Data Mining 2025

Association Analysis: Basic Concepts and Algorithms

Dept. of Computer Science and Information Engineering

National Cheng Kung University

Kun-Ta Chuang

ktchuang@mail.ncku.edu.tw

Continuous and Categorical Attributes

How to apply association analysis formulation to non-asymmetric binary variables?

Session Id	Country	Session Length (sec)	Number of Web Pages viewed	Gender	Browser Type	Buy
1	USA	982	8	Male	IE	No
2	China	811	10	Female	Netscape	No
3	USA	2125	45	Female	Mozilla	Yes
4	Germany	596	4	Male	IE	Yes
5	Australia	123	9	Male	Mozilla	No
...	...	...	...	...	...	...

Example of Association Rule:

{Number of Pages $\in$[5,10) $\wedge$ (Browser=Mozilla)} $\rightarrow$ {Buy = No}

Handling Categorical Attributes

- Transform categorical attribute into asymmetric binary variables
- Introduce a new "item" for each distinct attribute-value pair
    - Example: replace Browser Type attribute with
        - Browser Type = Internet Explorer
        - Browser Type = Mozilla
        - Browser Type = Mozilla

Handling Categorical Attributes

- Potential Issues
  - What if attribute has many possible values
    - Example: attribute country has more than 200 possible values
    - Many of the attribute values may have very low support
      - Potential solution: Aggregate the low-support attribute values
  - What if distribution of attribute values is highly skewed
    - Example: 95% of the visitors have Buy = No
    - Most of the items will be associated with (Buy=No) item
      - Potential solution: drop the highly frequent items

Handling Continuous Attributes

Different kinds of rules:

- $Age \in [21,35) \land Salary \in [70k,120k) \rightarrow Buy$
- $Salary \in [70k,120k) \land Buy \rightarrow Age: \mu=28, \sigma=4$

Different methods:

- Discretization-based
- Statistics-based
- Non-discretization based
  - minApriori

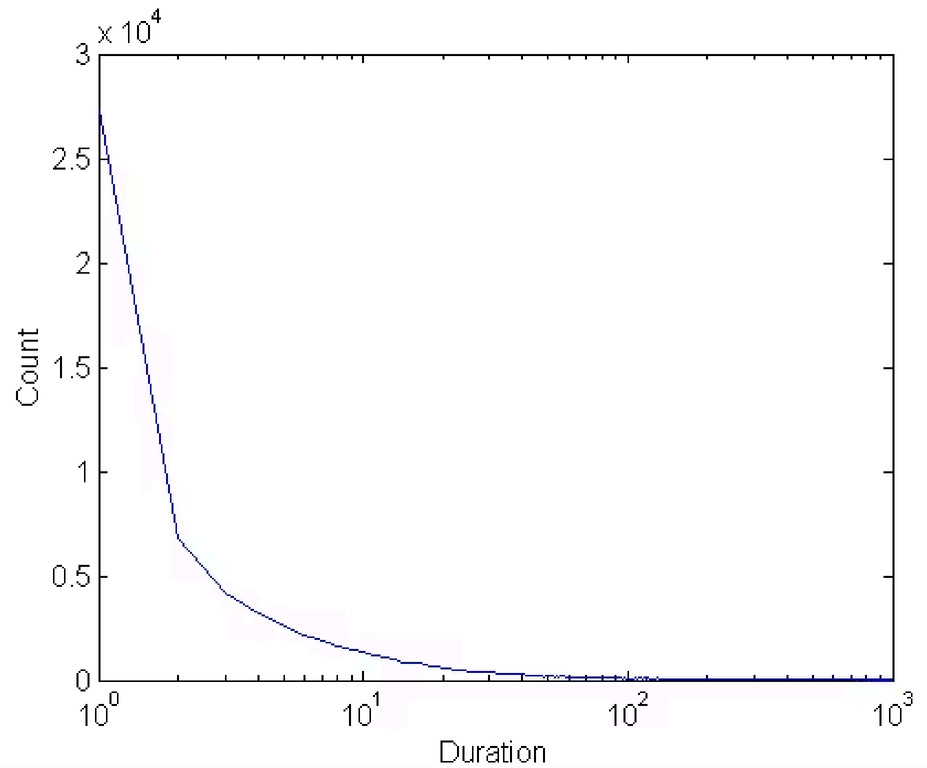
Handling Continuous Attributes

- Use discretization
- Unsupervised:
  - Equal-width binning
  - Equal-depth binning
  - Clustering
- Supervised:

Attribute values, v

Class	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
Anomalous	0	0	20	10	20	0	0	0	0
Normal	150	100	0	0	0	100	100	150	100

bin_1 bin_2 bin_3



Discretization Issues

- Size of the discretized intervals affect support & confidence

{Refund = No, (Income = $51,250)} → {Cheat = No}

{Refund = No, (60K ≤ Income ≤ 80K)} → {Cheat = No}

{Refund = No, (0K ≤ Income ≤ 1B)} → {Cheat = No}

  - If intervals too small
    - may not have enough support
  - If intervals too large
    - may not have enough confidence
- Potential solution: use all possible intervals

Discretization Issues

- Execution time

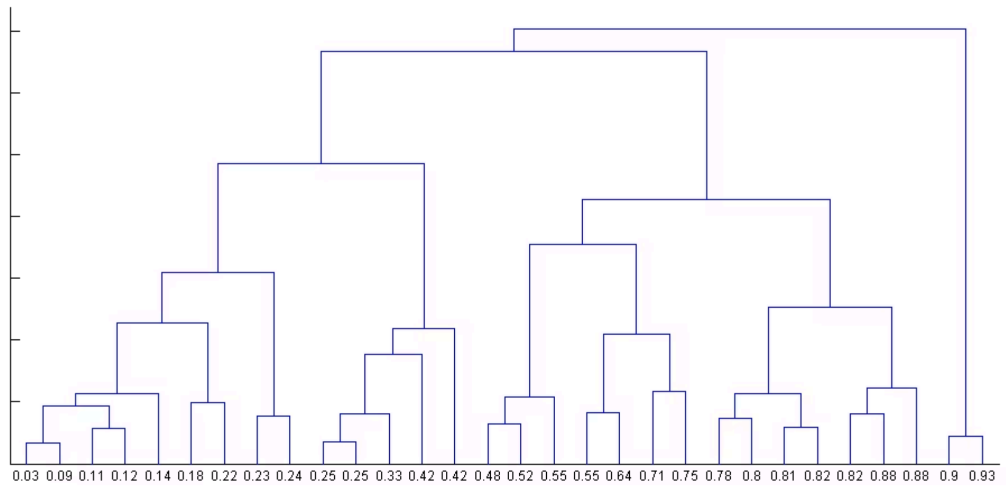
If intervals contain n values, there are on average O(n^2) possible ranges

- Too many rules

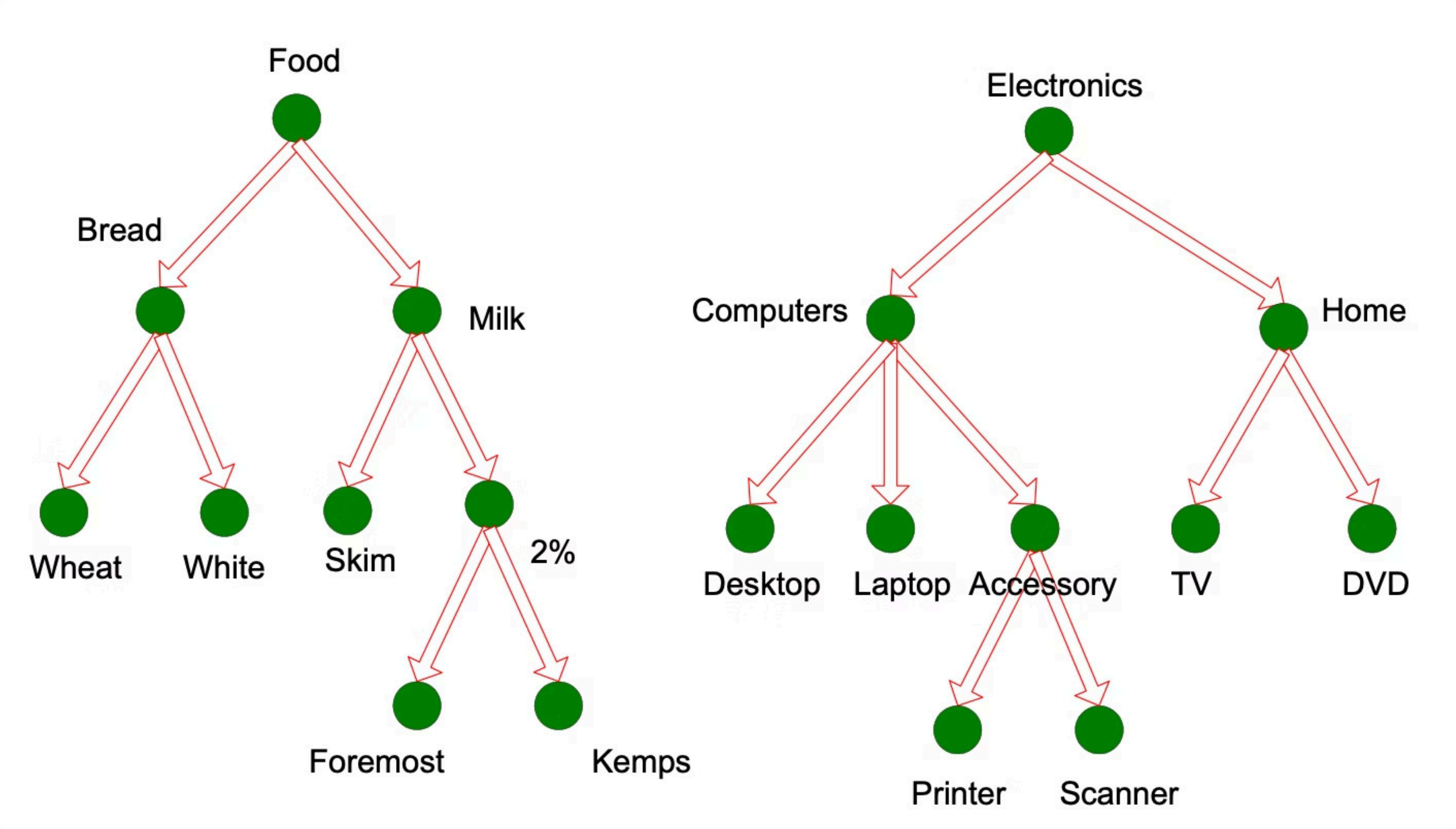
{Refund = No, (Income = $51,250)} → {Cheat = No}

{Refund = No, (51K ≤ Income ≤ 52K)} → {Cheat = No}

{Refund = No, (50K ≤ Income ≤ 60K)} → {Cheat = No}



Multi-level Association Rules



Multi-level Association Rules

- Why should we incorporate concept hierarchy?
  - Rules at lower levels may not have enough support to appear in any frequent itemsets
  - Rules at lower levels of the hierarchy are overly specific
    - e.g., skim milk → white bread, 2% milk → wheat bread, skim milk → wheat bread, etc. are indicative of association between milk and bread

Multi-level Association Rules

- How do support and confidence vary as we traverse the concept hierarchy?
  - If X is the parent item for both X1 and X2, then $\sigma(X) \leq \sigma(X1) + \sigma(X2)$
  - If $\sigma(X1 \cup Y1) \geq minsup$, and X is parent of X1, Y is parent of Y1 then $\sigma(X \cup Y1) \geq minsup$, $\sigma(X1 \cup Y) \geq minsup$ $\sigma(X \cup Y) \geq minsup$
  - If $conf(X1 \Rightarrow Y1) \geq minconf$, then $conf(X1 \Rightarrow Y) \geq minconf$

Multi-level Association Rules

Approach 1:

Extend current association rule formulation by augmenting each transaction with higher level items

Original Transaction: {skim milk, wheat bread}

Augmented Transaction: {skim milk, wheat bread, milk, bread, food}

Issues:

- Items that reside at higher levels have much higher support counts
  - if support threshold is low, too many frequent patterns involving items from the higher levels
- Increased dimensionality of the data

Multi-level Association Rules

Approach 2:

- Generate frequent patterns at highest level first
- Then, generate frequent patterns at the next highest level, and so on

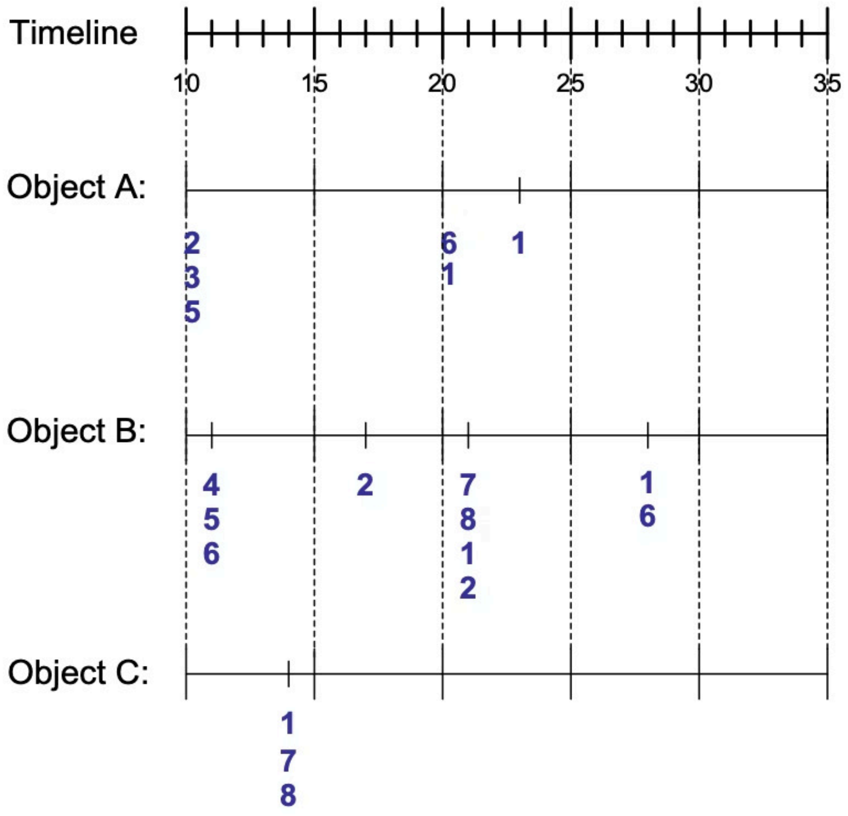
Issues:

- I/O requirements will increase dramatically because we need to perform more passes over the data
- May miss some potentially interesting cross-level association patterns

Sequence Data

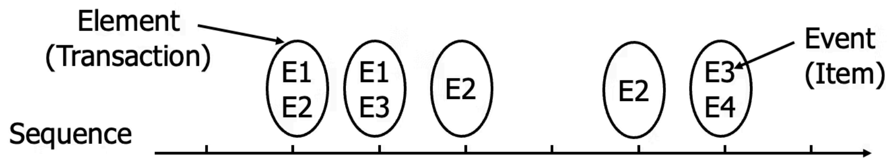
Sequence Database:

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7



Examples of Sequence Data

Sequence Database	Sequence	Element (Transaction)	Event (Item)
Customer	Purchase history of a given customer	A set of items bought by a customer at time t	Books, diary products, CDs, etc
Web Data	Browsing activity of a particular Web visitor	A collection of files viewed by a Web visitor after a single mouse click	Home page, index page, contact info, etc
Event data	History of events generated by a given sensor	Events triggered by a sensor at time t	Types of alarms generated by sensors
Genome sequences	DNA sequence of a particular species	An element of the DNA sequence	Bases A,T,G,C



Formal Definition of a Sequence

- A sequence is an ordered list of elements (transactions)

$$s = \langle e1\ e2\ e3\ \dots \rangle$$

  - Each element contains a collection of events (items)

$$ei = \{i1, i2, \dots, ik\}$$

  - Each element is attributed to a specific time or location
- Length of a sequence, |s|, is given by the number of elements of the sequence
- A k-sequence is a sequence that contains k events (items)

Examples of Sequence

- Web sequence:

  < {Homepage} {Electronics} {Digital Cameras} {Canon Digital Camera} {Shopping Cart} {Order Confirmation} {Return to Shopping} >

- Sequence of initiating events causing the nuclear accident at 3-mile Island:

  (http://stellar-one.com/nuclear/staff_reports/summary_SOE_the_initiating_event.htm)

  < {clogged resin} {outlet valve closure} {loss of feedwater} {condenser polisher outlet valve shut} {booster pumps trip} {main waterpump trips} {main turbine trips} {reactor pressure increases}>

- Sequence of books checked out at a library:

  <{Fellowship of the Ring} {The Two Towers} {Return of the King}>

Formal Definition of a Subsequence

A sequence <a1, a2 … an> is contained in another sequence <b1, b2 … bm> (m ≥ n) if there exist integers i1 < i2 < … < in such that a1 ⊆ bi1, a2 ⊆ bi_1, …, an ⊆ bi_n

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {8} >	< {2} {3,5} >	Yes
< {1,2} {3,4} >	< {1} {2} >	No
< {2,4} {2,4} {2,5} >	< {2} {4} >	Yes

- The support of a subsequence w is defined as the fraction of data sequences that contain w
- A *sequential pattern* is a frequent subsequence (i.e., a subsequence whose support is ≥ *minsup*)

Sequential Pattern Mining: Definition

Given:

- a database of sequences
- a user-specified minimum support threshold, *minsup*

Task:

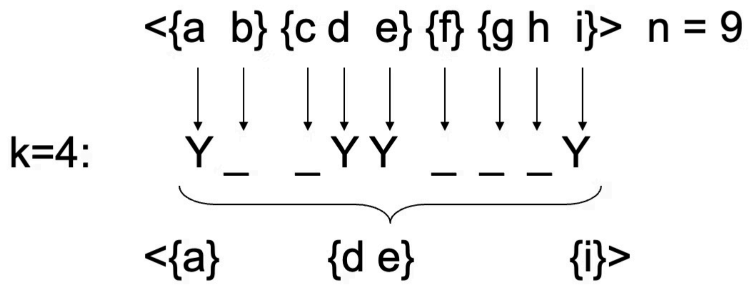
- Find all subsequences with support ≥ *minsup*

Sequential Pattern Mining: Challenge

- Given a sequence: <{a b} {c d e} {f} {g h i}>
- Examples of subsequences:

<{a} {c d} {f} {g} >, < {c d e} >, < {b} {g} >, etc.

- How many k-subsequences can be extracted from a given n-sequence?



Answer :

$$\binom{n}{k} = \binom{9}{4} = 126$$

Sequential Pattern Mining: Example

Object	Timestamp	Events
A	1	1,2,4
A	2	2,3
A	3	5
B	1	1,2
B	2	2,3,4
C	1	1, 2
C	2	2,3,4
C	3	2,4,5
D	1	2
D	2	3, 4
D	3	4, 5
E	1	1, 3
E	2	2, 4, 5

Minsup = 50%

Examples of Frequent Subsequences:

< {1,2} > s=60%
< {2,3} > s=60%
< {2,4}> s=80%
< {3} {5}> s=80%
< {1} {2} > s=80%
< {2} {2} > s=60%
< {1} {2,3} > s=60%
< {2} {2,3} > s=60%
< {1,2} {2,3} > s=60%

Extracting Sequential Patterns

- Given n events:

  i1, i2, i3, ..., in

- Candidate 1-subsequences:

  <{i1}>, <{i2}>, <{i3}>, ..., <{in}>

- Candidate 2-subsequences:

  <{i1, i2}>, <{i1, i3}>, ..., <{i1} {i1}>, <{i1} {i2}>, ..., <{in-1} {in}>

- Candidate 3-subsequences:

  <{i1, i2, i3}>, <{i1, i2, i4}>, ..., <{i1, i2} {i1}>, <{i1, i2} {i2}>, ...,

  <{i1} {i1, i2}>, <{i1} {i1, i3}>, ..., <{i1} {i1} {i1}>, <{i1} {i1} {i2}>, ...

Generalized Sequential Pattern (GSP)

Step 1:

Make the first pass over the sequence database D to yield all the 1-element frequent sequences

Step 2:

Repeat until no new frequent sequences are found

- **Candidate Generation**: Merge pairs of frequent subsequences found in the (k-1)*th* pass to generate candidate sequences that contain k items
- **Candidate Pruning**: Prune candidate *k*-sequences that contain infrequent (*k-1*)-subsequences
- **Support Counting**: Make a new pass over the sequence database D to find the support for these candidate sequences
- **Candidate Elimination**: Eliminate candidate *k*-sequences whose actual support is less than *minsup*

Candidate Generation

Base case (k=2):

Merging two frequent 1-sequences $<\{i_1\}>$ and $<\{i_2\}>$ will produce two candidate 2-sequences: $<\{i_1\} \{i_2\}>$ and $<\{i_1\ i_2\}>$

General case (k>2):

A frequent (*k-1*)-sequence w_1 is merged with another frequent (*k-1*)-sequence w_2 to produce a candidate *k*-sequence if the subsequence obtained by removing the first event in w_1 is the same as the subsequence obtained by removing the last event in w_2

- The resulting candidate after merging is given by the sequence w_1 extended with the last event of w_2.
- If the last two events in w_2 belong to the same element, then the last event in w_2 becomes part of the last element in w_1
- Otherwise, the last event in w_2 becomes a separate element appended to the end of w_1

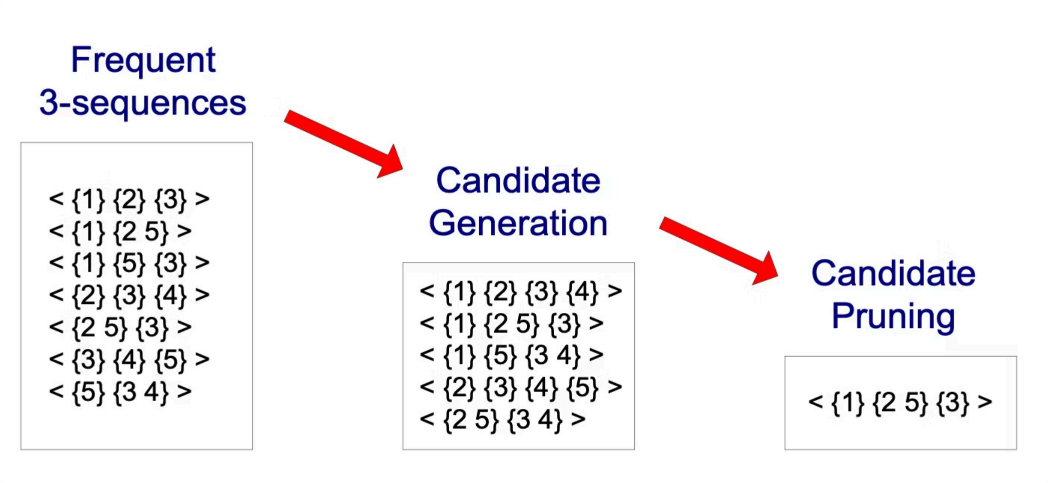
Candidate Generation Examples

- Merging the sequences w1=<{1} {2 3} {4}> and w2 =<{2 3} {4 5}> will produce the candidate sequence < {1} {2 3} {4 5}> because the last two events in w2 (4 and 5) belong to the same element

- Merging the sequences w1=<{1} {2 3} {4}> and w2 =<{2 3} {4} {5}> will produce the candidate sequence < {1} {2 3} {4} {5}> because the last two events in w2 (4 and 5) do not belong to the same element

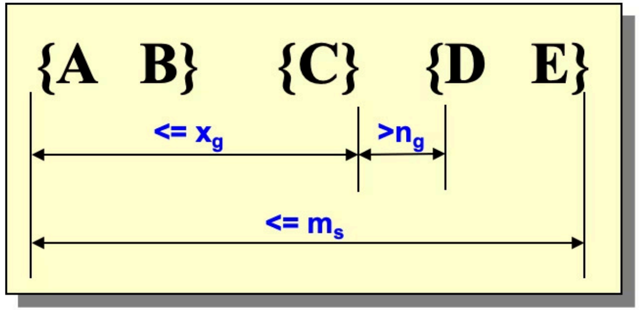
- We do not have to merge the sequences w1 =<{1} {2 6} {4}> and w2 =<{1} {2} {4 5}> to produce the candidate < {1} {2 6} {4 5}> because if the latter is a viable candidate, then it can be obtained by merging w1 with < {1} {2 6} {5}>

GSP Example



Frequent
3-sequences
< {1} {2} {3} >
< {1} {2 5} >
< {1} {5} {3} >
< {2} {3} {4} >
< {2 5} {3} >
< {3} {4} {5} >
< {5} {3 4} >
Candidate
Generation
< {1} {2} {3} {4} >
< {1} {2 5} {3} >
< {1} {5} {3 4} >
< {2} {3} {4} {5} >
< {2 5} {3 4} >
Candidate
Pruning
< {1} {2 5} {3} >

Timing Constraints (I)



x_g: max-gap

n_g: min-gap

m_s: maximum span

$x_g = 2$, $n_g = 0$, $m_s = 4$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {4,7} {4,5} {8} >	< {6} {5} >	Yes
< {1} {2} {3} {4} {5}>	< {1} {4} >	No
< {1} {2,3} {3,4} {4,5}>	< {2} {3} {5} >	Yes
< {1,2} {3} {2,3} {3,4} {2,4} {4,5}>	< {1,2} {5} >	No

Mining Sequential Patterns with Timing Constraints

Approach 1:

- Mine sequential patterns without timing constraints
- Postprocess the discovered patterns

Approach 2:

- Modify GSP to directly prune candidates that violate timing constraints
- Question: Does Apriori principle still hold?

Apriori Principle for Sequence Data

Object	Timestamp	Events
A	1	1,2,4
A	2	2,3
A	3	5
B	1	1,2
B	2	2,3,4
C	1	1, 2
C	2	2,3,4
C	3	2,4,5
D	1	2
D	2	3, 4
D	3	4, 5
E	1	1, 3
E	2	2, 4, 5

Suppose:

$x_g = 1$ (max-gap)

$n_g = 0$ (min-gap)

$m_s = 5$ (maximum span)

minsup = 60%

<{2} {5}> support = 40%

but

<{2} {3} {5}> support = 60%

Problem exists because of max-gap constraint

No such problem if max-gap is infinite

Contiguous Subsequences

- s is a contiguous subsequence of w = <e1>< e2>…< ek>

  if any of the following conditions hold:

1. s is obtained from w by deleting an item from either e1 or ek
2. s is obtained from w by deleting an item from any element ei that contains more than 2 items
3. s is a contiguous subsequence of s' and s' is a contiguous subsequence of w (recursive definition)

- Examples: s = < {1} {2} >
  - is a contiguous subsequence of < {1} {2 3}>, < {1 2} {2} {3}>, and < {3 4} {1 2} {2 3} {4} >
  - is not a contiguous subsequence of < {1} {3} {2}> and < {2} {1} {3} {2}>

Modified Candidate Pruning Step

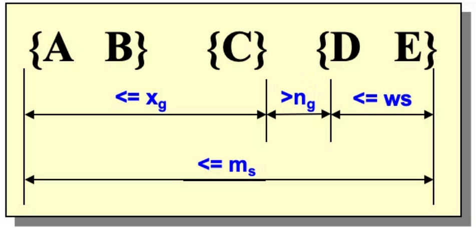
- Without maxgap constraint:

  A candidate k-sequence is pruned if at least one of its (k-1)-subsequences is infrequent

- With maxgap constraint:

  A candidate *k*-sequence is pruned if at least one of its **contiguous** (*k-1*)-subsequences is infrequent

Timing Constraints (II)



x_g: max-gap

n_g: min-gap

ws: window size

m_s: maximum span

x_g = 2, n_g = 0, ws = 1, m_s= 5

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {4,7} {4,6} {8} >	< {3} {5} >	No
< {1} {2} {3} {4} {5}>	< {1,2} {3} >	Yes
< {1,2} {2,3} {3,4} {4,5}>	< {1,2} {3,4} >	Yes

Modified Support Counting Step

- Given a candidate pattern: <{a, c}>
  - Any data sequences that contain

<... {a c} ... >,

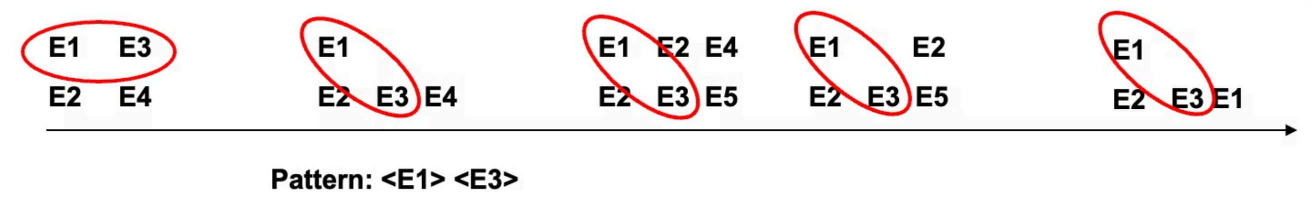
<... {a} ... {c}...> (where time({c}) – time({a}) ≤ ws)

<...{c} ... {a} ...> (where time({a}) – time({c}) ≤ ws)

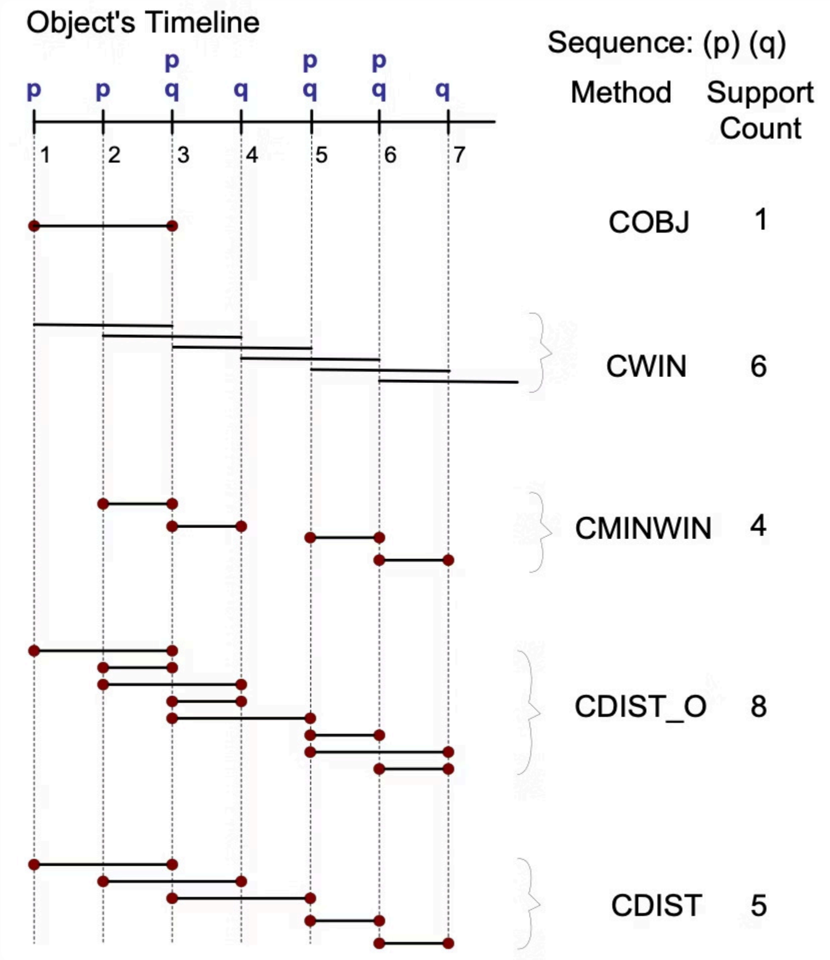
will contribute to the support count of candidate pattern

Other Formulation

- In some domains, we may have only one very long time series
  - Example:
    - monitoring network traffic events for attacks
    - monitoring telecommunication alarm signals
- Goal is to find frequent sequences of events in the time series
  - This problem is also known as frequent episode mining



General Support Counting Schemes



Assume:

$x_g = 2$ (max-gap)

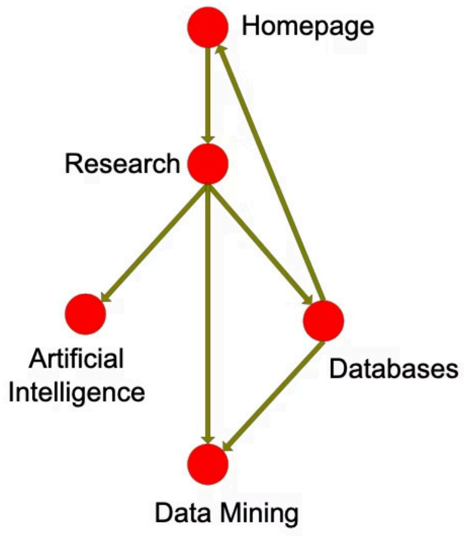
$n_g = 0$ (min-gap)

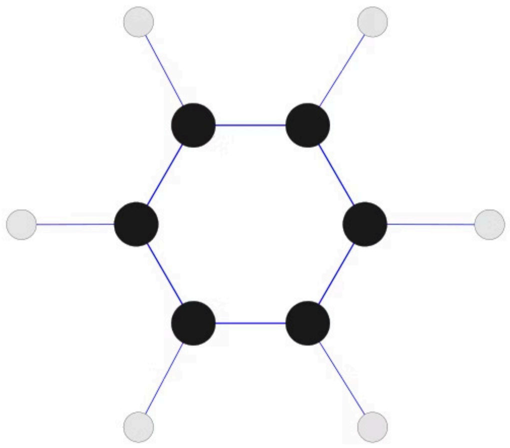
ws = 0 (window size)

$m_s = 2$ (maximum span)

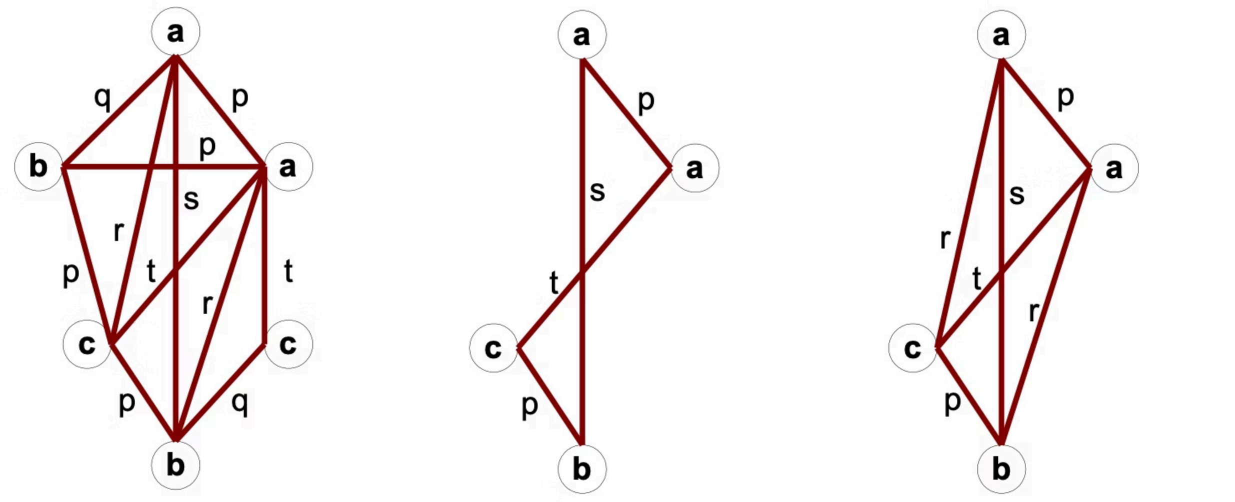
Frequent Subgraph Mining

- Extend association rule mining to finding frequent subgraphs
- Useful for Web Mining, computational chemistry, bioinformatics, spatial data sets, etc





Graph Definitions

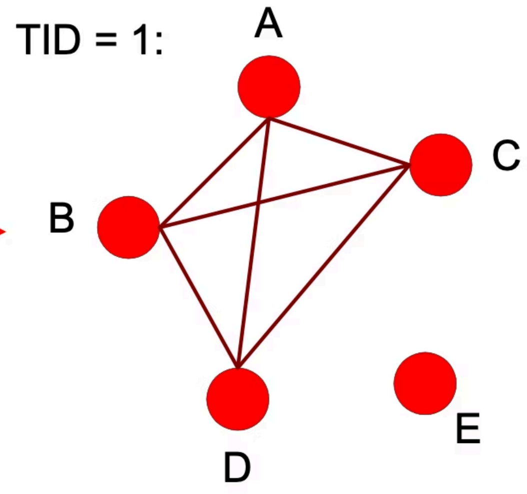


(a) Labeled Graph (b) Subgraph (c) Induced Subgraph

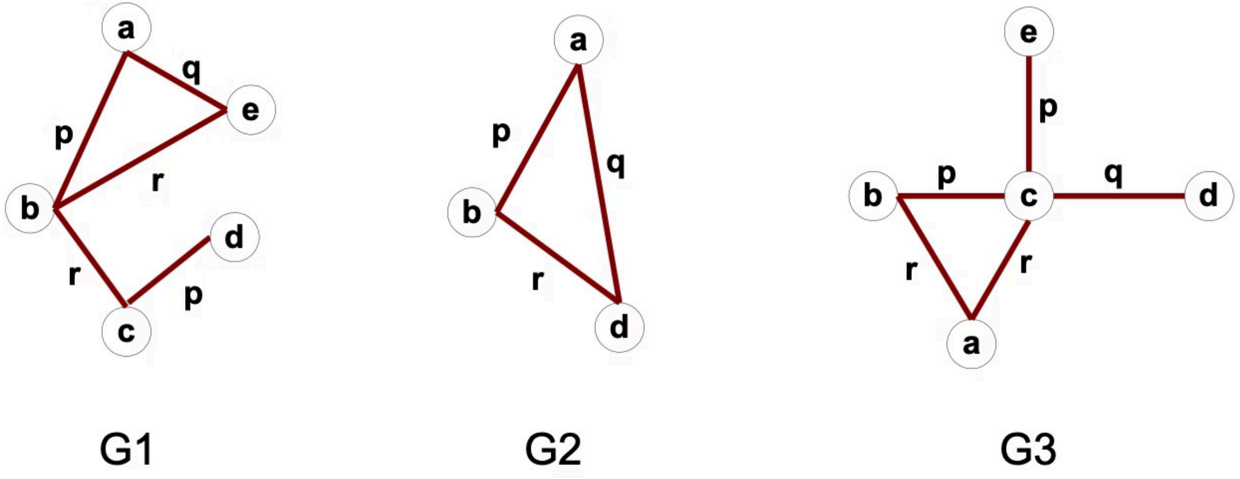
Representing Transactions as Graphs

- Each transaction is a clique of items

Transaction Id	Items
1	{A,B,C,D}
2	{A,B,E}
3	{B,C}
4	{A,B,D,E}
5	{B,C,D}



Representing Transactions as Graphs



	(a,b,p)	(a,b,q)	(a,b,r)	(b,c,p)	(b,c,q)	(b,c,r)	…	(d,e,r)
G1	1	0	0	0	0	1	…	0
G2	1	0	0	0	0	0	…	0
G3	0	0	1	1	0	0	…	0
G3	…	…	…	…	…	…	…	…

Challenges

Issues

- Node may contain duplicate labels
- Support and confidence

–How to define them?

- Additional constraints imposed by pattern structure

–Support and confidence are not the only constraints

–Assumption: frequent subgraphs must be connected

Approach

- Apriori-like approach:

–Use frequent k-subgraphs to generate frequent (k+1) subgraphs

- What is k?
- Support:

–number of graphs that contain a particular subgraph

–Apriori principle still holds

–Level-wise (Apriori-like) approach:

–Vertex growing:

k is the number of vertices

–Edge growing:

k is the number of edges