Data Mining

Chapter 6

Association Analysis: Advance Concepts

Introduction to Data Mining, 2nd Edition

by

Tan, Steinbach, Karpatne, Kumar

Data Mining
Association Analysis: Advanced Concepts

Extensions of Association Analysis to Continuous and Categorical Attributes and Multi-level Rules

Continuous and Categorical Attributes

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

Gender	···	Age	Annual Income	No of hours spent online per week	No of email accounts	Privacy Concern
Female	···	26	90K	20	4	Yes
Male	···	51	135K	10	2	No
Male	···	29	80K	10	3	Yes
Female	···	45	120K	15	3	Yes
Female	···	31	95K	20	5	Yes
Male	···	25	55K	25	5	Yes
Male	···	37	100K	10	1	No
Male	···	41	65K	8	2	No
Female	···	26	85K	12	1	No
···	···	···	···	···	···	···

Example of Association Rule:

{Gender=Male, Age $\in$ [21,30)} $\rightarrow$ {No of hours online $\geq$ 10}

Handling Categorical Attributes

- Example: Internet Usage Data

Gender	Level of Education	State	Computer at Home	Online Auction	Chat Online	Online Banking	Privacy Concerns
Female	Graduate	Illinois	Yes	Yes	Daily	Yes	Yes
Male	College	California	No	No	Never	No	No
Male	Graduate	Michigan	Yes	Yes	Monthly	Yes	Yes
Female	College	Virginia	No	Yes	Never	Yes	Yes
Female	Graduate	California	Yes	No	Never	No	Yes
Male	College	Minnesota	Yes	Yes	Weekly	Yes	Yes
Male	College	Alaska	Yes	Yes	Daily	Yes	No
Male	High School	Oregon	Yes	No	Never	No	No
Female	Graduate	Texas	No	No	Monthly	No	No
···	···	···	···	···	···	···	···

{Level of Education=Graduate, Online Banking=Yes}
$\rightarrow$ {Privacy Concerns = Yes}

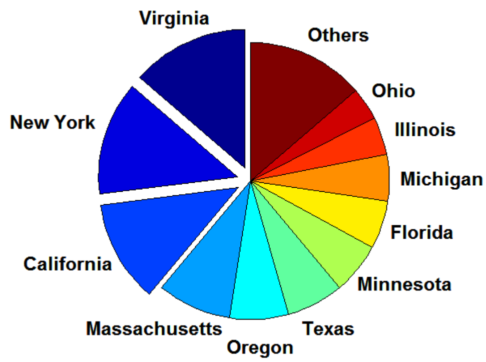
Handling Categorical Attributes

- Introduce a new "item" for each distinct attribute-value pair

Male	Female	Education = Graduate	Education = College	Education = High School	...	Privacy = Yes	Privacy = No
0	1	1	0	0	...	1	0
1	0	0	1	0	...	0	1
1	0	1	0	0	...	1	0
0	1	0	1	0	...	1	0
0	1	1	0	0	...	1	0
1	0	0	1	0	...	1	0
1	0	0	0	0	...	0	1
1	0	0	0	1	...	0	1
0	1	1	0	0	...	0	1
...	...	...	...	...	...	...	...

Handling Categorical Attributes

- Some attributes can have many possible values
  - Many of their attribute values have very low support
    - Potential solution: Aggregate the low-support attribute values



Handling Categorical Attributes

- Distribution of attribute values can be highly skewed
  - Example: 85% of survey participants own a computer at home
    - Most records have Computer at home = Yes
    - Computation becomes expensive; many frequent itemsets involving the binary item (Computer at home = Yes)
    - Potential solution:
      - discard the highly frequent items
      - Use alternative measures such as h-confidence
- Computational Complexity
  - Binarizing the data increases the number of items
  - But the width of the "transactions" remain the same as the number of original (non-binarized) attributes
  - Produce more frequent itemsets but maximum size of frequent itemset is limited to the number of original attributes

Handling Continuous Attributes

- Different methods:
  - Discretization-based
  - Statistics-based
  - Non-discretization based
    - minApriori
- Different kinds of rules can be produced:
  - {Age$\in$[21,30), No of hours online$\in$[10,20)} $\rightarrow$ {Chat Online =Yes}
  - {Age$\in$[15,30), Covid-Positive = Yes} $\rightarrow$ Full_recovery

Discretization-based Methods

Gender	···	Age	Annual Income	No of hours spent online per week	No of email accounts	Privacy Concern
Female	···	26	90K	20	4	Yes
Male	···	51	135K	10	2	No
Male	···	29	80K	10	3	Yes
Female	···	45	120K	15	3	Yes
Female	···	31	95K	20	5	Yes
Male	···	25	55K	25	5	Yes
Male	···	37	100K	10	1	No
Male	···	41	65K	8	2	No
Female	···	26	85K	12	1	No
···	···	···	···	···	···	···

Male	Female	··· ···	Age < 13	Age $\in [13, 21)$	Age $\in [21, 30)$	··· ···	Privacy $=$ Yes	Privacy $=$ No
0	1	···	0	0	1	···	1	0
1	0	···	0	0	0	···	0	1
1	0	···	0	0	1	···	1	0
0	1	···	0	0	0	···	1	0
0	1	···	0	0	0	···	1	0
1	0	···	0	0	1	···	1	0
1	0	···	0	0	0	···	0	1
1	0	···	0	0	0	···	0	1
0	1	···	0	0	1	···	0	1
···	···	···	···	···	···	···	···	···

Discretization-based Methods

- Unsupervised:
  - Equal-width binning <1 2 3> <4 5 6> <7 8 9>
  - Equal-depth binning <1 2 > <3 4 5 6 7 > < 8 9>
  - Cluster-based

- Supervised discretization

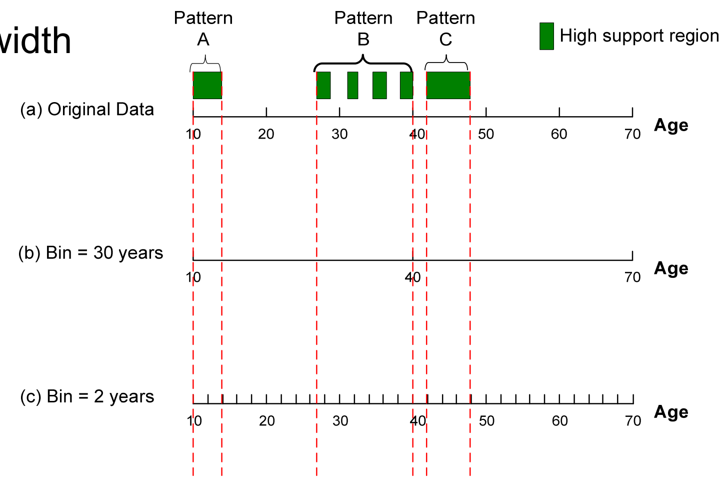
Continuous attribute, v

	1	2	3	4	5	6	7	8	9
Chat Online = Yes	0	0	20	10	20	0	0	0	0
Chat Online = No	150	100	0	0	0	100	100	150	100

bin_1 (1–2) bin_2 (3–5) bin_3 (6–9)

Discretization Issues

- Interval width



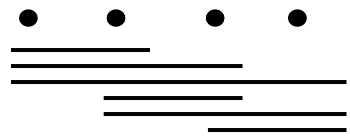
Pattern A: Age $\in [10, 15) \longrightarrow$ Chat Online = Never
Pattern B: Age $\in [26, 41) \longrightarrow$ Chat Online = Never
Pattern C: Age $\in [42, 48) \longrightarrow$ Online Banking = Yes

Discretization Issues

- Interval too wide (e.g., Bin size= 30)
  - May merge several disparate patterns
    - Patterns A and B are merged together
  - May lose some of the interesting patterns
    - Pattern C may not have enough confidence
- Interval too narrow (e.g., Bin size = 2)
  - Pattern A is broken up into two smaller patterns
    - Can recover the pattern by merging adjacent subpatterns
  - Pattern B is broken up into smaller patterns
    - Cannot recover the pattern by merging adjacent subpatterns
  - Some windows may not meet support threshold

Discretization: all possible intervals

Number of intervals = k
Total number of Adjacent intervals = k(k-1)/2



- Execution time
  - If the range is partitioned into k intervals, there are $O(k^2)$ new items
  - If an interval [a,b) is frequent, then all intervals that subsume [a,b) must also be frequent
    - E.g.: if {Age $\in$[21,25), Chat Online=Yes} is frequent,
      then {Age $\in$[10,50), Chat Online=Yes} is also frequent
  - Improve efficiency:
    - Use maximum support to avoid intervals that are too wide

Statistics-based Methods

- Example:

  {Income > 100K, Online Banking=Yes} $\rightarrow$ Age: μ=34
- Rule consequent consists of a continuous variable, characterized by their statistics
  - mean, median, standard deviation, etc.
- Approach:
  - Withhold the target attribute from the rest of the data
  - Extract frequent itemsets from the rest of the attributes
    - Binarize the continuous attributes (except for the target attribute)
  - For each frequent itemset, compute the corresponding descriptive statistics of the target attribute
    - Frequent itemset becomes a rule by introducing the target variable as rule consequent
  - Apply statistical test to determine interestingness of the rule

Statistics-based Methods

Gender	···	Age	Annual Income	No of hours spent online per week	No of email accounts	Privacy Concern
Female	···	26	90K	20	4	Yes
Male	···	51	135K	10	2	No
Male	···	29	80K	10	3	Yes
Female	···	45	120K	15	3	Yes
Female	···	31	95K	20	5	Yes
Male	···	25	55K	25	5	Yes
Male	···	37	100K	10	1	No
Male	···	41	65K	8	2	No
Female	···	26	85K	12	1	No
···	···	···	···	···	···	···

Frequent Itemsets:

{Male, Income > 100K}

{Income < 30K, No hours $\in$[10,15)}

{Income > 100K, Online Banking = Yes}

....

Association Rules:

{Male, Income > 100K} $\rightarrow$ Age: μ = 30

{Income < 40K, No hours $\in$[10,15)} $\rightarrow$ Age: μ = 24

{Income > 100K,Online Banking = Yes} $\rightarrow$ Age: μ = 34

....

Statistics-based Methods

- How to determine whether an association rule interesting?
  - Compare the statistics for segment of population covered by the rule vs segment of population not covered by the rule:

    $A \Rightarrow B: \mu$ versus $\bar{A} \Rightarrow B: \mu'$

  - Statistical hypothesis testing:

    $$Z = \frac{\mu' - \mu - \Delta}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

    - Null hypothesis: H0: $\mu' = \mu + \Delta$
    - Alternative hypothesis: H1: $\mu' > \mu + \Delta$
    - Z has zero mean and variance 1 under null hypothesis

Statistics-based Methods

- Example:

  r: Covid-Postive & Quick_Recovery=Yes → Age: μ=23

  – Rule is interesting if difference between μ and μ' is more than 5 years (i.e., Δ = 5)
  – For r, suppose n1 = 50, s1 = 3.5
  – For r' (complement): n2 = 250, s2 = 6.5

$$Z = \frac{\mu' - \mu - \Delta}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} = \frac{30 - 23 - 5}{\sqrt{\frac{3.5^2}{50} + \frac{6.5^2}{250}}} = 3.11$$

  – For 1-sided test at 95% confidence level, critical Z-value for rejecting null hypothesis is 1.64.
  – Since Z is greater than 1.64, r is an interesting rule

Min-Apriori

Document-term matrix:

TID	W1	W2	W3	W4	W5
D1	2	2	0	0	1
D2	0	0	1	2	2
D3	2	3	0	0	0
D4	0	0	1	0	1
D5	1	1	1	0	2

Example:

W1 and W2 tends to appear together in the same document

Min-Apriori

- Data contains only continuous attributes of the same "type"
  - e.g., frequency of words in a document

TID	W1	W2	W3	W4	W5
D1	2	2	0	0	1
D2	0	0	1	2	2
D3	2	3	0	0	0
D4	0	0	1	0	1
D5	1	1	1	0	2

- Potential solution:
  - Convert into 0/1 matrix and then apply existing algorithms
    - lose word frequency information
  - Discretization does not apply as users want association among words based on how frequently they co-occur, not if they occur with similar frequencies

Min-Apriori

- How to determine the support of a word?
  - If we simply sum up its frequency, support count will be greater than total number of documents!
    - Normalize the word vectors – e.g., using L_1 norms
    - Each word has a support equals to 1.0

TID	W1	W2	W3	W4	W5
D1	2	2	0	0	1
D2	0	0	1	2	2
D3	2	3	0	0	0
D4	0	0	1	0	1
D5	1	1	1	0	2

Normalize →

TID	W1	W2	W3	W4	W5
D1	0.40	0.33	0.00	0.00	0.17
D2	0.00	0.00	0.33	1.00	0.33
D3	0.40	0.50	0.00	0.00	0.00
D4	0.00	0.00	0.33	0.00	0.17
D5	0.20	0.17	0.33	0.00	0.33

Min-Apriori

- New definition of support:

$$\sup(C) = \sum_{i \in T} \min_{j \in C} D(i, j)$$

TID	W1	W2	W3	W4	W5
D1	0.40	0.33	0.00	0.00	0.17
D2	0.00	0.00	0.33	1.00	0.33
D3	0.40	0.50	0.00	0.00	0.00
D4	0.00	0.00	0.33	0.00	0.17
D5	0.20	0.17	0.33	0.00	0.33

Example:

Sup(W1,W2)

= .33 + 0 + .4 + 0 + 0.17

= 0.9

Anti-monotone property of Support

TID	W1	W2	W3	W4	W5
D1	0.40	0.33	0.00	0.00	0.17
D2	0.00	0.00	0.33	1.00	0.33
D3	0.40	0.50	0.00	0.00	0.00
D4	0.00	0.00	0.33	0.00	0.17
D5	0.20	0.17	0.33	0.00	0.33

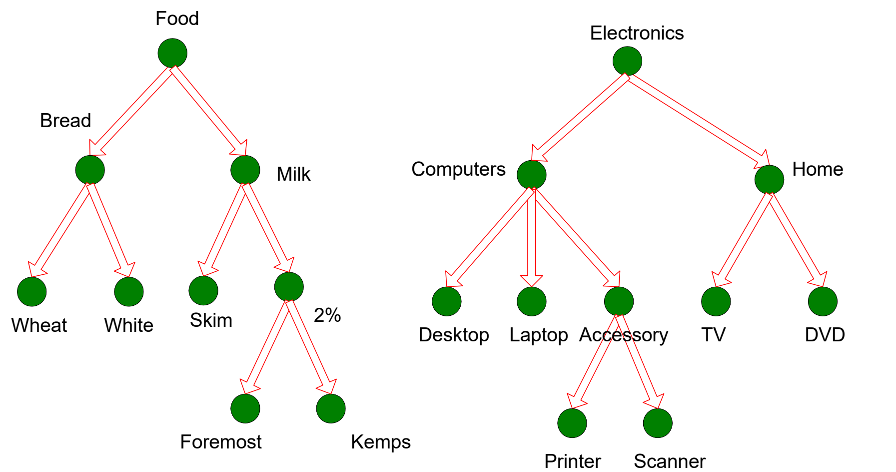
Example:

Sup(W1) = 0.4 + 0 + 0.4 + 0 + 0.2 = 1

Sup(W1, W2) = 0.33 + 0 + 0.4 + 0 + 0.17 = 0.9

Sup(W1, W2, W3) = 0 + 0 + 0 + 0 + 0.17 = 0.17

Concept Hierarchies

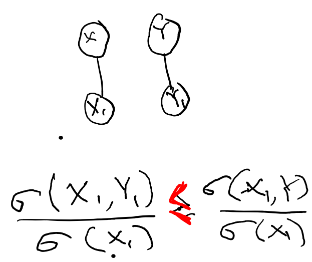


Multi-level Association Rules

- 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., following rules are indicative of association between milk and bread
      - skim milk → white bread,
      - 2% milk → wheat bread,
      - skim milk → wheat bread, etc.
  - Rules at higher level of hierarchy may be too generic
    - e.g., electronics → food

Multi-level Association Rules

- How do support and confidence vary as we traverse the concept hierarchy?
  - 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

Data Mining
Association Analysis: Advanced Concepts

Sequential Patterns

Examples of Sequence

- Sequence of different transactions by a customer at an online store:

  < {Digital Camera,iPad} {memory card} {headphone,iPad cover} >

- 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}>

Sequential Pattern Discovery: Examples

- In telecommunications alarm logs,
  - Inverter_Problem:

  (Excessive_Line_Current) (Rectifier_Alarm) --> (Fire_Alarm)

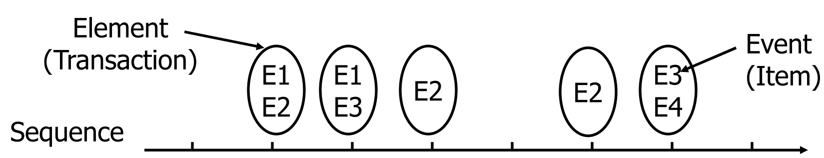
- In point-of-sale transaction sequences,
  - Computer Bookstore:

    (Intro_To_Visual_C) (C++_Primer) -->
    (Perl_for_dummies,Tcl_Tk)
  - Athletic Apparel Store:

    (Shoes) (Racket, Racketball) --> (Sports_Jacket)

Sequence Data

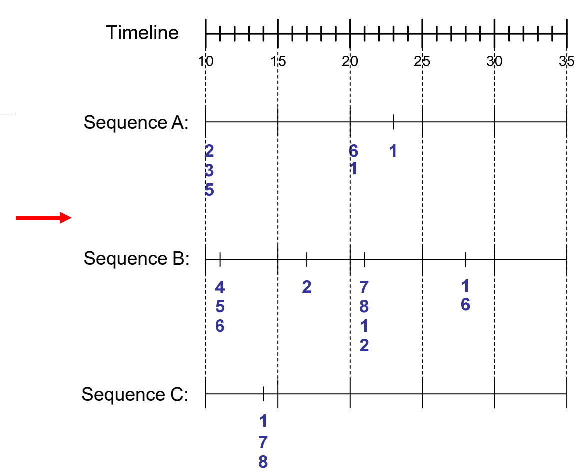
Sequence Database	Sequence	Element (Transaction)	Event (Item)
Customer	Purchase history of a given customer	A set of items bought by a customer at time t	Books, diary products, CDs, etc
Web Data	Browsing activity of a particular Web visitor	A collection of files viewed by a Web visitor after a single 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



Sequence Data

Sequence Database:

Sequence ID	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



Sequence Data vs. Market-basket Data

Sequence Database:

Customer	Date	Items bought
A	10	2, 3, 5
A	20	1,6
A	23	1
B	11	4, 5, 6
B	17	2
B	21	1,2,7,8
B	28	1, 6
C	14	1,7,8

Market- basket Data

Events
2, 3, 5
1,6
1
4,5,6
2
1,2,7,8
1,6
1,7,8

Sequence Data vs. Market-basket Data

Sequence Database:

Customer	Date	Items bought
A	10	2, 3, 5
A	20	1,6
A	23	1
B	11	4, 5, 6
B	17	2
B	21	1,2,7,8
B	28	1, 6
C	14	1,7,8

Market- basket Data

Events
2, 3, 5
1,6
1
4,5,6
2
1,2,7,8
1,6
1,7,8

Formal Definition of a Sequence

- A sequence is an ordered list of elements

  $s = < e_1 \, e_2 \, e_3 \ldots >$

  - Each element contains a collection of events (items)

    $e_i = \{i_1, i_2, \ldots, i_k\}$

- Length of a sequence, |s|, is given by the number of elements in the sequence
- A k-sequence is a sequence that contains k events (items)
  - <{a,b} {a}> has a length of 2 and it is a 3-sequence

Formal Definition of a Subsequence

- A sequence t: $<a_1 \, a_2 \ldots a_n>$ **is contained** in another sequence s: $<b_1 \, b_2 \ldots b_m>$ ($m \geq n$) if there exist integers $i_1 < i_2 < \ldots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, …, $a_n \subseteq b_{in}$
- Illustrative Example:

s:	b_1	b_2	b_3	b_4	b_5
t:		a_1	a_2		a_3

t **is a subsequence** of s if $a_1 \subseteq b_2$, $a_2 \subseteq b_3$, $a_3 \subseteq b_5$.

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

Sequential Pattern Mining: Definition

- 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*)

- Given:
  - a database of sequences
  - a user-specified minimum support threshold, *minsup*
- Task:
  - Find all subsequences with support ≥ *minsup*

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%

Sequence Data vs. Market-basket Data

Sequence Database:

Customer	Date	Items bought
A	10	2, 3, 5
A	20	1,6
A	23	1
B	11	4, 5, 6
B	17	2
B	21	1,2,7,8
B	28	1, 6
C	14	1,7,8

Market- basket Data

Events
2, 3, 5
1,6
1
4,5,6
2
1,2,7,8
1,6
1,7,8

{2} -> {1}

$$conf(\{2\} \rightarrow \{1\}) = \frac{\sigma(\{2\}\,\{1\})}{\sigma(\{2\})}$$

(1,8) -> (7)

$$conf(1,8) \rightarrow (7)) = \frac{\sigma(1,7,8)}{\sigma(\{1,8\})}$$

Extracting Sequential Patterns

- Given n events: $i_1, i_2, i_3, \ldots, i_n$
- Candidate 1-subsequences:
  $<\{i_1\}>, <\{i_2\}>, <\{i_3\}>, \ldots, <\{i_n\}>$
- Candidate 2-subsequences:
  $<\{i_1, i_2\}>, <\{i_1, i_3\}>, \ldots,$
  $<\{i_1\}\ \{i_1\}>, <\{i_1\}\ \{i_2\}>, \ldots, <\{i_n\}\ \{i_n\}>$
- Candidate 3-subsequences:
  $<\{i_1, i_2, i_3\}>, <\{i_1, i_2, i_4\}>, \ldots,$
  $<\{i_1, i_2\}\ \{i_1\}>, <\{i_1, i_2\}\ \{i_2\}>, \ldots,$
  $<\{i_1\}\ \{i_1, i_2\}>, <\{i_1\}\ \{i_1, i_3\}>, \ldots,$
  $<\{i_1\}\ \{i_1\}\ \{i_1\}>, <\{i_1\}\ \{i_1\}\ \{i_2\}>, \ldots$

Extracting Sequential Patterns: Simple example

- Given 2 events: a, b
- Candidate 1-subsequences:

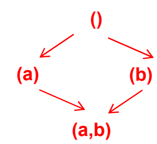
  <{a}>, <{b}>.

- Candidate 2-subsequences:

  <{a} {a}>, <{a} {b}>, <{b} {a}>, <{b} {b}>, <{a, b}>.

- Candidate 3-subsequences:

  <{a} {a} {a}>, <{a} {a} {b}>, <{a} {b} {a}>, <{a} {b} {b}>,
  <{b} {b} {b}>, <{b} {b} {a}>, <{b} {a} {b}>, <{b} {a} {a}>
  <{a, b} {a}>, <{a, b} {b}>, <{a} {a, b}>, <{b} {a, b}>



Item-set patterns

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 the following candidate 2-sequences: <{i_1} {i_1}>, <{i_1} {i_2}>, <{i_2} {i_2}>, <{i_2} {i_1}> and <{i_1, i_2}>. (**Note**: <{i_1}> can be merged with itself to produce: <{i_1} {i_1}>)
- 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 an event from the first element in w_1 is the same as the subsequence obtained by removing an event from the last element in w_2

Candidate Generation

- Base case (k=2):
  - Merging two frequent 1-sequences <{i_1}> and <{i_2}> will produce the following candidate 2-sequences: <{i_1} {i_1}>, <{i_1} {i_2}>, <{i_2} {i_2}>, <{i_2} {i_1}> and <{i_1 i_2}>. (**Note**: <{i_1}> can be merged with itself to produce: <{i_1} {i_1}>)
- 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 an event from the first element in w_1 is the same as the subsequence obtained by removing an event from the last element in w_2
    - The resulting candidate after merging is given by extending the sequence w_1 as follows-
      - If the last element of w_2 has only one event, append it to w_1
      - Otherwise add the event from the last element of w_2 (which is absent in the last element of w_1) to the last element of w_1

Candidate Generation Examples

- Merging w_1=<{1 2 3} {4 6}> and w_2 =<{2 3} {4 6} {5}>
  produces the candidate sequence < {1 2 3} {4 6} {5}> because the last element of w_2 has only one event
- Merging w_1=<{1} {2 3} {4}> and w_2 =<{2 3} {4 5}>
  produces the candidate sequence < {1} {2 3} {4 5}> because the last element in w_2 has more than one event
- Merging w_1=<{1 2 3} > and w_2 =<{2 3 4} >
  produces the candidate sequence < {1 2 3 4}> because the last element in w_2 has more than one event
- We do not have to merge the sequences
  w_1 =<{1} {2 6} {4}> and w_2 =<{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 w_1 with
  < {2 6} {4 5}>

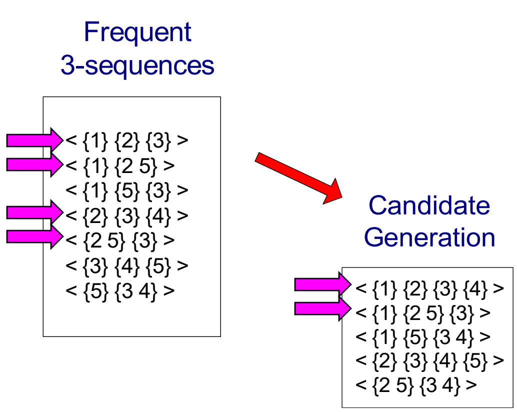
Candidate Generation: Examples (ctd)

- Can <{a},{b},{c}> merge with <{b},{c},{f}> ?
- Can <{a},{b},{c}> merge with <{b,c},{f}>?
- Can <{a},{b},{c}> merge with <{b},{c,f}>?
- Can <{a,b},{c}> merge with <{b},{c,f}> ?
- Can <{a,b,c}> merge with <{b,c,f}>?
- Can <{a}> merge with <{a}>?

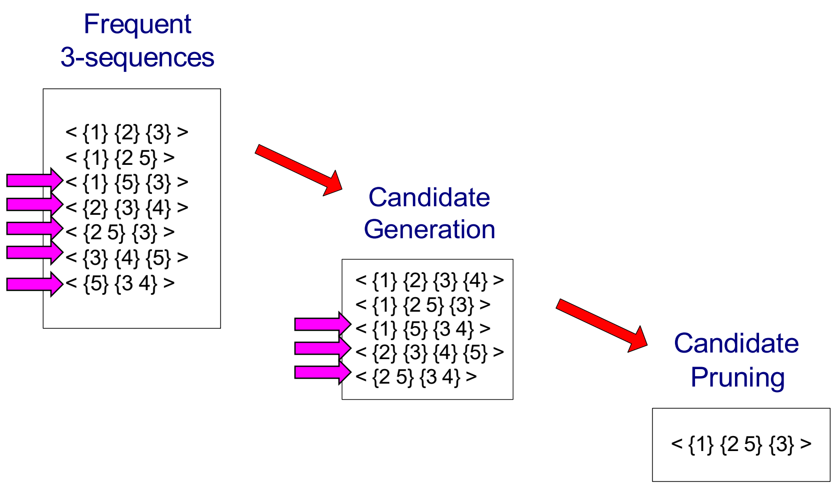
Candidate Generation: Examples (ctd)

- <{a},{b},{c}> can be merged with <{b},{c},{f}> to produce <{a},{b},{c},{f}>
- <{a},{b},{c}> cannot be merged with <{b,c},{f}>
- <{a},{b},{c}> can be merged with <{b},{c,f}> to produce <{a},{b},{c,f}>
- <{a,b},{c}> can be merged with <{b},{c,f}> to produce <{a,b},{c,f}>
- <{a,b,c}> can be merged with <{b,c,f}> to produce <{a,b,c,f}>
- <{a}{b}{a}> can be merged with <{b}{a}{b}> to produce <{a},{b},{a},{b}>
- <{b}{a}{b}> can be merged with <{a}{b}{a}> to produce <{b},{a},{b},{a}>

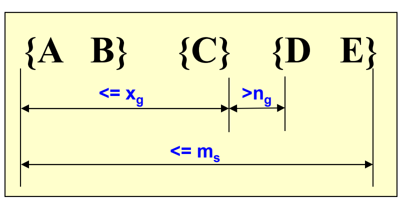
GSP Example



GSP Example



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, d	Sequential Pattern, s	d contains s?
< {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 = $<e_1>< e_2>\ldots< e_k>$

  if any of the following conditions hold:

  1. s is obtained from w by deleting an item from either e_1 or e_k
  2. s is obtained from w by deleting an item from any element e_i that contains at least 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

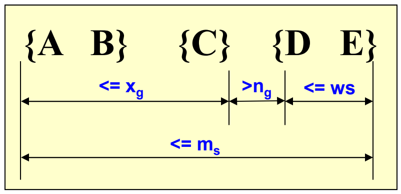
    < {1} {3} {2}> and < {2} {1} {3} {2}>

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, d	Sequential Pattern, s	d contains s?
< {2,4} {3,5,6} {4,7} {4,5} {8} >	< {3,4,5}>	Yes
< {1} {2} {3} {4} {5}>	< {1,2} {3,4} >	No
< {1,2} {2,3} {3,4} {4,5}>	< {1,2} {3,4} >	Yes

Modified Support Counting Step

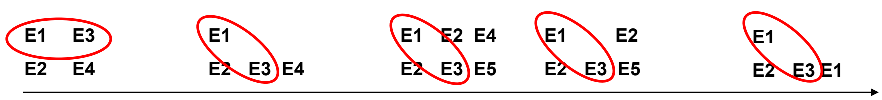
- Given a candidate sequential pattern: <{a, c}>
  - Any data sequences that contain

    <… {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

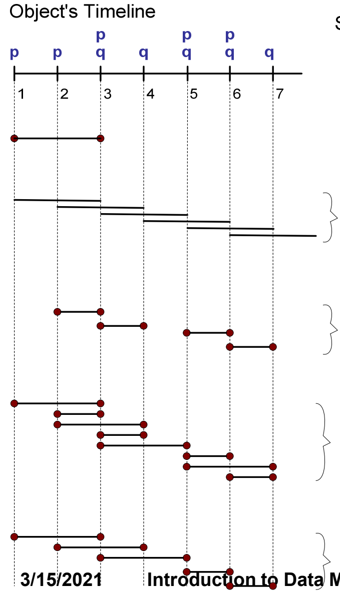
- In some domains, we may have only one very long time series
  - Example:
    - monitoring network traffic events for attacks
    - monitoring telecommunication alarm signals
- Goal is to find frequent sequences of events in the time series
  - This problem is also known as frequent episode mining



Pattern: <E1> <E3>

General Support Counting Schemes

Object's Timeline



Sequence: (p) (q)

Method	Support Count
COBJ	1
CWIN	6
CMINWIN	4
CDIST_O	8
CDIST	5

Assume:

x_g = 2 (max-gap)

n_g = 0 (min-gap)

ws = 0 (window size)

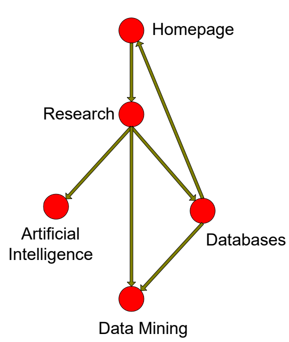
m_s = 2 (maximum span)

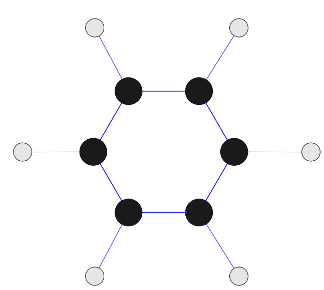
Data Mining
Association Analysis: Advanced Concepts

Subgraph Mining

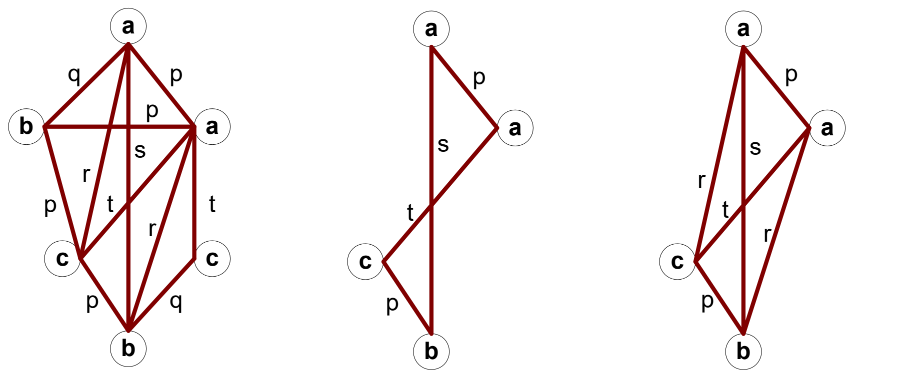
Frequent Subgraph Mining

- Extends association analysis to finding frequent subgraphs
- Useful for Web Mining, computational chemistry, bioinformatics, spatial data sets, etc





Graph Definitions

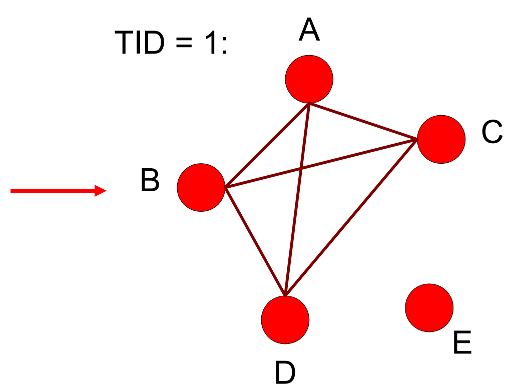


(a) Labeled Graph (b) Subgraph (c) Induced Subgraph

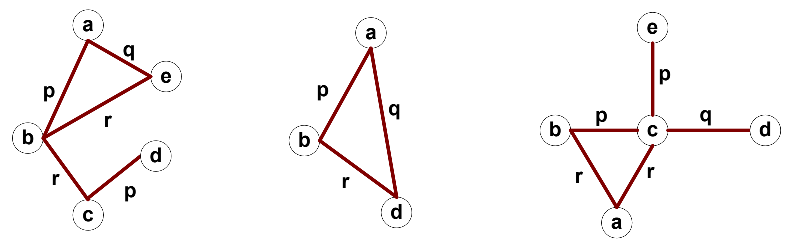
Representing Transactions as Graphs

- 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 Graphs as Transactions



G1 G2 G3

	(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	…	…	…	…	…	…	…	…

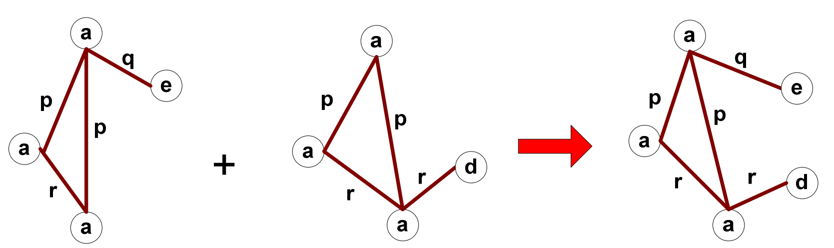
Challenges

- 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
- Apriori-like approach:
  - Use frequent k-subgraphs to generate frequent (k+1) subgraphs
    - What is k?

Challenges…

- 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

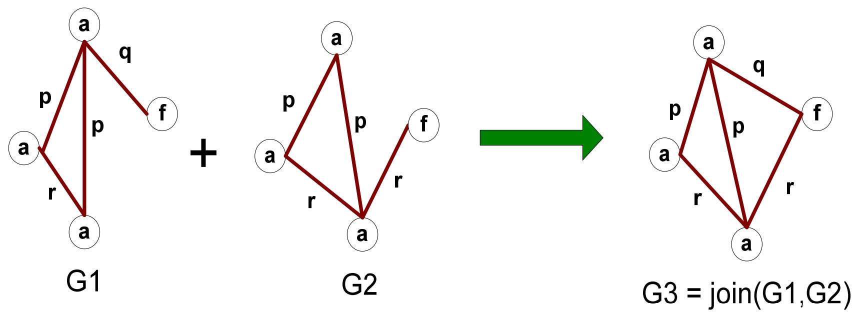
Vertex Growing



G1 G2 G3 = join(G1,G2)

$$M_{G1} = \begin{pmatrix} 0 & p & p & q \\ p & 0 & r & 0 \\ p & r & 0 & 0 \\ q & 0 & 0 & 0 \end{pmatrix} \quad M_{G2} = \begin{pmatrix} 0 & p & p & 0 \\ p & 0 & r & 0 \\ p & r & 0 & r \\ 0 & 0 & r & 0 \end{pmatrix} \quad M_{G3} = \begin{pmatrix} 0 & p & p & q & 0 \\ p & 0 & r & 0 & 0 \\ p & r & 0 & 0 & r \\ q & 0 & 0 & 0 & ? \\ 0 & 0 & r & ? & 0 \end{pmatrix}$$

Edge Growing

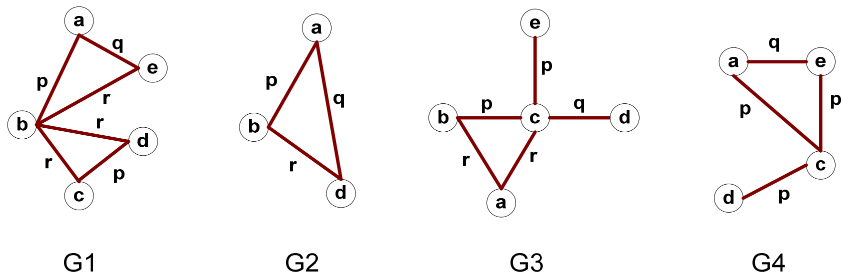


Apriori-like Algorithm

- Find frequent 1-subgraphs
- Repeat
  - Candidate generation
    - Use frequent (*k-1*)-subgraphs to generate candidate *k*-subgraph
  - Candidate pruning
    - Prune candidate subgraphs that contain infrequent (*k-1*)-subgraphs
  - Support counting
    - Count the support of each remaining candidate
  - Eliminate candidate *k*-subgraphs that are infrequent

In practice, it is not as easy. There are many other issues

Example: Dataset

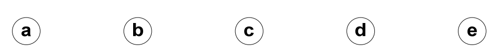


	(a,b,p)	(a,b,q)	(a,b,r)	(b,c,p)	(b,c,q)	(b,c,r)	…	(d,e,r)
G1	1	0	0	0	0	1	…	0
G2	1	0	0	0	0	0	…	0
G3	0	0	1	1	0	0	…	0
G4	0	0	0	0	0	0	…	0

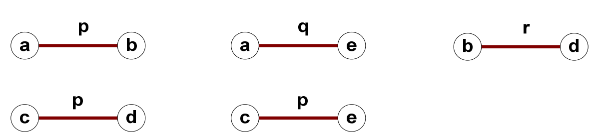
Example

Minimum support count = 2

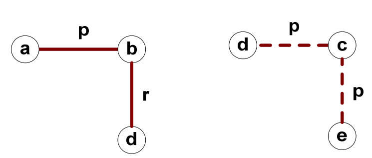
k=1
Frequent
Subgraphs



k=2
Frequent
Subgraphs



k=3
Candidate
Subgraphs

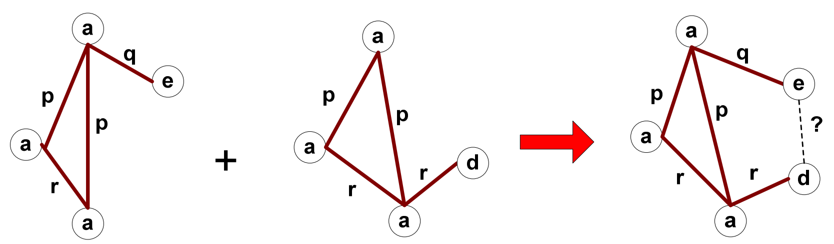


(Pruned candidate due to low support)

Candidate Generation

- In Apriori:
  - Merging two frequent *k*-itemsets will produce a candidate (*k+1*)-itemset
- In frequent subgraph mining (vertex/edge growing)
  - Merging two frequent *k*-subgraphs may produce more than one candidate (*k+1*)-subgraph

Multiplicity of Candidates (Vertex Growing)

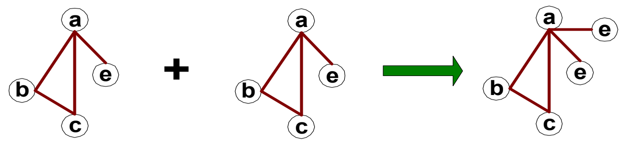


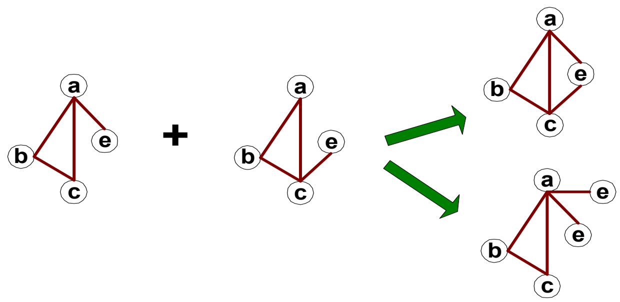
G1 G2 G3 = join(G1,G2)

$$M_{G1} = \begin{pmatrix} 0 & p & p & q \\ p & 0 & r & 0 \\ p & r & 0 & 0 \\ q & 0 & 0 & 0 \end{pmatrix} \quad M_{G2} = \begin{pmatrix} 0 & p & p & 0 \\ p & 0 & r & 0 \\ p & r & 0 & r \\ 0 & 0 & r & 0 \end{pmatrix} \quad M_{G3} = \begin{pmatrix} 0 & p & p & 0 & q \\ p & 0 & r & 0 & 0 \\ p & r & 0 & r & 0 \\ 0 & 0 & r & 0 & ? \\ q & 0 & 0 & ? & 0 \end{pmatrix}$$

Multiplicity of Candidates (Edge growing)

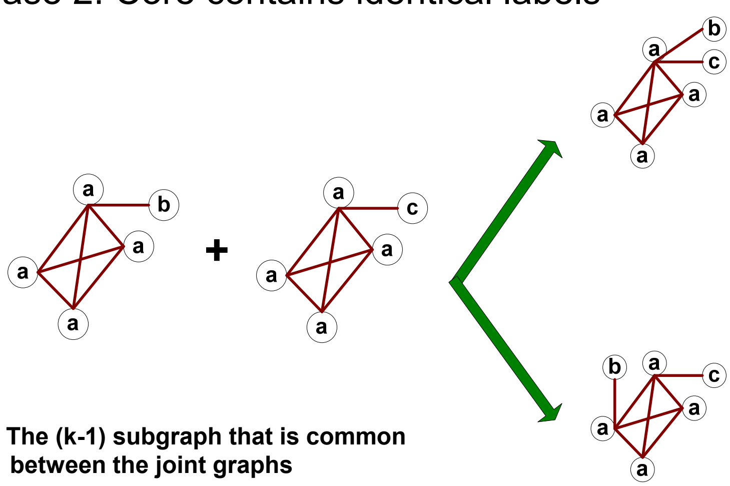
- Case 1: identical vertex labels





Multiplicity of Candidates (Edge growing)

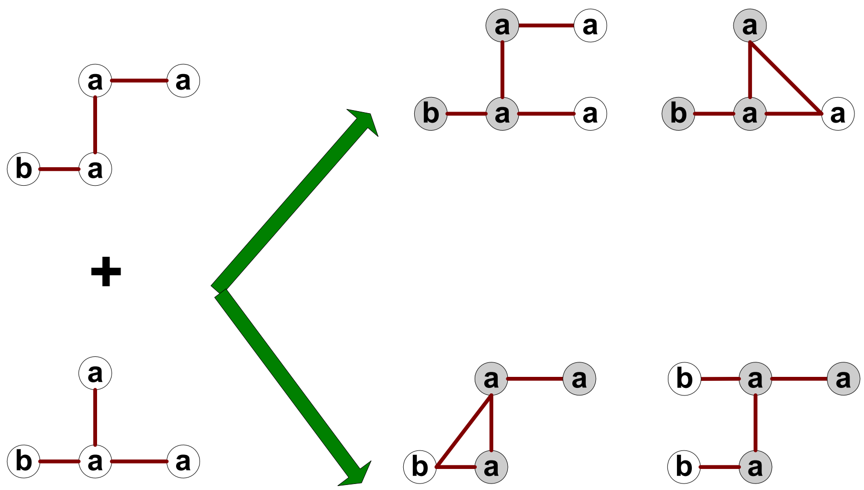
- Case 2: Core contains identical labels



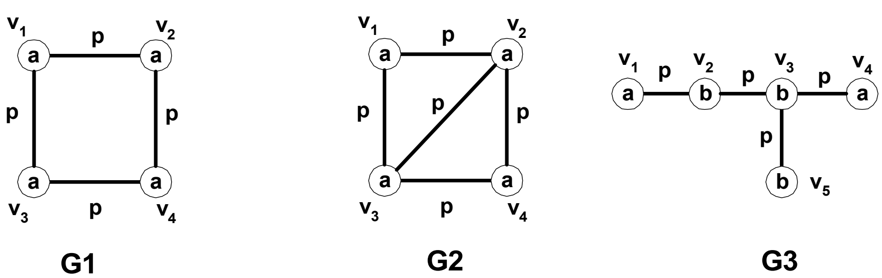
Core: The (k-1) subgraph that is common between the joint graphs

Multiplicity of Candidates (Edge growing)

- Case 3: Core multiplicity

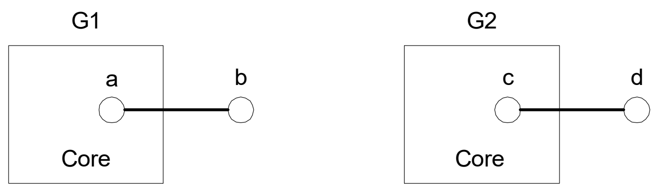


Topological Equivalence

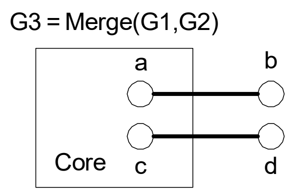


Candidate Generation by Edge Growing

- Given:

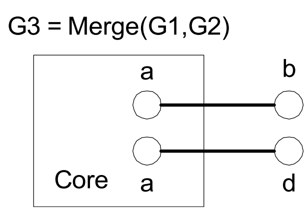


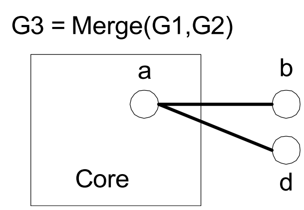
- Case 1: a ≠ c and b ≠ d



Candidate Generation by Edge Growing

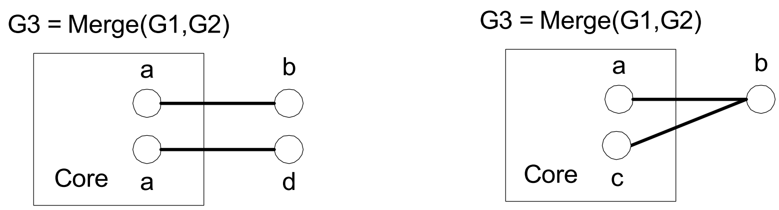
- Case 2: a = c and b ≠ d





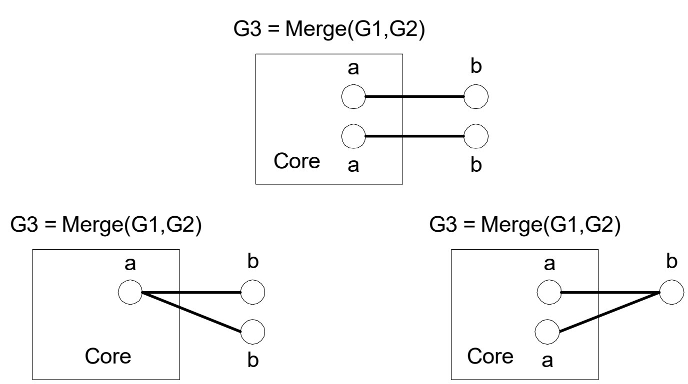
Candidate Generation by Edge Growing

- Case 3: a ≠ c and b = d



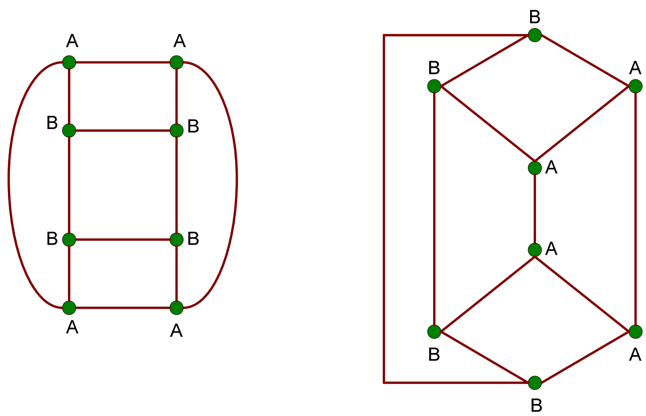
Candidate Generation by Edge Growing

- Case 4: a = c and b = d



Graph Isomorphism

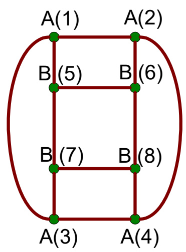
- A graph is isomorphic if it is topologically equivalent to another graph



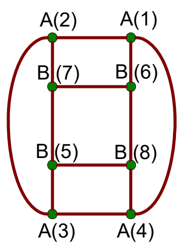
Graph Isomorphism

- Test for graph isomorphism is needed:
  - During candidate generation step, to determine whether a candidate has been generated
  - During candidate pruning step, to check whether its (*k-1*)-subgraphs are frequent
  - During candidate counting, to check whether a candidate is contained within another graph

Graph Isomorphism



	A(1)	A(2)	A(3)	A(4)	B(5)	B(6)	B(7)	B(8)
A(1)	1	1	1	0	1	0	0	0
A(2)	1	1	0	1	0	1	0	0
A(3)	1	0	1	1	0	0	1	0
A(4)	0	1	1	1	0	0	0	1
B(5)	1	0	0	0	1	1	1	0
B(6)	0	1	0	0	1	1	0	1
B(7)	0	0	1	0	1	0	1	1
B(8)	0	0	0	1	0	1	1	1

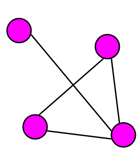


	A(1)	A(2)	A(3)	A(4)	B(5)	B(6)	B(7)	B(8)
A(1)	1	1	0	1	0	1	0	0
A(2)	1	1	1	0	0	0	1	0
A(3)	0	1	1	1	1	0	0	0
A(4)	1	0	1	1	0	0	0	1
B(5)	0	0	1	0	1	0	1	1
B(6)	1	0	0	0	0	1	1	1
B(7)	0	1	0	0	1	1	1	0
B(8)	0	0	0	1	1	1	0	1

- **The same graph can be represented in many ways**

Graph Isomorphism

- Use canonical labeling to handle isomorphism
  - Map each graph into an ordered string representation (known as its code) such that two isomorphic graphs will be mapped to the same canonical encoding
  - Example:
    - Lexicographically largest adjacency matrix

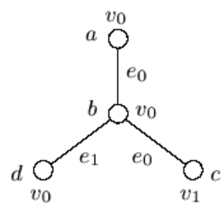


$$\longrightarrow \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

String: 011011 **Canonical: 111100**

Example of Canonical Labeling (Kuramochi & Karypis, ICDM 2001)

- Graph:



- Adjacency matrix representation:

id	a	b	c	d
label	v_0	v_0	v_1	v_0
a	0	e_0	0	0
b	e_0	0	e_0	e_1
c	0	e_0	0	0
d	0	e_1	0	0

Example of Canonical Labeling (Kuramochi & Karypis, ICDM 2001)

- Order based on vertex degree:

id	a	c	d	b
label	v_0	v_1	v_0	v_0
partition		0		1
a	0	0	0	e_0
c	0	0	0	e_0
d	0	0	0	e_1
b	e_0	e_0	e_1	0

- Order based on vertex labels:

id	d	a	c	b
label	v_0	v_0	v_1	v_0
partition	0		1	2
d	0	0	0	e_1
a	0	0	0	e_0
c	0	0	0	e_0
b	e_1	e_0	e_0	0

Example of Canonical Labeling (Kuramochi & Karypis, ICDM 2001)

- Find canonical label:

id	d	a	c	b
label	v_0	v_0	v_1	v_0
partition	0		1	2
d	0	0	0	e_1
a	0	0	0	e_0
c	0	0	0	e_0
b	e_0	e_1	e_0	0

id	a	d	c	b
label	v_0	v_0	v_1	v_0
partition	0		1	2
a	0	0	0	e_0
d	0	0	0	e_1
c	0	0	0	e_0
b	e_0	e_1	e_0	0

$0\ 0\ 0\ e_1\ e_0\ e_0$ > $0\ 0\ 0\ e_0\ e_1\ e_0$

(Canonical Label)