  - When there are in total 200 transactions, we have
    - s(A U B) = 0.005, s(A) × s(B) = 0.25, s(A U B) << s(A) × s(B)
  - But when there are 10^5 transactions, we have
    - s(A U B) = $1/10^5$, s(A) × s(B) = $1/10^3 \times 1/10^3$, s(A U B) > s(A) × s(B)
  - What is the problem?—Null transactions: The support-based definition is not null-invariant!

Defining Negative Correlation: Need Null-Invariance in Definition

- A good definition on negative correlation should take care of the null-invariance problem
  - Whether two itemsets A and B are negatively correlated should not be influenced by the number of null-transactions
- A Kulczynski measure-based definition
  - If itemsets A and B are frequent but $(P(A|B) + P(B|A))/2 < \epsilon$, where ϵ is a negative pattern threshold, then A and B are negatively correlated
- For the same needle package problem:
  - No matter there are in total 200 or 10^5 transactions
  - If $\epsilon = 0.01$, we have $(P(A|B) + P(B|A))/2 = (0.01 + 0.01)/2 < \epsilon$

Mining Compressed Patterns

Mining Compressed Patterns

Pat-ID	Item-Sets	Support
P1	{38,16,18,12}	205227
P2	{38,16,18,12,17}	205211
P3	{39,38,16,18,12,17}	101758
P4	{39,16,18,12,17}	161563
P5	{39,16,18,12}	161576

- Closed patterns
  - P1, P2, P3, P4, P5
  - Emphasizes too much on support
  - There is no compression
- Max-patterns
  - P3: information loss
- Desired output (a good balance):
  - P2, P3, P4

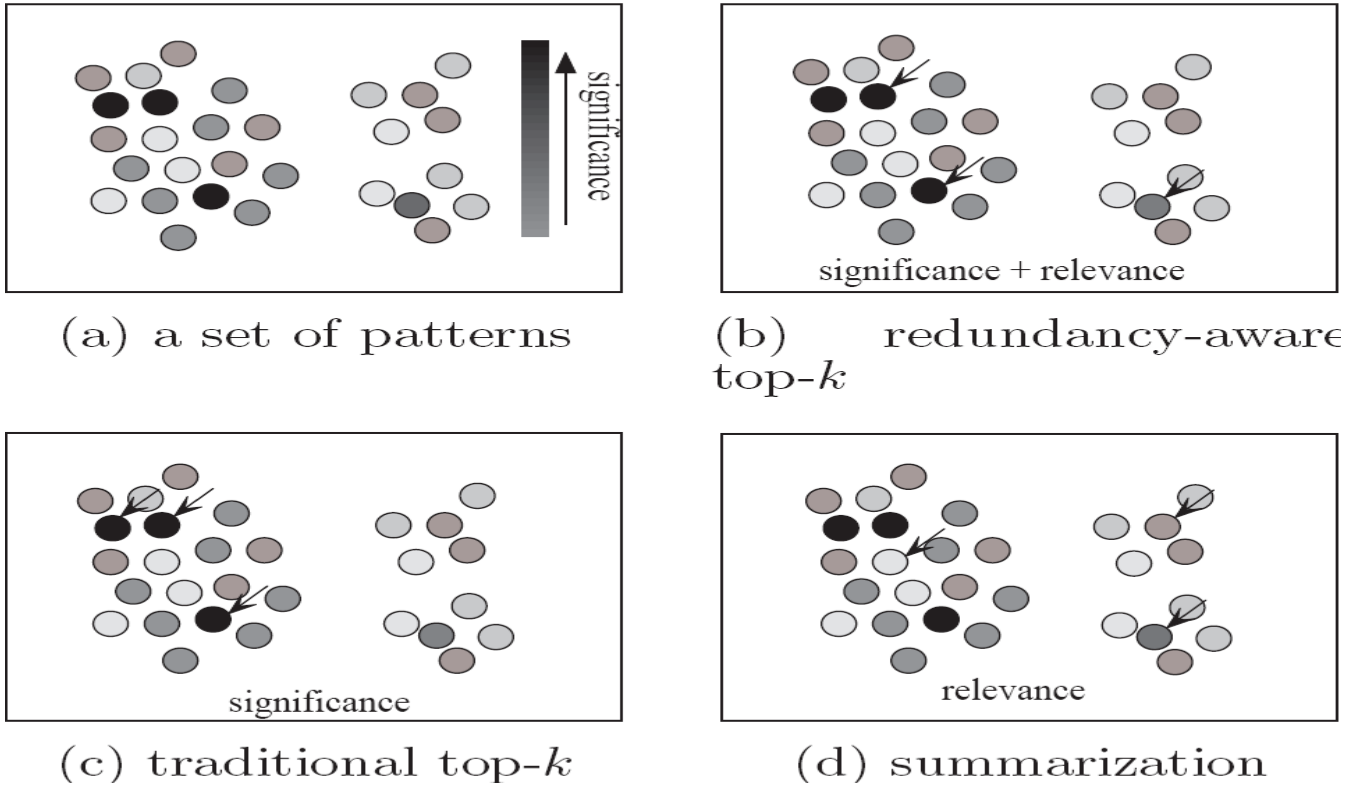
- Why mining compressed patterns?
  - Too many scattered patterns but not so meaningful
- Pattern distance measure

$$Dist(P_1, P_2) = 1 - \frac{|T(P_1) \cap T(P_2)|}{|T(P_1) \cup T(P_2)|}$$

- δ-clustering: For each pattern P, find all patterns which can be expressed by P and whose distance to P is within δ (δ-cover)
- All patterns in the cluster can be represented by P
- Method for efficient, direct mining of compressed frequent patterns (e.g., D. Xin, J. Han, X. Yan, H. Cheng, "On Compressing Frequent Patterns", Knowledge and Data Engineering, 60:5-29, 2007)

Redundancy-Aware Top-k Patterns

- Desired patterns: high significance & low redundancy



- Method: Use MMS (Maximal Marginal Significance) for measuring the combined significance of a pattern set
- Xin et al., Extracting Redundancy-Aware Top-K Patterns, KDD'06

Summary

Summary: Mining Diverse Patterns

- Efficient methods have been developed for mining various kinds of patterns
  - Mining Multiple-Level Associations
  - Mining Multi-Dimensional Associations
  - Mining Quantitative Associations
  - Mining Negative Correlations
  - Mining Compressed and Redundancy-Aware Patterns

Recommended Readings

- R. Srikant and R. Agrawal, “Mining generalized association rules”, VLDB'95
- Y. Aumann and Y. Lindell, “A Statistical Theory for Quantitative Association Rules”, KDD'99
- K. Wang, Y. He, J. Han, “Pushing Support Constraints Into Association Rules Mining”, IEEE Trans. Knowledge and Data Eng. 15(3): 642-658, 2003
- D. Xin, J. Han, X. Yan and H. Cheng, "On Compressing Frequent Patterns", Knowledge and Data Engineering, 60(1): 5-29, 2007
- D. Xin, H. Cheng, X. Yan, and J. Han, "Extracting Redundancy-Aware Top-K Patterns", KDD'06
- 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

Mining Colossal Patterns

Mining Long Patterns: Challenges

- Mining long patterns is needed in bioinformatics, social network analysis, software engineering, …
  - But the methods introduced so far mine only short patterns (e.g., length < 10)
- Challenges of mining long patterns
  - The curse of "downward closure" property of frequent patterns
    - Any sub-pattern of a frequent pattern is frequent
    - If $\{a_1, a_2, …, a_{100}\}$ is frequent, then $\{a_1\}, \{a_2\}, …, \{a_{100}\}, \{a_1, a_2\}, \{a_1, a_3\}, …, \{a_1, a_{100}\}, \{a_1, a_2, a_3\}, …$ are all frequent! There are about 2^{100} such frequent itemsets!
  - Whether searching in breadth-first (e.g., Apriori) or depth-first (e.g., FPgrowth), if we still adopt the "small to large" step-by-step growing paradigm, we have to examine so many patterns, which leads to combinatorial explosion!

Colossal Patterns: A Motivating Example

T_1 = 2 3 4 ….. 39 40
T_2 = 1 3 4 ….. 39 40
: .
: .
: .
: .
T_{40}=1 2 3 4 …… 39
T_{41}= 41 42 43 ….. 79
T_{42}= 41 42 43 ….. 79
: .
: .
T_{60}= 41 42 43 … 79

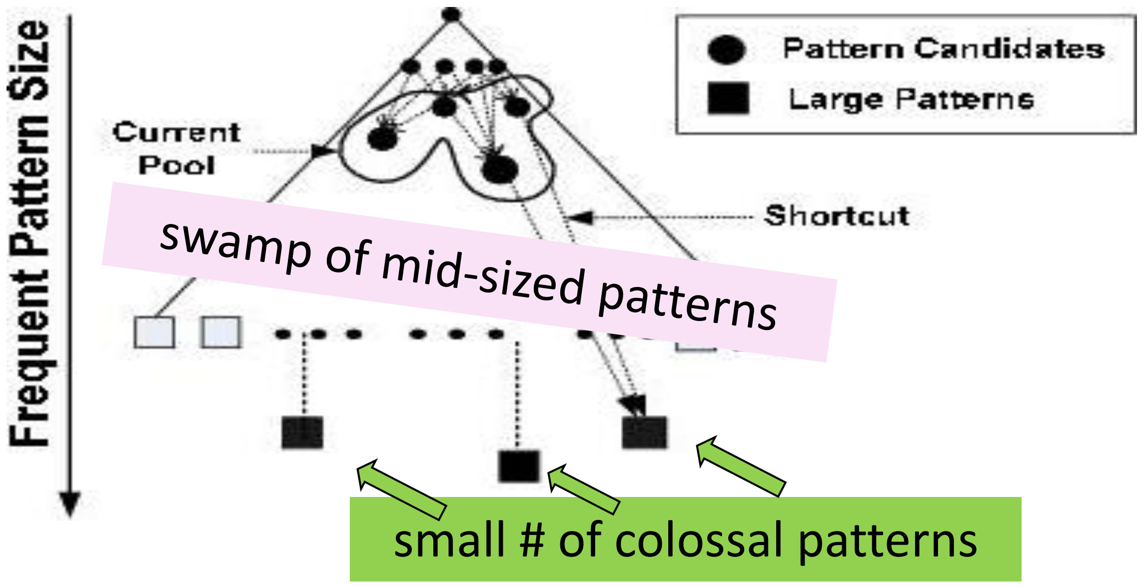
- Let min-support σ= 20
- # of closed/maximal patterns of size 20: About $\binom{40}{20}$
- But there is only one pattern with size close to 40 (*i.e., long* or *colossal*)
  - α= {41,42,…,79} of size 39
- Q: How to find it without generating an exponential number of size-20 patterns?

The existing fastest mining algorithms (*e.g.,* FPClose, LCM) fail to complete running

A new algorithm, *Pattern-Fusion*, outputs this colossal pattern in seconds

What Is Pattern-Fusion?

- Do not strive for completeness (why?)
- Jump out of the swamp of the mid-sized intermediate “results”
- Strive for mining almost complete and representative colossal patterns: Identify “short-cuts” and take “leaps”
- Key observation
  - The larger the pattern or the more distinct the pattern, the greater chance it will be generated from small ones
- Philosophy: Collection of small patterns hint at the larger patterns
- Pattern-fusion strategy (“not crawl but jump”): Fuse small patterns together in one step to generate new pattern candidates of significant sizes



Observation: Colossal Patterns and Core Patterns

- Suppose dataset D contains 4 colossal patterns (below) plus many small patterns
  - $\{a_1, a_2, \ldots, a_{50}\}$: 40, $\{a_3, a_6, \ldots, a_{99}\}$: 60, $\{a_5, a_{10}, \ldots, a_{95}\}$: 80, $\{a_{10}, a_{20}, \ldots, a_{100}\}$: 100
- If you check the pattern pool of size-3, you may likely find
  - $\{a_2, a_4, a_{45}\}$: ~40; $\{a_3, a_{34}, a_{39}\}$: ~40; …, $\{a_5, a_{15}, a_{85}\}$: ~80, …, $\{a_{20}, a_{40}, a_{85}\}$: ~80, …
- If you merge the patterns with similar support, you may obtain candidates of much bigger size and easily validate whether they are true patterns
- *Core patterns* of a colossal pattern α: A set of subpatterns of α that cluster around α by sharing a similar support
- A colossal pattern has far more core patterns than a small-sized pattern
- A random draw from a complete set of patterns of size c would be more likely to pick a core pattern (or its descendant) of a colossal pattern
- A colossal pattern can be generated by merging a set of core patterns

Robustness of Colossal Patterns

- Core patterns: For a frequent pattern α, a subpattern β is a τ-core pattern of α if β shares a similar support set with α, i.e.,

  $$\frac{|D_\alpha|}{|D_\beta|} \geq \tau \qquad 0 < \tau \leq 1$$ where τ is called the core ratio

- (d, τ)-robustness: A pattern α is *(d, τ)-robust* if *d* is the maximum number of items that can be removed from α for the resulting pattern to remain a τ-core pattern of α

- For a (d, τ)-robust pattern α, it has $\Omega(2^d)$ core patterns

- Robustness of colossal patterns: A colossal pattern tends to have many more core patterns than small patterns

- Such core patterns can be clustered together to form "dense balls" based on pattern distance defined by

  $$Dist(\alpha, \beta) = 1 - \frac{|D_\alpha \cap D_\beta|}{|D_\alpha \cup D_\beta|}$$

A random draw in the pattern space will hit somewhere in the ball with high probability

The Pattern-Fusion Algorithm

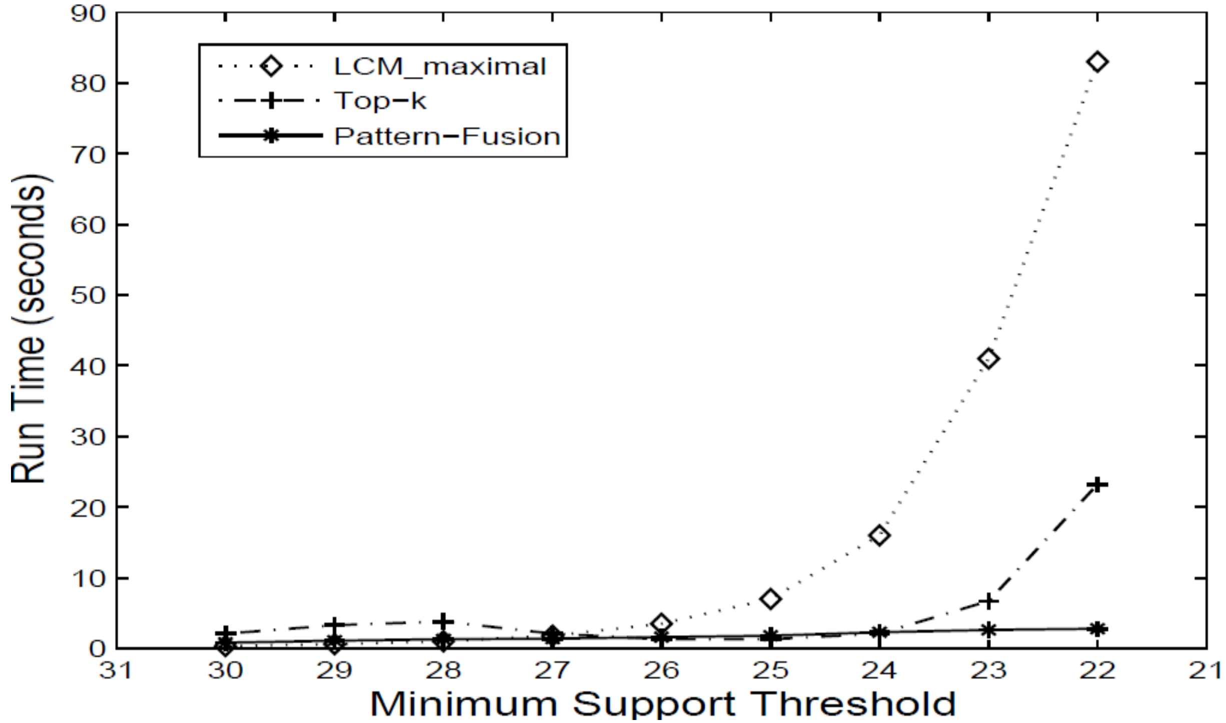
- Initialization (creating initial pool): Use an existing algorithm to mine all frequent patterns up to a small size, e.g., 3
- Iteration (iterative pattern fusion):
  - At each iteration, K seed patterns are randomly picked from the current pattern pool
  - For each seed pattern thus picked, we find all the patterns within a bounding ball centered at the seed pattern
  - All these patterns found are fused together to generate a set of super-patterns
  - All the super-patterns thus generated form a new pool for the next iteration
- Termination: When the current pool contains no more than K patterns at the beginning of an iteration

Experimental Results on Data Set: ALL

- ALL: A popular gene expression clinical data set on ALL-AML leukemia, with 38 transactions, each with 866 columns. There are 1736 items in total.
  - When minimum support is high (e.g., 30), Pattern-Fusion gets all the largest colossal patterns with size greater than 85

Pattern Size	110	107	102	91	86	84	83
The complete set	1	1	1	1	1	2	6
Pattern-Fusion	1	1	1	1	1	1	4
Pattern Size	**82**	**77**	**76**	**75**	**74**	**73**	**71**
The complete set	1	2	1	1	1	2	1
Pattern-Fusion	0	2	0	1	1	1	1

Mining colossal patterns on a leukemia dataset



Algorithm runtime comparison on another dataset

Appendix: Correction in Original Textbook and Helpful Slides for Mining Colossal Patterns

Note: Correction in the Original Textbook

- **Example 7.11 Core patterns.** Line 4 should be
  - Therefore, $|D_{\alpha 1}|/|D_{(ab)}| = 200/200 \geq \tau$
- **Figure 7.9. A transaction database, which contains duplicates, and core patterns for each distinct transactions.** The corrected table contents should be as follows:

Transaction (# of transactions)	Core Patterns (τ = 0.5)
(abe) (100)	(abe), (ab), (be), (ae), (a), (b), (e)
(bcf) (100)	(bcf), (bc), (bf), (cf), (b), (c), (f)
(acf) (100)	(acf), (ac), (af), (a), (c), (f)
(abcef) (100)	(ab), (ac), (ae), (af), (bc), (be), (bf), (ce), (ef), (e), (abc), (abf), (abe) (ace), (acf), (aef), (bcf), (bce), (bef) (cef), (abcf), (abce), (abef), (acef), (bcef), (abcef)