Data Mining
-- Special Data Mining

Instructor: Jen-Wei Huang

Office: 92528 in the EE building
jwhuang@mail.ncku

Mining Data Streams

- Data streams : continuous, ordered, changing, fast, huge amount
- Characteristics
  - Huge volumes of continuous data, possibly infinite
  - Fast changing and requires fast, real-time response
  - Data stream captures nicely our data processing needs of today
  - Random access is expensive—single scan algorithm (*can only have one look*)
  - Store only the summary of the data seen thus far
  - Most stream data are at pretty low-level or multi-dimensional in nature, needs multi-level and multi-dimensional processing

Data Stream Applications

- Telecommunication calling records
- Business: credit card transaction flows
- Network monitoring and traffic engineering
- Financial market: stock exchange
- Sensor, monitoring & surveillance: video streams, RFIDs
- Security monitoring
- Web logs and Web page click streams

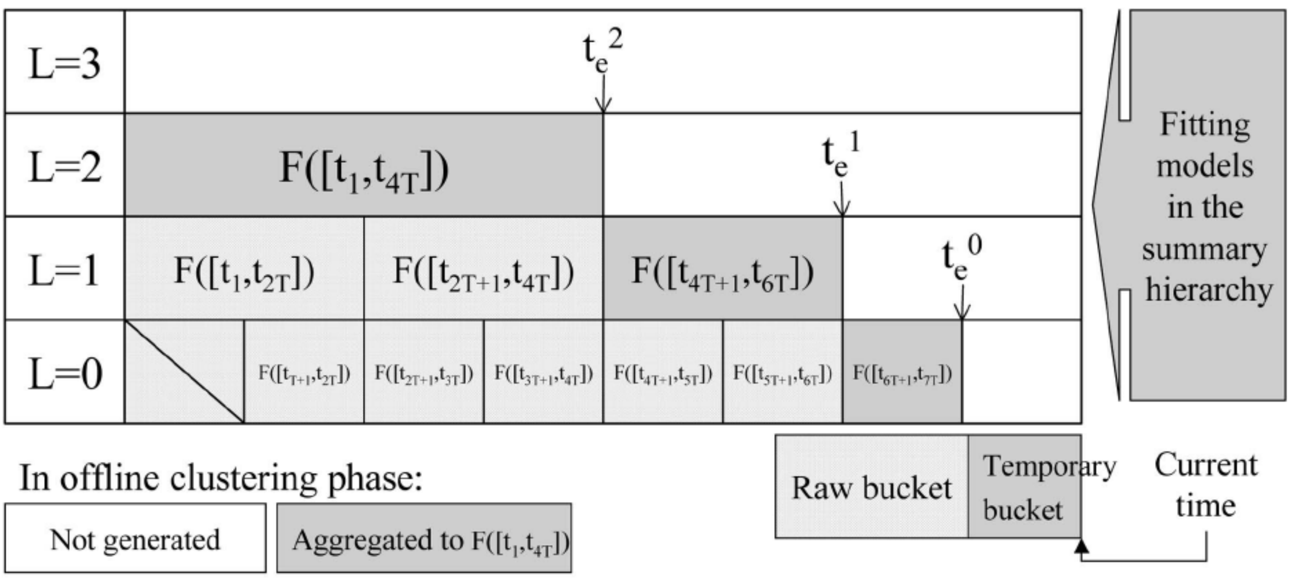
Challenges

- Queries are often continuous
  - Evaluated continuously as stream data arrives
  - Answer updated over time
- Queries are often complex
  - Beyond element-at-a-time processing
  - Beyond stream-at-a-time processing
  - Beyond relational queries (scientific, data mining, OLAP)
- Multi-level/multi-dimensional processing and data mining
  - Most stream data are at low-level or multi-dimensional in nature

Methodologies

- Methodology
  - Synopses (trade-off between accuracy and storage)
  - Use *synopsis data structure*, much smaller ($O(\log^k N)$ space) than their base data set ($O(N)$ space)
  - Compute an *approximate answer* within a *small error range* (factor ϵ of the actual answer)
- Major methods
  - Random sampling
  - Histograms
  - Sliding windows
  - Multi-resolution model
  - Sketches
  - Radomized algorithms

Wavelet Transform



Graph Mining

- Modeling
- Circuits
- Images
- Chemical compounds
- Biological networks
- Social networks
- Web structures

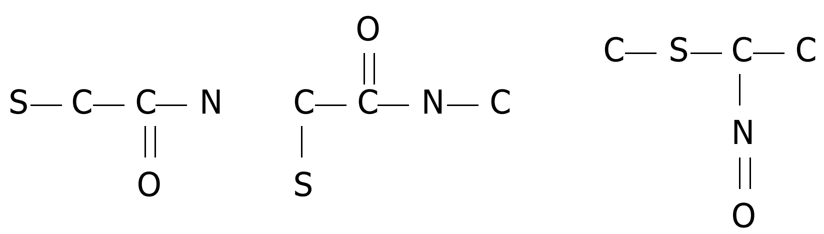
Categories

- Frequent subgraph patterns
- Graph classification
- Graph clustering
- Indices building
- Similarity search

Definition

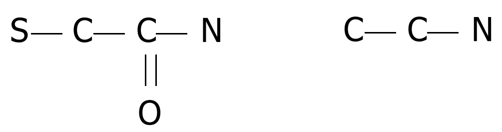
- A graph *g* is a subgraph of another graph *g'* if there exists a subgraph isomorphism from *g* to *g'*.
- Given a labeled graph data se $D = \{G_1, G_2, \ldots, G_n\}$, *support(g)* is the percentage of graphs in *D* where *g* is a subgraph
- A frequent graph is a graph whose support is no less than *min_sup*.

Example



(g_1) (g_2) (g_3)

Subgraphs :



Frequency = 2 Frequency = 3

* Example from Han's book

Methods

- Apriori-based method
  - AGM, FSG, path-join method
- Pattern-growth method
  - gSpan

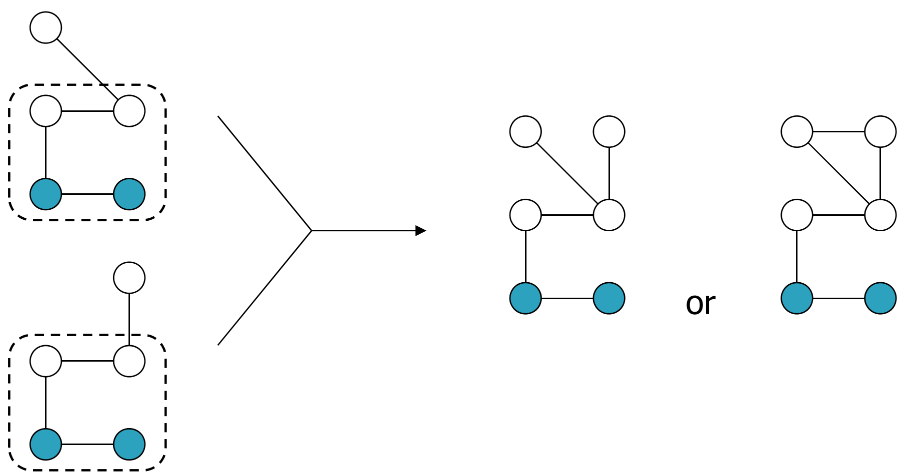
Apriori-based Method

- AprioriGraph(*D, min_sup,* S_k)
  1. $S_{k+1} \leftarrow \psi$
  2. For each size *(k+1)* graph *g* formed by merging frequent g_i and g_j in S_k
  3. If *g* is frequent in *D* and *g* is not in S_{k+1}
  4. Insert *g* into S_{k+1}
  5. If S_{k+1} *!=* ψ
  6. Apriori(*D, min_sup,* S_{k+1});
  7. Return;

Algorithm AGM

- AGM uses vertex-based candidate generation.
- Two size-*k* frequent graphs are joined if they have the same size-*(k-1)* subgraph
- Size is the number of vertices in the graph.

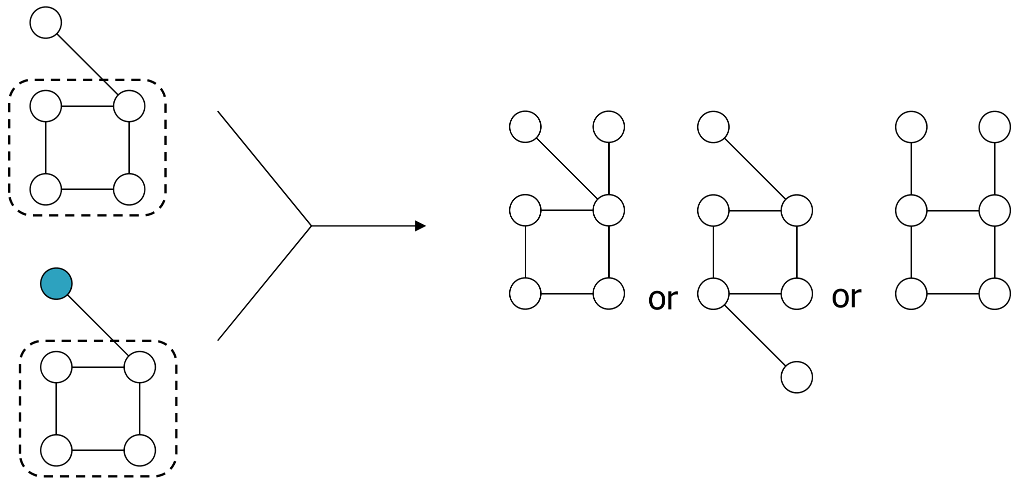
AGM join



Algorithm FSG

- FSG adopts edge-based candidate generation.
- Two size-*k* frequent graphs are merged if they share the same subgraph having *(k-1)* edges, which is called the core.
- Size is the number of edges in the graph.

FSG merge



Edge-disjoint path method

- A subgraph with k+1 disjoint paths is generated by joining subgraphs with k disjoint paths.
- Two paths are edge-disjoint if they do not share any common edge.

Pattern-Growth Method

- PatternGrowth(*g, D, min_sup, S*)
  1. If g is in S then return;
  2. Else insert g into S;
  3. Scan D to find all the edges such that g can be extended to $g \diamond_x e$
  4. For each frequent $g \diamond_x e$
  5. PatternGrowth($g \diamond_x e$, *D, min_sup, S*)
  6. Return;

Graph Extension

- A graph g can be extended by adding a new edge e. The new graph is denoted by $g \diamond_x e$.
- If e introduces a new vertex, the new graph is denoted by $g \diamond_{xf} e$, otherwise $g \diamond_{xb} e$, where f or b indicates a forward or backward extension.

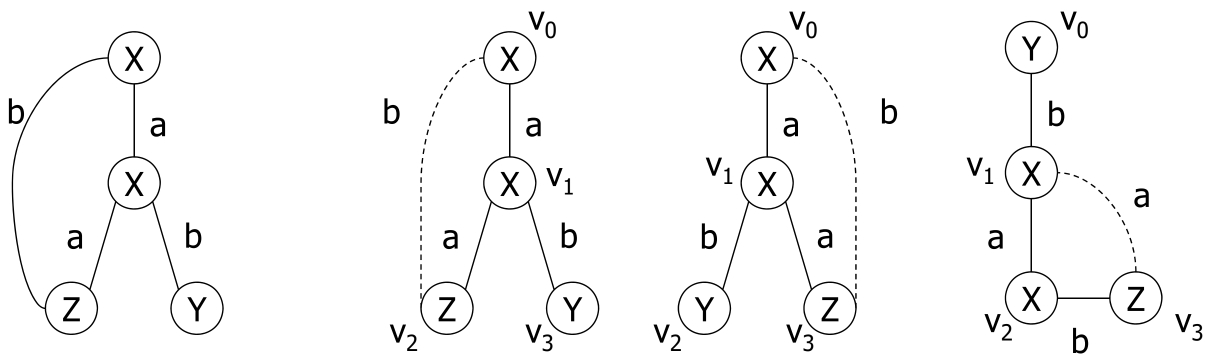
Duplicate Graph

- An n-edge graph can be extended from n different (n-1)-edge graphs.
- We call a graph that is discovered a second time a duplicate graph.
- The generation and detection of duplicate graph introduce huge overheads.
- gSpan algorithm is designed to remedy this problem.

gSpan Algorithm

- To traverse a graph, a starting vertex is randomly selected.
- The visited vertex set is expanded repeatedly until a full depth-first search tree is built.
- A DFS tree T of a graph G is called a DFS subscripting of G.

Example

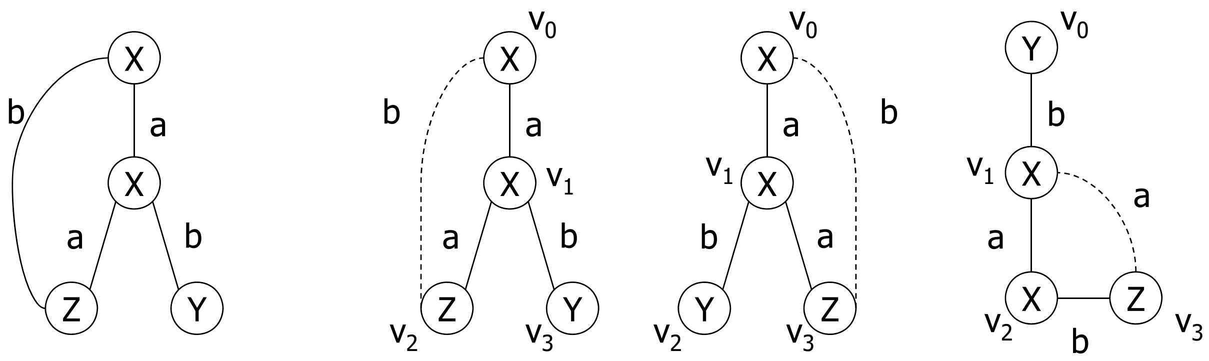


(G) (1) (2) (3)

Right-most path

- The root of T is set to v_0 and the last visited vertex is set to v_n.
- v_n is called the right-most vertex.
- The path from v_0 to v_n is called the right-most path.

Example



(G)

(1) (v_0,v_1,v_3)

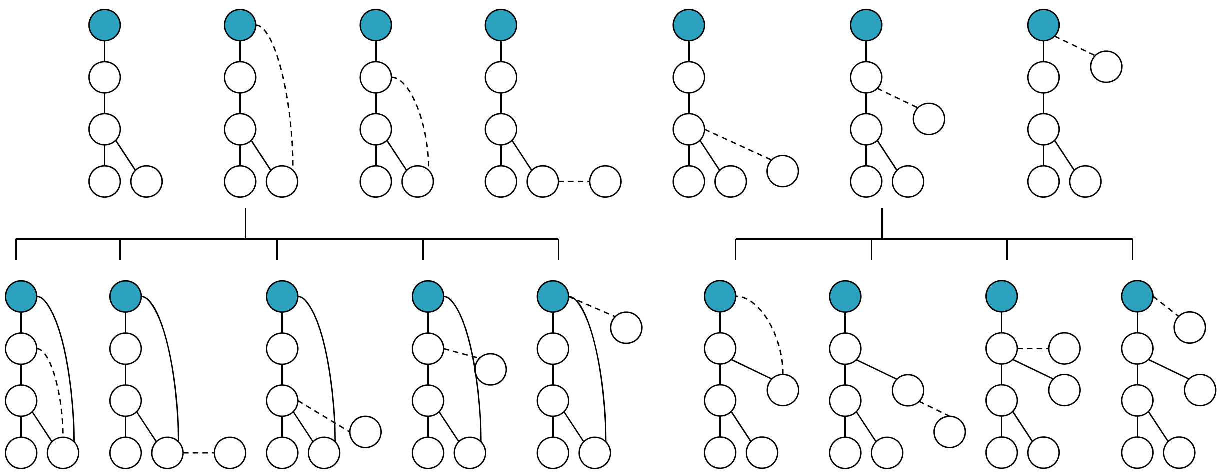
(2) (v_0,v_1,v_3)

(3) (v_0,v_1,v_2,v_3)

Right-most extension

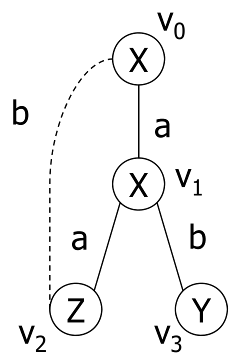
- A new edge e can be added in two ways :
  - Link between the right-most vertex (backward extension)
  - Introduce a new vertex on the right-most path (forward extension)
- They are called right-most extension, denoted by $G \diamond_r e$.

Example



Edge order

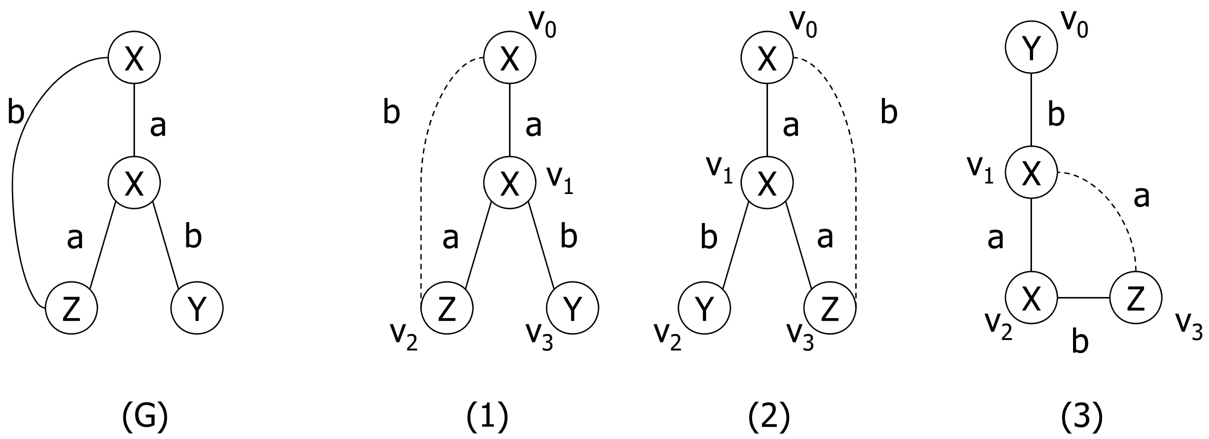
- Backward edges should appear before the forward ones of a vertex in the edge code.
- The forward edges are visited in the order of (0,1), (1,2), (1,3).
- A complete sequence is (0,1), (1,2), (2,0), (1,3).



DFS code

- Each edge is transformed into an edge code – $(i, j, l_i, l_{(i,j)}, l_j)$.
- For a given edge order, we can obtain an DFS code for a subscripting of a graph g.

Example



Example

Edge order	(1)	(2)	(3)
0	(0,1,X,a,X)	(0,1,X,a,X)	(0,1,Y,b,X)
1	(1,2,X,a,Z)	(1,2,X,b,Y)	(1,2,X,a,X)
2	(2,0,Z,b,X)	(1,3,X,a,Z)	(2,3,X,b,Z)
3	(1,3,X,b,Y)	(3,0,Z,b,X)	(3,1,Z,a,X)

DFS Lexicographic Order

- (1)<(2)<(3)
- The subscripting that generates the minimum DFS code is called the base subscripting.
- According to the base subscripting, gSpan creates a lexicographic search tree.

Other Interesting Issues

- Mining variant and constrained substructure patterns
- Mining closed frequent substructures
- Mining alternative substructure patterns
- Constraint-based mining
- Mining approximate frequent substructures

Other Interesting Issues

- Mining coherent substructures
- Mining dense substructure
- Graph indexing, similarity search, classification and clustering

Social Network Analysis

Mining Biomedical Data

Text Mining

Web Mining

- Web usage mining
- Web contents mining
- Web structure mining

Multimedia Data Mining

- Progressive Image Search and Recommendation System
  - http://pisar.cse.yzu.edu.tw

References

- Slides from Prof. J.-W. Han, UIUC
- Slides from Prof. M.-S. Chen, NTU
- Slides from Prof. W.-Z. Peng, NCTU